THE CHICKEN AND EGG DILEMMA: CO-OPTIMIZING DATA AND MODEL CONFIGURATIONS FOR LLMS

Zhiliang Chen[1*], **Alfred Wei Lun Leong**[1*], **Shao Yong Ong**[1*], **Apivich Hemachandra**[1], **Gregory Kang Ruey Lau**[1], **Chuan-Sheng Foo**[2], **Zhengyuan Liu**[2], **Nancy F. Chen**[2], **Bryan Kian Hsiang Low**[1]
[1]Department of Computer Science, National University of Singapore, Singapore
[2]Institute for Infocomm Research, A*STAR, Singapore
{chenzhiliang,alfred_leong,ongshaoyong,apivich,gregorylau}@u.nus.edu
lowkh@comp.nus.edu.sg

ABSTRACT

Co-optimizing data and model configurations for training LLMs presents a classic chicken-and-egg dilemma: The best training data configuration (e.g., data mixture) for a downstream task depends on the chosen model configuration (e.g., model architecture), and vice versa. However, jointly optimizing both data and model configurations is often deemed intractable, and existing methods focus on either data or model optimization without considering their interaction. We introduce JoBS, an approach that uses a scaling-law-inspired performance predictor to aid Bayesian optimization (BO) in *jointly* optimizing LLM training data and model configurations efficiently. JoBS allocates a portion of the optimization budget to learn an LLM performance predictor that predicts how promising a training configuration is from a small number of training steps. The remaining budget is used to perform BO *entirely* with the predictor, effectively amortizing the cost of running full-training runs. We study JoBS's *average regret* and devise the optimal budget allocation to minimize regret. JoBS outperforms existing multi-fidelity BO baselines, as well as data and model optimization approaches across diverse LLM tasks under the same optimization budget. Our code is available at https://github.com/a35453779/JoBS.

1 INTRODUCTION

There is great commercial and practical interest in maximizing the performance of an LLM for downstream tasks. To do so, practitioners typically optimize the LLM *training components*, particularly the *training data* and *model architecture*. From the *data* perspective, better training data can be chosen via data optimization (Koh & Liang, 2020; Xia et al., 2024; Xie et al., 2023b) and mixing (Chen et al., 2025a;c; Liu et al., 2025; Xie et al., 2023a; 2025). From the *model* perspective, various model optimization methods (Raschka, 2020; White et al., 2020; Zhang et al., 2024b; He et al., 2024) have been introduced to select the most appropriate model for a given task.

In practice, optimizing training data and model architecture is a highly interdependent process. Finding the best training data configuration (e.g., data mixture) depends on the chosen model configuration (e.g., model fine-tuning architecture), but determining the best model configuration also depends on the chosen data. This presents a classic *chicken-and-egg dilemma* where the optimal choice of training data depends on the optimal choice of model configuration, and vice versa. Optimizing data and model *independently* often leads to sub-optimal LLM performance (Chen et al., 2024). This is demonstrated in Sec. 6 where we naively combined data and model optimization methods.

Unfortunately, jointly optimizing training data and model configurations is often deemed challenging and budget-intensive. Prior scaling law works (Hoffmann et al., 2022; Kaplan et al., 2020; Shukor et al., 2025; Zhang et al., 2024a) attempt to quantify the effects of each training component on

[*]Equal contribution

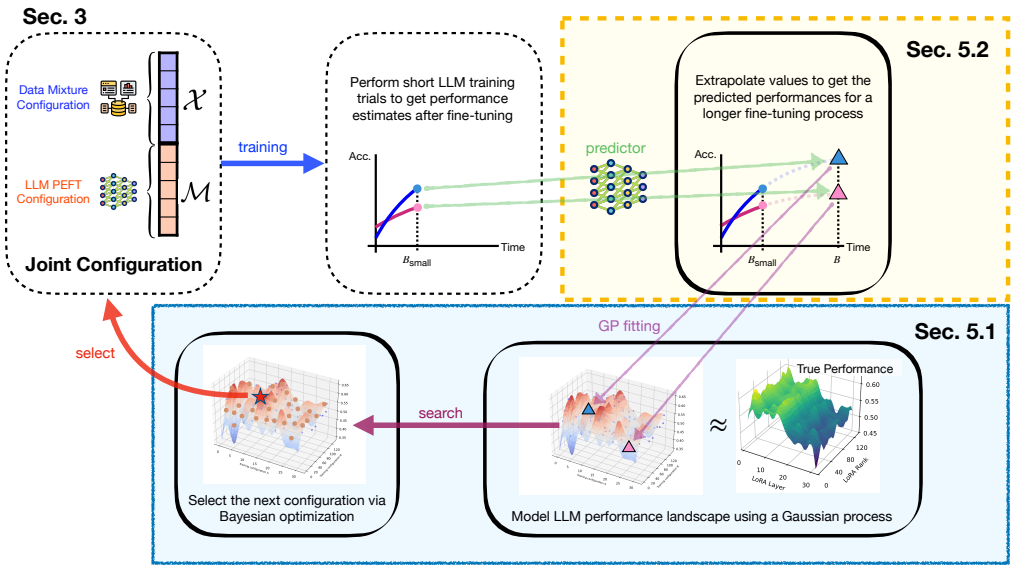


Figure 1: JoBS optimally balances budget between learning a performance predictor and running Bayesian Optimization iterations.

downstream performance while prescribing simple guidelines for choosing good training components. However, they require an exhaustive search over a large number of training configurations, which is infeasible in practice.

Our paper frames this chicken-and-egg dilemma as a joint optimization problem w.r.t. training data and model configurations for LLMs. To solve it efficiently, we present **Joint Bayesian Optimization with a Scaling-law-inspired predictor** (JoBS), an algorithm that uses a novel performance predictor to aid BO in accelerating joint optimization. We show that the predictor does not need to be perfectly accurate; it only needs to provide sufficiently informative signals during the optimization process. We summarize JoBS in Fig. 1 and present our main contributions:

- We propose JoBS (Sec. 5), an algorithm that efficiently optimizes LLM training configurations by interleaving BO (Sec. 5.1) with a performance predictor that predicts how promising a training configuration is, amortizing the cost of performing full-training runs. This enables far more BO iterations within the same optimization budget.
- We show that learning an accurate performance predictor requires us to fully train an LLM over a good coverage of initial training configurations (Sec. 4.2). We demonstrate an inherent tradeoff under a fixed budget: performing more full-training runs to train the predictor improves the quality of each BO observation, but reduces the subsequent number of BO acquisition steps available.
- We theoretically analyze how the performance predictor's error propagates into JoBS's average regret. From this, we find the optimal budget allocation that one should use to train the predictor and minimize regret (Sec. 5.4).
- We show that JoBS attains performance improvement over existing data optimization, model optimization and multi-fidelity BO baselines on a variety of LLM tasks (Sec. 6). We also run ablations to demonstrate how the performance of JoBS varies w.r.t. the accuracy of our performance predictor, affirming our theoretical findings.

2 Related Work

Multi-fidelity BO. BO has been widely adopted for optimizing black-box functions where evaluations are costly (Srinivas et al., 2010). In particular, multi-fidelity BO (Wu et al., 2019; Yen et al., 2025; Swersky et al., 2014; Lee et al., 2025) models the relationship between low-fidelity and high-fidelity observations to strategically select which fidelity to perform trials in. In our setting, the LLM performance at a small training step can naturally be viewed as a low-fidelity observation. JoBS differs from multi-fidelity BO by first using an optimal portion of the optimization budget to learn a performance predictor, before performing BO *entirely* using this neural network predictor *without*

revisiting any high-fidelity evaluations. In App. A, we provide a detailed analysis of the difference between our work and existing multi-fidelity BO works, such as freeze-thaw methods (Swersky et al., 2014) and early-stopping (Dai et al., 2019).

Data and model optimization. Conventional data (Koh & Liang, 2020; Xia et al., 2024; Chen et al., 2025c; Wang et al., 2024b) and model (Liu et al., 2019; He et al., 2024; Zhang et al., 2024b) optimization works have only optimized either the data or model component *independently*. To our knowledge, JoBS is the first algorithm that *jointly* optimizes LLM data and model components.

LLM Scaling Law. Existing scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022; Zhang et al., 2024a; Shukor et al., 2025; Wu & Tang, 2024; Chen et al., 2025b) establish symbolic formulas that extrapolate LLM performance from a small number of training steps. Unfortunately, as shown in Sec. 4.2, these formulas do not work well in our problem setting because they are defined w.r.t. a fixed training configuration (e.g., the same training data pool), while we need to predict LLM performance for a wide range of different LLM training configurations. Notably, Yen et al. (2025) explored data mixtures optimization by using neural networks to predict LLM performance, but did not consider the cost of training the predictor as part of their optimization budget.

3 Problem Setup

We consider two types of training components: **training data** $\mathcal{X}$ and **model** $\mathcal{M}$. Given these training components, we define a training process P_B that fine-tunes an LLM for B training steps to produce trained LLM weights $\theta_{\mathcal{X},\mathcal{M},B} \triangleq P_B(\mathcal{X},\mathcal{M})$, which can be evaluated over a predefined performance metric $\mathcal{L}$. Our formulation is flexible enough to accommodate different, possibly black-box metrics (e.g., question-answering accuracy, evaluation loss). This is similar to the setting introduced by Chen et al. (2025c) where we do not have any fine-grained knowledge of data involved in the evaluation task. Our goal is to find training component configurations $\mathcal{X},\mathcal{M}$ that maximize the LLM performance metric:

$$\max_{\mathcal{X},\mathcal{M}} \mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B}) \,. \tag{1}$$

Data $\mathcal{X}$. Assume we have N training datasets $\mathcal{D} \triangleq D_1 \bigcup D_2 \bigcup \cdots \bigcup D_N$ from N different domains (e.g., Wikipedia, TruthfulQA (Lin et al., 2022) for language tasks). The training data component consists of a subset of data $\mathcal{X} \subseteq \mathcal{D}$. In general, the selection of $\mathcal{X}$ ensures the selected data points are more relevant to the given task (Chen et al., 2025c) or of higher quality (Wang et al., 2024a; Xia et al., 2024; Zhang et al., 2025). In our work, we overload the notation $\mathcal{X}$ to represent a data mixture's mixing ratio (Chen et al., 2025c), represented by a probability simplex of dimension $N-1$ ($\mathcal{X} \in \Delta^{N-1} \subset \mathbb{R}^N$). This implies that optimizing the data mixture requires us to find the optimal mixing ratio across the N data domains.

Model $\mathcal{M}$. Our formulation is flexible enough to incorporate any model configuration that needs to be optimized. To simplify our formulation, our work focuses on parameter-efficient fine-tuning (PEFT) of LLMs with LoRA (Hu et al., 2021), but we also extend our work to the full-parameter fine-tuning setting in App. H. The optimization problem takes as inputs: (1) the LLM *modules* where PEFT is applied (e.g., Q,V-projection (Vaswani et al., 2017)), (2) the *layer(s)* where LoRA is applied (e.g., layer 30), and (3) the *PEFT hyperparameters*, including LoRA rank, α, and dropout (Hu et al., 2021). These inputs can be concatenated into an M-dimensional vector $\mathcal{M} \in (\mathbb{Z}^+)^M \subset \mathbb{R}^M$.

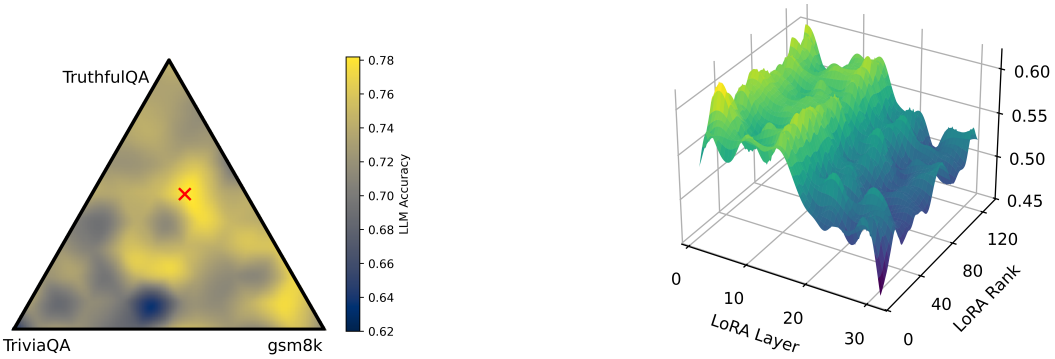
We also consider a total optimization budget $C \geq B$, which prior work (Yen et al., 2025) does not. A practitioner needs to optimize the training configuration within this budget. For instance, if our optimization budget is $C = 50000$ training steps and each full-training run consumes $B = 1000$ training steps, then we can only evaluate $C/B = 50$ different training configurations. JoBS amortizes the cost of each training run to $B_{\text{small}} < B$ with a performance predictor, allowing it to run many more BO iterations. Our work also theoretically analyzes the tradeoff between the cost of training the predictor and BO iterations in Sec. 5.4.

4 Preliminary Findings

The objective function of Problem 1 that describes the relationship between selected training components $\mathcal{X},\mathcal{M}$ and fine-tuned LLM performance $\mathcal{L}$ has no closed-form expression. As mentioned,

JoBS uses BO and a performance predictor to solve the problem. To motivate these design choices, we present several empirical findings that give us a clearer understanding of the optimization problem at hand.

4.1 Performance Landscape



(a) LLM performance varies with data mixtures.

(b) LLM performance varies with LoRA configurations.

Figure 2: LLM performance varies with different configurations.

We first investigated the performance landscape of our optimization problem. In Fig. 2, we fine-tuned a `Llama-3-8B-Instruct` (Grattafiori et al., 2024) model with varying LoRA configurations and data mixtures and plotted its performance on the GSM8K (Cobbe et al., 2021) task (averaged over 5 trials). The landscape shows that certain LoRA layer, rank, and data mixture yield better performance (more than 10%) than an arbitrarily chosen configuration. While optimizing LLM training configurations leads to significant performance improvements, the optimal training configuration is difficult to find via heuristics (Radford et al., 2019; Gao et al., 2020). In addition, the performance landscape varies rather smoothly, making BO a suitable black-box optimization approach, since we can effectively model the landscape with a smooth surrogate function (Frazier, 2018).

4.2 Amortization by Predicting Performance

We then investigated how well a small number of training steps can predict LLM performance at larger training steps in the fine-tuning setting. Fig. 3 shows that the change in LLM evaluation performance varies widely depending on the chosen training configuration. For instance, if a good training configuration is chosen (green), performance improves with more training steps. In contrast, training with a suboptimal training configuration (orange) causes performance to *decrease* as training steps increase. This is unsurprising as we expect certain data mixtures to degrade LLM performance for a mismatched task. In our setting, we need to predict LLM performance for different training configurations that we do not know beforehand. While existing scaling law formulas (Zhang et al., 2024a; Chen et al., 2025b) attempt to predict LLM performance, we cannot reuse them since they are defined for a *fixed* training configuration. Therefore, JoBS uses a more expressive and generalizable neural network (Sec. 5.2) to predict LLM performance from a small number of training steps across different training configurations. While this network does not admit an interpretable or symbolic representation like scaling laws, its primary role in our work is to provide useful signals and accelerate the optimization process.

5 Introducing JoBS

JoBS features two main components: (i) We use a surrogate Gaussian process (Williams & Rasmussen, 2006) to model the empirically smooth performance function landscape $\mathcal{L}$ (shown in Fig. 2), allowing BO to search for the optimal training configuration in a sample-efficient manner (Sec. 5.1). (ii) Before optimization, we allocate a portion of the optimization budget to learn a performance predictor that estimates the trained LLM performance from a small number of training steps (Sec. 5.2), amortizing the cost of full-training runs and increasing the BO iterations available.

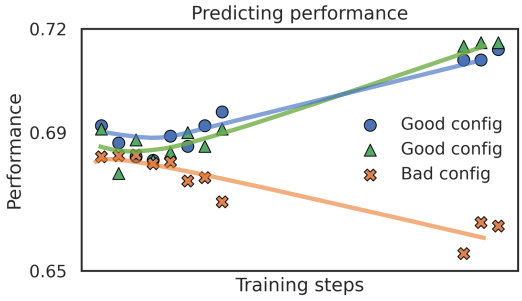


Figure 3: A neural network can be trained to predict LLM performance from small training steps.

A key question is how much optimization budget we should allocate to learn our predictor. Suppose we have an optimization budget of $C = 50000$ training steps and each full-training run consumes a budget of $B = 1000$ training steps. Running BO without a predictor admits only 50 BO iterations. However, if we use 60% of the optimization budget to collect full-training runs and train our performance predictor to predict the full-training performance from $B_{\text{small}} = 100$ training steps, we can run four times as many BO iterations: $(C - 30000)/B_{\text{small}} = 200$.

A clear tradeoff exists: allocating too much budget to train the predictor leaves fewer BO acquisition steps available to effectively search for the optimal training configuration, while collecting too few full-training runs leads to an inaccurate predictor that does not faithfully recover the true performance landscape and degrades the optimization performance. We analyze the performance tradeoff between the number of initial full-training runs and remaining BO iterations theoretically (Sec. 5.4) and empirically (Sec. 6.4), allowing us to mathematically derive the optimal budget allocation. Notably, the performance prediction does not need to be perfect; the BO framework gracefully treats any prediction noise as *observation noise*, allowing JoBS to converge to the optimal training configuration (Theorem 5.1).

5.1 BO as the Backbone of JoBS

We consider LLM performance as a function $\mathcal{L} : \mathbb{R}^d \mapsto \mathbb{R}$ over the space of inputs $x = [\mathcal{X}, \mathcal{M}] \in \mathbb{R}^d$ where $d = N + M$ (see Sec. 3). We treat our objective function in Problem 1 as a *black-box function* whose maximum $x^* \triangleq \text{argmax}_x \mathcal{L}(x)$ we want to recover. In line with existing work, we model $\mathcal{L}$ as a surrogate *Gaussian process* (GP) (Williams & Rasmussen, 2006). In each iteration $t = 1, 2, \ldots, T$, we use an LLM training configuration x_t to obtain a *noisy* realization of the LLM performance (after training with x_t) $y_t \triangleq \mathcal{L}(x_t) + \epsilon_t$, which we assume is corrupted with sub-Gaussian noise ϵ_t (e.g., normally-distributed noise) to form the sample (x_t, y_t). Consistent with the work of Chowdhury & Gopalan (2017), our GP is specified by its *prior* mean $\mu(x)$ and covariance $\kappa(x, x')$ for all $x, x' \in \mathbb{R}^d$, where κ is a *kernel* function that characterizes the correlation between two inputs x and x'.

Given the noisy observations $\boldsymbol{y}_t \triangleq [y_\tau]^\top_{\tau=1,\ldots,t}$ at inputs $x_1, \ldots, x_t$, the posterior belief of $\mathcal{L}$ at any new input x' is a Gaussian distribution with the *posterior* mean and variance given by

$$\begin{aligned} \mu_t(x') &\triangleq \kappa_t^\top(x')(K_t + \zeta I)^{-1}\boldsymbol{y}_t \\ \sigma_t(x') &\triangleq \kappa(x', x') - \kappa_t^\top(x')(K_t + \zeta I)^{-1}\kappa_t(x') \end{aligned} \tag{2}$$

where $\kappa_t(x') \triangleq [\kappa(x', x_\tau)]^\top_{\tau=1,\ldots,t}$ is a column vector, $K_t \triangleq [\kappa(x_\tau, x_{\tau'})]_{\tau,\tau' \in 1,\ldots,t}$ is a $t \times t$ covariance matrix, and $\zeta > 0$ is viewed as a free hyperparameter (Chowdhury & Gopalan, 2017). By learning the correlation between inputs and observations, modeling $\mathcal{L}$ with a GP allows us to model the relationship between training configurations and LLM performance.

Using BO for our joint optimization problem. To determine the best configuration x^*, we strategically choose an LLM training configuration to evaluate at each iteration to determine its performance and continually update the GP in (2) to better estimate $\mathcal{L}$. In round t, the BO algorithm proposes the next configuration x_{t+1} that maximizes an acquisition function, e.g., the *upper confidence bound* (UCB) (Srinivas et al., 2010). More details on BO can be found in App. B. We assess the convergence of a BO algorithm by analyzing its average regret after T BO iterations, given by

$\mathcal{R}_T \triangleq T^{-1} \sum_{t=1}^{T} (\mathcal{L}(x^*) - \mathcal{L}(x_t))$ (Tay et al., 2023) where $\mathcal{L}(x^*)$ is the optimum. We provide a theoretical analysis of JoBS's average regret in Sec. 5.4.

5.2 Amortization with Performance Predictor

We found that BO alone can already achieve a considerable improvement in LLM performance over other data and model optimization baselines (Sec. 6.2). However, directly applying BO requires us to fully train an LLM at each iteration. To amortize the cost of running these expensive trials, we learn a performance predictor to estimate the full fine-tuned LLM performance from a small number of training steps. This allows us to run significantly more BO iterations and enables us to find better-performing training configurations. Contrary to previous work (Swersky et al., 2020; Yen et al., 2025), we additionally provide a detailed analysis of the computational cost tradeoff from training this predictor.

For our predictor to work, we need to predict LLM performance for different LLM training configurations (which we do not know in advance) at each BO iteration. To do so, JoBS learns a neural network, which takes *any* LLM training configuration $[\mathcal{X}, \mathcal{M}]$ and its performance $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B_{\text{small}}})$ at timestep $B_{\text{small}} < B$ as inputs and predicts the final fine-tuned LLM performance. In our work, we use the LLM performance at a few earlier, sequential training steps to make more accurate predictions. For example, in our experiments we use the observed LLM performances before 100 training steps to predict the final LLM performance at 1000 training steps. Some examples of the predictor's predictions are shown in Fig. 3. In our ablations, we found that a larger B_{small} also allows the predictor to make more accurate predictions at the expense of consuming more optimization budget, which is consistent with Theorem 5.1.

JoBS learns the predictor in two stages. *First*, it collects a random Sobol sequence (Nguyen et al., 2018) of N LLM training configurations over $\mathcal{X}$ and $\mathcal{M}$. This ensures the predictor is trained with a diverse set of data. For each configuration, the LLM is trained to completion, and we observe LLM performance at a small number of training steps $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B_{\text{small}}})$ and after a full-training run $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B})$. These observations are also used to fit the GP surrogate that approximates the performance landscape (Sec. 5.1). *Second*, using this initial set of full-training observations, JoBS fits a neural network $\mathcal{F} : (\mathcal{X}, \mathcal{M}, \mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B_{\text{small}}})) \mapsto \mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B})$, which predicts the final LLM performance from a short training run. After learning the predictor, JoBS proceeds with BO normally (Sec. 5.1) but at every iteration, it only trains the LLM for B_{small} steps and uses $\mathcal{F}$ to predict the corresponding full-training performance $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B})$. The predicted full-training performance is then used to update the GP model for that iteration.

5.3 Prediction Error and Performance Tradeoffs

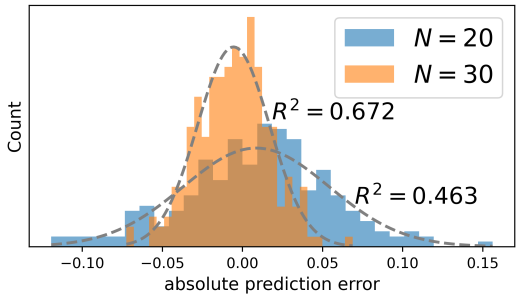


Figure 4: Prediction error of the predictor in JoBS w.r.t. different number of initial full-training runs N

Prediction error. Fig. 4 illustrates how the prediction error of $\mathcal{F}$ varies with the number N of initial full-training runs. Increasing N improves the predictor $\mathcal{F}$'s accuracy, as shown by the higher R^2 scores and tighter error concentration around zero. Empirically, we observe that the prediction error is well approximated by a normal distribution, whose variance decreases as more full-training observations are collected (i.e., increasing N). It is worth mentioning that despite prediction errors, JoBS is still shown to converge theoretically (Theorem 5.1) and empirically (Fig. 5).

Tradeoffs. It is not practical to use an excessively large N to reduce these errors. Under a given optimization budget, increasing the number of full-training observations N reduces the number of BO iterations that JoBS can perform, and vice versa. This tradeoff can be characterized analytically. Collecting N full-training runs, each requiring B training steps, consumes a budget of $N \times B$ initially. After the predictor is learned, each BO iteration incurs only B_{small} training steps. Hence, given an optimization budget C, the remaining budget $C - NB$ allows for $\lfloor (C - NB)/B_{\text{small}} \rfloor$ BO iterations (i.e., acquisition steps). In the next section, we formally analyze how the choice of N influences the convergence of JoBS.

5.4 Theoretical Analysis of JoBS

We analyze how the regret bound of JoBS varies w.r.t. the predictor error and the choice of N in JoBS. Although we cannot precisely quantify the prediction error, Fig. 4 suggests that it is R-sub-Gaussian (Chowdhury & Gopalan, 2017) whose variance increases with smaller N. The following assumption describes this characteristic.

Assumption. Let ϵ be the prediction error of the performance predictor, learned from N full-training runs over random training configurations, and its empirical distribution is shown in Fig. 4. We assume ϵ is R-sub-Gaussian with $R = \frac{k}{\sqrt{N}}$ for some constant k, suggesting that the prediction error variance decreases with a larger N.

Using this assumption, the following theorem captures JoBS's average regret given an optimization budget C and a choice on the number of initial full-training runs N.

Theorem 5.1. *Let $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B})$ be the performance landscape of the LLM training configuration with bounded RKHS norm: $\|\mathcal{L}\|_\kappa = \sqrt{\langle \mathcal{L}, \mathcal{L} \rangle_\kappa} \leq \mathcal{B}$ w.r.t. kernel κ. Assume our performance predictor is learnt from N full-training runs and makes prediction $\mathcal{F}(\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B_{small}})) = \mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B}) + \epsilon$ from $B_{small} < B$ training steps and satisfies the assumption above. Then, running JoBS over LLM training configurations $\mathcal{X}, \mathcal{M}$ until an optimization budget of C is exhausted yields $T = \left\lfloor \frac{C-NB}{B_{small}} \right\rfloor$ BO iterations, and with the IGP-UCB acquisition function (Chowdhury & Gopalan, 2017), JoBS yields the following average regret with probability at least $1 - \delta$:*

$$\mathcal{R}_T = \mathcal{O}\left(\sqrt{\frac{B_{small}}{C - NB - B_{small}}} \left(\mathcal{B}\sqrt{\gamma_T} + \frac{k}{\sqrt{N}} \sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)} \right) \right), \tag{3}$$

where γ_T is the maximum information gain of $\mathcal{L}$ after $T = \left\lfloor \frac{C-NB}{B_{small}} \right\rfloor$ BO iterations.

The proof is provided in App. C and considers how the choice of N affects the prediction error and the number of BO iterations available for optimization. We also show in App. C.1 that this bound is smaller than the average regret of vanilla BO (Chowdhury & Gopalan, 2017) under some reasonable assumptions.

Optimal budget allocation. Theorem 5.1 presents a compute-performance tradeoff: collecting more (N) initial full-training runs allows JoBS to make a more accurate performance prediction at each BO iteration. This *decreases* the $k/\sqrt{N}$ term in the average regret given by Eq. 3. However, a larger N also reduces the number of BO iterations available and *increases* the left term in Eq. 3. Using $B_{\text{small}} = 100$, $C = 50000$, and $B = 1000$ (the values adopted in our experiments), **the optimal** N (used to train our predictor) **that minimizes the average regret** $\mathcal{R}_T$ **lies approximately between** 25 **and** 35 (depending on other constants). In our ablation studies (Sec. 6.4), we empirically illustrate this tradeoff and provide practical guidelines for choosing N.

6 Experiments

We use JoBS to jointly optimize training configurations for LLMs in a variety of language tasks and LLM model types. Our experiments are divided into three parts. First, we compare JoBS against

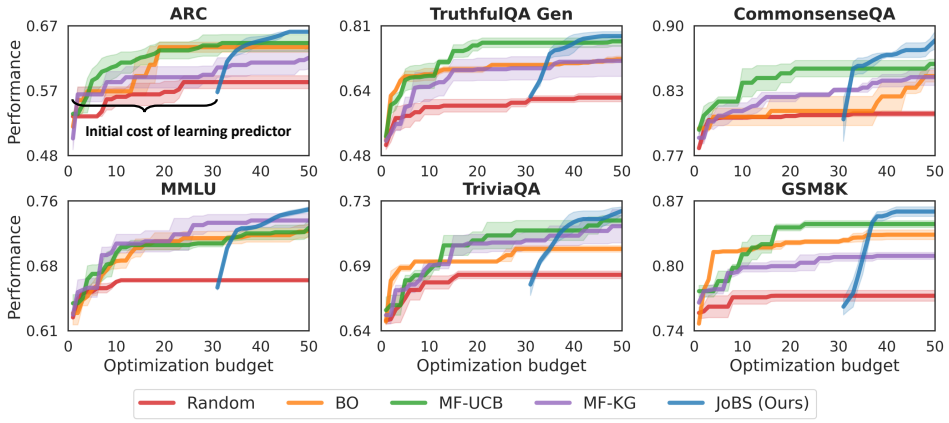


Figure 5: Comparison of LLM performance of best-found configuration at each iteration of JoBS as compared with other BO-centric approaches under the same total optimization budget of 50000 training steps, across different language tasks. JoBS's plot begins later because it allocates an initial budget to learn a performance predictor. Optimization budget is 50000 training steps.

independent data and model optimization methods. Next, we compare JoBS's convergence with various multi-fidelity BO baselines. Lastly, we perform ablations to tease apart several components in JoBS, illustrating the tradeoff between the choice of N and the number of BO iterations available for optimization.

In our main results, we used LoRA (Hu et al., 2021) to fine-tune the `Llama-3-8B-Instruct` LLM. We also extend our experiments to different model families (`Qwen`), larger models (14B, 32B), and full-parameter fine-tuning in App. H, where the results remain consistent with our key findings. Our goal is to maximize the LLM's evaluation task performance after $B = 1000$ training steps, given an overall optimization budget of $C = 50000$ training steps. This evaluation is done using `lm-evaluation-harness` (Gao et al., 2024) and evaluates the LLM performance across six language tasks (See Fig. 5). We used $N = 30$ initial full-training runs to train our performance predictor to make predictions from $B_{\text{small}} = 100$ training steps. This allows us to run $T = \frac{C-NB}{B_{\text{small}}} = 200$ BO iterations of cheaper trials in JoBS. We also used the UCB acquisition function (Srinivas et al., 2010) in JoBS. We perform ablations in Sec. 6.4 to investigate and justify our choice of N and B_{small}. More details on the experimental setup, including batch size, learning rate, and BO parameters can be found in App. D.

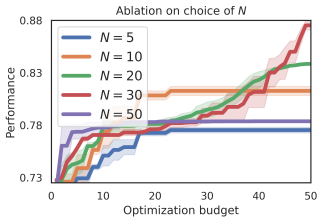
Data configuration. There are 9 data domains in our training data mixture: **Wikitext** (Merity et al., 2016), **GSM8K** (Cobbe et al., 2021), **PubmedQA** (Jin et al., 2019), **SciQ** (Welbl et al., 2017), **TriviaQA** (Joshi et al., 2017), **TruthfulQA** (Lin et al., 2022), **MMLU** (Hendrycks et al., 2021), **AI2 ARC** (Clark et al., 2018) and **CommonsenseQA** (Talmor et al., 2019). We construct a fine-tuning dataset consisting of 10000 data points by randomly sampling data points from the training datasets (Chen et al., 2025c; Xie et al., 2023a; Ye et al., 2024). As mentioned earlier, we define the data configuration $\mathcal{X} \in \mathbb{R}^9$ as the mixing ratio (a probability simplex) across these data domains.

Model configuration. The model configurations $\mathcal{M}$ we consider are LoRA hyperparameters. These include: which LLM layers to apply LoRA to, which LLM modules to apply LoRA to (e.g., Q-projection), LoRA rank, LoRA dropout and LoRA alpha, resulting in 10 model configuration dimensions in total. More information on the training configurations is provided in App. D.

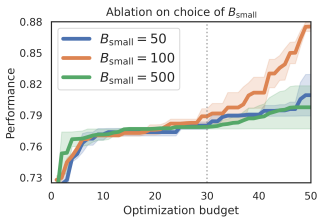
6.1 Baselines

Data optimization. LESS (Xia et al., 2024) searches for more relevant data points based on their training gradients. **DoReMi** (Xie et al., 2023a) adopts a distributionally robust approach to produce data-mixtures that work well for all evaluation task distributions. Influence Function (**IF**) (Koh & Liang, 2020) selects data points with the highest influence scores. **Diversity** (Wang et al., 2024b) finds the subset of data points with the highest log-determinant score.

Model optimization. We use a variant of Differentiable Architecture Search (**DARTS**) (Liu et al., 2019) applied to our LoRA weights by tuning an additional mixture coefficient on each LLM layer (so, when this coefficient approaches zero for a layer, we do not apply LoRA to that layer). **AutoLoRA**



(a) Varying N



(b) Varying B_{small}

Method	Time (hours)
LESS	16.3
DoReMi	18.5
IF	52
Diversity	122
JoBS	13.3

(c) Wall-clock runtime

Figure 6: (a) Ablation on choice of N, the amount of budget used to learn the performance predictor. (b) Ablation on choice of B_{small}. (c) Wall-clock runtime of JoBS compared to other data optimization approaches.

(Zhang et al., 2024b) is a baseline that automatically tunes the LoRA rank, but does not consider how we should select the layers to apply LoRA to. **RoBoT** (He et al., 2024) adopts a training-free approach to select different model configurations by aggregating different training-free metrics to measure how promising a given configuration is.

Multi-fidelity BO. We would like to emphasize that our work is the first to use BO to jointly optimize data and model components. Hence, our aim is to compare using a predictor to amortize the cost of full-training runs against other fidelity regimes. We consider multi-fidelity BO baselines that use two discrete fidelities at $B_{\text{small}} = 100$ and $B = 1000$. MF-KG uses Knowledge Gradients (Wu et al., 2019) and MF-UCB uses cost-aware UCB (Srinivas et al., 2010). Both implementations adopt an acquisition formula that is cost-aware (`https://botorch.org/docs/tutorials/multi_fidelity_bo/`). More details on baseline implementations can be found in App. E.

6.2 Comparison with Data and Model Optimization Approaches

Table 1: Combination matrix of different model and data optimization methods on GSM8K (higher is better). Subscripts denote standard deviations over 5 trials.

↓ **Model | Data** →	Default	LESS	DoReMi	IF	Diversity	JoBS
Default	$71.9_{\pm 1.5}$	$71.1_{\pm 1.2}$	$72.3_{\pm 3.2}$	$68.7_{\pm 1.0}$	$74.4_{\pm 2.0}$	-
DARTS	$73.1_{\pm 0.9}$	$71.7_{\pm 0.7}$	$74.7_{\pm 1.4}$	$69.3_{\pm 0.5}$	$66.8_{\pm 0.8}$	-
AutoLoRA	$72.9_{\pm 1.2}$	$75.2_{\pm 0.4}$	$70.9_{\pm 0.8}$	$68.5_{\pm 0.5}$	$74.0_{\pm 0.6}$	-
RoBoT	$71.8_{\pm 0.7}$	$72.7_{\pm 1.6}$	$74.0_{\pm 1.9}$	$73.1_{\pm 1.6}$	$70.2_{\pm 1.8}$	-
JoBS	-	-	-	-	-	**86.4**$_{\pm 1.2}$

First, we mixed and matched conventional data optimization and model architecture optimization methods and applied them to `Llama-3-8B-Instruct`'s training components independently. Due to space constraints, we only display the results for **GSM8K** here. Our result in Table 1 shows that pairing data and model optimization methods independently yields worse LLM performance than JoBS. This is because independently applying data and model optimization methods does not consider their joint interaction. In addition, these baselines (compared to BO-centric methods) cannot exploit performance feedback effectively. In contrast, JoBS models the complex interaction between data and model configurations and uses performance feedback directly to optimize them jointly. By doing so, JoBS performs significantly better than other baselines. The results for other tasks are shown in App. H and are generally consistent with our findings.

6.3 Comparison with Multi-Fidelity BO (MF-BO) Approaches

Next, we compare JoBS with a variety of MF-BO baselines mentioned earlier in Sec. 6.1 in Fig. 5 across all six language tasks. Given the two discrete fidelities of $B_{\text{small}} = 100$ and $B = 1000$ mentioned earlier, we allocated the same budget of $C = 50000$ steps for all methods to ensure fairness. Fig. 5 shows the best LLM evaluation performance achieved by each method under the same amount of total optimization budget. It is important to note that JoBS allocates a portion of the total

optimization budget to learn the predictor initially and hence has a shorter line plot that starts later. The result shows that after a fixed amount of optimization budget, JoBS consistently outperforms MF-BO baselines across all six tasks in terms of optimizing the evaluation task performance (Fig. 5). This suggests that using a neural network is more effective than using a covariance kernel to model the relationship between low and high fidelity performance observations (Wu et al., 2019). Furthermore, another weakness of MF-BO lies in the handcrafted cost-aware acquisition function (Yen et al., 2025) that cannot directly control how many low and high fidelity observations are queried. In contrast, our work allows us to directly control the number (N) of high-fidelity observations used to build the performance predictor. Notably, we can even drive an *optimal* N (approximately 25 to 35) in our setting.

Additional results on larger models, training settings, and performance metric. We also extended our experiments to larger models (Qwen3-14B) and the full-parameter fine-tuning setting. Additionally, we repeated the experiment in Fig. 5 to minimize evaluation loss to showcase the flexibility of our approach. The results are shown in App. H and are generally consistent with our main results, suggesting JoBS is equally effective across different model families, sizes, and training settings.

6.4 Ablation and Additional Analysis

We run several ablations to investigate key design choices in JoBS. For instance, how does the choice of N influence the convergence of JoBS under a fixed optimization budget? What happens if we adjusted B_{small}? Unless otherwise stated, our ablations are performed with Llama-3-8B-Instruct and evaluated on the CommonsenseQA task.

Optimal number of initial training samples N. Fig. 6a illustrates the tradeoff (mentioned earlier in Sec. 5.3) between the number of initial samples N used to train the performance predictor $\mathcal{F}$ and the number of BO iterations under a fixed optimization budget. Increasing N improves the accuracy of predictor $\mathcal{F}$, but also consumes more optimization budget, leaving fewer BO iterations available. As such, a value of N that is too small or too large degrades JoBS's performance. Empirically, under the optimization budget described in Sec. 6, we find that $N = 30$ strikes a good balance between predictor quality and number of BO iterations; this also agrees with our theoretical results in Sec. 5.4. This suggests that this N value optimizes the average regret bounds shown in Theorem 5.1.

Effects of varying B_{small}. Fig. 6b shows the tradeoff in the choice of B_{small}. A large B_{small} consumes more optimization budget and reduces the number of available BO iterations. On the other hand, a small B_{small} makes it difficult for us to predict the LLM performance. Both cases degrade optimization performance. In our experiments, we find that using $B_{\text{small}} = 100$ (10% of $B = 1000$) serves as a good balance, allowing JoBS to accurately predict future LLM performance while leaving sufficient budget for optimization later on.

Computational cost and other qualitative discussion. Lastly, Table 6c shows that the wall-clock runtime of JoBS (for budget of $C = 50000$) is lower than that of existing data optimization baselines (model optimization baselines are anytime algorithms so runtime comparison is not necessary). We provide a detailed computational cost analysis in App. F. We also present a few interesting case studies of the optimal LLM training configurations found by JoBS in App. G.

7 Conclusion and Future Work

We illustrated the chicken-and-egg dilemma in LLM training and introduced JoBS, an algorithm that leverages BO and a novel scaling-law-inspired performance predictor to jointly optimize data and model configurations efficiently. We analyzed the tradeoff between the budget used to learn the predictor and the leftover budget for optimization, showing that JoBS enjoys theoretical guarantees and better empirical performance than other baselines.

We foresee several exciting future research directions that address JoBS's limitations. For instance, can we incorporate existing scaling law formulas into GP priors directly to improve optimization efficiency? Can we reuse the same performance predictor *across* different tasks to improve amortization efficiency? Answering these questions can help reduce the computational cost of JoBS even further.

8 ETHICS STATEMENT

Our work strives to improve the performance of LLMs for the greater good. We do not foresee any ethical concerns related to our work. From our theoretical findings and experiments, our method does indeed improve the performance of LLMs.

REFERENCES

Mayee F. Chen, Michael Y. Hu, Nicholas Lourie, Kyunghyun Cho, and Christopher Ré. Aioli: A unified optimization framework for language model data mixing. *arXiv:2411.05735*, 2025a.

Yangyi Chen, Binxuan Huang, Yifan Gao, Zhengyang Wang, Jingfeng Yang, and Heng Ji. Scaling laws for predicting downstream performance in llms. *arXiv:2410.08527*, 2025b.

Zhiliang Chen, Chuan-Sheng Foo, and Bryan Kian Hsiang Low. Towards AutoAI: Optimizing a machine learning system with black-box and differentiable components. In *Proc. ICML*, 2024.

Zhiliang Chen, Gregory Kang Ruey Lau, Chuan-Sheng Foo, and Bryan Kian Hsiang Low. Duet: Optimizing training data mixtures via feedback from unseen evaluation tasks. *arXiv:2502.00270*, 2025c.

Sayak Ray Chowdhury and Aditya Gopalan. On kernelized multi-armed bandits. In *Proc. ICML*, 2017.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv:2110.14168*, 2021.

Zhongxiang Dai, Haibin Yu, Bryan Kian Hsiang Low, and Patrick Jaillet. Bayesian optimization meets Bayesian optimal stopping. In *Proc. ICML*, 2019.

Peter I. Frazier. A tutorial on bayesian optimization. *arXiv:1807.02811*, 2018.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling. *arXiv:2101.00027*, 2020.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL `https://zenodo.org/records/12608602`.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie

Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani,

Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

Zhenfeng He, Yao Shu, Zhongxiang Dai, and Bryan Kian Hsiang Low. Robustifying and boosting training-free neural architecture search. *arXiv:2403.07591*, 2024.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv:2009.03300*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv2203.15556*, 2022.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv:2106.09685*, 2021.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. Pubmedqa: A dataset for biomedical research question answering. *arXiv:1909.06146*, 2019.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv:1705.03551*, 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv:2001.08361*, 2020.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. *arXiv:1703.04730*, 2020.

Dong Bok Lee, Aoxuan Silvia Zhang, Byungjoo Kim, Junhyeon Park, Steven Adriaensen, Juho Lee, Sung Ju Hwang, and Hae Beom Lee. Cost-sensitive freeze-thaw bayesian optimization for efficient hyperparameter tuning, 2025. URL `https://arxiv.org/abs/2510.21379`.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv:2109.07958*, 2022.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv:1806.09055*, 2019.

Qian Liu, Xiaosen Zheng, Niklas Muennighoff, Guangtao Zeng, Longxu Dou, Tianyu Pang, Jing Jiang, and Min Lin. Regmix: Data mixture as regression for language model pre-training. *arXiv:2407.01492*, 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv:1609.07843*, 2016.

Vu Nguyen, Sunil Gupta, Santu Rana, Cheng Li, and Svetha Venkatesh. Practical batch bayesian optimization for less expensive functions. *arXiv:1811.01466*, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Sebastian Raschka. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv:1811.12808*, 2020.

Mustafa Shukor, Louis Bethune, Dan Busbridge, David Grangier, Enrico Fini, Alaaeldin El-Nouby, and Pierre Ablin. Scaling laws for optimal data mixtures. *arXiv:2507.09404*, 2025.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *Proc. ICML*, 2010.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization, 2014. URL `https://arxiv.org/abs/1406.3896`.

Kevin Swersky, Yulia Rubanova, David Dohan, and Kevin Murphy. Amortized bayesian optimization over discrete spaces. In *Conference on Uncertainty in Artificial Intelligence*, pp. 769–778. PMLR, 2020.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv:1811.00937*, 2019.

Sebastian Shenghong Tay, Chuan-Sheng Foo, Daisuke Urano, Richalynn Leong, and Bryan Kian Hsiang Low. Bayesian optimization with cost-varying variable subsets. In *Proc. NeurIPS*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. Neurips*, 2017.

Jingtan Wang, Xiaoqiang Lin, Rui Qiao, Chuan-Sheng Foo, and Bryan Kian Hsiang Low. Helpful or harmful data? fine-tuning-free shapley attribution for explaining language model predictions. *arXiv:2406.04606*, 2024a.

Peiqi Wang, Yikang Shen, Zhen Guo, Matthew Stallone, Yoon Kim, Polina Golland, and Rameswar Panda. Diversity measurement and subset selection for instruction tuning datasets. *arXiv:2402.02318*, 2024b.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Workshop on Noisy User-generated Text*, 2017.

Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. *arXiv:1910.11858*, 2020.

Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

Chuhan Wu and Ruiming Tang. Performance law of large language models. *arXiv:2408.09895*, 2024.

Jian Wu, Saul Toscano-Palmerin, Peter I. Frazier, and Andrew Gordon Wilson. Practical multi-fidelity bayesian optimization for hyperparameter tuning. *arXiv:1903.04703*, 2019.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv:2402.04333*, 2024.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy Liang, Quoc V. Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *arXiv:2305.10429*, 2023a.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling. *arXiv:2302.03169*, 2023b.

Wanyun Xie, Francesco Tonin, and Volkan Cevher. Chameleon: A flexible data-mixing framework for language model pretraining and finetuning, 2025.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Yunhua Zhou, Jun Zhan, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance. *arXiv:2403.16952*, 2024.

Thomson Yen, Andrew Wei Tung Siah, Haozhe Chen, Tianyi Peng, Daniel Guetta, and Hongseok Namkoong. Data mixture optimization: A multi-fidelity multi-scale bayesian framework, 2025.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When scaling meets llm finetuning: The effect of data, model and finetuning method. *arXiv:2402.17193*, 2024a.

Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Qiu Jiantao, Lei Cao, Ju Fan, Ye Yuan, Guoren Wang, and Conghui He. Harnessing diversity for important data selection in pretraining large language models. In *Proc. ICLR*, 2025.

Ruiyi Zhang, Rushi Qiang, Sai Ashish Somayajula, and Pengtao Xie. Autolora: Automatically tuning matrix ranks in low-rank adaptation based on meta learning. *arXiv:2403.09113*, 2024b.

A Relationship with Multi-Fidelity Bayesian Optimization

Multi-fidelity BO works typically have the option to query the black-box function at high-fidelity or low-fidelity, with low-fidelity queries being cheaper (and the acquisition function is incentivized to query them). High-fidelity function queries follow the real black-box function faithfully, while low-fidelity queries are inaccurate estimations of the real function. Multi-fidelity BO considers an *additional dimension of fidelity* in the input variables and uses a cost-aware acquisition function (Wu et al., 2019) to propose which fidelity to query at every BO iteration.

In our setting, observing the LLM performance after a fraction of the training steps can be considered a low-fidelity evaluation, while observing the LLM performance after full-training can be viewed as a high-fidelity evaluation. JoBS first uses many high-fidelity evaluations to build a performance predictor. Then, we perform BO entirely using low-fidelity evaluations given by the predictor. JoBS models the relationship between low-fidelity observations and high-fidelity observations explicitly with an expressive neural network (called a performance predictor in our work), *which is decoupled from the GP surrogate model*. In contrast, Multi-fidelity BO *jointly* models the relationship between low and high-fidelity observations and inputs with a GP (by treating fidelity as an additional input dimension). While this sounds practical on paper, a GP is often not expressive enough to model them jointly and hence Multi-fidelity BO on its own does not work well in our experiments. The same can be said for early-stopping BO (Dai et al., 2019), which uses a GP to model different fidelities. In addition, most of these works do not work out the optimal queries to each fidelity. To overcome these limitations, our work provides theoretical guidelines for choosing an optimal number of high-fidelity observations to query (Sec. 5.4).

There is another line of work called freeze-thaw BO (Swersky et al., 2014; Lee et al., 2025) that decides whether to continue training/fine-tuning an old training configuration or to explore a new set of training configurations. We do not find this approach suitable for LLM training for a few reasons. *First*, they require storing multiple copies of fine-tuned LLMs with different training configurations, which is memory-intensive. *Second*, they typically use an exponential decay kernel to model the increase in LLM performance (or decrease in loss) with more training time. However, Fig. 3 shows that in our setting, the change in LLM performance *can even decrease* with more training steps because a chosen data mixture might be irrelevant for the downstream task. As such, existing exponential decay kernel introduced in these works does not work well in our setting.

B Additional Background on Bayesian Optimization

At every BO iteration, the BO algorithm proposes the next configuration x_{t+1} as the configuration which maximizes some acquisition function, such as the *upper confidence bound* (UCB) (Srinivas et al., 2010), given by $x_{t+1} = \text{argmax}_x \, \mu_t(x) + \beta_{t+1}\sigma_t(x)$, where β_{t+1} is an exploration parameter.

A common kernel choice is the *squared exponential* (SE) kernel $\kappa(x, x') \triangleq \exp(-\|x - x'\|_2^2/(2m^2))$ with a *length-scale* hyperparameter m that can be learned via maximum likelihood estimation from observations.

C Proof of Theorem 5.1

Theorem 5.1. *Let $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B})$ be the performance landscape of the LLM training configuration with bounded RKHS norm: $\|\mathcal{L}\|_\kappa = \sqrt{\langle \mathcal{L}, \mathcal{L} \rangle_\kappa} \leq \mathcal{B}$ w.r.t. kernel κ. Assume our performance predictor is learnt from N full-training runs and makes prediction $\mathcal{F}(\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B_{small}})) = \mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B}) + \epsilon$ from $B_{small} < B$ training steps and satisfies the assumption above. Then, running JoBS over LLM training configurations $\mathcal{X}, \mathcal{M}$ until an optimization budget of C is exhausted yields $T = \left\lfloor \frac{C - NB}{B_{small}} \right\rfloor$ BO iterations, and with the IGP-UCB acquisition function (Chowdhury & Gopalan, 2017), JoBS yields the following average regret with probability at least $1 - \delta$:*

$$\mathcal{R}_T = \mathcal{O}\Bigg(\sqrt{\frac{B_{small}}{C - NB - B_{small}}} \Big(\mathcal{B}\sqrt{\gamma_T} + \frac{k}{\sqrt{N}}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\Big)\Bigg), \tag{3}$$

where γ_T is the maximum information gain of $\mathcal{L}$ after $T = \left\lfloor \frac{C-NB}{B_{small}} \right\rfloor$ BO iterations.

Proof. To begin, recall that we are trying to maximize our LLM performance, a black-box function $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B})$ (Sec. 3). Using our scaling law predictor (Sec. 5.2), we instead train our LLM for B_{small} training steps (or time) and observe $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B_{\text{small}}})$. We then use the performance predictor to produce $\mathcal{F}(\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B_{\text{small}}}))$, estimating the LLM performance if we trained it fully for B training steps. Since we are predicting the LLM performance, our model prediction is noisy with $\mathcal{F}(\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B_{\text{small}}})) = \mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B}) + \epsilon$. Hence, we only have access to a noisy estimate of our black-box function: $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B}) + \epsilon$. The prediction error is also R-sub-Gaussian as suggested from our empirical findings (Fig. 4). As such, this formulation is in line with the BO algorithmic framework introduced in Sec. 5.1, where we observe noisy observations of the true underlying function.

Next, we aim to draw relationship between the choice of N with our prediction error ϵ and consequentially, JoBS's regret. First, we present the following lemma from (Chowdhury & Gopalan, 2017):

Lemma C.1. *Let $||f||_\kappa = \sqrt{\langle f, f\rangle_\kappa} \leq \mathcal{B}$. Also, assume that the observation noise associated with each BO iteration is R-sub-Gaussian with $R > 0$. Then with probability at least $1 - \delta$, the following holds for BO iteration $t \leq T$:*

$$|\mu_t(x) - f(x)| \leq \left(\mathcal{B} + R\sqrt{2(\gamma_t + 1 + \ln(1/\delta))}\right)\sigma_t(x) \tag{4}$$

where γ_t is the maximum information gain after t observations and $\mu_t(x), \sigma_t^2(x)$ are mean and variance of posterior distribution of GP defined in Eq. 2 with $\zeta = 1 + 2/T$.

In our setting, set $f = \mathcal{L}$ (our LLM performance after fine-tuning) and $x = \mathcal{X}, \mathcal{M}$ (our LLM training configuration). This lemma indicates that our estimated mean $\mu_t(x)$ of our performance landscape from our fitted GP over historical observations of LLM performance deviates from the true LLM performance $f(x) = \mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B})$ by at most the term in Eq. C.1.

We are now ready to prove Theorem 5.1. First, we observe that the next LLM training configuration x_t at each BO iteration t is chosen via the IGP-UCB acquisition function (i.e., $x_t = \text{argmax}_x\, \mu_{t-1}(x) + \beta_t\sigma_{t-1}(x)$ and $\beta_t = \mathcal{B} + R\sqrt{2(\gamma_{t-1} + 1 + \ln(1/\delta))}$ where the observation noise associated with each BO iteration is R-sub Gaussian). Thus, we can see that at each iteration $t \geq 1$, we have $\mu_{t-1}(x_t) + \beta_t\sigma_{t-1}(x_t) \geq \mu_{t-1}(x^*) + \beta_t\sigma_{t-1}(x^*)$. It then follows that for all $t \geq 1$ and with probability at least $1 - \delta$,

$$\begin{aligned} |f(x^*) - f(x_t)| &\overset{(1)}{\leq} \beta_t\sigma_{t-1}(x_t) + \mu_{t-1}(x_t) - f(x_t) \\ &\overset{(2)}{\leq} \beta_t\sigma_{t-1}(x_t) + \mu_{t-1}(x_t) + (\beta_t\sigma_{t-1}(x_t) - \mu_{t-1}(x_t)) \\ &\leq 2\beta_t\sigma_{t-1}(x_t) \end{aligned} \tag{5}$$

where $\overset{(1)}{\leq}$ uses the fact that via Lemma C.1 and our acquisition function, $f(x^*) \leq \beta_t\sigma_{t-1}(x^*) + \mu_{t-1}(x^*) \leq \beta_t\sigma_{t-1}(x_t) + \mu_{t-1}(x_t)$ and $\overset{(2)}{\leq}$ once again uses Lemma C.1.

Next, we adjust the total number of BO iterations in our problem setting, depending on the chosen N. As mentioned in the analysis of Sec. 5.3, under an overall optimization budget of C, collecting N initial full-training runs leaves enough budget for $\lfloor \frac{C-NB}{B_{\text{small}}} \rfloor$ BO iterations. Hence, we analyze how JoBS's cumulative regret grows for up to $T = \lfloor \frac{C-NB}{B_{\text{small}}} \rfloor$ iterations. Using Eq. 5, we can bound the

cumulative regret by

$$\sum_{t=1}^{T} r_t = \sum_{t=1}^{T} (f(x^*) - f(x_t)) \leq 2 \sum_{t=1}^{T} \beta_t \sigma_{t-1}(x_t) \,. \tag{6}$$

Since we know that $\sum_{t=1}^{T} \sigma_{t-1}(x_t) = \mathcal{O}(\sqrt{T\gamma_T})$ and used $\beta_t = \mathcal{B} + R\sqrt{2(\gamma_{t-1} + 1 + \ln(1/\delta))}$, the cumulative regret in Theorem 5.1 can be written as

$$\begin{aligned}
\mathcal{R}_T &= \sum_{t=1}^{T} r_t && (7)\\
&\leq 2 \sum_{t=1}^{T} \beta_t \sigma_{t-1}(x_t) && (8)\\
&\leq 2\mathcal{O}(\sqrt{T\gamma_T})(\mathcal{B} + R\sqrt{2(\gamma_T + 1 + \ln(1/\delta))}) && (9)\\
&= \mathcal{O}\left(\mathcal{B}\sqrt{T\gamma_T} + R\sqrt{T}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right) && (10)\\
&\overset{(1)}{=} \mathcal{O}\left(\mathcal{B}\sqrt{\left\lfloor \frac{C - NB}{B_{\text{small}}} \right\rfloor \gamma_T} + R\sqrt{\left\lfloor \frac{C - NB}{B_{\text{small}}} \right\rfloor}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right) && (11)\\
&\overset{(2)}{=} \mathcal{O}\left(\mathcal{B}\sqrt{\left\lfloor \frac{C - NB}{B_{\text{small}}} \right\rfloor \gamma_T} + \frac{k}{\sqrt{N}}\sqrt{\left\lfloor \frac{C - NB}{B_{\text{small}}} \right\rfloor}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right), && (12)
\end{aligned}$$

where $\overset{(1)}{=}$ uses the fact that if we collected N full-training runs initially, the number of BO iterations available is $T = \left\lfloor \frac{C-NB}{B_{\text{small}}} \right\rfloor$ and $\overset{(2)}{=}$ uses **Assumption 1** that our prediction error is R-sub-Gaussian with $R = \frac{k}{\sqrt{N}}$.

Lastly, because different number of initial full-training runs N will influence the number of remaining BO iterations, we derive the average regret w.r.t. the number of BO iterations by dividing the cumulative regret bounds obtained in Eq. (14) by $\left\lfloor \frac{C-NB}{B_{\text{small}}} \right\rfloor$ throughout:

$$\begin{aligned}
\frac{\mathcal{R}_T}{T} &\overset{(1)}{=} \mathcal{O}\left(\mathcal{B}\sqrt{\frac{\gamma_T B_{\text{small}}}{C - NB - B_{\text{small}}}} + \frac{k}{\sqrt{N}}\sqrt{\frac{B_{\text{small}}}{C - NB - B_{\text{small}}}}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right) && (13)\\
&= \mathcal{O}\left(\left(\sqrt{\frac{B_{\text{small}}}{C - NB - B_{\text{small}}}}\right)\left(\mathcal{B}\sqrt{\gamma_T} + \frac{k}{\sqrt{N}}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right)\right) && (14)
\end{aligned}$$

where $\overset{(1)}{=}$ divides the cumulative regret derived in Eq. (14) throughout by $T = \left\lfloor \frac{C-NB}{B_{\text{small}}} \right\rfloor$ and considers the fact that $\frac{1}{\lfloor \frac{a}{b} \rfloor} \leq \frac{b}{a-b}$ if $a \neq 0$.

One interesting point to note is that our algorithm incurs some regret when collecting N random training configurations initially. However, since our black-box function is bounded (under the SE-kernel) and N is not assumed to be dependent on the total optimization budget T, then the average regret incurred when collecting training configurations initially can be removed in the big-O notation. □

C.1 Comparison of Regret Bound in Vanilla BO

We will show the conditions in which the average regret upper-bound of JoBS is smaller than that of the vanilla BO case. To simplify notation, let T be the number of BO iterations available in the vanilla BO case. Recall that in normal BO (without any performance predictor), the cumulative regret

is (Chen et al., 2024; Chowdhury & Gopalan, 2017)

$$\mathcal{O}\left(\mathcal{B}\sqrt{T\gamma_T} + R\sqrt{T}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right). \tag{15}$$

Dividing by T, we obtain an average regret of:

$$\mathcal{O}\left(\frac{\mathcal{B}\sqrt{\gamma_T}}{\sqrt{T}} + \frac{R\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}}{\sqrt{T}}\right). \tag{16}$$

Consider that in our case, JoBS devotes a constant number of iterations (e.g., $N << T$) to learn the performance predictor, which yields noisy observations with error distribution that is R_1-sub-Gaussian, where $R_1 > R$. This means the predictor does not recover the original performance landscape perfectly. By doing so, each BO iteration consumes 10% of the original resources (since we only need to fine-tune an LLM to 10% of the original number of training steps) and hence, we can perform $10(T - N)$ more BO iterations. Assuming that the average regret of the first N iteration where we make random queries is constant (we can ignore it in the asymptotic bound), this implies the average regret bound of JoBS is:

$$\mathcal{O}\left(\frac{\mathcal{B}\sqrt{\gamma_T}}{\sqrt{10(T-N)}} + \frac{R_1\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}}{\sqrt{10(T-N)}}\right). \tag{17}$$

Note that this is essentially the same bound as Theorem 5.1 but without the complication of deriving the expression for the number of BO acquisition steps available under a limited optimization budget. Comparing Eq.(16) and Eq.(17), we see that the average regret bound of JoBS is smaller than vanilla BO with probability at least $1 - \delta$ if and only if:

$$\left(\mathcal{B}\sqrt{\gamma_T} + R_1\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right) < \left(\mathcal{B}\sqrt{\gamma_T} + R\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right)\sqrt{\frac{10(T-N)}{T}}. \tag{18}$$

The differences between the left and right side are highlighted in red and this inequality admits a neat form that is easy to interpret: R_1 is larger than R (i.e., using a predictor gives noisier observations), but as long as $\sqrt{\frac{10(T-N)}{T}}$ is much larger (i.e., we can run more BO iterations; it is almost 10 times larger in this case), the average regret of JoBS will be smaller than that found in the vanilla BO case.

D Additional Experimental Details

We provide details of how we performed our experiments for JoBS. Our data configuration consists of 9 parameters representing the mixing ratio (a probability simplex). Our model configuration consists of 10 parameters, which represent the following:

1. Number of LLM layers to apply LoRA to $\in [1, 31]$ (this varies for different LLMs, depending on how many transformer layers are present),
2. Whether to apply LoRA to front layers or rear layers (binary decision),
3. Whether to apply LoRA to Q-projection layer (binary decision),
4. Whether to apply LoRA to V-projection layer (binary decision),
5. Whether to apply LoRA to K-projection layer (binary decision),
6. Whether to apply LoRA to MLP-Up-projection layer (binary decision),
7. Whether to apply LoRA to MLP-Down-projection layer (binary decision),
8. LoRA rank $\in [1, 256]$,
9. LoRA dropout $\in [0, 1]$, and
10. LoRA alpha $\in [1, 500]$.

In the baselines in which we do not optimize a certain component, we adopt the following **default fine-tuning configuration**: (a) Apply LoRA on all LLM layers, (b) apply LoRA to all Q-, V-, K-, MLP-Up-, MLP-Down-projection layers, (c) LoRA rank $= 16$, (d) dropout $= 0.1$, (e) alpha $= 16$, and (f) uniform training data mixture. These component configurations align with many off-the-shelf LLM training configurations provided in existing papers and online tutorials. So, in the case where we only optimize data configuration, we will use the default model configuration above. In all our main results, we used 8-shot prompting with chain-of-thought. We used a shortened training time of $B_{\text{small}} = 100$ training steps and an optimization budget of $C = 50000$ training steps.

To build our performance scaling law predictor (Sec. 5.2), we collected a random Sobol sequence (Nguyen et al., 2018) of 30 LLM training configurations and fine-tuned the LLM for $B = 1000$ training steps. This is around 1.5 training epochs. During training, we collected the LLM validation loss or performance (depending on the experimental setting) at training step intervals of 25. We build a dataset consisting of the observed loss or performance at training steps $= 25, 50, 75, 100$ as the neural network input. We also observe the full fine-tuning loss or performance at a large timestep $B = 1000$ training steps and treat it as the prediction ground-truth. Using this dataset, we train a densely-connected, 64-width, 3 layers neural network $\mathcal{F}$ to predict the full fine-tuning performance.

This random sequence of ground-truths collected initially is also added to our initial GP model to warm-start BO in `JoBS`. This leaves an optimization budget for $\frac{50000-30\times1000}{100} = 200$ BO iterations of cheaper trials in `JoBS`. We used a SE-kernel for the GP used to model our LLM performance landscape, and ran our experiments with the `BoTorch` library. At the end of every iteration, we use maximum-likelihood to estimate the hyperparameters in kernel. We normalize and rescale all our LLM training configuration parameters to be between 0 and 1 when fitting our GP. We also adopted a SE-kernel for our GP. Throughout our experiments, we used 4 `H200` GPUs and a training batch size of 64.

E Implementation Details of Baselines

LESS (Xia et al., 2024). We pass each training data from each data domain into the LLM to construct a gradient store (Xia et al., 2024). Since we do not know the exact evaluation task (just like the setting considered in Chen et al. (2025c)), we form a validation mixture consisting of uniformly sampled data from each data domain, before retrieving a subset of 10000 data points from the gradient store with the highest feature similarity with the evaluation dataset. We follow the implementation given in `https://github.com/princeton-nlp/LESS`.

DoReMi (Xie et al., 2023a). We use DoReMi to optimize the data mixing ratio across the training data domains, according to the implementation given in `https://github.com/sangmichaelxie/doremi`. Again, since we do not have direct access to the evaluation task (as DoReMi requires an explicit validation dataset), we form the validation dataset using a uniform validation mixture similar to LESS (see above).

IF (Koh & Liang, 2020). We compute the influence scores of each data point (following the implementation in `https://github.com/alstonlo/torch-influence` w.r.t. a uniform validation mixture constructed from the training data domains (see above). Then, we retrieve the top 10000 data points with the highest influence scores.

Diversity (Wang et al., 2024b). We first project every data point into a continuous semantic space using an off-the-shelf embedding model. Then, we retrieve the data subset with the highest log-determinant value, which is derived by arranging data point's embedding in a matrix and computing the determinant. There is a greedy implementation (Wang et al., 2024b) that improves the computational efficiency.

DARTS (Liu et al., 2019). We optimize the same model configurations detailed in App. D by introducing a differentiable parameter for each model component (e.g., LoRA rank). For example, LoRA rank is controlled by a single differentiable parameter, while LoRA layer selection is controlled by a softmax layer with weights, allowing us to drop certain LoRA layers (if the weight is decreased to 0 after update). We alternate between freezing the differentiable parameters that control the model configuration and model parameters that control the LLM behavior every 5 training steps. We optimize these objectives in an alternating fashion, switching between updating configuration parameters and model parameters every 5 steps.

AutoLoRA (Zhang et al., 2024b). We follow the implementation in `https://github.com/ruz048/AutoLoRA`, which tunes the LoRA rank for each layer.

RoBoT (He et al., 2024). We utilize the training-free proxies given in (He et al., 2024) and perform BO over the weights assigned to each training-free proxies. With the optimized weights, we then iterate through the model configurations to find the one with the best (weighted) training-free measure.

MF-KG (Wu et al., 2019). The implementation is reproduced from `https://botorch.org/docs/tutorials/multi_fidelity_bo/`.

MF-UCB. We use a cost-aware UCB acquisition function that follows the framework in Yen et al. (2025).

F Computational Cost of JoBS vs. Baselines

Quantitative comparison of wall-clock hours. JoBS is run for 50000 training steps in total. This equates to approximately 50000 seconds of wall-clock time (13.3 hours) using a H200 GPU. All model selection methods (He et al., 2024; Liu et al., 2019) used in our paper are iterative in nature and require repeated fine-tuning of LLMs. Equal time budget was allocated to both JoBS and all baseline methods for a fair comparison. We found that given equal computation time, JoBS achieves better performance than all model selection methods. For data selection methods (LESS, DoReMi, IF, Diversity), we record their wall-clock runtime in Table 2. In general, we found data selection methods to be more computationally expensive than JoBS.

Table 2: Wall-clock runtime comparison of data selection methods vs. JoBS.

Method	Time (hours)
LESS	16.3
DoReMi	18.5
IF	52
Diversity	122
JoBS	**13.3**

G Optimal LLM Training Configurations Found by JoBS Compared to Baselines

We show some of the optimal LLM training configurations found by JoBS vs. other baselines. We divided the configurations into two tables detailing the best data (Table 3) and model (Table 4) configurations found for the **GSM8K** evaluation task.

Table 3: Optimal data mixing ratio found by JoBS vs. other baselines. The columns denote the ratio allocated to each training domain.

	CQA	GSM8K	PubmedQA	SciQ	TrivQA	TruthQA	Wiki	MMLU	ARC
JoBS	0	0.34	0	0.10	0.19	0	0.21	0.16	0
DoReMi	0.08	0.11	0.18	0.05	0.08	0.14	0.04	0.16	0.13

Of particular interest is that JoBS optimizes the data mixture by placing greater weights on some data domains based on their evaluation performance on the downstream task (in this case, GSM8K). Specifically, JoBS successfully inferred (without knowing that the evaluation task is GSM8K) that domains such as GSM8K, SciQ, TriviaQA, Wikipedia, and MMLU contain some math information and therefore includes them in the optimized data mixture.

On the other hand, DoReMi is a distributionally robust data mixing method and results in a more uniform data mixing ratio. As a result, the output data mixtures are not specialized for any single

Table 4: Optimal model configuration found by JoBS vs. other baselines.

	Rank	NumLayers	Order	Q	K	V	Up	Down	dropout	α
JoBS	36	25	1	1	0	1	1	0	0.112	64
DARTS	12	13	0	1	1	1	1	0	0.058	45

evaluation task, leading to lower performance when evaluated on specific individual tasks, such as GSM8K.

Next, we examine the optimal model configurations found by JoBS. We noticed that JoBS prefers a higher LoRA rank and number of layers (i.e., how many layers to apply LoRA), but chooses to apply LoRA to only certain transformer layers. In particular, JoBS found that for the GSM8K evaluation task, fine-tuning Q-, V-, and MLP-Up-projection layers is sufficient to achieve good fine-tuning performance, and we should fine-tune the rear layers instead of the front layers (Order $= 1$).

H Additional Experimental Results and Discussion

In Tables 5, 6, 7, 8, and 9, we repeated the experimental set-up from Table 1 by performing mix-and-match on different model and data selection methods over 5 other evaluation tasks (TruthfulQA, CommonsenseQA, MMLU, ARC, and TriviaQA). The results show that JoBS outperforms all combinations of data and model selection works. This suggests that jointly optimizing both data and model configurations does indeed produce *interaction improvement* over optimizing the configurations independently. In addition, from running our experiments, we find our approach significantly easier to implement in code.

Table 5: TruthfulQA (Lin et al., 2022).

↓ **Model | Data** →	Default	LESS	DoReMi	IF	Diversity	BO	JoBS
Default	$59.8_{\pm 1.9}$	$60.7_{\pm 1.0}$	$62.9_{\pm 2.6}$	$62.3_{\pm 1.3}$	$64.5_{\pm 1.2}$	$74.8_{\pm 1.0}$	-
DARTS	$61.9_{\pm 1.3}$	$62.4_{\pm 0.6}$	$66.7_{\pm 1.3}$	$63.8_{\pm 0.5}$	$64.8_{\pm 1.2}$	$77.0_{\pm 0.9}$	-
AutoLoRA	$61.7_{\pm 1.0}$	$66.9_{\pm 1.2}$	$64.5_{\pm 1.1}$	$65.3_{\pm 1.2}$	$65.9_{\pm 0.6}$	$72.5_{\pm 0.5}$	-
RoBoT	$64.2_{\pm 0.6}$	$65.8_{\pm 0.7}$	$58.9_{\pm 1.3}$	$56.9_{\pm 1.0}$	$65.8_{\pm 0.6}$	$73.9_{\pm 1.3}$	-
BO	$66.8_{\pm 1.2}$	$67.9_{\pm 0.5}$	$69.8_{\pm 0.9}$	$70.9_{\pm 1.0}$	$64.7_{\pm 1.4}$	$76.1_{\pm 2.0}$	-
JoBS	-	-	-	-	-	-	$\mathbf{80.6}_{\pm 1.1}$

Table 6: CommonsenseQA (Talmor et al., 2019).

↓ **Model | Data** →	Default	LESS	DoReMi	IF	Diversity	BO	JoBS
Default	$76.3_{\pm 1.0}$	$73.0_{\pm 0.8}$	$74.2_{\pm 1.7}$	$79.3_{\pm 0.7}$	$77.4_{\pm 1.7}$	$80.6_{\pm 0.8}$	-
DARTS	$79.6_{\pm 1.3}$	$76.3_{\pm 1.7}$	$76.1_{\pm 1.1}$	$73.7_{\pm 1.2}$	$80.1_{\pm 1.1}$	$79.6_{\pm 0.6}$	-
AutoLoRA	$78.9_{\pm 0.9}$	$79.8_{\pm 0.4}$	$76.1_{\pm 0.5}$	$77.9_{\pm 1.2}$	$78.0_{\pm 1.0}$	$81.5_{\pm 1.0}$	-
RoBoT	$74.9_{\pm 0.8}$	$75.5_{\pm 0.9}$	$77.1_{\pm 0.9}$	$79.4_{\pm 1.5}$	$76.3_{\pm 0.9}$	$80.2_{\pm 0.2}$	-
BO	$79.7_{\pm 1.3}$	$79.4_{\pm 0.3}$	$77.0_{\pm 0.4}$	$81.1_{\pm 0.9}$	$79.4_{\pm 1.1}$	$80.7_{\pm 1.2}$	-
JoBS	-	-	-	-	-	-	$\mathbf{84.3}\pm 2.4$

Table 7: MMLU (Hendrycks et al., 2021).

↓ **Model | Data** →	Default	LESS	DoReMi	IF	Diversity	BO	JoBS
Default	$61.2_{\pm 1.3}$	$63.5_{\pm 0.9}$	$59.7_{\pm 1.8}$	$57.9_{\pm 0.6}$	$62.1_{\pm 1.4}$	$64.2_{\pm 1.2}$	-
DARTS	$58.3_{\pm 0.7}$	$61.0_{\pm 2.1}$	$62.9_{\pm 1.0}$	$55.7_{\pm 1.6}$	$60.1_{\pm 0.5}$	$63.4_{\pm 2.0}$	-
AutoLoRA	$62.5_{\pm 1.4}$	$64.3_{\pm 0.6}$	$60.8_{\pm 2.2}$	$58.2_{\pm 1.9}$	$63.7_{\pm 0.8}$	$61.5_{\pm 1.1}$	-
RoBoT	$59.9_{\pm 0.9}$	$60.7_{\pm 1.2}$	$63.4_{\pm 1.7}$	$61.5_{\pm 1.5}$	$58.3_{\pm 2.3}$	$62.1_{\pm 0.7}$	-
BO	$55.8_{\pm 1.8}$	$57.2_{\pm 0.4}$	$61.3_{\pm 1.2}$	$60.5_{\pm 1.6}$	$63.9_{\pm 1.0}$	$59.6_{\pm 1.5}$	-
JoBS	-	-	-	-	-	-	$\mathbf{69.5}_{\pm 0.8}$

Table 8: ARC (Clark et al., 2018).

↓ **Model | Data** →	Default	LESS	DoReMi	IF	Diversity	BO	JoBS
Default	$54.7_{\pm 1.3}$	$59.2_{\pm 0.7}$	$61.4_{\pm 2.0}$	$52.8_{\pm 1.5}$	$60.6_{\pm 0.9}$	$62.3_{\pm 1.2}$	-
DARTS	$58.1_{\pm 0.8}$	$61.0_{\pm 1.6}$	$62.8_{\pm 0.5}$	$54.3_{\pm 2.1}$	$57.9_{\pm 1.7}$	$60.5_{\pm 0.6}$	-
AutoLoRA	$60.4_{\pm 1.1}$	$63.2_{\pm 0.9}$	$58.6_{\pm 1.9}$	$55.1_{\pm 0.8}$	$62.1_{\pm 2.0}$	$59.8_{\pm 1.0}$	-
RoBoT	$56.8_{\pm 1.5}$	$58.7_{\pm 1.4}$	$61.1_{\pm 1.2}$	$60.3_{\pm 0.7}$	$55.7_{\pm 2.2}$	$61.4_{\pm 1.3}$	-
BO	$52.6_{\pm 2.0}$	$55.9_{\pm 0.6}$	$59.7_{\pm 1.3}$	$58.5_{\pm 1.4}$	$63.4_{\pm 1.0}$	$57.4_{\pm 0.9}$	-
JoBS	-	-	-	-	-	-	$\mathbf{70.4}_{\pm 1.3}$

Table 9: TriviaQA Gen (Joshi et al., 2017).

↓ **Model | Data** →	Default	LESS	DoReMi	IF	Diversity	BO	JoBS
Default	$55.5_{\pm 1.4}$	$57.2_{\pm 0.8}$	$53.1_{\pm 0.9}$	$55.8_{\pm 0.7}$	$58.9_{\pm 0.8}$	$65.0_{\pm 0.6}$	-
DARTS	$58.2_{\pm 0.8}$	$61.3_{\pm 1.2}$	$61.0_{\pm 0.7}$	$63.3_{\pm 1.0}$	$59.2_{\pm 0.6}$	$66.7_{\pm 1.8}$	-
AutoLoRA	$67.8_{\pm 1.4}$	$64.7_{\pm 0.9}$	$70.6_{\pm 2.2}$	$68.6_{\pm 1.7}$	$66.2_{\pm 1.5}$	$69.7_{\pm 2.4}$	-
RoBoT	$58.4_{\pm 1.5}$	$62.3_{\pm 1.7}$	$64.2_{\pm 1.4}$	$57.2_{\pm 1.2}$	$63.4_{\pm 1.5}$	$68.2_{\pm 1.3}$	-
BO	$70.7_{\pm 1.4}$	$66.7_{\pm 0.8}$	$72.5_{\pm 0.8}$	$71.7_{\pm 0.9}$	$74.7_{\pm 1.0}$	$72.7_{\pm 2.3}$	-
JoBS	-	-	-	-	-	-	$\mathbf{76.2}\pm 1.9$

H.1 Additional Baselines and Extension to Larger Model and Different Evaluation Metrics

Additional baselines. In Table 10, we jointly optimized LLM training configurations using several other naive approaches in our experiments. We tried 3 naive approaches: (a) **Random** randomly picking 500 different LLM training configurations, fine-tune them for 100 training steps each, use our performance scaling law predictor to predict and select the best-performing LLM training configuration. (b) **Random Data** perform JoBS on model configurations for only 10 iterations and repeat the experiment with 10 randomly chosen data configurations (this ensures the same amount of compute as performing JoBS on all LLM training configurations for 100 iterations). (c) **Random Model** repeat approach (b) on LLM training configurations instead. While these approaches serve as good sanity checks, they do not yield good LLM performances, largely because randomly selecting LLM training configurations does not exploit the learnt performance landscape from historically observed LLM performances.

Extension to different model sizes, full-parameter fine-tuning and different performance metrics. We extended our experiments to larger model sizes (Qwen3-14B (Fig. 8) and also to full-parameter fine-tuning (Table 11). In particular, when performing full-parameter fine-tuning, we do not use LoRA, but instead optimize which LLM layer to fine-tune.

Table 10: Comparison of some naive baselines with JoBS (Higher is better), averaged over 5 trials. **Random Data** means that we randomly selected data mixtures and applied JoBS only on the model configurations (and vice versa for **Random Model**).

Model	Task	**Random Data**	**Random Model**	JoBS
Llama-3-8B-Instruct	GSM8K	$67.3_{\pm 1.6}$	$71.5_{\pm 0.9}$	$86.4_{\pm 1.2}$
	TruthfulQA	$59.8_{\pm 1.5}$	$64.2_{\pm 1.4}$	$80.6_{\pm 1.1}$
	CommonsenseQA	$76.4_{\pm 1.2}$	$76.3_{\pm 1.2}$	$84.3_{\pm 2.4}$
	MMLU	$66.4_{\pm 0.7}$	$63.1_{\pm 1.1}$	$69.5_{\pm 0.8}$
	ARC	$65.2_{\pm 1.7}$	$64.6_{\pm 0.6}$	$70.4_{\pm 1.3}$
	TriviaQA	$61.7_{\pm 2.4}$	$63.2_{\pm 1.5}$	$76.2_{\pm 1.2}$

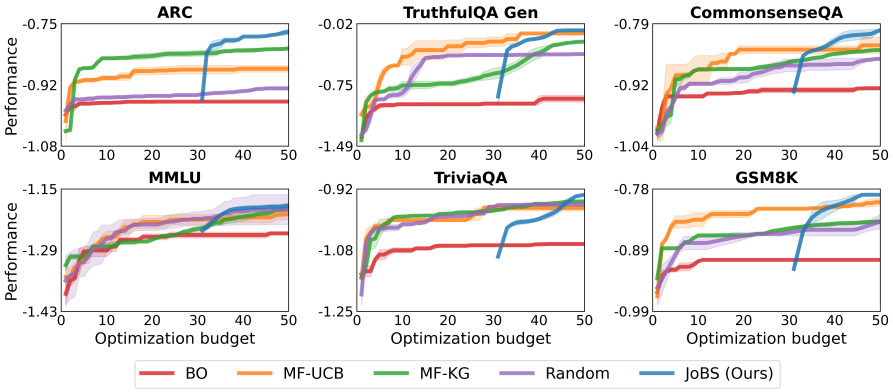


Figure 7: JoBS performance for LLM evaluation loss (instead of performance).

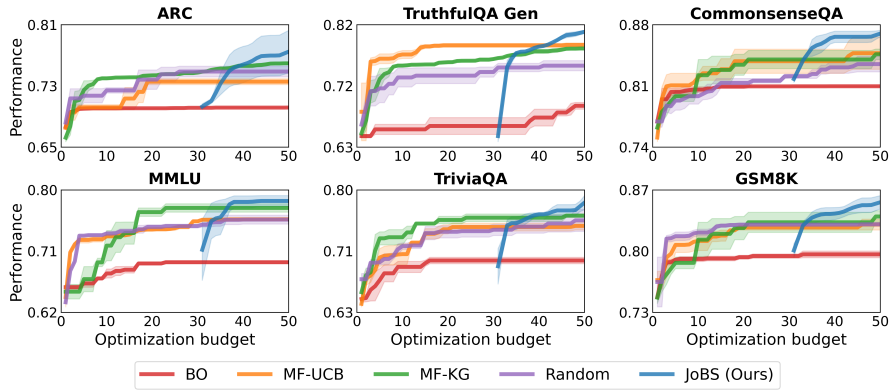


Figure 8: JoBS performance for LLM evaluation performance with Qwen-14B.

Table 11: JoBS applied to full-parameter fine-tuning of larger models.

↓ Model | Method →	LESS + AutoLoRA	DoReMi + DARTS	JoBS
Llama-3-8B-Instruct	0.73	0.76	**0.81**
Qwen3-14B	0.76	0.81	**0.83**
Qwen3-32B	0.86	0.82	**0.88**