Hyperband-based Bayesian Optimization for Black-box Prompt Selection

Lennart Schneider [1] **Martin Wistuba** [2] **Aaron Klein** [3] **Jacek Golebiowski** [4] **Giovanni Zappella** [2]
Felice Antonio Merra [5]

Abstract

Optimal prompt selection is crucial for maximizing large language model (LLM) performance on downstream tasks, especially in black-box settings where models are only accessible via APIs. Black-box prompt selection is challenging due to potentially large, combinatorial search spaces, absence of gradient information, and high evaluation cost of prompts on a validation set. We propose `HbBoPs`, a novel method that combines a structural-aware deep kernel Gaussian Process with Hyperband as a multi-fidelity scheduler to efficiently select prompts. `HbBoPs` uses embeddings of instructions and few-shot exemplars, treating them as modular components within prompts. This enhances the surrogate model's ability to predict which prompt to evaluate next in a sample-efficient manner. Hyperband improves query-efficiency by adaptively allocating resources across different fidelity levels, reducing the number of validation instances required for evaluating prompts. Extensive experiments across ten diverse benchmarks and three LLMs demonstrate that `HbBoPs` outperforms state-of-the-art methods in both performance and efficiency.

1. Introduction

In recent years, pre-trained auto-regressive large language models (LLMs) have demonstrated remarkable capabilities in addressing a wide range of machine learning tasks involving natural language (Brown et al., 2020; Liu et al., 2023), such as Q&A (Joshi et al., 2017; Clark et al., 2018), text summarization (Koupaee & Wang, 2018), text generation (Hendrycks et al., 2021), and mathematical problem-solving (Hendrycks et al., 2021; Cobbe et al., 2021). As LLMs are highly sensitive to their input (Zhou et al., 2023; Honovich et al., 2023; Lu et al., 2022; Liu et al., 2023; Ye et al., 2023; Wu et al., 2024), performance on these tasks relies on prompt engineering, where the input is formatted within a carefully designed prompt that may include an *instruction*, a *few-shot exemplar*, and additional information.

The goal of static black-box prompt optimization and selection (Sun et al., 2022b; Chen et al., 2024; Lin et al., 2024; Wu et al., 2024; Shi et al., 2024) is to construct or identify a single prompt for a black-box LLM that, in expectation, performs well across all instances of a downstream task. This process involves evaluating different prompts on a validation set and using derivative-free techniques to guide optimization or selection. The static black-box setting allows for offline optimization, with the resulting optimal prompt being used for the downstream task.

While much research has focused on automatically *generating* new prompts (Sun et al., 2022b; Xu et al., 2022; Zhou et al., 2023; Chen et al., 2024; Fernando et al., 2024; Lin et al., 2024), there is growing interest in efficiently *selecting* prompts from a predefined pool of candidates (Shi et al., 2024). This is because many prompt optimization techniques involve generating a large candidate pool a priori before identifying the best prompt (Xu et al., 2022; Zhou et al., 2023; Fernando et al., 2024; Prasad et al., 2023). Moreover, recent empirical findings indicate that few-shot exemplars often contribute more strongly to prompt performance than the instruction of a prompt itself, with the best results typically achieved through the optimal selection of both components (Wan et al., 2024).

The task of black-box prompt selection is challenging. First, the search space can be extensive, as instructions and few-shot exemplars form a combinatorial set of candidates. Second, black-box LLMs make it impossible to directly optimize based on gradient information. Third, evaluating a prompt is time-consuming and costly, as each evaluation involves querying the LLM on multiple validation instances of a task. This calls for the development of selection methods that can efficiently explore the space of candidate prompts while keeping the total number of LLM calls low.

[1]Work done during an internship at Amazon Web Services, Berlin, Germany. [2]Amazon Web Services, Berlin, Germany. [3]ScaDS.AI, University of Leipzig, Germany; work done while at Amazon. [4]distil labs, Berlin, Germany; work done while at Amazon. [5]Cognism, Remote, Italy; work done while at Amazon. Correspondence to: Lennart Schneider <lennart.sch@web.de>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

Existing methods for prompt selection in this setting include MIPROv2 (Opsahl-Ong et al., 2024), EASE (Wu et al., 2024), TRIPLE-SH, and TRIPLE-GSE (Shi et al., 2024). We identify the following limitations among these approaches: (1) Except for MIPROv2, these methods are not explicitly designed to address the problem of *jointly* selecting instructions and few-shot exemplars. Specifically, EASE primarily focuses on exemplar selection, while TRIPLE-SH and TRIPLE-GSE focus on instruction selection. Although EASE can be applied to joint selection by treating the entire prompt as an unstructured block of text, it does not exploit the compositional structure of prompts. (2) No method is both *sample-efficient* (allowing for evaluating fewer prompts by relying on a surrogate model) and *query-efficient* (reducing the total number of LLM calls by not evaluating prompts on all available validation instances).

In this work, we propose HbBoPs (**H**yper**b**and-based **B**ayesian **o**ptimization for black-box **P**rompt **s**election) addressing these limitations. Our main contributions are the following: (1) We introduce a structural-aware deep kernel Gaussian Process (GP) that learns a low-dimensional prompt representation from separate embeddings of instructions and few-shot exemplars in an end-to-end fashion to predict prompt performance. (2) We adopt Hyperband (Li et al., 2018) as a *multi-fidelity* scheduler for prompt selection that governs the number of validation instances prompts are evaluated on. (3) We introduce a novel method, HbBoPs, that relies on our structural-aware deep kernel GP to make a Bayesian Optimization (BO) proposal within Hyperband and as a result is both sample- and query-efficient. (4) We compare HbBoPs against four baselines and four state-of-the-art methods across ten benchmarks and three LLMs, demonstrating that HbBoPs performs best in identifying a well-performing prompt after a given budget of total LLM calls but also exhibits strongest anytime performance during the selection process. (5) We perform an ablation study of the components of HbBoPs gaining insight into their inner workings and further demonstrate its robustness to the choice of the encoder model used to obtain embeddings.

2. Problem Statement

Let $\mathcal{I} = \{i_1, \ldots, i_l\}$ denote a finite set of instructions (task descriptions) and $\mathcal{E} = \{e_1, \ldots, e_m\}$ a finite set of few-shot exemplars. Note that by *exemplar* we refer to an ordered tuple of a given number of input-output examples of a task. Let $\mathcal{P} = \mathcal{I} \times \mathcal{E}$ be the set of prompts that are generated by combining each $i \in \mathcal{I}$ with each $e \in \mathcal{E}$.

Instructions can be generated either manually by experts or automatically by LLM-based methods, e.g., Automatic Prompt Engineering (APE; Zhou et al. 2023). Few-shot exemplars can be generated by selecting different input-output instances from a training set of the task.

A prompt $p \in \mathcal{P}$ is instantiated by combining it with a given task input or query $x \in \mathcal{X}$ for which the LLM, $h : (\mathcal{P} \times \mathcal{X}) \to \mathcal{Y}, h([p, x]) \mapsto \hat{y}$, produces an output $\hat{y} \in \mathcal{Y}$. We will use $h_p(x)$ to denote $h([p, x])$ as a shorthand.

We make no assumptions regarding the nature of the LLM and treat it as a black-box. The LLM returns output given input without any additional information, i.e., no access to model parameters, gradients, or token probabilities.

Having access to a validation set $\{(x_i, y_i)\}_{i=1}^{n_{\text{valid}}}$, evaluating a prompt is performed by comparing the ground truth output y_i to the output $\hat{y}_i = h_p(x_i)$ generated by the LLM based on a pointwise loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}, (y, \hat{y}) \mapsto l(y, \hat{y})$. This loss function quantifies how close the output generated by the LLM is to the ground truth. For example, a loss function based on the widely used exact match (Chang et al., 2024) scoring function is given by:

$$l(y, \hat{y}) = \begin{cases} 0 & \text{if } y = \hat{y} \\ 1 & \text{otherwise.} \end{cases} \tag{1}$$

Our task is to identify a single prompt $p \in \mathcal{P}$ that is optimal with respect to the loss in expectation:

$$\arg\min_{p \in \mathcal{P}} \mathbb{E}_{(x,y)\sim\mathbb{P}_{xy}} \left[l(y, h_p(x)) \right]. \tag{2}$$

Here, the expectation is taken over all input-output instances (x, y). In practice, however, Equation (2) can only be approximated based on the validation instances available:

$$f(p) \coloneqq \frac{1}{n_{\text{valid}}} \sum_{i=1}^{n_{\text{valid}}} l(y_i, h_p(x_i)). \tag{3}$$

We refer to this setting as the *static* setting, as we are searching offline for a single optimal prompt on the target downstream task. Note that due to the non-deterministic nature of LLMs (depending on temperature), f itself can in general only be observed with noise.

We will denote by

$$v = f(p) + \epsilon, \ \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{4}$$

the observed validation error of the LLM configured to use prompt p [1]. f is a black-box as the LLM is a black-box and no analytic description or gradient information is available.

The core challenge of static black-box prompt selection is balancing exploration (searching the prompt space $\mathcal{P}$) and efficiency (minimizing costly LLM evaluations). Our goal is to identify the best prompt using as few LLM calls for evaluation as possible.

[1] We model the observation noise as homoscedastic Gaussian for analytical convenience.

3. Method

We want to learn a surrogate model of the black-box function f in Equation (3) that predicts the validation error of prompts on a downstream task based on observed data collected during optimization. This surrogate is used to predict the validation error of the unevaluated prompts during the optimization process. We first describe how GPs can serve as surrogate models in black-box prompt selection (Section 3.1), but highlight the limitations of vanilla GPs on raw, high-dimensional prompt embeddings. To address this, we introduce a structural-aware deep kernel GP (Section 3.2) that relies on structural information of prompts via separate embeddings of their building blocks. We then adopt Hyperband (Li et al., 2018) for query-efficient multi-fidelity prompt selection (Section 3.3) and integrate our structural-aware deep kernel GP via a BO proposal, which results in our final `HbBoPs` algorithm (Section 3.4).

3.1. Gaussian Process as Surrogate Model

To learn a surrogate model, we collect, at each optimization step t, a set of design data $\mathcal{D}_t := \{(p_j, v_j)\}_{j=1}^t$, where each tuple consists of a prompt $p_j \in \mathcal{P}$ and its corresponding validation error v_j, as defined in Equation (4), recorded at the j-th previous step. This design data captures the history of prompts evaluated and their observed performance throughout the sequential optimization process. To learn a model that maps prompts to their validation errors, we embed each prompt into a d-dimensional numeric space, making use of pre-trained language encoders. Let $enc : \mathcal{P} \to \mathbb{R}^d, p \mapsto \mathbf{z}$ be the encoding function. We then augment the design data, $\mathcal{D}_t = \{(p_j, \mathbf{z}_j, v_j)\}_{j=1}^t$, where $\mathbf{z}_j$ is the embedding of prompt p_j. Since we are concerned with black-box prompt selection, it is natural to use embeddings as feature representations of prompts. Notably, recent work (Tang et al., 2024) has shown that (LLM) embeddings can in general serve as effective features for high-dimensional regression tasks, even in domains where the input data is not textual.

We want to use a GP as a surrogate model since it allows for flexible probabilistic modeling of black-box functions by returning a point estimate and well-calibrated uncertainty estimates in the form of a Gaussian posterior predictive distribution (Williams & Rasmussen, 2006). In the following, we assume a GP prior over f in the d-dimensional space of embedded prompts, $f(\mathbf{z}) \sim \mathcal{GP}(m, k)$; $\mathbf{f} \sim \mathcal{N}(m(\mathbf{Z}), k(\mathbf{Z}, \mathbf{Z}|\theta))$, where m is the prior mean function, usually set to zero, k is the covariance function depending on kernel parameters θ, and $\mathbf{Z}$ is a matrix of prompt embeddings.

Given the design data $\mathcal{D}_t$ and new prompts $\mathbf{p}_\star$ with their embeddings $\mathbf{Z}_\star$, the function $\mathbf{f}_\star$ is modeled as a random variable that is jointly Gaussian distributed with all previously observed validation errors $\mathbf{v} = (v_1, \ldots, v_t)$.

In short, this can be written as

$$\begin{bmatrix} \mathbf{v} \\ \mathbf{f}_\star \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{Z}) \; m(\mathbf{Z}_\star) \end{bmatrix}, \begin{bmatrix} \mathbf{K}_t & \mathbf{K}_\star \\ \mathbf{K}_\star^\intercal & \mathbf{K}_{\star\star}, \end{bmatrix} \right),$$

where $\mathbf{K}_t = k(\mathbf{Z}, \mathbf{Z}|\theta) + \sigma^2 \mathbf{I}_t$, $\mathbf{K}_\star = k(\mathbf{Z}, \mathbf{Z}_\star|\theta)$, and $\mathbf{K}_{\star\star} = k(\mathbf{Z}_\star, \mathbf{Z}_\star|\theta)$ are the kernel matrices.

The posterior predictive distribution under the (zero mean) GP is obtained as

$$\begin{aligned} \mathbb{E}[\mathbf{f}_\star|\mathbf{Z}, \mathbf{v}, \mathbf{Z}_\star] &= \mathbf{K}_\star^\intercal (\mathbf{K}_t)^{-1} \mathbf{v}, \\ \mathrm{cov}[\mathbf{f}_\star|\mathbf{Z}, \mathbf{Z}_\star] &= \mathbf{K}_{\star\star} - \mathbf{K}_\star^\intercal (\mathbf{K}_t)^{-1} \mathbf{K}_\star, \end{aligned} \tag{5}$$

where common choices for the kernel function k are given by the squared exponential kernel or variations of the Matérn kernel (see, e.g., Williams & Rasmussen, 2006, Chapter 4).

At this point, we could proceed and train a vanilla GP as outlined above on the d-dimensional space of embedded prompts. However, as stated in many previous works on BO (Kandasamy et al., 2015; Wang et al., 2016; Gardner et al., 2017; Eriksson et al., 2019; Eriksson & Jankowiak, 2021), GPs struggle with high-dimensional input such as that found in our design data $\mathcal{D}_t$, e.g., the dimensionality of BERT's (Devlin et al., 2019) [CLS] token embedding is already 768. In general, dimensionality reduction techniques such as principal component analysis (PCA) or random projections could be used. However, such techniques are unsupervised and will not yield a lower-dimensional representation that is aligned with the downstream performance of prompts (see also Figure 3 in Appendix A for an illustration). Additionally, using a single embedding of the whole prompt treated as a block of text ignores that the prompt is composed of different building blocks with distinct structural information. Below, we present our solution.

3.2. Structural-aware Deep Kernel

To learn a lower-dimensional representation of the embedded prompts aligned with the downstream task, we propose to use a deep kernel (Wilson et al., 2016) within the GP. We design a feature extractor, $\phi : \mathbb{R}^d \to \mathbb{R}^p$, $p \ll d$ to learn a flexible kernel transformation function $k(\phi(\mathbf{z}, \mathbf{w}), \phi(\mathbf{z}', \mathbf{w})|\theta)$, where θ and $\mathbf{w}$ are the parameters of the kernel and respectively the extractor.

Given that prompts consist of two distinct components (instructions and few-shot exemplars), we hypothesize that embedding these components separately can improve the deep kernel GP's (DK-GP) ability to use both structural and semantic differences effectively. Instructions typically exhibit consistent patterns across prompts, whereas few-shot exemplars are more variable due to diverse input-output pairs and flexible ordering. To address this, we propose learning a structural-aware latent representation of prompts. Our approach involves embedding the instructions $i \in \mathcal{I}$

and few-shot exemplars $e \in \mathcal{E}$ separately. Each component embedding is processed through distinct feed-forward neural networks, each consisting of two fully connected layers with ReLU activations, as defined below:

$\phi_{\text{enc}(\cdot)}$:
`Lin(d, 64)` $\rightarrow$ `ReLU()` $\rightarrow$ `Lin(64, 32)` $\rightarrow$ `ReLU()`

After processing the instructions and exemplars independently, we concatenate their outputs and input the combined representation into another feed-forward neural network to learn a joint latent representation:

$\phi_{(\phi_{\text{enc}(i)}, \phi_{\text{enc}(e)})}$:
`Lin(32` $\cdot$ `2, 32)` $\rightarrow$ `ReLU()` $\rightarrow$ `Lin(32, 10)`

During training the GP, we obtain both the optimal kernel parameters and the parameters of the neural network feature extractor ϕ by optimizing the log marginal likelihood criterion (up to constants):

$$\hat{\theta}, \hat{\mathbf{w}} = \arg\max_{\theta, \mathbf{w}} -\mathbf{v}^\intercal \mathbf{K}_t(\theta, \mathbf{w})^{-1}\mathbf{v} - \log|\mathbf{K}_t(\theta, \mathbf{w})| \quad (6)$$

An additional advantage of using a deep kernel GP is its ability to model non-stationary functions, as the feature extractor learns input-dependent transformations that allow the surrogate to capture varying levels of smoothness across the input space, which standard stationary kernels in vanilla GPs cannot accomplish (Li et al., 2024). Before illustrating how we use our structural-aware DK-GP during optimization to achieve sample-efficiency via a BO proposal, we explain how we ensure query-efficiency via Hyperband.

3.3. Hyperband for Multi-Fidelity Scheduling

To improve query-efficiency identified as one of the key limitations of existing works, we want to terminate the evaluation of poor-performing prompts early, saving cost during the evaluation process. Similarly to `TRIPLE` (Shi et al., 2024), we model the number of validation instances prompts are evaluated on as a fidelity parameter. Full-fidelity methods evaluate prompts on all validation instances, while multi-fidelity methods adaptively schedule evaluations on varying numbers of instances. `TRIPLE` implements Successive Halving (SH; Karnin et al. 2013) as a multi-fidelity scheduler. In contrast, we use Hyperband (HB; Li et al. 2018) as it will generally evaluate fewer prompts and hedges against a poorly configured SH as explained below.

Given a total budget of B LLM calls to evaluate prompts on validation instances, SH allocates a budget of $b = B/(|\mathcal{P}| \log_2(|\mathcal{P}|))$ to each prompt (see details in Appendix C). After having evaluated the prompts on b validation instances, the lower half of bad performing prompts is discarded, and the process repeats, doubling the number of calls for the remaining prompts in the next stage, until a single prompt remains.

This strategy is affected by the "budget vs. number of configurations" dilemma (Li et al., 2018), since, at the beginning of the algorithm, it is not clear if we should evaluate many (by default all) prompts on few instances (good exploration but noisy performance estimates) or few prompts on many instances (less exploration but accurate performance estimates). When many prompts need to be evaluated with a limited total budget, SH's initial budget is low, which risks discarding a prompt based on noisy performance estimation. HB in contrast hedges against a poor choice of the number of starting prompts and their budget by repeatedly running SH in different brackets with different numbers of starting prompts and starting budgets. This results in HB being robust under various scenarios without knowing the optimal resource allocation, making it ideal for prompt selection.

To tailor HB to prompt selection, we make the following design decisions (for an ablation study, see Appendix E): (1) We extend previous evaluations when advancing stages within a bracket, ensuring validation instances of higher stages subsume those of lower stages, and (2) return the prompt with the lowest validation error among those evaluated on the entire validation set.

Finally, we combine HB with our structural-aware DK-GP by employing a sequential BO proposal mechanism for candidate prompts in each bracket, which we outline next.

3.4. `HbBoPs`

`HbBoPs` combines HB with our structural-aware DK-GP. While the vanilla HB algorithm for prompt selection samples prompts uniformly at random, `HbBoPs` replaces the random proposal mechanism of HB with a sequential BO proposal (highlighted in gray in Algorithm 1). This is similar to the approach proposed by Falkner et al. (2018) for hyperparameter optimization. During the execution of HB, `HbBoPs` trains the GP on a subset of the design data $\mathcal{D}_{t|b}$ for a given fidelity-level b. We use the highest fidelity b for which "enough" observations are available. This design decision stems from the observation that validation errors are estimated more accurately with more instances (see Appendices B and C). Importantly, we train the GP entirely online during the selection process and are not relying on a pre-trained surrogate model.

After training the GP on $\mathcal{D}_{t|b}$, we select the next candidate prompt p_{t+1} by maximizing the Expected Improvement (EI; Kushner 1964; Mockus et al. 1978; Jones et al. 1998) acquisition function:

$$\begin{aligned} \alpha_{\text{EI}}(p|\mathcal{D}_{t|b}) &:= \mathbb{E}[\max\{v_{\min,b} - f(\mathbf{z}_p), 0\}] \\ p_{t+1} &= \arg\max_{p \in \mathcal{P}} \alpha_{\text{EI}}(p|\mathcal{D}_{t|b}), \end{aligned} \quad (7)$$

Algorithm 1 HbBoPs

input n_{valid}, $b_{\min}$ (lower limit to #validation instances), η (halving parameter)
  $r = n_{\text{valid}}/b_{\min}$
  $s_{\max} = \lfloor \log_\eta(r) \rfloor$
  $B = (s_{\max}+1)n_{\text{valid}}$
  for $s \in \{s_{\max}, s_{\max}-1, \dots, 0\}$ **do**
    $n = \left\lceil \frac{B}{n_{\text{valid}}} \frac{\eta^s}{(s+1)} \right\rceil$
    $b = n_{\text{valid}}\eta^{-s}$
    $P = \{\}, V = \{\}$
    for $j \in \{0, \dots, n-1\}$ **do**
      $p =$ get_prompt()
      $v =$ get_validation_error(p, b)
      $P \leftarrow P \cup \{p\}, V \leftarrow V \cup \{v\}$
    end for
    $P =$ top_k$(P, V, \lfloor n/\eta \rfloor)$
    for $i \in \{1, \dots, s\}$ **do**
      $n_i = \lfloor n\eta^{-i} \rfloor$
      $b_i = b\eta^i$
      $V = \{$get_validation_error$(p, b_i) : p \in P\}$
      $P =$ top_k$(P, V, \lfloor n_i/\eta \rfloor)$
    end for
  end for
output Prompt with the lowest validation error evaluated on the whole validation set

In words, given the incumbent (the best-performing prompt evaluated at the highest fidelity level so far) and its validation error $v_{\min,b}$ at fidelity level b, the EI quantifies the expected improvement of a candidate prompt over the incumbent (considering only actual improvement due to the max operator), based on the GP's posterior predictive distribution (Equation (5)). Since our search space is given by a finite set of candidate prompts, we can evaluate the EI exhaustively for all candidate prompts.

4. Experimental Setup

4.1. Benchmark Tasks

We benchmark HbBoPs on ten tasks commonly used for LLM evaluation (Zhou et al., 2023; Lin et al., 2024; Chen et al., 2024; Wu et al., 2024; Shi et al., 2024). *AI2's Reasoning Challenge (ARC)* (Clark et al., 2018): multiple-choice Q&A problems; *Grade School Math 8K* (Cobbe et al., 2021): math problems taking between two and eight steps to solve; *Eight Tasks from the BBII subset of the BIG-bench and instruction induction benchmarks* (Srivastava et al., 2023; Honovich et al., 2023) used in Zhou et al. (2023); Wu et al. (2024); Shi et al. (2024): antonyms, larger animal, negation, second word letter, sentiment, object counting, orthography starts with, and word unscrambling. Task statistics are reported in Table 5 in Appendix D.

4.2. Methods

We compare HbBoPs against full-fidelity and multi-fidelity methods described in Table 1. Additional details on the methods are reported in Section 6 and Appendix D. RS is a simple random search. All methods that rely on embeddings of prompts use BERT's [CLS] token embedding to ensure fair comparison. All full-fidelity BO methods (vanilla BO, HDBO, BOPCA) use an ARD Matérn $5/2$ kernel and Expected Improvement as acquisition function and normalize inputs to the unit cube and standardize outputs to have zero mean and unit variance. HDBO is a simple but well-performing high-dimensional BO algorithm using adjusted priors on GP kernel and likelihood parameters, as described in Hvarfner et al. (2024). BOPCA uses a ten component PCA inspired by Zhang et al. (2024). We run MIPROv2, NUCB (Zhou et al., 2020) as used by EASE ($\nu = 0.1$), TRIPLE-SH and TRIPLE-GSE as implemented in their publicly available code bases. All full-fidelity methods use the same initial design of ten prompts sampled uniformly at random. HbBoPs uses an ARD Matérn $5/2$ kernel, normalizes inputs to the unit cube and standardizes outputs. We always train the DK-GP on the highest fidelity level for which at least four observations are available. To optimize the log marginal likelihood in Equation (6), we use AdamW (Loshchilov & Hutter, 2019) with learning rate $= 0.01$, maximum number of epochs $= 3000$, and early termination with a patience $= 10$. Within the HB schedule, we use a lower limit on the number of validation instances $b_{\min} = 10$ and a halving parameter $\eta = 2.0$.

4.3. Experimental Protocol

For each task, we generate a search space $\mathcal{P}$ of candidate prompts by combining five task-specific instructions with 50 few-shot exemplars. Instructions are generated using APE's forward mode (Zhou et al., 2023), where Claude 3 Sonnet (Anthropic, 2024) generates instructions based on ten input-output samples from each task's training set. For exemplars, we sample 25 sets of *five* input-output instances from the training set of each task, then permute each set twice to create 50 exemplar tuples, allowing assessment of example ordering effects. We fix the five-shot setting throughout our experiments to ensure consistency with prior work (Wu et al., 2024). The final $|\mathcal{P}| = 250$ prompts are formed by the Cartesian product of instructions and exemplars.

For LLMs, we use Claude 3 Haiku (Anthropic, 2024), LLAMA3 8B Instruct (Llama Team, AI @ Meta, 2024), and Mistral 7B Instruct (Jiang et al., 2023).

For each benchmark scenario (a given task and LLM), we run each method for a total budget of 25 full-fidelity evaluations (i.e., being allowed as many LLM calls as 25 prompts evaluated on all validation instances require for a given task) to mimic a budget-constrained scenario. We use the number

Table 1. Overview of baselines, competitors and our HbBoPs in the static black-box prompt selection setting.

Method	Fidelity Level	Efficiency		Surrogate Model	Bandit Algorithm	Prompt Representation
		sample	query			
RS	Full	-	-	-	-	p
Vanilla BO	Full	✓	-	vanilla GP	-	$enc(p)$
HDBO	Full	✓	-	GP (Hvarfner et al., 2024)	-	$enc(p)$
BOPCA	Full	✓	-	vanilla GP	-	PCA($enc(p)$) (Zhang et al., 2024)
EASE (Wu et al., 2024)	Full	✓	-	NN	NUCB	$enc(p)$
MIPROv2 (Opsahl-Ong et al., 2024)	Full	✓	-	TPE	-	ID_i ID_e
TRIPLE-SH (Shi et al., 2024)	Multi	-	✓	-	SH	p
TRIPLE-GSE (Shi et al., 2024)	Multi	-	✓	LM/GLM	GSE	$enc(p)$
HbBoPs (ours)	Multi	✓	✓	structural-aware DK-GP	HB	$enc(i), enc(e)$

of LLM calls as our cost metric, rather than actual monetary cost, since LLM calls are a directly interpretable and model-agnostic measure of cost that allows for aggregating over different benchmark scenarios. We repeat each method run 30 times on each benchmark scenario. We evaluate prompts using the loss function described in Equation (1) which is based on the exact match scoring function.

5. Results

We report the validation and test errors computed for the best prompt identified by each method given a specific budget. For instance, given *GSM8k* with a validation set of 1319 instances, a budget of 0.25 means that we report the results of the methods after performing $\lceil 0.25 \cdot 25 \cdot 1319 \rceil = 8244$ LLM calls. Therefore, full-fidelity methods always start after having executed a fraction of $1/25$ total LLM calls.

5.1. Analysis of Overall Performance

We start by analyzing the overall performance of the methods averaged across all benchmark tasks and LLMs. To allow for averaging results, we normalize validation and test errors for each benchmark scenario by the performance of the worst and best prompt. Figure 1 visualizes the results. We observe that HbBoPs outperforms all full-fidelity and multi-fidelity methods, particularly in terms of anytime performance on both the validation and test set.

Beginning with an analysis of test error at full budget (i.e., a fraction of LLM calls equal to 1.0), we can see that our HbBoPs on average outperforms all full-fidelity and multi-fidelity approaches with an average normalized test error of 0.150. In detail, we observe that all full-fidelity methods surpass the RS baseline (0.214) with the following errors: Vanilla BO (0.211), MIPROv2 (0.198), EASE (0.195), BOPCA (0.192), and HDBO (0.185). However, they all have higher error values than HbBoPs (0.150). Additionally, HbBoPs also outperforms all multi-fidelity methods. Although both TRIPLE-GSE (0.158) and TRIPLE-SH (0.159) exhibit superior performance compared to their best-in-class full-fidelity counterpart, i.e., HDBO (0.185), they on average have identified prompts that yield error values higher than the ones obtained for HbBoPs's prompts.

Table 2. Median relative validation and test improvement of HbBoPs over TRIPLE-SH across ten benchmarks per LLM at different fractions of total LLM calls. IQR in parentheses.

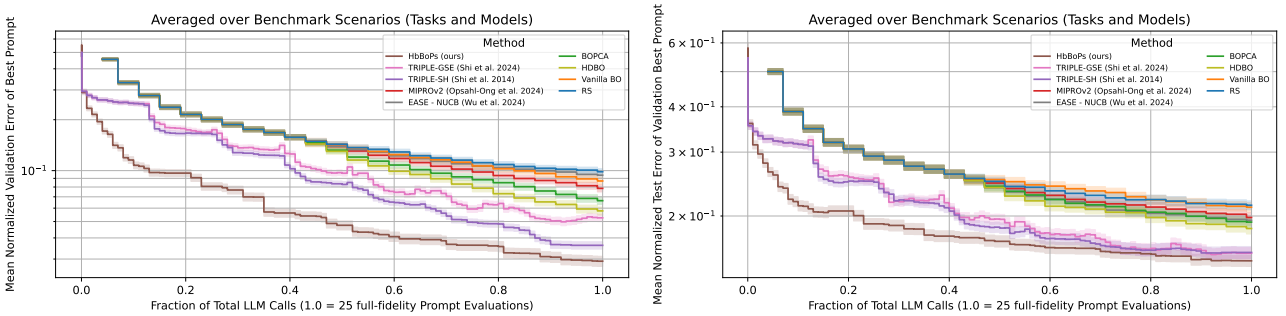
		Fraction of Total LLM Calls		
		0.25	0.50	1.00
Claude 3 Haiku	Valid	0.121 (0.145)	0.059 (0.093)	0.018 (0.066)
	Test	0.066 (0.105)	0.027 (0.045)	-0.006 (0.035)
LLAMA3 8B Instruct	Valid	0.120 (0.140)	0.042 (0.086)	0.001 (0.010)
	Test	0.036 (0.088)	0.010 (0.036)	0.000 (0.024)
Mistral7B Instruct	Valid	0.068 (0.079)	0.036 (0.036)	0.003 (0.044)
	Test	0.039 (0.022)	0.016 (0.033)	-0.001 (0.047)

Looking at the anytime performance with a more limited budget, e.g., a fraction of 0.25 LLM calls, we can confirm HbBoPs's improvements over the baselines. Indeed, HbBoPs on average outperforms HDBO, the best full-fidelity method, by approximately 35% and TRIPLE-SH, the best multi-fidelity method, by 24%. For additional statistical analyses, we refer to Appendix E.

5.2. Analysis of the Performance for each LLM

As shown in Section 5.1, TRIPLE-SH emerges as the strongest competitor. To assess whether HbBoPs's improvements over TRIPLE-SH are consistent across different LLMs, we present in Table 2 the median relative improvement over the ten benchmark tasks for each LLM.

The table reveals that HbBoPs consistently outperforms TRIPLE-SH in terms of both *anytime* validation and test error. For instance, when using Claude 3 Haiku, the average test error is reduced by a median factor of 0.066 and 0.027 at 0.25 and 0.50 of the total budget, respectively. While we further observe positive improvements over TRIPLE-SH on the validation set with a full budget, these gains are less pronounced on the test set. One possible reason for this discrepancy is that both methods, when given a sufficient budget, successfully identify prompts that have low or optimal validation error. However, optimal validation error does



(a) Validation (b) Test

Figure 1. Normalized error (log scale) of the best prompt per method, averaged over benchmarks. Lower is better. Ribbons represent SE.

Table 3. Normalized validation and test error of HbBoPs with different encoders at different fractions of total LLM calls averaged over all 30 benchmarks. SE in parentheses.

		Fraction of Total LLM Calls		
		0.25	0.50	1.00
BERT	Valid	0.081 (0.004)	0.048 (0.003)	0.029 (0.002)
	Test	0.190 (0.006)	0.170 (0.006)	0.150 (0.005)
MPNet	Valid	0.083 (0.004)	0.049 (0.003)	0.031 (0.002)
	Test	0.193 (0.006)	0.173 (0.006)	0.158 (0.006)
DistillRoBERTa	Valid	0.071 (0.003)	0.045 (0.002)	0.026 (0.002)
	Test	0.185 (0.006)	0.166 (0.006)	0.150 (0.005)

not necessarily guarantee optimal test error. We provide further analysis of the generalization gap between validation and test performance that is influenced by the size of the validation set in Appendix B.

5.3. Ablation Study

To better understand the contributions of the individual components of HbBoPs, we conduct a comprehensive ablation study. Our ablation focuses on four key aspects: the use of a GP with a deep kernel (DK-GP), the incorporation of a structural-aware DK-GP, the integration of HB for multi-fidelity scheduling, and the final HbBoPs. We aim to answer the following main research questions: (RQ1) Does a structural-aware DK-GP improve over a non-structural-aware DK-GP and a vanilla GP? (RQ2) Does multi-fidelity scheduling with HB improve over full-fidelity methods? (RQ3) Does combining our structural-aware DK-GP with HB improve over HB with a random proposal?

Figure 2 presents the average anytime normalized validation and test errors of the best prompt found by systematically removing specific components of our HbBoPs such that we can quantify their importance and answer above research questions. We focus on the validation error as shown in Figure 2a to describe results, as improvement of the validation error is a direct consequence of the change of components.

First, we observe that using a DK-GP on prompts embedded as a block of text (BoPs (non structural-aware DK-GP)) in a full-fidelity setting improves over vanilla BO by 11% and 38% at 0.5 and 1.0 budget with respect to the average normalized validation error. This highlights the importance of handling the high-dimensional embedded space properly. The structural-aware deep kernel (BoPs (structural-aware DK-GP)) further enhances performance by 9% and 13% at 0.5 and 1.0 budget, demonstrating the value of directly incorporating structural information into the GP which answers (RQ1). Note that the structural-aware DK-GP improves over HDBO by 19% and 8% with respect to final average normalized validation and test error.

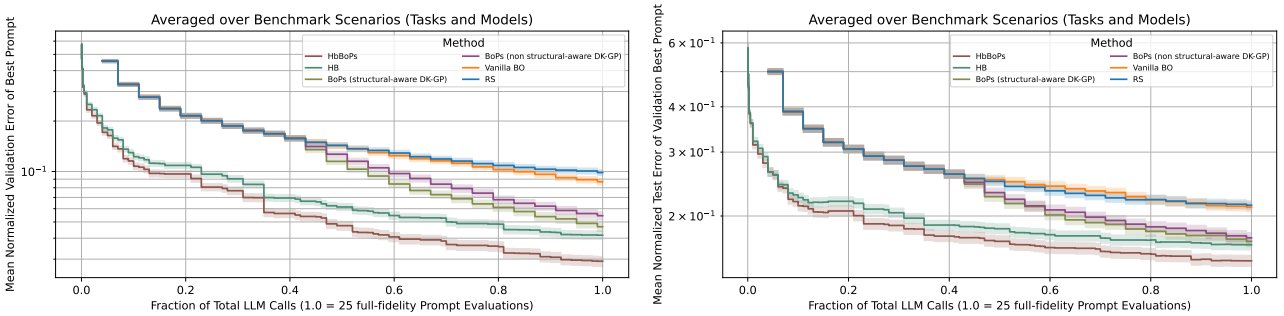
The integration of HB for multi-fidelity scheduling (HB using a random proposal mechanism) provides orthogonal boosts to both anytime and final performance. We observe improvements of 47% and 11% at 0.5 and 1.0 budget over the full-fidelity BoPs (structural-aware DK-GP) answering (RQ2).

Our complete HbBoPs further increases performance, achieving a 21% improvement over HB at 0.5 budget and 31% at 1.0 budget, answering (RQ3). Compared to our starting point of vanilla BO, HbBoPs demonstrates a substantial 66% improvement at 0.5 budget and 67% at 1.0 budget. For additional statistical analyses, we refer to Appendix E.

5.4. Analysis of Varying the Encoder

As HbBoPs relies on embeddings of prompts, we conduct a sensitivity analysis to evaluate the effect of different encoder models on the performance of HbBoPs. While our primary results were obtained using BERT's (Devlin et al., 2019) [CLS] token embedding, we extend our analysis to include two more encoder models that are MPNet (Song et al., 2020) and DistillRoBERTa (Liu et al., 2019). For each encoder model, we rerun the entire set of benchmarks.

We report in Table 3 the average normalized validation and test error for each encoder when used within HbBoPs for different fractions of total LLM calls over all 30 benchmark scenarios. Results show that HbBoPs maintains consistent validation and test error across all encoder models, indicat-



(a) Validation (b) Test

Figure 2. Normalized error (log scale) of the best prompt per `HbBoPs` ablation variant, `RS`, and vanilla `BO`, averaged over benchmarks. Lower is better. Ribbons represent SE.

ing robustness to the choice of encoder. This is expected, as none were specifically fine-tuned for predicting prompt performance. `HbBoPs`'s effectiveness stems from its ability to learn a mapping from prompts to performance through the structural-aware DK-GP, provided the embeddings capture meaningful distinctions between prompts.

6. Related Work

Automating Prompt Engineering. Recent work has been concerned with the general topic of automating prompt engineering. This work can be classified into *prompt optimization*, i.e., automating the creation of prompts (Prasad et al., 2023; Sun et al., 2022a; Zhou et al., 2023; Xu et al., 2022; Diao et al., 2023; Chen et al., 2024; Lin et al., 2024; Fernando et al., 2024; Pryzant et al., 2023; Guo et al., 2024; Pan et al., 2024; Schnabel & Neville, 2024; Shen et al., 2023; Hu et al., 2024), and *prompt selection*, i.e., finding the best prompt within a finite candidate set (Wu et al., 2024; Shi et al., 2024; Opsahl-Ong et al., 2024; Do et al., 2024).

Another dimension to categorize the related literature is given by the *white-box* vs. *black-box* setting. The white-box setting assumes access to the LLM, so that gradient-based methods for prompt optimization or selection are applicable (Shin et al., 2020). The black-box setting assumes no access to the LLM which only returns output given input (Sun et al., 2022b; Diao et al., 2023).

Finally, another differentiation is given by the *static* vs. *dynamic* setting. The goal of the static setting (Wu et al., 2024; Shi et al., 2024; Khattab et al., 2024) is to obtain a single prompt offline that in expectation performs well for all instances during test time. In contrast, the goal of the dynamic setting (Zhang et al., 2023; Do et al., 2024; Luo et al., 2024) is to select a prompt for each test instance (Rubin et al., 2022) in an online fashion.

Static Black-box Prompt Selection. Our work falls into the category of *static black-box prompt selection*. We summarize existing works below.

`MIPROv2` (Opsahl-Ong et al., 2024) is *DSPy*'s (Khattab et al., 2024) state-of-the-art *teleprompter* for joint instruction and few-shot exemplar selection. It searches over a finite set of candidate prompts by combining instructions with few-shot exemplars (which *DSPy* first constructs automatically). The method is a variant of BO using a Tree-structured Parzen Estimator (TPE; Bergstra et al. 2011) based on the categorical indices of instructions and exemplars (ID_i and ID_e) that compose a prompt. A downside is that learning a surrogate model based on indices does not use any semantic information of prompts, which may result in suboptimal predictive performance. Moreover, `MIPROv2` does not directly address query-efficiency. While *DSPy* can be configured to use a smaller random subset of the validation set to evaluate prompts, this risks suboptimal selection due to noisy performance estimates (see also Appendix B).

`EASE` proposed by Wu et al. (2024) mainly focuses on few-shot exemplar selection. It uses NeuralUCB (NUCB; Zhou et al. 2020) with embeddings of prompts as blocks of text as features, allowing for sequential evaluation of promising prompts based on the UCB criteria. `EASE`'s main contribution is to make the combinatorial problem of selecting examples to build the few-shot exemplar from an *extensive* training set computationally feasible. It prunes the candidate space using an optimal transport inspired heuristic before applying UCB. `EASE` is affected by query-inefficiency since it evaluates prompts on all validation instances (or a random subset, again risking suboptimal selection).

`TRIPLE` proposed by Shi et al. (2024) is a class of query-efficient algorithms for static black-box prompt selection using a multi-armed bandit approach. It makes use of Successive Halving (Karnin et al., 2013) or Generalized Successive Elimination (Azizi et al., 2022) to accelerate prompt evaluation by discarding poor-performing prompts early, reducing the need to evaluate all prompts on all validation instances. However, `TRIPLE-SH` is sensitive to the initial evaluation budget and may prematurely discard promising prompts due to noisy performance estimates. `TRIPLE-GSE` tries to mitigate this by (non-linear) modeling of expected prompt

performance using embeddings of prompts as blocks of text projected to a lower-dimensional space. While this approach introduces flexibility, the Generalized Successive Elimination algorithm has been formally analyzed only in the generalized linear setting (Azizi et al., 2022). Moreover, both TRIPLE-SH and TRIPLE-GSE begin by evaluating all prompts, whereas our HbBoPs employs a sample-efficient BO proposal to select candidate prompts for evaluation.

7. Conclusion

We introduced HbBoPs, a method for static black-box prompt selection in which prompts are composed of instructions and few-shot exemplars. HbBoPs employs a structural-aware deep kernel Gaussian Process to model the downstream performance of prompts based on separate embeddings of instructions and exemplars. This enables the identification of promising, unevaluated prompts during the selection process, making HbBoPs highly sample-efficient. Furthermore, HbBoPs integrates Hyperband as a multi-fidelity scheduler that governs the number of validation instances used for prompt evaluation, ensuring query-efficiency. In extensive experiments, we have demonstrated that HbBoPs improves upon baselines and state-of-the-art competitors in the limited budget regime while showing strong performance at any stage of the selection process.

While HbBoPs demonstrates strong performance, some limitations remain. Our method depends on embeddings from pre-trained encoders. Although we have demonstrated that HbBoPs is largely robust to the concrete choice of encoder model, obtaining embeddings induces some minor computational overhead. Additionally, our analysis focused on prompts composed of instructions and few-shot exemplars. While these components are highly relevant in practice, prompts may also include additional elements such as output guidance, formatting constraints, or other structural cues, which remain unexplored in this work. Nevertheless, the use of a deep kernel Gaussian Process as a surrogate model makes our framework, in principle, flexible enough to incorporate such additional prompt parameters by, for example, including categorical variables for formatting styles (e.g., JSON, Markdown).

In our experiments, we evaluated HbBoPs in a static setting with a fixed set of candidate prompts generated a priori. This enabled fair comparisons across baselines and methods operating in the static black-box prompt selection setting. However, we emphasize that HbBoPs can also be applied in more flexible prompt optimization settings where the candidate pool evolves over time. For example, it could be integrated with mutation-based prompt generation strategies (Fernando et al., 2024), where new prompts are generated iteratively, or with similar end-to-end prompt optimization pipelines (Pryzant et al., 2023; Yang et al., 2024).

Future work could extend HbBoPs to multiple objectives. Selecting more examples to include in a prompt may enhance performance but also increases response latency. Balancing the number and composition of examples in a few-shot exemplar introduces a trade-off between performance and efficiency, giving rise to a multi-objective optimization problem.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Anthropic. Claude 3 model card, 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Azizi, M. J., Kveton, B., and Ghavamzadeh, M. Fixed-budget best-arm identification in structured bandits. In de Raedt, L. (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pp. 2798–2804, 2022.

Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A. G., and Bakshy, E. BoTorch: A framework for efficient monte-carlo Bayesian optimization. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21524–21538, 2020.

Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. Algorithms for hyper-parameter optimization. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 24, pp. 2546–2554, 2011.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901, 2020.

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., and Xie, X. A survey on

evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

Chen, L., Chen, J., Goldstein, T., Huang, H., and Zhou, T. InstructZero: Efficient instruction optimization for black-box large language models. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 6503–6518, 2024.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? Try ARC, the AI2 Reasoning Challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Diao, S., Huang, Z., Xu, R., Li, X., Lin, Y., Zhou, X., and Zhang, T. Black-box prompt learning for pre-trained language models. *Transactions on Machine Learning Research*, 2023.

Do, V.-T., Hoang, V.-K., Nguyen, D.-H., Sabahi, S., Yang, J., Hotta, H., Nguyen, M.-T., and Le, H. Automatic prompt selection for large language models, 2024. URL https://arxiv.org/abs/2404.02717.

Eriksson, D. and Jankowiak, M. High-dimensional Bayesian optimization with sparse axis-aligned subspaces. In de Campos, C. and Maathuis, M. H. (eds.), *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161, pp. 493–503, 2021.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. Scalable global optimization via local Bayesian optimization. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32, 2019.

Falkner, S., Klein, A., and Hutter, F. BOHB: Robust and efficient hyperparameter optimization at scale. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 1437–1446, 2018.

Fernando, C., Banarse, D., Michalewski, H., Osindero, S., and Rocktäschel, T. Promptbreeder: Self-referential self-improvement via prompt evolution. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 13481–13544, 2024.

Gardner, J., Guo, C., Weinberger, K., Garnett, R., and Grosse, R. Discovering and exploiting additive structure for Bayesian optimization. In Singh, A. and Zhu, J. (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54, pp. 1311–1319, 2017.

Gardner, J., Pleiss, G., Weinberger, K. Q., Bindel, D., and Wilson, A. G. GPyTorch: Blackbox matrix-matrix Gaussian Process inference with GPU acceleration. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31, pp. 7576–7586, 2018.

Guo, Q., Wang, R., Guo, J., Li, B., Song, K., Tan, X., Liu, G., Bian, J., and Yang, Y. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. In *The Ninth International Conference on Learning Representations*, 2021.

Honovich, O., Shaham, U., Bowman, S. R., and Levy, O. Instruction induction: From few examples to natural language task descriptions. In *61st Annual Meeting of the Association for Computational Linguistics, ACL 2023*, pp. 1935–1952, 2023.

Hu, W., Shu, Y., Yu, Z., Wu, Z., Lin, X., Dai, Z., Ng, S.-K., and Low, B. K. H. Localized zeroth-order prompt optimization. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 86309–86345, 2024.

Hvarfner, C., Hellsten, E. O., and Nardi, L. Vanilla Bayesian optimization performs great in high dimensions. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 20793–20817, 2024.

Jamieson, K. and Talwalkar, A. Non-stochastic best arm identification and hyperparameter optimization. In Gretton, A. and Robert, C. C. (eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51, pp. 240–248, 2016.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Le Scao, T., Lavril, T., Wang, T., Lacroix, T., and El Sayed, W. Mistral 7B, 2023. URL https://arxiv.org/abs/2310.06825.

Jones, D. R., Schonlau, M., and Welch, W. J. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998.

Joshi, M., Choi, E., Weld, D., and Zettlemoyer, L. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Barzilay, R. and Kan, M.-Y. (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

Kandasamy, K., Schneider, J., and Poczos, B. High dimensional Bayesian optimisation and bandits via additive models. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 295–304, 2015.

Karnin, Z., Koren, T., and Somekh, O. Almost optimal exploration in multi-armed bandits. In Dasgupta, S. and McAllester, D. (eds.), *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pp. 1238–1246, 2013.

Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., Haq, S., Sharma, A., Joshi, T. T., Moazam, H., Miller, H., Zaharia, M., and Potts, C. DSPy: Compiling declarative language model calls into self-improving pipelines. In *The Twelfth International Conference on Learning Representations*, 2024.

Koupaee, M. and Wang, W. Y. WikiHow: A large scale text summarization dataset, 2018. URL https://arxiv.org/abs/1810.09305.

Kushner, H. J. 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, 1964.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

Li, L., Jamieson, K., Rostamizadeh, A., Gonina, E., Bentzur, J., Hardt, M., Recht, B., and Talwalkar, A. A system for massively parallel hyperparameter tuning. In Dhillon, I., Papailiopoulos, D., and Sze, V. (eds.), *Proceedings of Machine Learning and Systems*, volume 2, pp. 230–246, 2020.

Li, Y. L., Rudner, T. G. J., and Wilson, A. G. A study of Bayesian neural network surrogates for Bayesian optimization. In *The Twelfth International Conference on Learning Representations*, 2024.

Lin, X., Wu, Z., Dai, Z., Hu, W., Shu, Y., Ng, S.-K., Jaillet, P., and Low, B. K. H. Use your INSTINCT: INSTruction optimization usIng Neural bandits Coupled with Transformers. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, pp. 6503–6518, 2024.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A robustly optimized BERT pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Llama Team, AI @ Meta. The Llama 3 Herd of Models, 2024. URL https://arxiv.org/abs/2407.21783.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *The Seventh International Conference on Learning Representations*, 2019.

Lu, Y., Bartolo, M., Moore, A., Riedel, S., and Stenetorp, P. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 8086–8098, 2022.

Luo, M., Xu, X., Liu, Y., Pasupat, P., and Kazemi, M. In-context learning with retrieved demonstrations for language models: A survey. *Transactions on Machine Learning Research*, 2024.

Mockus, J., Tiesis, V., and Zilinskas, A. The application of Bayesian methods for seeking the extremum. *Towards Global Optimization*, 2(2):117–129, 1978.

Opsahl-Ong, K., Ryan, M. J., Purtell, J., Broman, D., Potts, C., Zaharia, M., and Khattab, O. Optimizing instructions and demonstrations for multi-stage language model programs. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 9340–9366, 2024.

Pan, R., Xing, S., Diao, S., Sun, W., Liu, X., Shum, K., Zhang, J., Pi, R., and Zhang, T. Plum: Prompt learning using metaheuristics. In Ku, L.-W.and Martins, A. and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2177–2197, 2024.

Prasad, A., Hase, P., Zhou, X., and Bansal, M. GrIPS: Gradient-free, edit-based instruction search for prompting large language models. In Vlachos, A. and Augenstein, I. (eds.), *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pp. 3845–3864, 2023.

Pryzant, R., Iter, D., Li, J., Lee, Y., Zhu, C., and Zeng, M. Automatic prompt optimization with "gradient descent" and beam search. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7957–7968, 2023.

Rubin, O., Herzig, J., and Berant, J. Learning to retrieve prompts for in-context learning. In Carpuat, M., de Marneffe, M.-C., and Meza Ruiz, I. V. (eds.), *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 2655–2671, 2022.

Schnabel, T. and Neville, J. Symbolic prompt program search: A structure-aware approach to efficient compile-time prompt optimization. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 670–686, 2024.

Shen, M., Ghosh, S., Sattigeri, P., Das, S., Bu, Y., and Wornell, G. Reliable gradient-free and likelihood-free prompt tuning. In Vlachos, A. and Augenstein, I. (eds.), *Findings of the Association for Computational Linguistics: EACL 2023*, pp. 2416–2429, 2023.

Shi, C., Yang, K., Chen, Z., Li, J., Yang, J., and Shen, C. Efficient prompt optimization through the lens of best arm identification. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 99646–99685, 2024.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4222–4235, 2020.

Song, K., Tan, X., Qin, T., Lu, J., and Liu, T.-Y. MPNet: Masked and permuted pre-training for language understanding. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 16857–16867, 2020.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., Kluska, A., Lewkowycz, A., Agarwal, A., Power, A., Ray, A., Warstadt, A., Kocurek, A. W., Safaya, A., Tazarv, A., Xiang, A., Parrish, A., Nie, A., Hussain, A., Askell, A., Dsouza, A., Slone, A., Rahane, A., Iyer, A. S., Andreassen, A. J., Madotto, A., Santilli, A., Stuhlmüller, A., Dai, A. M., La, A., Lampinen, A. K., Zou, A., Jiang, A., Chen, A., Vuong, A., Gupta, A., Gottardi, A., Norelli, A., Venkatesh, A., Gholamidavoodi, A., Tabassum, A., Menezes, A., Kirubarajan, A., Mullokandov, A., Sabharwal, A., Herrick, A., Efrat, A., Erdem, A., Karakaş, A., Roberts, B. R., Loe, B. S., Zoph, B., Bojanowski, B., Özyurt, B., Hedayatnia, B., Neyshabur, B., Inden, B., Stein, B., Ekmekci, B., Lin, B. Y., Howald, B., Orinion, B., Diao, C., Dour, C., Stinson, C., Argueta, C., Ferri, C., Singh, C., Rathkopf, C., Meng, C., Baral, C., Wu, C., Callison-Burch, C., Waites, C., Voigt, C., Manning, C. D., Potts, C., Ramirez, C., Rivera, C. E., Siro, C., Raffel, C., Ashcraft, C., Garbacea, C., Sileo, D., Garrette, D., Hendrycks, D., Kilman, D., Roth, D., Freeman, C. D., Khashabi, D., Levy, D., González, D. M., Perszyk, D., Hernandez, D., Chen, D., Ippolito, D., Gilboa, D., Dohan, D., Drakard, D., Jurgens, D., Datta, D., Ganguli, D., Emelin, D., Kleyko, D., Yuret, D., Chen, D., Tam, D., Hupkes, D., Misra, D., Buzan, D., Mollo, D. C., Yang, D., Lee, D.-H., Schrader, D., Shutova, E., Cubuk, E. D., Segal, E., Hagerman, E., Barnes, E., Donoway, E., Pavlick, E., Rodolà, E., Lam, E., Chu, E., Tang, E., Erdem, E., Chang, E., Chi, E. A., Dyer, E., Jerzak, E., Kim, E., Manyasi, E. E., Zheltonozhskii, E., Xia, F., Siar, F., Martínez-Plumed, F., Happé, F., Chollet, F., Rong, F., Mishra, G., Winata, G. I., de Melo, G., Kruszewski, G., Parascandolo, G., Mariani, G., Wang, G. X., Jaimovitch-Lopez, G., Betz, G., Gur-Ari, G., Galijasevic, H., Kim, H., Rashkin, H., Hajishirzi, H., Mehta, H., Bogar, H., Shevlin, H. F. A., Schuetze, H., Yakura, H., Zhang, H., Wong, H. M., Ng, I., Noble, I., Jumelet, J., Geissinger, J., Kernion, J., Hilton, J., Lee, J., Fisac, J. F., Simon, J. B., Koppel, J., Zheng, J., Zou, J., Kocon, J., Thompson, J.,

Wingfield, J., Kaplan, J., Radom, J., Sohl-Dickstein, J., Phang, J., Wei, J., Yosinski, J., Novikova, J., Bosscher, J., Marsh, J., Kim, J., Taal, J., Engel, J., Alabi, J., Xu, J., Song, J., Tang, J., Waweru, J., Burden, J., Miller, J., Balis, J. U., Batchelder, J., Berant, J., Frohberg, J., Rozen, J., Hernandez-Orallo, J., Boudeman, J., Guerr, J., Jones, J., Tenenbaum, J. B., Rule, J. S., Chua, J., Kanclerz, K., Livescu, K., Krauth, K., Gopalakrishnan, K., Ignatyeva, K., Markert, K., Dhole, K., Gimpel, K., Omondi, K., Mathewson, K. W., Chiafullo, K., Shkaruta, K., Shridhar, K., McDonell, K., Richardson, K., Reynolds, L., Gao, L., Zhang, L., Dugan, L., Qin, L., Contreras-Ochando, L., Morency, L.-P., Moschella, L., Lam, L., Noble, L., Schmidt, L., He, L., Oliveros-Colón, L., Metz, L., Senel, L. K., Bosma, M., Sap, M., Hoeve, M. T., Farooqi, M., Faruqui, M., Mazeika, M., Baturan, M., Marelli, M., Maru, M., Ramirez-Quintana, M. J., Tolkiehn, M., Giulianelli, M., Lewis, M., Potthast, M., Leavitt, M. L., Hagen, M., Schubert, M., Baitemirova, M. O., Arnaud, M., McElrath, M., Yee, M. A., Cohen, M., Gu, M., Ivanitskiy, M., Starritt, M., Strube, M., Swędrowski, M., Bevilacqua, M., Yasunaga, M., Kale, M., Cain, M., Xu, M., Suzgun, M., Walker, M., Tiwari, M., Bansal, M., Aminnaseri, M., Geva, M., Gheini, M., T, M. V., Peng, N., Chi, N. A., Lee, N., Krakover, N. G.-A., Cameron, N., Roberts, N., Doiron, N., Martinez, N., Nangia, N., Deckers, N., Muennighoff, N., Keskar, N. S., Iyer, N. S., Constant, N., Fiedel, N., Wen, N., Zhang, O., Agha, O., Elbaghdadi, O., Levy, O., Evans, O., Casares, P. A. M., Doshi, P., Fung, P., Liang, P. P., Vicol, P., Alipoormolabashi, P., Liao, P., Liang, P., Chang, P. W., Eckersley, P., Htut, P. M., Hwang, P., Miłkowski, P., Patil, P., Pezeshkpour, P., Oli, P., Mei, Q., Lyu, Q., Chen, Q., Banjade, R., Rudolph, R. E., Gabriel, R., Habacker, R., Risco, R., Millière, R., Garg, R., Barnes, R., Saurous, R. A., Arakawa, R., Raymaekers, R., Frank, R., Sikand, R., Novak, R., Sitelew, R., Bras, R. L., Liu, R., Jacobs, R., Zhang, R., Salakhutdinov, R., Chi, R. A., Lee, S. R., Stovall, R., Teehan, R., Yang, R., Singh, S., Mohammad, S. M., Anand, S., Dillavou, S., Shleifer, S., Wiseman, S., Gruetter, S., Bowman, S. R., Schoenholz, S. S., Han, S., Kwatra, S., Rous, S. A., Ghazarian, S., Ghosh, S., Casey, S., Bischoff, S., Gehrmann, S., Schuster, S., Sadeghi, S., Hamdan, S., Zhou, S., Srivastava, S., Shi, S., Singh, S., Asaadi, S., Gu, S. S., Pachchigar, S., Toshniwal, S., Upadhyay, S., Debnath, S. S., Shakeri, S., Thormeyer, S., Melzi, S., Reddy, S., Makini, S. P., Lee, S.-H., Torene, S., Hatwar, S., Dehaene, S., Divic, S., Ermon, S., Biderman, S., Lin, S., Prasad, S., Piantadosi, S., Shieber, S., Misherghi, S., Kiritchenko, S., Mishra, S., Linzen, T., Schuster, T., Li, T., Yu, T., Ali, T., Hashimoto, T., Wu, T.-L., Desbordes, T., Rothschild, T., Phan, T., Wang, T., Nkinyili, T., Schick, T., Kornev, T., Tunduny, T., Gerstenberg, T., Chang, T., Neeraj, T., Khot, T., Shultz, T., Shaham, U., Misra, V., Demberg, V., Nyamai, V., Raunak, V., Ramasesh, V. V., vinay uday prabhu, Padmakumar, V., Srikumar, V., Fedus, W., Saunders, W., Zhang, W., Vossen, W., Ren, X., Tong, X., Zhao, X., Wu, X., Shen, X., Yaghoobzadeh, Y., Lakretz, Y., Song, Y., Bahri, Y., Choi, Y., Yang, Y., Hao, S., Chen, Y., Belinkov, Y., Hou, Y., Hou, Y., Bai, Y., Seid, Z., Zhao, Z., Wang, Z., Wang, Z. J., Wang, Z., and Wu, Z. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

Sun, T., He, Z., Qian, H., Zhou, Y., Huang, X., and Qiu, X. BBTv2: Towards a gradient-free future with large language models. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 3916–3930, 2022a.

Sun, T., Shao, Y., Qian, H., Huang, X., and Qiu, X. Black-box tuning for language-model-as-a-service. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pp. 20841–20855, 2022b.

Tang, E., Yang, B., and Song, X. Understanding LLM embeddings for regression. *Transactions on Machine Learning Research*, 2024.

van der Maaten, L. and Hinton, G. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9(86): 2579–2605, 2008.

Wan, X., Sun, R., Nakhost, H., and Arı k, S. O. Teach better or show smarter? On instructions and exemplars in automatic prompt optimization. In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 58174–58244, 2024.

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(1):361–387, 2016.

Williams, C. K. I. and Rasmussen, C. E. *Gaussian Processes for Machine Learning*. MIT press Cambridge, MA, 2 edition, 2006.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. Deep kernel learning. In Gretton, A. and Robert, C. C. (eds.), *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51, pp. 370–378, 2016.

Wu, Z., Lin, X., Dai, Z., Hu, W., Shu, Y., Ng, S.-K., Jaillet, P., and Low, B. K. H. Prompt optimization with EASE? efficient ordering-aware automated selection of exemplars.

In Globerson, A., Mackey, L., Belgrave, D., Fan, A., Paquet, U., Tomczak, J., and Zhang, C. (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 122706–122740, 2024.

Xu, H., Chen, Y., Du, Y., Shao, N., Yanggang, W., Li, H., and Yang, Z. GPS: Genetic prompt search for efficient few-shot learning. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8162–8171, 2022.

Yang, C., Wang, X., Lu, Y., Liu, H., Le, Q. V., Zhou, D., and Chen, X. Large language models as optimizers. In *The Twelfth International Conference on Learning Representations*, 2024.

Ye, J., Wu, Z., Feng, J., Yu, T., and Kong, L. Compositional exemplars for in-context learning. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 39818–39833, 2023.

Zhang, H., He, J., Righter, R., and Zheng, Z. Language model prompt selection via simulation optimization, 2024. URL https://arxiv.org/abs/2404.08164.

Zhang, T., Wang, X., Zhou, D., Schuurmans, D., and Gonzalez, J. E. TEMPERA: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2023.

Zhou, D., Li, L., and Gu, Q. Neural contextual bandits with UCB-based exploration. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 11492–11502, 2020.

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., and Ba, J. Large language models are human-level prompt engineers. In *The Eleventh International Conference on Learning Representations*, 2023.

A. On the Latent Space of the Structural-Aware Deep Kernel Gaussian Process

As described in Section 3.1, unsupervised dimensionality reduction techniques such as PCA or random projections will not result in a lower-dimensional latent representation of prompt embeddings that is aligned with the downstream performance of prompts. To illustrate this, we perform the following experiment: We collect the validation error of 250 prompts on *GSM8K* (according to the splits described in Table 5) using LLAMA3 8B Instruct as LLM. We embed each prompt using the [CLS] token embedding of BERT ($d = 768$). We then split the prompts $p_1, \ldots, p_{250}$ and their corresponding validation errors $v_1, \ldots, v_{250}$ in a train (80%) and test set (20%). Using the train split, we perform a PCA and retain 10 principal components as features. Moreover, we train our structural-aware DK-GP introduced in Section 3.2 on the training split and extract the 10 latent features from the output of the feature extractor $\phi_{\left(\phi_{enc(i)}, \phi_{enc(e)}\right)}$. We visualize the raw 768 dimensional embedding features of prompts, the 10 dimensional PCA features and the 10 dimensional deep kernel features for the training split using a two component t-SNE (van der Maaten & Hinton, 2008) in the top row of Figure 3. The x- and y-axis represent the two t-SNE components, whereas color indicates the validation error of prompts (lighter color indicates better performance). We can see that for both the raw embedding features and the PCA features, it is difficult to visually discern any meaningful clusters or structure how closeness in feature space relates to closeness in performance space. For the deep kernel features, however, we can see that the feature space is well aligned with the performance space (well-performing prompts being closer together with a continuous transition into poorer performing prompts) - of course, this is on the training split on which the GP has been trained on, and therefore these results are not surprising. However, when looking at the test split (that was neither used to perform the PCA nor to train the GP) in the bottom row of Figure 3, we can see that similar conclusions as for the train split hold: The latent representation the feature extractor of the deep kernel has learned during training does generalize to the test split, and it has effectively learned a low-dimensional embedding of prompts aligned with the downstream task.

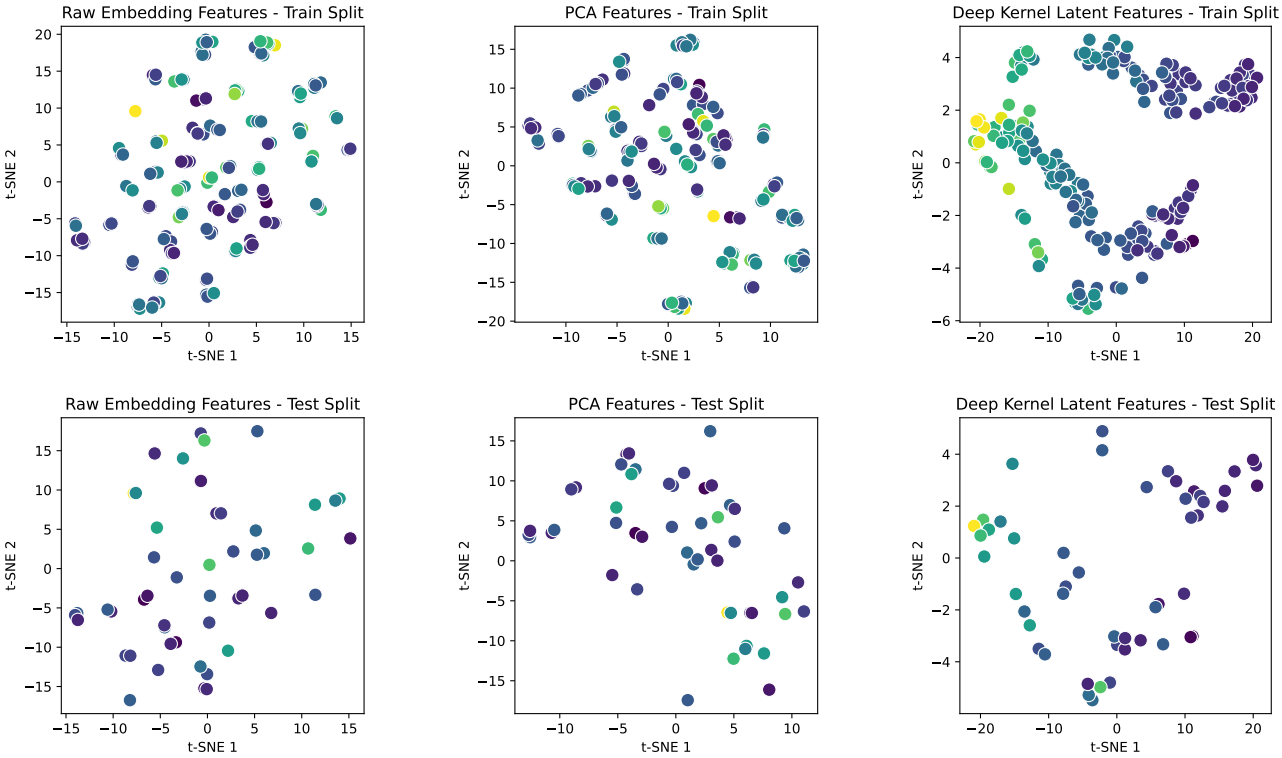


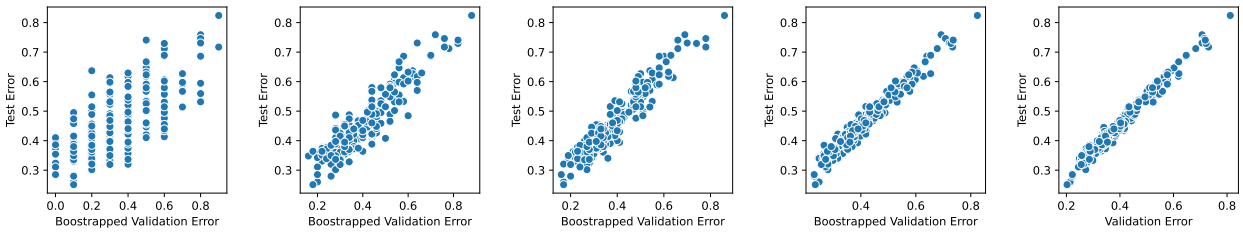
Figure 3. Visualization of the 768 dimensional BERT [CLS] token embeddings of prompts via a two component t-SNE. Left: Raw, unprocessed features. Middle: Features of a 10 component PCA solution. Right: Latent features (10 dimensional) from the feature extractor of our structural-aware DK-GP. Top row: Train split. Bottom row: Test split. Color indicates the performance of prompts for LLAMA3 8B Instruct on *GSM8K*.

B. On the Generalization from Validation to Test

When performing black-box prompt selection, we evaluate prompts on a validation set and iteratively improve over the current best prompt (the incumbent), trying to identify a better one. While progress on the validation set is expected (i.e., if we perform full-fidelity evaluations using the same validation instances for each prompt, the validation error of the incumbent will be monotonically decreasing as optimization progresses), it must not necessarily be the case that we also improve performance on a held-out test set of instances, i.e., the prompt identified as being validation optimal might not necessarily be optimal on the test set.

In this section, we provide additional insights regarding generalization gaps from validation to test performance. Recall that existing methods for black-box prompt selection (e.g., `EASE` and `MIPROv2`) are by design not query-efficient but evaluate all prompts on all validation instances or a random subset (e.g., Wu et al. 2024 used sub-sampled validation sets with as few as 20 instances). We now empirically demonstrate that using small random subsets during optimization is not a sensible choice, because this will result in increased variance of the estimate of the validation error, which prevents us from making correct decisions on the validation set. This can result in generalization issues when moving from the validation set to the test set.

We perform the following experiment: We collect the validation and test error (according to the splits described in Table 5) of 250 prompts on the *GSM8K* task using LLAMA3 8B Instruct as LLM. We vary the number of validation instances used to evaluate the performance of prompts via bootstrapping, using $k = 10, 50, 100, 500$ instead of the original $n_{\text{valid}} = 1319$ validation instances. Note that to compute the test error, we always use the full test set. In Figure 4, we provide scatter plots of the validation and test errors of the prompts with mean validation errors obtained via bootstrapping using $k = 10, 50, 100, 500$ validation instances vs. validation errors obtained on the full validation set of 1319 instances. We perform 1000 bootstrap replicates.



(a) $k = 10$. (b) $k = 50$. (c) $k = 100$. (d) $k = 500$. (e) Full validation set.

Figure 4. Scatter plots of the validation and test errors of 250 prompts evaluated with LLAMA3 8B Instruct on *GSM8K* using differently sized ($k = 10, 50, 100, 500$) bootstrap samples of validation instances (a) to (d) or the full validation set (e).

We can observe that if we use too few validation instances ($k = 10$ but also $k = 50$ and $k = 100$ to some extent) even if we select the validation optimal prompt, its test error can be far from optimal, because noise dominates the estimate of the validation error.

In Figure 5, we provide box plots of the bootstrapped variance estimates of the mean validation error over prompts when using smaller validation sets. As expected, the variance of the mean validation error can be substantial when using few validation instances. Note that the bootstrap results align with theoretical expectations under the assumption that the point-wise loss (based on exact match) is a binary random variable following a Bernoulli distribution with success probability p (corresponding to a loss of 0). In this case, the average validation error over n_{valid} instances follows a Binomial distribution, and the variance of the estimated validation error is given by $p(1-p)/n_{\text{valid}}$. For example, if a prompt has a true "success probability" of $p = 0.5$, then using $n_{\text{valid}} = 10$ validation instances yields an expected variance of $0.5^2/10 = 0.025$.

This has serious practical implications, depending on the variation in the performance of prompts on a downstream task. If the true performance of many prompts is similar, we cannot tell them apart based on their estimated validation error, as noise dominates the signal when using too few validation instances. Moreover, for benchmarking methods for black-box prompt selection this is highly relevant as when using too few validation instances, we cannot determine whether a generalization gap from the validation set to the test set results solely from noisy performance estimates or from internal method mechanisms (such as the optimal transport inspired heuristic employed by `EASE` to only consider examples that are similar to the validation set) which may further result in overfitting to the validation set.

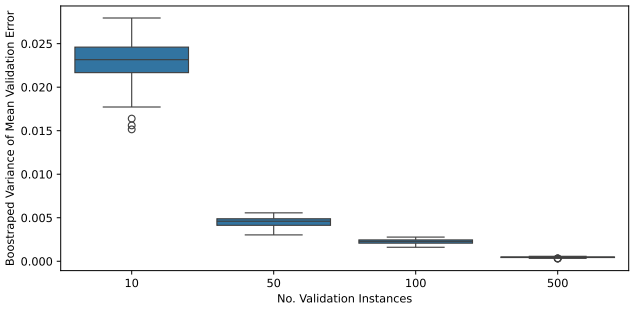


Figure 5. Box plots of the bootstrapped variance estimates of the mean validation error of 250 prompts evaluated with LLAMA3 8B Instruct on *GSM8K* varying the number of validation instances used to estimate the mean validation error.

The empirical results we have presented here further provide justification for using a multi-fidelity scheduler over the validation instances during prompt selection, such as Hyperband. Poor-performing prompts can be differentiated using few validation instances, however, well-performing prompts need to be evaluated on larger sets so that one can effectively tell their validation errors apart.

C. Multi-Fidelity over Validation Instances

In this section, we discuss how one can define a multi-fidelity schedule for prompt selection, in which the fidelity parameter is the number of validation instances. Let $\mathcal{D}_{\text{valid}} = \{(x_i, y_i)\}_{i=1}^{n_{\text{valid}}}$ denote the validation set containing n_{valid} input-output instances in natural language on which the LLM h_p configured to use a given prompt $p \in \mathcal{P}$ is evaluated. Recall our goal of identifying the optimal prompt: $\arg\min_{p \in \mathcal{P}} \mathbb{E}_{(x,y)\sim\mathbb{P}_{xy}} [l(y, h_p(x))]$. Here, the expectation is taken over all input output instances from a data-generating distribution $\mathbb{P}_{xy}$ and l is the pointwise loss function used to compare the LLM's output $h_p(x)$ to the ground truth y. We want to identify the best performing prompt while minimizing the number of LLM calls for evaluation, given the significant costs in both time, but especially query expenses associated with LLM black-box APIs.

In practice, we can only approximate this expectation by the full-fidelity evaluation on the whole validation set given by $\frac{1}{n_{\text{valid}}} \sum_{i=1}^{n_{\text{valid}}} l(y_i, h_p(x_i))$.

The idea of multi-fidelity techniques is to speed up and reduce the cost of the evaluation of prompts by using fewer validation instances during evaluation. Let $\mathcal{V} = \{1, \ldots, n_{\text{valid}}\}$ denote the index set corresponding to the indices of validation instances. A simple way to reduce cost of evaluation is to use a random subset, $\mathcal{S} \subset \mathcal{V}$ of validation instances: $\frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} l(y_i, h_p(x_i))$. This was for example done in the evaluation protocol of the benchmark study in Wu et al. (2024). However, it comes with the following downsides: (1) It is a priori not clear how many instances are needed to obtain an accurate estimate of the validation error of prompts as this depends on the concrete LLM, downstream task, and variation of the true performance over prompts. (2) Using a fixed (sub-)sample is inefficient because all prompts are evaluated on the same number of validation instances. However, poor-performing prompts can be identified using few validation instances, whereas well-performing prompts should be evaluated on many validation instances to differentiate between them and not risk suboptimal selection.

Shi et al. (2024) made the connection between multi-fidelity prompt selection and best arm identification from the multi-armed bandit literature. In this setting, "pulling an arm" refers to evaluating a prompt on a validation instance. The goal is to identify the best arm within a limited budget of evaluations. A well-known algorithm from the bandit literature is given by Successive Halving (SH, Karnin et al. 2013; Jamieson & Talwalkar 2016). In our setting of prompt selection, the idea of SH is to efficiently identify the best performing prompt (arm) under a given budget constraint of total LLM calls (pulls). Given a total budget B of LLM calls and $n := |\mathcal{P}|$ prompts, SH starts by allocating a budget of $b := B/(n \log_2(n))$ LLM calls to each prompt. After having evaluated each prompt on b instances, the lower half of bad performing prompts are discarded and the process repeats, doubling the number of LLM calls for the remaining prompts in the next stage. This process in general repeats until a single prompt remains. SH can be employed in both a stochastic and non-stochastic setting. The stochastic setting (Karnin et al., 2013) is formally characterized by the following: (1) Losses are i.i.d. samples from a probability distribution. (2) Each arm has a fixed expected loss μ_p. (3) The goal is to identify the arm with the lowest expected loss.

Jamieson & Talwalkar (2016) introduced SH in the non-stochastic setting when applying it to the problem of hyperparameter optimization, where the budget can, for example, be the size of the training set used to train an algorithm or the number of epochs used to train a neural network. This non-stochastic setting formally is characterized by the following: (1) Losses are real numbers chosen by an oblivious adversary. (2) Each arm has a limit ν_p of its loss sequence as the number of evaluations go to infinity. (3) The goal is to identify the arm with the lowest loss limit.

Within the non-stochastic setting, Li et al. (2018) introduced Hyperband (HB) although it is in general also applicable to the stochastic setting. Recall that SH requires the overall budget B and the number of prompts n as input parameters. These determine the starting budget b (number of validation instances) prompts are evaluated on. However, for a given overall budget B it is a priori not clear whether one should evaluate many prompts using on average few validation instances (resulting in better exploration of the search space at the risk of noisy validation error estimates) or whether one should consider a smaller number of prompts using on average more validation instances (focusing on fewer prompts but obtaining more accurate validation error estimates).

Our proposal to use HB as a multi-fidelity scheduler for prompt selection improves over SH by addressing this trade-off between the number of prompts to explore and the amount of resources to allocate to each configuration. It does so by running multiple SH brackets, each with a different initial number of prompts and initial per-prompt budget. This allows HB to efficiently explore the search space, quickly discarding poor prompts while allocating more resources to promising ones, therefore hedging against a poor choice of the number of prompts and per-prompt budget.

We present pseudocode for HB adapted to the problem of black-box prompt selection in Algorithm 1 (as explained in Section 3.4, `HbBoPs` replaces the random proposal mechanism of HB by a BO proposal) and in Table 4, we present an exemplary schedule describing how the number of prompts used in each stage of each bracket of the algorithm relates to the number of validation instances used to evaluate the prompts. Regarding the output of the algorithm, we note that there is a critical design decision: The vanilla HB (Li et al., 2018) algorithm would output the configuration with the smallest validation error observed so far. However, in the context of prompt selection, validation errors are determined based on different numbers of validation instances varying between stages of brackets ($\{s_{\max}, s_{\max}-1, \dots, 0\}$). To allow for a robust selection of the optimal prompt, we always return the prompt with the lowest validation error among all prompts that were evaluated on the full validation set[2]. This is crucial for robust performance of HB in the context of prompt selection, which we demonstrate in Appendix E.4.

Table 4. Exemplary HB schedule for black-box prompt selection assuming a minimum budget of $b_{\min} = 10$ validation instances, a maximum number of $n_{\text{valid}} = 80$ validation instances being available in total, and a halving parameter of $\eta = 2.0$.

Bracket (s)	**Stage** (i)	**#Instances** (b)	**#Prompts** (n)
3	0	10	8
3	1	20	4
3	2	40	2
3	3	80	1
2	0	20	6
2	1	40	3
2	2	80	1
1	0	40	4
1	1	80	2
0	0	80	4

Our HB adapted to black-box prompt selection has three inputs: n_{valid} the total number of validation instances available (depending on the task), $b_{\min}$ a lower bound on the number of validation instances used for prompt evaluation, and η the halving parameter of the SH subroutine. HB makes use of three functions:
(1) get_prompt() returns a candidate prompt from the search space $\mathcal{P}$. In vanilla HB, we would sample uniformly at random. In our `HbBoPs` we obtain the next candidate prompt via a BO proposal (Section 3.4).
(2) get_validation_error(p, b_i) evaluates a prompt p using b_i validation instances. We further discuss this below.
(3) top_k$(P, V, \lfloor n_i/\eta \rfloor)$ reduces the n_i prompts in the active set P by only keeping the $\lfloor n_i/\eta \rfloor$ best performing ones.

[2]Or evaluated on the largest subset used so far when assessing the anytime performance of the algorithm.

As mentioned above, there is another critical detail when adapting HB to the problem of prompt selection, which is concerned with the selection of validation instances for a given stage of a bracket but also when moving from a given stage to the next stage within a bracket. In principle, the validation instances in get_validation_error(p, b_i) could be different for each prompt (sampled uniformly at random from the set of validation instances). However, to allow for a fairer comparison of the performance of prompts, we decided to use the same, fixed subset of validation instances for each prompt evaluated at a given stage of a bracket. This ensures that we perform a paired comparison of the performance of prompts when discarding the worst performing half. We ablate this design decision in Appendix E.4.

Moreover, when moving from one stage to another stage, there are two possibilities how to construct the subset of validation instances used in the next stage: (1) Simply draw a sample. (2) Keep the already used validation instances of the previous stage and only sample the remaining number of additional validation instances needed to fill the current stage from the remaining yet not used validation instances. The second option is highly desirable to reduce the total number of LLM calls if we cache the evaluation of prompts. The validity of this modeling choice naturally depends on the degree of stochasticity in the LLM's output for a given prompt-instance pair, which in turn is influenced by the sampling temperature. However, if we assume reasonably deterministic outcomes, we can reduce the total number of LLM calls used in HB drastically (i.e., roughly by a factor of η), In `HbBoPs` we cache the output for a given prompt and validation instance and reuse the already used validation instances of the previous stage and only sample the remaining needed validation instances. We ablate this design decision in Appendix E.4.

We want to note that `HbBoPs` employs a fully sequential HB schedule, i.e., prompts are proposed sequentially, and we evaluate brackets and their stages in their given order (e.g., as described in Table 4). While plenty possibilities exist to parallelize SH or HB in the context of hyperparameter optimization where evaluating a configuration involves a training step, we argue that in the context of prompt selection, there is little gain made by performing a batch proposal of prompts and evaluating batch-parallel (as in vanilla HB for hyperparameter optimization) or using asynchronous multi-fidelity schedulers (Li et al., 2020). This is because parallelization can be directly performed on the lowest level of evaluating a prompt on the validation set, i.e., in get_validation_error(p, b_i) by parallelizing LLM calls when evaluating a prompt p on the b_i validation instances. As a final comment on overhead, note that it may initially seem that training the structural-aware DK-GP in each iteration of a BO proposal within `HbBoPs` leads to computational costs that increase proportionally as optimization progresses. However, the GP is trained only on the subset of the highest fidelity design data for which enough observations are available. This approach not only ensures that the design data used to train the GP has accurately estimated validation errors but also keeps the subset size manageable, mitigating scaling issues of the GP even when `HbBoPs` is executed repeatedly.

D. Details on the Experimental Setup

D.1. Tasks

Table 5 reports characteristics on the benchmark tasks used in our experiments. For *AI2 ARC*, we use the official train, validation and test splits from AI2's Reasoning Challenge. However, during inspection we noticed that for some reason, AI2's Reasoning Challenge includes a few instances with choices named "1", "2", "3", "4" instead of "A", "B", "C", "D" and a few instances with five choices instead of four which we excluded from the splits for consistency. *GSM8K* officially only contains a train and test split. We sampled 1319 instances from the train split uniformly at random to create a validation set of comparable size to the test set. For all other tasks from the BBII subset of the BIG-bench and instruction induction benchmarks, we use the splits as proposed by Wu et al. (2024). Unlike Wu et al. (2024), who used only 20 validation instances by further sub-sampling the validation splits, we retain larger validation splits to reduce noise and improve the reliability of performance estimates. For each task, the training split was used to generate instructions for the prompts via APE's (Zhou et al., 2023) forward mode and to select instances for few-shot exemplars. Instructions are generated using APE's forward mode, where Claude 3 Sonnet (Anthropic, 2024), configured with a temperature of 1.0 and default settings otherwise (as in the main paper for Claude 3 Haiku), produces 100 candidate instructions from ten input-output examples per task; these are then embedded using BERT's [CLS] token representation, and five representative instructions are selected via 5-medoid clustering. The validation split was used during optimization and the test split was used to assess unbiased performance. We perform standard sanitization of LLM outputs to be able to employ the loss function in Equation (1) based on the exact match as a scoring function. For *GSM8K*, we determine the prediction for the exact match loss function by selecting the last number contained in the LLM's output. This approach aligns with the Chain-of-Thought prompting and the typical output behavior of LLMs for this task.

Table 5. Characteristics of tasks used in the experiments.

Task	Setting	n_{train}	n_{valid}	n_{test}
AI2 ARC	multiple choice question answering	1094	291	1144
GSM8K	grade school math questions	6154	1319	1319
antonyms	find antonym of word	2073	519	100
larger animal	select larger of two animals	2422	606	100
negation	negate a sentence	723	181	100
object counting	count number of objects	560	140	100
orthography starts with	output all words starting with a given letter	2400	600	100
second word letter	output the second letter of a word	2644	662	100
sentiment	sentiment analysis of movie rating	933	234	100
word unscrambling	build a word from scrambled letters	5627	1407	100

D.2. LLMs

We use Claude 3 Haiku (Anthropic, 2024), LLAMA3 8B Instruct (Llama Team, AI @ Meta, 2024), and Mistral 7B Instruct (Jiang et al., 2023) with default hyperparameters (Claude 3 Haiku: max tokens $= 200$, temperature $= 0.5$, top p $= 1.0$, top $k = 250$; LLAMA3 8B Instruct: max tokens $= 512$, temperature $= 0.5$, top $p = 0.9$; Mistral 7B Instruct: max tokens $= 512$, temperature $= 0.5$, top p $= 0.9$, top $k = 50$). For *GSM8K* we increase max tokens for all LLMs to 1024.

D.3. Methods

We run all methods as described in Section 4. All full-fidelity BO methods are implemented within BoTorch (Balandat et al., 2020). We include `HDBO` (Hvarfner et al., 2024) to have a simple yet well-performing "high-dimensional" BO baseline. Hvarfner et al. (2024) recently challenged the general belief that vanilla BO does not perform well for high-dimensional functions by training a GP via MAP with priors over kernel and likelihood parameters adjusted to reflect the dimensionality of the problem which resulted in strong BO performance on high-dimensional functions. We run `EASE` as implemented in the official code base[3], `MIPROv2` as implemented in `DSPy`[4] and `TRIPLE-SH` and `TRIPLE-GSE` as implemented in the official code base[5]. Regarding `TRIPLE-GSE` we noticed that Shi et al. (2024) did not provide detailed descriptions of the model used to predict expected prompt performance (in the main paper, they state that one can use a linear model or MLP). In their implementation, however, they use an ensemble of a Bayesian ridge regression model, a gradient boosting regression model, and an MLP where weights for the ensemble are determined based on each model's R^2 performance on a separate validation set. We did not change this modeling approach when running `TRIPLE-GSE`. We implement our `HbBoPs` in GPyTorch (Gardner et al., 2018) and run it as described in Section 4. Moreover, to hedge against poor model-based proposals, we perform random interleaving as described in Falkner et al. (2018) for each proposal with a probability of $\rho = 0.1$. Since a single SH or HB schedule may require less budget than the total pre-defined LLM call budget per task, we repeatedly run all methods that use SH or HB as multi-fidelity schedulers until their total LLM call usage reaches this pre-defined budget.

E. Additional Results

E.1. Main Results

Here, we provide additional analyses of the main results reported in Section 5.1. To test whether `HbBoPs` outperforms all other methods with respect to validation and test error for different fractions of budget, we conduct a linear mixed effects model analysis. We model the unaggregated performance as a function involving random intercepts for each benchmark scenario (benchmark task and LLM combination). This approach is sensible as each method has been run repeatedly on the

[3] https://github.com/ZhaoxuanWu/EASE-Prompt-Optimization/blob/e3514de58bd682ebc5ea46fe890481f2b92e5589/experiments/LlamaForMLPRegression.py

[4] https://github.com/stanfordnlp/dspy/blob/425b6f07d5cf0530f5a5566ad4f247b15aecb522/dspy/teleprompt/mipro_optimizer_v2.py

[5] https://github.com/ShenGroup/TRIPLE/blob/06264a97b4dd766c9a88afc24058627fac0f223d/src/bandit/contextual/gse.py

same benchmark scenario with different random seeds that affect, for example, initial designs. To test the global hypothesis that there is an effect of the method on performance, we test an intercept model against a model including an effect of the factor method. If we reject the null hypothesis, we proceed with a Tukey post hoc test (corrected for multiple testing) to test each method against HbBoPs. We test at the conservative $\alpha = 0.01$ level.

For the validation error at a fraction of 1.00, we reject the global null hypothesis of no effect of methods ($\chi^2(8) = 672.95, p < 1\text{e}{-4}$). The pairwise results are:

- RS vs. HbBoPs, $z = 18.52, p < 1\text{e}{-4}$
- vanilla BO vs. HbBoPs, $z = 15.40, p < 1\text{e}{-4}$
- HDBO vs. HbBoPs, $z = 7.65, p < 1\text{e}{-4}$
- BOPCA vs. HbBoPs, $z = 9.95, p < 1\text{e}{-4}$
- EASE vs. HbBoPs, $z = 17.29, p < 1\text{e}{-4}$
- MIPROv2 vs. HbBoPs, $z = 13.18, p < 1\text{e}{-4}$
- TRIPLE-SH vs. HbBoPs, $z = 1.87, p = 0.236$
- TRIPLE-GSE vs. HbBoPs, $z = 6.26, p = 1\text{e}{-4}$

We conclude that HbBoPs outperforms all methods significantly with respect to final performance, except for TRIPLE-SH. While the effect is positive, i.e., HbBoPs improves over TRIPLE-SH, it is not strong enough to be considered statistically significant at the $\alpha = 0.01$ level. For brevity, we do not include results for fractions of 0.25 or 0.50 here, where we observed similar results but HbBoPs more strongly outperforming the other methods.

Table 6. Normalized validation error of each method on each benchmark for Claude 3 Haiku. Averaged over repetitions. Standard errors in parentheses. For visual analysis, we highlight all methods that have a mean error that is less or equal to the mean error of the best method plus two times its standard error.

	Benchmark									
	AI2 ARC	GSM8K	antonyms	larger animal	negation	object counting	orthography starts with	second word letter	sentiment	word unscrambling
RS	0.053 (0.007)	0.087 (0.010)	0.018 (0.002)	0.011 (0.002)	**0.000 (0.000)**	0.093 (0.011)	0.033 (0.005)	0.076 (0.011)	0.021 (0.003)	0.085 (0.007)
vanilla BO	**0.046 (0.007)**	0.084 (0.010)	0.020 (0.002)	0.012 (0.002)	**0.000 (0.000)**	0.075 (0.010)	0.033 (0.005)	0.097 (0.011)	0.021 (0.004)	0.084 (0.009)
HDBO	**0.039 (0.007)**	0.050 (0.007)	**0.014 (0.002)**	**0.004 (0.001)**	**0.000 (0.000)**	**0.040 (0.009)**	0.028 (0.004)	0.029 (0.008)	**0.014 (0.003)**	0.039 (0.008)
BOPCA	**0.047 (0.008)**	**0.041 (0.007)**	**0.014 (0.002)**	**0.005 (0.001)**	**0.000 (0.000)**	0.062 (0.011)	0.035 (0.005)	0.044 (0.010)	**0.015 (0.003)**	0.081 (0.009)
EASE	0.080 (0.010)	0.063 (0.003)	0.024 (0.002)	0.017 (0.002)	**0.000 (0.000)**	0.099 (0.010)	0.039 (0.005)	0.070 (0.010)	0.027 (0.003)	0.103 (0.007)
MIPROv2	**0.048 (0.008)**	0.052 (0.004)	0.016 (0.001)	0.011 (0.002)	**0.000 (0.000)**	**0.050 (0.009)**	0.019 (0.003)	0.043 (0.009)	0.025 (0.003)	0.059 (0.007)
TRIPLE-SH	0.109 (0.014)	**0.043 (0.008)**	0.029 (0.003)	**0.005 (0.002)**	0.003 (0.002)	**0.043 (0.010)**	0.035 (0.006)	**0.006 (0.002)**	0.051 (0.007)	**0.025 (0.006)**
TRIPLE-GSE	0.178 (0.014)	0.072 (0.010)	0.031 (0.005)	0.021 (0.003)	0.000 (0.000)	0.087 (0.014)	0.049 (0.008)	**0.007 (0.002)**	0.060 (0.009)	**0.016 (0.006)**
HbBoPs	**0.040 (0.008)**	**0.035 (0.005)**	**0.011 (0.002)**	**0.005 (0.001)**	**0.000 (0.000)**	**0.055 (0.011)**	**0.013 (0.002)**	**0.008 (0.002)**	0.023 (0.003)	**0.026 (0.005)**

Table 7. Normalized validation error of each method on each benchmark for LLAMA3 8B Instruct. Averaged over repetitions. Standard errors in parentheses. For visual analysis, we highlight all methods that have a mean error that is less or equal to the mean error of the best method plus two times its standard error.

	Benchmark									
	AI2 ARC	GSM8K	antonyms	larger animal	negation	object counting	orthography starts with	second word letter	sentiment	word unscrambling
RS	0.078 (0.010)	0.067 (0.006)	0.093 (0.010)	0.023 (0.004)	0.011 (0.001)	0.156 (0.015)	0.230 (0.031)	0.275 (0.021)	0.022 (0.003)	0.271 (0.037)
vanilla BO	0.051 (0.005)	0.061 (0.007)	0.079 (0.009)	0.016 (0.003)	0.010 (0.001)	0.115 (0.017)	0.184 (0.024)	0.226 (0.023)	0.024 (0.005)	0.248 (0.036)
HDBO	0.042 (0.006)	0.019 (0.006)	0.073 (0.009)	0.015 (0.002)	**0.008 (0.001)**	**0.066 (0.008)**	0.148 (0.023)	0.186 (0.020)	**0.010 (0.003)**	0.163 (0.027)
BOPCA	0.049 (0.006)	0.026 (0.007)	0.074 (0.010)	0.014 (0.002)	0.010 (0.001)	**0.074 (0.015)**	0.225 (0.028)	0.160 (0.023)	**0.013 (0.004)**	0.149 (0.023)
EASE	0.059 (0.006)	0.060 (0.005)	0.115 (0.012)	0.032 (0.005)	0.010 (0.001)	**0.076 (0.005)**	0.129 (0.022)	0.271 (0.015)	0.023 (0.004)	0.357 (0.033)
MIPROv2	0.063 (0.007)	0.056 (0.007)	0.062 (0.008)	0.017 (0.002)	**0.008 (0.001)**	0.092 (0.011)	0.183 (0.026)	0.218 (0.022)	0.023 (0.004)	0.265 (0.033)
TRIPLE-SH	**0.023 (0.005)**	0.007 (0.002)	**0.024 (0.005)**	**0.008 (0.002)**	0.012 (0.002)	**0.068 (0.008)**	**0.034 (0.008)**	**0.047 (0.014)**	0.025 (0.005)	0.095 (0.019)
TRIPLE-GSE	**0.022 (0.006)**	**0.002 (0.001)**	**0.026 (0.006)**	0.020 (0.003)	0.015 (0.001)	**0.080 (0.008)**	0.074 (0.011)	**0.071 (0.016)**	0.035 (0.004)	0.184 (0.033)
HbBoPs	**0.019 (0.005)**	0.006 (0.002)	**0.024 (0.004)**	**0.008 (0.002)**	**0.008 (0.001)**	0.092 (0.018)	**0.041 (0.009)**	**0.043 (0.014)**	0.020 (0.004)	**0.048 (0.014)**

In Tables 6, 7 and 8 we report the average normalized validation error of the best prompt found by each method after having used a fraction of 1.00 LLM calls, separately for each benchmark task, separately for each LLM.

We perform the same analysis for the test error at a fraction of 1.00 and reject the global null hypothesis of no effect of methods ($\chi^2(8) = 288.36, p < 1\text{e}{-4}$). The pairwise results are:

- RS vs. HbBoPs, $z = 11.51, p < 1\text{e}{-4}$
- vanilla BO vs. HbBoPs, $z = 11.03, p < 1\text{e}{-4}$

Table 8. Normalized validation error of each method on each benchmark for Mistral 7B Instruct. Averaged over repetitions. Standard errors in parentheses. For visual analysis, we highlight all methods that have a mean error that is less or equal to the mean error of the best method plus two times its standard error.

	Benchmark									
	AI2 ARC	GSM8K	antonyms	larger animal	negation	object counting	orthography starts with	second word letter	sentiment	word unscrambling
RS	0.329 (0.035)	0.160 (0.021)	0.020 (0.002)	0.043 (0.006)	0.006 (0.003)	0.040 (0.004)	0.164 (0.018)	0.171 (0.016)	0.098 (0.013)	0.224 (0.017)
vanilla BO	0.359 (0.038)	0.166 (0.021)	0.017 (0.002)	0.040 (0.008)	0.008 (0.003)	**0.033 (0.005)**	0.117 (0.020)	0.136 (0.013)	0.085 (0.016)	0.161 (0.019)
HDBO	0.217 (0.045)	0.105 (0.019)	**0.013 (0.002)**	0.021 (0.005)	0.009 (0.003)	**0.028 (0.005)**	0.120 (0.018)	0.085 (0.017)	**0.042 (0.013)**	0.108 (0.016)
BOPCA	0.246 (0.044)	0.108 (0.019)	0.015 (0.002)	0.021 (0.006)	0.011 (0.004)	**0.032 (0.006)**	0.112 (0.021)	0.125 (0.019)	**0.048 (0.013)**	0.136 (0.021)
EASE	0.384 (0.033)	0.133 (0.014)	0.022 (0.002)	0.053 (0.008)	**0.000 (0.000)**	0.035 (0.005)	0.147 (0.023)	0.183 (0.014)	0.110 (0.014)	0.095 (0.021)
MIPROv2	0.323 (0.041)	0.122 (0.017)	**0.013 (0.002)**	0.032 (0.005)	0.011 (0.004)	**0.032 (0.005)**	0.125 (0.014)	0.158 (0.016)	0.068 (0.014)	0.163 (0.017)
TRIPLE-SH	**0.024 (0.011)**	0.008 (0.003)	0.041 (0.005)	**0.012 (0.003)**	0.044 (0.008)	0.057 (0.006)	0.005 (0.004)	**0.044 (0.012)**	0.055 (0.010)	0.103 (0.018)
TRIPLE-GSE	**0.038 (0.016)**	0.001 (0.000)	0.042 (0.005)	0.022 (0.006)	0.055 (0.010)	0.067 (0.010)	**0.002 (0.000)**	0.072 (0.015)	0.084 (0.017)	0.145 (0.022)
HbBoPs	0.107 (0.036)	**0.000 (0.000)**	**0.011 (0.001)**	**0.012 (0.003)**	0.012 (0.005)	**0.025 (0.005)**	0.024 (0.010)	0.067 (0.013)	**0.030 (0.011)**	**0.061 (0.014)**

- HDBO vs. HbBoPs, $z = 6.19$, $p < 1\text{e}{-4}$
- BOPCA vs. HbBoPs, $z = 7.58$, $p < 1\text{e}{-4}$
- EASE vs. HbBoPs, $z = 7.98$, $p < 1\text{e}{-4}$
- MIPROv2 vs. HbBoPs, $z = 8.61$, $p < 1\text{e}{-4}$
- TRIPLE-SH vs. HbBoPs, $z = 1.49$, $p = 1.000$
- TRIPLE-GSE vs. HbBoPs, $z = 1.34$, $p = 1.000$

Conclusions are largely consistent with the analysis with respect to validation error, however, while HbBoPs improves over TRIPLE-SH and TRIPLE-GSE also with respect to test error, the effects are not strong enough to be considered statistically significant at the $\alpha = 0.01$ level. For brevity, we do not include results for fractions of 0.25 or 0.50 here, where we observed similar results but HbBoPs again more strongly outperforming the other methods.

Table 9. Normalized test error of each method on each benchmark for Claude 3 Haiku. Averaged over repetitions. Standard errors in parentheses. For visual analysis, we highlight all methods that have a mean error that is less or equal to the mean error of the best method plus two times its standard error.

	Benchmark									
	AI2 ARC	GSM8K	antonyms	larger animal	negation	object counting	orthography starts with	second word letter	sentiment	word unscrambling
RS	**0.224 (0.023)**	0.092 (0.012)	**0.074 (0.003)**	0.096 (0.008)	0.124 (0.007)	0.140 (0.014)	**0.277 (0.011)**	0.058 (0.010)	0.244 (0.019)	0.163 (0.011)
vanilla BO	**0.213 (0.020)**	0.092 (0.011)	0.084 (0.004)	0.113 (0.010)	0.124 (0.007)	0.147 (0.019)	0.296 (0.011)	0.063 (0.008)	**0.207 (0.016)**	0.156 (0.012)
HDBO	**0.203 (0.020)**	0.087 (0.010)	0.078 (0.003)	0.105 (0.008)	0.124 (0.007)	**0.083 (0.015)**	0.294 (0.011)	0.030 (0.008)	**0.237 (0.014)**	0.133 (0.009)
BOPCA	**0.209 (0.018)**	0.095 (0.009)	0.078 (0.003)	0.105 (0.009)	0.124 (0.007)	0.129 (0.019)	0.302 (0.012)	0.039 (0.009)	**0.230 (0.014)**	0.150 (0.011)
EASE	0.243 (0.023)	**0.036 (0.006)**	**0.070 (0.004)**	0.108 (0.007)	0.124 (0.007)	0.135 (0.015)	**0.282 (0.018)**	0.058 (0.009)	0.263 (0.016)	0.135 (0.010)
MIPROv2	**0.200 (0.020)**	0.068 (0.006)	**0.077 (0.003)**	0.099 (0.008)	0.124 (0.007)	**0.100 (0.017)**	**0.271 (0.011)**	0.039 (0.008)	**0.219 (0.016)**	0.125 (0.008)
TRIPLE-SH	0.256 (0.024)	0.099 (0.009)	0.078 (0.006)	**0.076 (0.006)**	**0.114 (0.011)**	**0.084 (0.012)**	**0.272 (0.015)**	**0.008 (0.005)**	0.248 (0.019)	**0.116 (0.005)**
TRIPLE-GSE	0.307 (0.024)	0.080 (0.010)	0.082 (0.005)	0.115 (0.010)	**0.106 (0.009)**	0.130 (0.019)	0.299 (0.014)	**0.009 (0.005)**	0.322 (0.022)	**0.122 (0.005)**
HbBoPs	**0.182 (0.026)**	0.093 (0.008)	0.080 (0.003)	0.089 (0.007)	**0.118 (0.008)**	0.117 (0.021)	**0.262 (0.010)**	**0.018 (0.007)**	**0.219 (0.016)**	**0.125 (0.004)**

Table 10. Normalized test error of each method on each benchmark for LLAMA3 8B Instruct. Averaged over repetitions. Standard errors in parentheses. For visual analysis, we highlight all methods that have a mean error that is less or equal to the mean error of the best method plus two times its standard error.

	Benchmark									
	AI2 ARC	GSM8K	antonyms	larger animal	negation	object counting	orthography starts with	second word letter	sentiment	word unscrambling
RS	0.066 (0.010)	0.112 (0.011)	0.147 (0.012)	0.085 (0.011)	**0.104 (0.010)**	0.238 (0.021)	0.296 (0.041)	0.270 (0.029)	0.356 (0.013)	0.633 (0.054)
vanilla BO	0.038 (0.009)	0.099 (0.012)	0.145 (0.013)	**0.075 (0.007)**	0.129 (0.008)	0.236 (0.023)	0.218 (0.034)	0.292 (0.031)	0.356 (0.015)	0.700 (0.054)
HDBO	0.030 (0.008)	0.028 (0.009)	0.133 (0.014)	0.086 (0.007)	0.138 (0.007)	0.165 (0.019)	0.184 (0.035)	0.248 (0.027)	0.321 (0.010)	0.767 (0.056)
BOPCA	0.033 (0.004)	0.038 (0.011)	0.124 (0.014)	**0.075 (0.007)**	0.126 (0.009)	0.216 (0.014)	0.283 (0.038)	0.227 (0.025)	0.344 (0.010)	0.667 (0.048)
EASE	0.047 (0.007)	0.094 (0.009)	0.181 (0.015)	0.076 (0.009)	**0.112 (0.008)**	**0.014 (0.009)**	0.171 (0.037)	0.248 (0.027)	**0.274 (0.013)**	0.622 (0.055)
MIPROv2	0.041 (0.008)	0.088 (0.012)	0.114 (0.013)	0.076 (0.006)	0.135 (0.009)	0.160 (0.019)	0.239 (0.040)	0.248 (0.024)	0.356 (0.014)	0.700 (0.051)
TRIPLE-SH	**0.016 (0.003)**	0.008 (0.003)	**0.070 (0.015)**	**0.075 (0.008)**	0.125 (0.011)	0.177 (0.017)	**0.037 (0.009)**	0.152 (0.011)	0.362 (0.014)	0.678 (0.059)
TRIPLE-GSE	**0.017 (0.004)**	**0.003 (0.002)**	**0.052 (0.012)**	**0.059 (0.008)**	**0.110 (0.009)**	0.167 (0.019)	0.078 (0.019)	**0.140 (0.016)**	0.369 (0.014)	**0.544 (0.059)**
HbBoPs	**0.016 (0.006)**	0.009 (0.003)	0.093 (0.015)	**0.064 (0.007)**	0.152 (0.010)	0.207 (0.027)	0.078 (0.019)	**0.120 (0.011)**	0.341 (0.012)	**0.489 (0.044)**

In Tables 9, 10 and 11 we report the average normalized test error of the best prompt found by each method after having used a fraction of 1.00 LLM calls, separately for each benchmark task, separately for each LLM.

Table 11. Normalized test error of each method on each benchmark for Mistral 7B Instruct. Averaged over repetitions. Standard errors in parentheses. For visual analysis, we highlight all methods that have a mean error that is less or equal to the mean error of the best method plus two times its standard error.

	Benchmark									
	AI2 ARC	GSM8K	antonyms	larger animal	negation	object counting	orthography starts with	second word letter	sentiment	word unscrambling
RS	0.277 (0.037)	0.198 (0.021)	0.099 (0.006)	0.239 (0.033)	0.502 (0.025)	**0.188 (0.015)**	0.293 (0.026)	0.194 (0.015)	0.156 (0.019)	0.476 (0.039)
vanilla BO	0.366 (0.042)	0.200 (0.022)	**0.092 (0.006)**	0.201 (0.025)	0.531 (0.025)	**0.199 (0.013)**	0.229 (0.026)	0.172 (0.018)	0.125 (0.017)	0.448 (0.031)
HDBO	0.215 (0.045)	0.112 (0.020)	0.098 (0.005)	**0.140 (0.009)**	**0.481 (0.027)**	0.226 (0.013)	0.237 (0.031)	0.112 (0.021)	**0.059 (0.018)**	0.386 (0.028)
BOPCA	0.266 (0.047)	0.125 (0.019)	**0.088 (0.007)**	0.182 (0.019)	0.500 (0.028)	0.232 (0.013)	0.226 (0.031)	0.140 (0.022)	**0.063 (0.016)**	0.357 (0.034)
EASE	0.356 (0.036)	0.174 (0.014)	**0.076 (0.008)**	0.181 (0.021)	0.631 (0.022)	**0.209 (0.013)**	0.257 (0.029)	0.211 (0.018)	0.158 (0.019)	**0.290 (0.025)**
MIPROv2	0.327 (0.043)	0.165 (0.016)	0.094 (0.004)	0.178 (0.020)	0.519 (0.026)	**0.195 (0.013)**	0.248 (0.025)	0.193 (0.018)	0.115 (0.022)	0.429 (0.029)
TRIPLE-SH	**0.016 (0.008)**	0.078 (0.010)	0.104 (0.007)	**0.140 (0.009)**	0.526 (0.026)	0.230 (0.014)	**0.044 (0.014)**	**0.061 (0.016)**	0.116 (0.019)	0.395 (0.033)
TRIPLE-GSE	**0.028 (0.012)**	0.050 (0.011)	0.095 (0.006)	0.169 (0.011)	**0.421 (0.038)**	0.226 (0.013)	**0.061 (0.011)**	**0.071 (0.015)**	0.104 (0.022)	0.400 (0.029)
HbBoPs	0.096 (0.032)	**0.011 (0.006)**	**0.083 (0.005)**	0.162 (0.013)	**0.462 (0.028)**	0.238 (0.011)	**0.053 (0.023)**	**0.083 (0.017)**	**0.088 (0.021)**	0.367 (0.030)

E.2. Ablation Study

Here, we provide additional analyses of the ablation results reported in Section 5.3. To test the significance of each component on the validation and test error of HbBoPs, we again conduct a linear mixed effects model analysis. We model the unaggregated performance as a function of the fraction of LLM calls (i.e., over time; starting after the initial design of full-fidelity methods, i.e., after a fraction of 0.40) and include random intercepts for each benchmark scenario (benchmark task and LLM combination). To test the global hypothesis that there is an effect of the components on performance, we test an intercept model against a model including an effect of the factor method (corresponding to an ablation variant). If we reject the null hypothesis, we proceed with a Tukey post hoc test (corrected for multiple testing) to perform pairwise comparisons. We test at the conservative $\alpha = 0.01$ level.

Examining the validation error, we reject the global null hypothesis of no effect of the method on the anytime performance ($\chi^2(5) = 35184.28, p < 1\text{e}{-4}$). The relevant pairwise comparison results state as follows:

- vanilla BO vs. BoPs (non structural-aware DK-GP), $z = 48.80, p < 1\text{e}{-4}$
- BoPs (non structural-aware DK-GP) vs. BoPs (structural-aware DK-GP), $z = 16.67, p < 1\text{e}{-4}$
- BoPs (structural-aware DK-GP) vs. HB, $z = 52.92, p < 1\text{e}{-4}$
- HB vs. HbBoPs, $z = 24.00, p < 1\text{e}{-4}$

We can conclude that using a DK-GP significantly improves over vanilla BO, that a structural-aware DK-GP improves over the non structural-aware DK-GP, that HB improves over the structural-aware DK-GP and that HbBoPs further improves over HB.

We perform the same analysis for the test error and reject the global hypothesis of no effect ($\chi^2(5) = 14578.73, p < 1\text{e}{-4}$). The relevant pairwise comparison results state as follows:

- vanilla BO vs. BoPs (non structural-aware DK-GP), $z = 39.35, p < 1\text{e}{-4}$
- BoPs (non structural-aware DK-GP) vs. BoPs (structural-aware DK-GP), $z = 6.51, p < 1\text{e}{-4}$
- BoPs (structural-aware DK-GP) vs. HB, $z = 31.59, p < 1\text{e}{-4}$
- HB vs. HbBoPs, $z = 18.85, p < 1\text{e}{-4}$

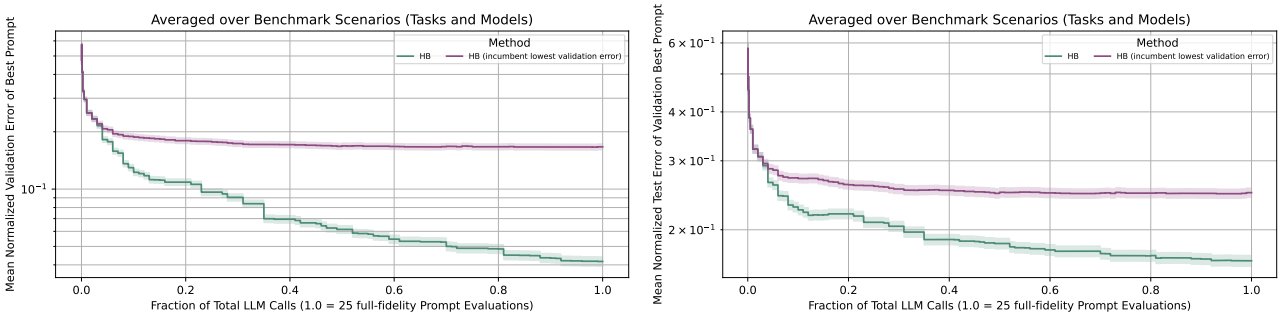
Conclusions are the same as for the validation error.

E.3. Encoder Sensitivity

Here, we provide additional analyses of the encoder sensitivity results reported in Section 5.4. To test whether the choice of encoder model affects the final performance of HbBoPs, we again conduct a linear mixed effects model analysis. We model the unaggregated performance at a fraction of 1.00 total LLM calls involving random intercepts for each benchmark scenario (benchmark task and LLM combination). To test the global hypothesis that there is an effect of the encoder on performance, we test an intercept model against a model including an effect of the factor encoder. We test at the conservative $\alpha = 0.01$ level. For both the validation and test error, we cannot reject the null hypothesis of the encoder making no difference, $\chi^2(2) = 4.69, p = 0.096$ and $\chi^2(2) = 3.85, p = 0.146$ respectively. We therefore conclude that HbBoPs is robust to the choice of encoder model.

E.4. Hyperband Design Choices

As mentioned in Section 3 and Appendix C, adapting HB to prompt selection involves several design decisions. Here, we provide an ablation of these decisions. While vanilla HB for hyperparameter optimization would return the configuration with the lowest validation error as the (anytime) incumbent, this is not sensible for prompt selection as the fidelity directly influences the noise of the validation error. We therefore always return the prompt with the lowest validation error among all prompts that have been evaluated on the (current) highest fidelity level. To analyze the effect of this design decision, we run HB for prompt selection with this incumbent selection mechanism and compare to the incumbent selection mechanism that simply selects the prompt with the lowest validation error. The experimental setup is exactly the same as for the results reported in the main paper. We visualize the (oracle) normalized validation and test error of the best prompt found by HB under each incumbent selection scheme in Figure 6. As before, for visualization purposes the validation error of the incumbent is computed here in an oracle setting (i.e., using all validation instances), whereas during the selection process the anytime incumbent itself was selected based on its validation error computed on fewer validation instances. Examining the validation error (Figure 6a) of `HB` as used by us (selecting the incumbent as the prompt with the lowest validation error among all prompts evaluated on the highest fidelity level), we observe that the validation error of the incumbent keeps decreasing as optimization progresses. In contrast, if we would perform the incumbent selection simply by choosing the prompt with the lowest validation error (ignoring the fidelity level), as in `HB (incumbent lowest validation error)`, we observe that optimization progress stagnates quickly. This occurs because the incumbent is no longer updated, since, at lower fidelity levels, noisy performance estimates can result in artificially low validation errors. Similar conclusions hold for the test error (Figure 6b).



(a) Validation (b) Test

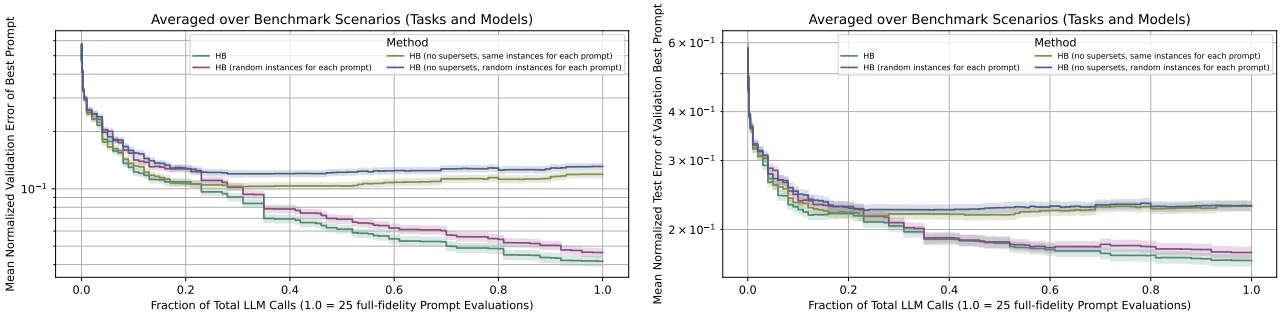
Figure 6. Normalized error (log scale) of the best prompt found by each HB incumbent selection mechanism, averaged over benchmarks. Lower is better. Ribbons represent SE.

To test whether the choice of selection mechanism does make a difference for the final performance of HB, we again conduct a linear mixed effects model analysis. We model the unaggregated performance at a fraction of 1.00 total LLM calls involving random intercepts for each benchmark scenario (benchmark task and LLM combination). To test the global hypothesis that there is an effect of the incumbent selection mechanism on performance, we test an intercept model against a model including an effect of the factor selection mechanism. We test at the conservative $\alpha = 0.01$ level. For both validation and test error we reject the null hypothesis of no effect of the selection mechanism $\chi^2(1) = 448.05, p < 1\text{e}{-4}$ and $\chi^2(1) = 164.17, p < 1\text{e}{-4}$ respectively. We therefore conclude that our incumbent selection mechanism is superior.

Another design decision for adapting HB to prompt selection is concerned with whether prompts should be evaluated on the same random validation instances within a stage or on their own random samples. The final related design decision involves whether validation instances of higher stages for a given bracket are constructed to be supersets of the validation instances used in lower stages (as described in Appendix C) which allows for further speed-ups due to caching.

To investigate the effect of using the same random vs. truly random instances for each prompt and the effect of validation instances used in higher stages of a bracket being supersets of the validation instances used in lower stages, we run HB for prompt selection varying these two components. The experimental setup is exactly the same as for the results reported in the main paper. We visualize the (oracle) normalized validation and test error of the best prompt found by HB under each incumbent selection scheme in Figure 7. As before, for visualization purposes the validation error of the incumbent is computed here in an oracle setting (i.e., using all validation instances), whereas during the selection process the anytime incumbent itself was selected based on its validation error computed on fewer validation instances. Examining the

validation error (Figure 7a) of our proposed HB (same random instances and supersets), we can see that this variant performs best. If we use truly random instances (but keep the superset structure) as in HB (random instances for each prompt), performance is slightly worse. Giving up the superset structure (HB (no supersets, same instances for each prompt) and HB (no supersets, random instances for each prompt) we can see that performance is substantially worse, even more so when using truly random instances for each prompt. In general, we can conclude that the effect of using supersets for higher stages within a given bracket boosts the performance of HB. Moreover, using truly random validation instances for each prompt instead of using the same random validation instances for all prompt evaluations in a stage (i.e., the paired setting) generally worsens performance. Examining the test performance, we observe that these conclusions generalize (Figure 7b).



(a) Validation (b) Test

Figure 7. Normalized error (log scale) of the best prompt found by each HB validation instances sampling variant, averaged over benchmarks. Lower is better. Ribbons represent SE.

To test whether the design decisions of using supersets and using the same random validation instances for each prompt evaluation within a stage make a difference for the final performance of HB, we again conduct a linear mixed effects model analysis. We model the unaggregated performance at a fraction of 1.00 total LLM calls involving random intercepts for each benchmark scenario (benchmark task and LLM combination). To test the global hypothesis that there is an effect of the design choices on performance, we test an intercept model against a model including an effect of the factor method (corresponding to an ablation variant). We test at the conservative $\alpha = 0.01$ level.

For the validation error at a fraction of 1.00, we reject the global null hypothesis of no effect of methods ($\chi^2(3) = 730.85, p < 1\text{e}{-4}$). The pairwise results are:

- HB (random instances for each prompt) vs. HB, $z = 1.13, p = 0.259$
- HB (no supersets, same instances for each prompt) vs. HB, $z = 19.12, p < 1\text{e}{-4}$
- HB (no supersets, random instances for each prompt) vs. HB, $z = 22.03, p < 1\text{e}{-4}$
- HB (no supersets, same instances for each prompt) vs. HB (random instances for each prompt), $z = 17.99, p < 1\text{e}{-4}$
- HB (no supersets, random instances for each prompt) vs. HB (random instances for each prompt), $z = 20.90, p < 1\text{e}{-4}$
- HB (no supersets, random instances for each prompt) vs. HB (no supersets, same instances for each prompt), $z = 2.91, p = 0.007$

Examining test error at a fraction of 1.00 we also reject the global null hypothesis of no effect of methods ($\chi^2(3) = 207.12, p < 1\text{e}{-4}$). The pairwise results are:

- HB (random instances for each prompt) vs. HB, $z = 1.39, p = 0.328$
- HB (no supersets, same instances for each prompt) vs. HB, $z = 10.92, p < 1\text{e}{-4}$
- HB (no supersets, random instances for each prompt) vs. HB, $z = 11.02, p < 1\text{e}{-4}$
- HB (no supersets, same instances for each prompt) vs. HB (random instances for each prompt), $z = 9.53, p < 1\text{e}{-4}$
- HB (no supersets, random instances for each prompt) vs. HB (random instances for each prompt), $z = 9.63, p < 1\text{e}{-4}$
- HB (no supersets, random instances for each prompt) vs. HB (no supersets, same instances for each prompt), $z = 0.11, p = 0.916$

Summarizing, our results confirm that the design decisions made to adapt HB to prompt selection are effective: (1) The incumbent should be selected as the prompt with the lowest validation error among all prompts evaluated on the highest fidelity level. (2) Validation instances used to evaluate prompts in higher stages of a given bracket should be supersets of the validation instances used in lower stages. (3) Using the same (random) validation instances to evaluate prompts in each stage in general is beneficial compared to using truly random validation instances for each prompt, albeit this effect is comparably small.