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

Anonymous authors
Paper under double-blind review

ABSTRACT

Co-optimizing data and model configurations for LLM training presents a classic chicken-and-egg dilemma: The best training data configuration (e.g., training data mixture) for a downstream task depends on the chosen model configuration (e.g., model architecture, fine-tuning configuration), and vice versa. However, jointly optimizing both data and model configurations is conventionally deemed intractable, and existing methods focus only on data or model selection in isolation without considering their complex interdependence. We introduce JoBS, an efficient approach that uses Bayesian Optimization (BO) to *jointly* optimize LLM training data and model configurations. JoBS allocates a portion of the overall optimization budget to run LLM full-training runs to learn an explicit performance predictor that predicts how promising a particular training configuration is after a small number of training steps. The remaining budget is used to perform BO entirely with the predictor without needing any full-training runs. By studying JoBS's regret, we analyze the tradeoff between how much initial budget we should devote to learning the predictor and the remaining BO acquisition steps available. JoBS not only outperforms existing baselines across diverse LLM tasks, but also runs up to $12.4\times$ faster. Our code is available here.

1 INTRODUCTION

LLMs have become ubiquitous in our lives with great commercial and practical interest in maximizing their performance for specific tasks. Much effort has been put into optimizing the *training components* to maximize LLM performance, particularly the *training data* and *model architecture*. From the data perspective, better training data can be chosen via data selection (Chen et al., 2025c; 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) techniques. From the model perspective, various model selection methods (He et al., 2024; Raschka, 2020; White et al., 2020; Zhang et al., 2024b) 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. For example, finding the best training data configuration (e.g., training data mixture) depends on the chosen model configuration (e.g., model architecture, fine-tuning configuration), but determining the best model configuration also depends on the chosen training data. This presents a classic *chicken-and-egg dilemma*, where the optimal choice of training data depends on the optimal choice of model architecture, and vice versa. Furthermore, due to their interdependency, optimizing data and model *independently* would often lead to sub-optimal LLM performance (Chen et al., 2024). This is demonstrated in Sec. 6 where we naively combined data and model selection methods. Therefore, to address the interdependent nature of data and model architecture and maximize LLM performance, we should *jointly* optimize these two training components.

Unfortunately, jointly optimizing training data and model configurations is conventionally 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) have tried to quantify the effects of each training component on downstream performances, while prescribing simple guidelines on the optimal choices of training components given fixed computational budgets. However, they require exhaustive search over a large number of configurations, which is infeasible in practice.

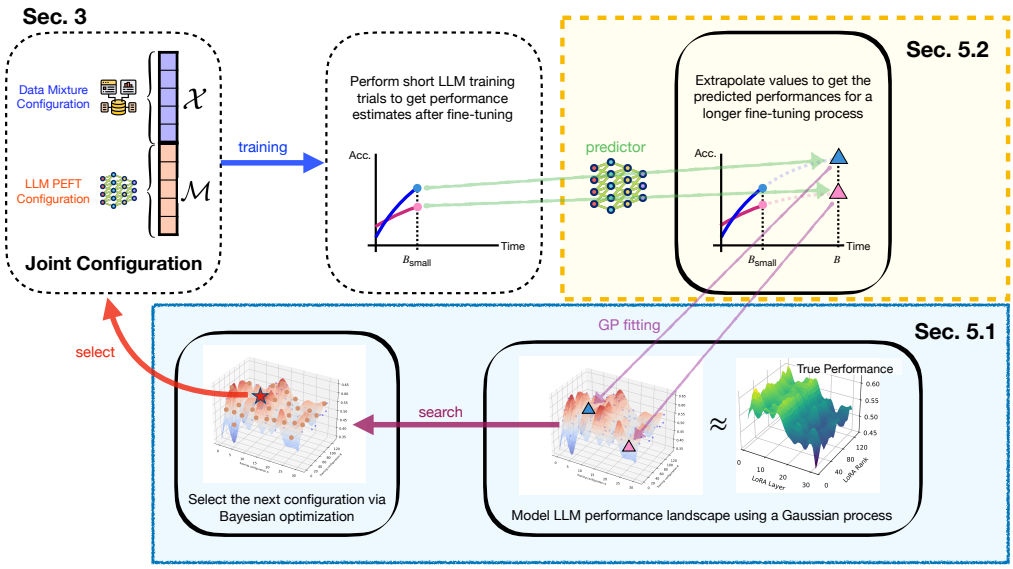


Figure 1: Overview of JoBS.

Our paper aims to study this chicken-and-egg dilemma and joint optimization problem for a scenario commonly faced by practitioners, namely *parameter-efficient fine-tuning* (Hu et al., 2021) (PEFT) of LLMs under different *training data mixtures* and model configurations. We present **Joint Bayesian Optimization with Scaling Laws** (JoBS), an approach that efficiently co-optimizes LLM training configurations by using Bayesian Optimization (BO) and a novel performance predictor to predict the full training performance of an LLM from a small amount of training time. We summarize JoBS in Fig. 1 and state our main contributions below:

- We formulate our chicken-and-egg dilemma as a black-box optimization problem (Sec. 3) and propose JoBS (Sec. 5), an algorithm that efficiently optimize LLM training configurations by interleaving *Bayesian Optimization* (BO) (Sec. 5.1) with a novel performance predictor (modeled with a flexible neural network) that amortizes the cost of full-training runs by predicting how promising a training configuration is based on a small number of training steps.
- We show that learning an accurate (i.e., low error) performance predictor requires us to run LLM full-training over a good coverage of initial random training configurations. We demonstrate an inherent tradeoff between the accuracy of the predictor and the number of BO acquisition steps — under a fixed BO budget, collecting more full-training samples improves the predictor and quality of each BO observation but reduces the remaining optimization budget available for BO.
- We theoretically analyze how the performance predictor's error propagates into JoBS's cumulative regret. From this, we establish practical guidelines towards, under a fixed BO budget, balancing the number of random full-training samples used to learn our predictor and the number of BO acquisition steps to minimize JoBS's overall cumulative regret.
- We show that JoBS (Sec. 6) attains a 6-7% flat-performance improvement over existing data, model optimization and multi-fidelity BO baselines on a variety of LLM tasks. 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

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 a family of formulas that indicate that LLM performance can be extrapolated from a small number of training steps. These scaling law formulas cannot be used as the performance predictor in our problem setting because these formulas are defined w.r.t. a fixed training configuration (e.g., using the same training data pool). Our setting requires us to predict LLM performance for a wide range of LLM training configurations, none of which is known beforehand, during the BO process.

Multi-fidelity BO. BO has been widely adopted for optimizing black-box functions where evaluations are costly (Srinivas et al., 2010) and multi-fidelity BO (Wu et al., 2019; Yen et al., 2025) models the relationship between low-fidelity and high-fidelity observations by interleaving them during the BO process. JoBS differs from multi-fidelity BO in a fundamental way: JoBS first uses a portion of the overall BO budget to learn a high-quality performance predictor using a neural network and then perform BO (using the remaining budget) *entirely* based on the fast-to-evaluate neural network predictions without revisiting any high-fidelity evaluations. While (Yen et al., 2025) explored using neural networks to predict LLM performance in BO, they only considered optimizing data component and does not consider the cost of learning the neural network as part of the overall BO budget.

Joint data and model optimization. Conventional data (Xia et al., 2024; Chen et al., 2025c; Koh & Liang, 2020; Wang et al., 2024b) and model (Liu et al., 2019; He et al., 2024; Zhang et al., 2024b) optimization works have only optimized either data or model components independently. As far as we know, JoBS is the first algorithm to *jointly* optimize data and model components of an LLM.

3 Problem Setup

We consider two kinds 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 a fixed number of training steps (e.g., B epochs, B training steps) produce fine-tuned LLM weights $\theta_{\mathcal{X},\mathcal{M},B} \triangleq P_B(\mathcal{X}, \mathcal{M})$, which can be evaluated over any predefined performance metric $\mathcal{L}$ (e.g., question-answering accuracy, evaluation loss). While we assume B is kept fixed, we also discuss the possibility of varying it w.r.t. the overall optimization budget C (defined later) in App. C. We want to find training component configurations $\mathcal{X}, \mathcal{M}$ that maximize the LLM performance metric:

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

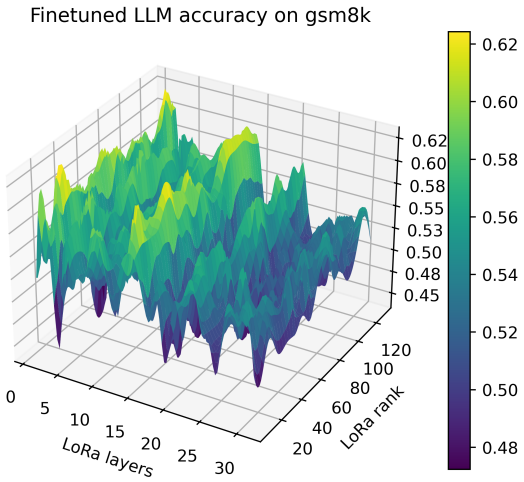
Data $\mathcal{X}$. Assume we have N training datasets $\mathcal{D} \triangleq D_1 \bigcup D_2 \bigcup \cdots \bigcup D_N$ from N different domains (e.g., Wikipedia, TruthfulQA (Lin et al., 2022) for language tasks). The training data component consists of a subset of data $\mathcal{X} \subseteq \mathcal{D}$. 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). However, this is done by assuming that a fixed model architecture is used. In our work, we overload the notation and parameterize our selected data mixture with a training data mixing ratio represented by a probability simplex of dimension N ($\mathcal{X} \in \Delta^{N-1} \subset \mathbb{R}^N$).

Model $\mathcal{M}$. Under the LLM PEFT regime, the optimization problem takes as inputs: (1) the *module* of the LLM to which PEFT is applied (e.g., Q, V projection (Vaswani et al., 2017)), (2) the *layer(s)* where PEFT 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 a M-dimensional vector $\mathcal{M} \in \mathbb{R}^M$ with $M \in \mathbb{Z}^+$. Our formulation is flexible enough to incorporate any model configuration that needs to be optimized.

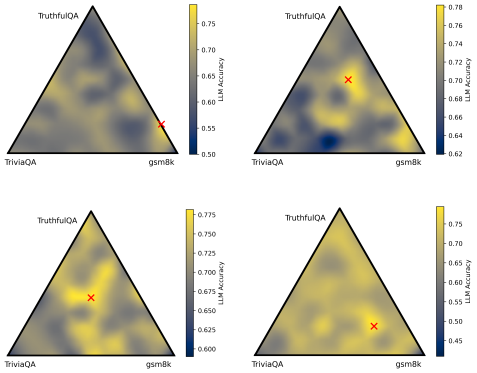
Additionally, we also consider a total optimization budget C, which serves as a practical constraint for practitioners and has the same unit as B. For instance, if our optimization budget is $C = 50000$ training steps and each full-training run consumes $B = 1000$ training steps, then we can only perform $C/B = 50$ BO iterations. One of our key contributions is to amortize the cost of each training run to $B_{\text{small}} \leq B$ with a performance predictor, allowing us to run many more BO iterations. Our work will analyze and balance the cost of training the predictor with the amortization cost savings.

4 Preliminary findings

The performance landscape of Problem 1 that describes the relationship between selected training components $\mathcal{X}, \mathcal{M}$ and the fine-tuned LLM performance $\mathcal{L}$ has no closed-form expression. To motivate the design choices in `JoBS` we present several illustrative empirical findings that give us a clearer understanding of the LLM performance landscape and justifying why our chicken-and-egg dilemma deserves attention in the first place.



(a) LLM performance after fine-tuning with LoRA applied on varying number of layers and with varying LoRA rank.



(b) LLM performance after fine-tuning with varying data mixtures and LoRA configurations. Each chart corresponds to a different LoRA configuration, while the red cross denotes optimal mixture for that LoRA configuration.

Figure 2: How data and model configurations jointly affect LLM performance.

4.1 Performance landscape

In Fig. 2a, we fine-tuned a Llama-3-8b-Instruct (Touvron et al., 2023) model for one epoch and plot its performance on the GSM8K (Cobbe et al., 2021) task (averaged over 5 trials). The empirical landscape shows that certain LoRA layer, rank, and data configurations yield *drastically* better performance (almost 20%) than arbitrarily choosing a configuration. This result shows that the optimal training configuration is difficult to find via heuristics (Radford et al., 2019; Gao et al., 2020), and it is worthwhile to jointly optimize them. Beyond just LoRA rank and layers considered in prior works (Zhang et al., 2024b), our paper also considered other model configurations, such as *which LLM modules* to apply LoRA to and more (Sec. 6).

4.2 Scaling laws and predicting performance

To fill with Scaling law examples and prediction examples.

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 earlier), whose maximum can be obtained in a sample-efficient manner by Bayesian optimization (Sec. 5.1). (ii) Inspired by prior works that suggest that LLM performance during training can be extrapolated (Wu & Tang, 2024; Chen et al., 2025b), we learn a novel performance predictor that estimates the fine-tuned LLM performance from a small number of training steps (Sec. 5.2). Unlike existing rigid symbolic scaling law formulas which are fixed to a small group of LLM training configurations, our predictor is a flexible neural network that is capable of predicting LLM performance w.r.t. *any LLM training configuration*, effectively amortizing the cost of expensive trials in BO.

A key question is how much optimization budget we should allocate to learn our predictor. For instance, if we have an optimization budget $C = 100000$ and each full-training run consumes a budget of $B = 1000$. Running BO without a predictor affords us 100 BO iterations. In contrast, we can consider using 60% of the optimization budget (60000) to perform 60 full-training runs over random training configurations, yielding 60 high-fidelity LLM performance observations. If we use these observations to train our performance predictor to predict the trained performance from $B_{\text{small}} = 100$ training steps, then we can run four times as many BO iterations: $(C - 60000)/B_{\text{small}} = 400$.

A clear tradeoff exists — using too many initial full-training runs to train the predictor leaves fewer BO acquisition steps available to actually search for the optimal training configuration. One of our

key contributions is to study how the tradeoff between number of initial training runs and remaining BO iterations influences JoBS's performance by studying its regret. We show theoretically (in Sec. 5.4) and empirically (in Sec. 6) that even if our LLM performance predictions are noisy, the BO framework handles them gracefully as *observation noise*, eventually allowing JoBS to converge to the optimal training component configuration.

5.1 BO AS THE BACKBONE OF JoBS

We consider the 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 $x^* \triangleq \operatorname{argmax}_x \mathcal{L}(x)$ we want to recover. In line with existing works, we model $\mathcal{L}$ as a surrogate *Gaussian process* (GP) (Williams & Rasmussen, 2006). In each iteration $t = 1, 2, \ldots, T$, we can use some LLM training configuration x_t to obtain a potentially *noisy* realization of the LLM performance (after fine-tuning 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., Gaussian or bounded noise) to form the sample (x_t, y_t).

Consistent with the work of Chowdhury & Gopalan (2017), our GP is fully 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 chosen to characterize the correlation of the observations between any 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 component configurations and LLM performance.

Using BO for our joint optimization problem. To determine the best configuration x^*, we trial different LLM training configurations in each round to determine their performance and continually update the GP in (2) to have a better estimate of $\mathcal{L}$. In round t, 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 $x_{t+1} = \operatorname{argmax}_x \mu_t(x) + \beta_{t+1}\sigma_t(x)$, where β_{t+1} is an exploration parameter. We can assess the convergence of a BO algorithm by analyzing its average regret after T BO iterations, given by $\mathcal{R}_T \triangleq \sum_{t=1}^{T}(T^{-1}\mathcal{L}(x^*) - \mathcal{L}(x_t))$ (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 USING PERFORMANCE PREDICTOR TO IMPROVE COMPUTATION TIME

While BO searches through different LLM training configurations avoids performing exhaustive search over all possible x (Srinivas et al., 2010), naively applying BO still requires lengthy fine-tuning in each iteration. JoBS introduces a performance predictor to estimate the full fine-tuning LLM performance from a shorter training trial (see Fig. 1), improving the overall efficiency of BO.

For our predictor to work, we need to predict LLM performance for different LLM training configurations (that we do not know in advance) at each BO iteration. Hence, we cannot use existing scaling laws (Kaplan et al., 2020; Wu & Tang, 2024; Chen et al., 2025b), which are defined for a *fixed* LLM training configuration. To address this issue, 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 time $B_{\text{small}} < B$ as inputs and predicts the final fine-tuned LLM performance.

JoBS learns this predictor in two steps. *First*, before the start of BO, it collects a random Sobol sequence (Nguyen et al., 2018) of N initial LLM training configurations in $\mathcal{X}$, $\mathcal{M}$ and observes LLM performance after small training steps $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B_{\text{small}}})$ and large training steps $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B})$. These observations are also used to fit our GP to approximate our performance landscape (Sec. 5.1), and therefore are not wasted after the predictor has been learnt. *Second*, using the initial collection of

full-training observations, it fits a predictor neural network $\mathcal{F} : \mathcal{X}, \mathcal{M}, \mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B_{\text{small}}}) \mapsto \mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B})$ that extrapolates how well an LLM performs from a small amount of training time B_{small}. At each BO iteration, JoBS fine-tunes the LLM for time B_{small} to observe $\mathcal{L}(\theta_{\mathcal{X},\mathcal{M},B_{\text{small}}})$, before using $\mathcal{F}$ to estimate the full fine-tuning performance $\hat{\mathcal{L}}(\theta_{\mathcal{X},\mathcal{M},B})$.

5.3 Prediction error and Performance tradeoffs

Prediction error. Fig. 3 shows how the prediction error of $\mathcal{F}$ varies w.r.t. different number N of initial full-training runs. We found that our predictor $\mathcal{F}$ makes more accurate predictions (higher R^2) if we collected more full-training observations at the beginning. Empirically, the predictor's prediction error distribution can be approximately fitted with a normal distribution, whose variance decreases with larger N.

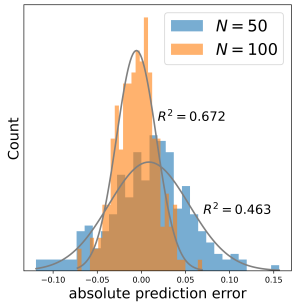


Figure 3: $\mathcal{F}$ prediction error varies w.r.t. number of training samples N

Tradeoffs. Using more full-training observations N decreases the number of BO iterations JoBS can perform later (and vice versa): Collecting N full-training trials, each consuming B training budget, incurs BN optimization budget initially. After learning a performance predictor, we only need to train an LLM for B_{small} training steps each time. Hence, the remaining optimization budget $C - NB$ can be used for $\lfloor (C - NB)/B_{\text{small}} \rfloor$ BO iterations (i.e., number of acquisition steps). We will formally analyze how this tradeoff affects JoBS's convergence in the next section.

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. 3 indicates that it is R-sub-Gaussian (Chowdhury & Gopalan, 2017) and the error distribution variance increases with smaller N. We cover use the following assumption to describe the characteristic of the prediction error.

Assumption. Let ϵ be the prediction error of the performance predictor, learnt from N full-training runs over randomly sampled training configurations, whose empirical distribution is shown in Fig. 3. Assume ϵ is R-sub-Gaussian with $R = \frac{k}{\sqrt{N}}$ for some constant k, indicating that the prediction error variance decreases with a larger choice of N.

Using this assumption, the following Theorem captures JoBS's average regret given an optimization budget C and number of initial full-training runs N used to train our performance predictor.

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 $1 - \delta$:*

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

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. D and considers how the choice of N affects the prediction error and the number of BO iterations available for optimization.

Our Theorem showcases a compute-performance tradeoff: collecting more (N) initial full-training runs allows JoBS to fit a more accurate predictor and enjoy smaller observation errors at each BO iteration. This decreases the $k/\sqrt{N}$ term in the average regret (Eq. 3). However, a larger N also reduces the number of BO iterations available, decreasing the expression on the left of Eq. 3. Although we can theoretically find an optimal N that minimizes $\mathcal{R}_T$, it is difficult to do so practically because

of the presence of other constants such as γ_T. In our ablation studies (Sec. 6.3), we empirically illustrate this tradeoff and provide some practical guidelines in choosing N.

6 EXPERIMENTS

We use JoBS to jointly optimize training configurations for LLM in a variety of language tasks and LLM model types. Our experiments are divided into three parts. First, we compared JoBS to independent data and model optimization methods. Next, we compared JoBS's convergence with various BO-centric and Multi-fidelity BO baselines. Lastly, we run several 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 an LLM. Our goal is to maximize its evaluation task performance after $B = 1000$ training steps. We consider an overall optimization budget of $C = 50000$ training steps. The performance evaluation is done using `lm-evaluation-harness` (Gao et al., 2024). We used $N = 40$ initial full-training runs to train our performance predictor to make predictions from $B_{\text{small}} = 100$ training steps. This allows us to run $T = \frac{C-NB}{B_{\text{small}}} = 100$ BO iterations (of cheaper trials) in JoBS. We perform ablations in Sec. 6.3 to investigate and justify our choice of N and B_{small}. More information on the experiment setup, including batch size, learning rate, and BO details can be found in App. E. Our main results are shown w.r.t. the `Llama-3-8b-Instruct` model, but we also extended our experiments to different model families (`Qwen`, `Mistral`), larger models (14B, 32B) and full-parameter training in App.XXX, where the results remain consistent with our key findings.

Data configuration. There are 10 data domains in our training data mixture: **Wikitext** (Merity et al., 2016), **GSM8K** (Cobbe et al., 2021), **PubmedQA** (Jin et al., 2019), **HeadQA** (Vilares & Gómez-Rodríguez, 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). One of these domains is designated as the evaluation task. 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 $\mathcal{X}$ as the mixing ratio (a probabilty simplex) across these data domains. This gives us 10 data configuration dimensions in total.

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. E.

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 generally well against every distribution of evaluation task. Influence Function (**IF**) (Koh & Liang, 2020) selects data points with the higher influence scores. **Diversity** (Wang et al., 2024b) finds the subset of data points with the largest 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, it implies we do not apply LoRA weight to that LLM layer). **AutoLoRA** (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 towards selecting different model configurations by aggregating different training-free metrics to measure how promising a given configuration is.

Multi-fidelity BO. We compared JoBS with Multi-fidelity BO baselines that consider two discrete fidelities at $B_{\text{small}} = 100$ and $B = 1000$. MF-KG uses Knowledge Gradients (KG) (Wu et al., 2019) and MF-EI uses cost-aware Expected Improvement (Yen et al., 2025; Jones et al., 1998). More details are found in App. E.

Table 1: Combination matrix of different model and data optimization methods on LLM performance compared to JoBS on the GSM8K (Cobbe et al., 2021) task (higher is better). Subscript represents standard deviations across 5 trials. The results for other tasks are shown in App. H.

↓ **Model | Data** →	Default	LESS	DoReMi	IF	Diversity	JoBS
Default	$68.1_{\pm 2.1}$	$70.4_{\pm 1.1}$	$71.6_{\pm 3.1}$	$67.9_{\pm 0.9}$	$73.8_{\pm 1.8}$	-
DARTS	$72.4_{\pm 0.8}$	$71.0_{\pm 0.6}$	$74.1_{\pm 1.3}$	$68.7_{\pm 0.4}$	$66.1_{\pm 0.7}$	-
AutoLoRA	$72.3_{\pm 1.1}$	$74.6_{\pm 0.3}$	$70.3_{\pm 0.7}$	$67.9_{\pm 0.4}$	$73.4_{\pm 0.5}$	-
RoBoT	$71.1_{\pm 0.6}$	$72.0_{\pm 1.5}$	$73.4_{\pm 1.8}$	$72.4_{\pm 1.5}$	$69.6_{\pm 1.7}$	-
JoBS	-	-	-	-	-	$\mathbf{80.4}_{\pm 1.9}$

6.2 Main results and key takeaways

Comparison with data and model optimization approaches. We mixed and matched conventional data optimization and model architecture optimization methods and applied them on Llama-3-8b-Instruct's training components independently. Such a combination does not consider the joint interaction between data and model configurations. Our result shows that simply pairing data and model optimization methods independently yields worse LLM performance than JoBS, which considers the complex interaction between data and model configurations. By jointly optimizing both components, JoBS attain a flat 6-7% "interaction improvement" over other baselines. Due to space constraints, we only display the results for **GSM8K**. The results for other tasks are shown in App. H and are generally consistent with our findings.

We also selected a few optimization pairings from Table 1 and applied them in an alternating fashion to our training configurations for 5 iterations. While it partially accounts for the interaction between data and model, this method is computationally expensive (because we need to repeatedly apply data and model optimization) and Table 5 in App. H shows that it still performs worse than JoBS.

Comparison with Multi-fidelity BO (MF-BO) approaches. We found that under same amount of total optimization budget, JoBS consistently outperforms Multi-fidelity BO baselines across all six LLM tasks in terms of optimizing the evaluation task loss (Fig. 4). In addition, JoBS also consistently outperforms baselines when optimizing the evaluation task performance (Fig.XXX).

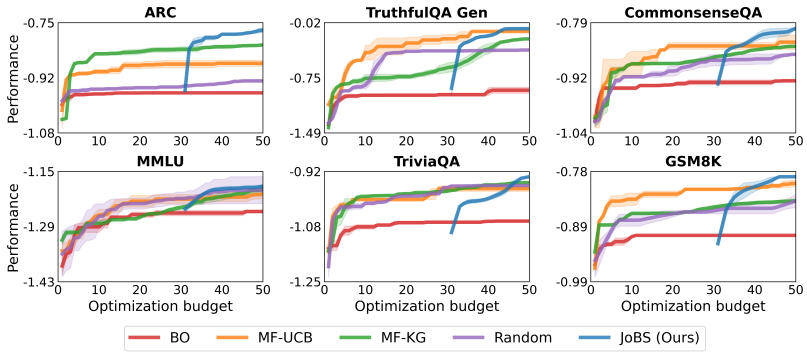


Figure 4: JoBS consistently outperforms other BO-centric approaches under the same total-optimization budget.

6.3 Ablation and additional analysis

In the previous sections, we showed that JoBS outperforms baselines in a variety of evaluation tasks. However, several questions remain regarding the performance-compute tradeoff 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}? To address these questions, we ran ablations on the Llama-3-8b-Instruct model and the CommonsenseQA evaluation task performance.

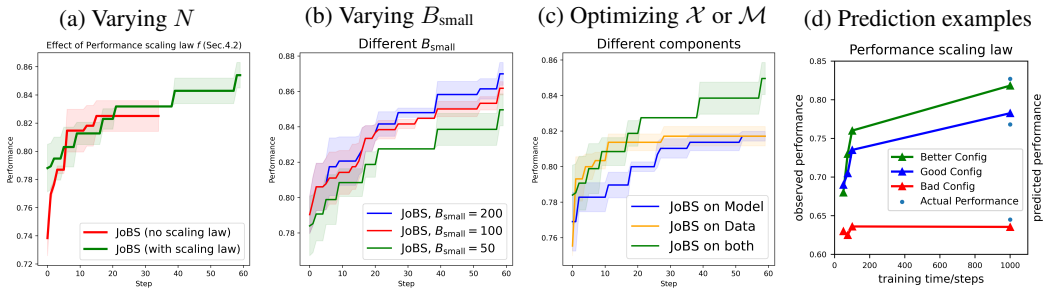


Figure 5: Various ablation studies for the effect of the performance predictor on JoBS.

Choice of number of initial training samples N. Fig. XXX illustrates the tradeoff 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, and vice versa. Empirically, under the optimization budget described in Sec. 6, we find that $N = 30$ strikes a good balance between predictor quality and optimization capacity.

Effect of varying B_{small}. Fig. 5b shows that, ceteris paribus, increasing B_{small} improves the convergence of JoBS. This is because B_{small} corresponds to the small number of training steps used for our performance predictor to predict the full-training performance at B. If B_{small} increases, the predictor makes more accurate predictions and hence JoBS's convergence improves. In practice, increasing B_{small} reduces the number of BO acquisition steps available; in our experiments, we find that using $B_{\text{small}} = 100$ serves as a good balance between the predictor's accuracy and the number of BO acquisition steps available.

Computational cost and other qualitative discussion. Lastly, we found that JoBS has a smaller runtime than existing baselines, achieving a speedup of $1.7\times$ to $12.4\times$ over different baselines. We provide a computational cost analysis in App. F, where we find that our performance scaling law predictor is the main reason why JoBS has a smaller runtime, and existing data selection methods are generally expensive. We also present a few interesting analyses of the optimal LLM training configurations found by JoBS vs. other baselines in App. G.

7 Conclusion and Future Work

We illustrated the chicken-and-egg dilemma in LLMs, showing that the interdependence between data and model components makes it challenging for conventional methods to optimize model performance efficiently. We introduced JoBS, an efficient algorithm that leverages BO and a novel performance scaling law predictor to jointly optimize data and model configurations by efficiently learning the LLM performance landscape under the fine-tuning regime. Despite noisy estimates from the predictor, JoBS still has theoretical guarantees and shows promising empirical results in our experiments. Across different evaluation tasks and LLM models, JoBS attains substantial "interaction improvement" over prior baselines, showing that jointly optimizing data and model configurations performs better than independent optimization.

We believe JoBS can also be adapted for LLM pretraining by considering LLM pretraining configurations, such as LLM architecture and pretraining mixtures. However, there are unique challenges specific to pretraining - for instance, can we still accurately predict the LLM performance at the pretraining stage? At this point, it is not immediately clear if we can still learn the LLM performance predictor well. However, considering the importance of model and data configurations in LLM pretraining, we can safely speculate that the same chicken-and-egg dilemma is equally relevant for LLM pretraining.

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.

Samuel Daulton, Xingchen Wan, David Eriksson, Maximilian Balandat, Michael A. Osborne, and Eytan Bakshy. Bayesian optimization over discrete and mixed spaces via probabilistic reparameterization. *arXiv:2210.10199*, 2022.

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`.

Guillaume Garrigos and Robert M Gower. Handbook of convergence theorems for (stochastic) gradient methods. *arXiv:2301.11235*, 2023.

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.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

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.

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.

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.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 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.

David Vilares and Carlos Gómez-Rodríguez. HEAD-QA: A healthcare dataset for complex reasoning. In *Proc. ACL*, 2019.

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 THEORETICAL INSIGHTS INTO OPTIMAL LLM TRAINING DATA AND MODEL CONFIGURATIONS

We provide theoretical insights on why an optimal model size and training data size exists in our problem setting. To do so, we analyze the convergence of mini-batch Stochastic Gradient Descent (SGD) (Garrigos & Gower, 2023) over a convex loss function w.r.t. varying model size and training data size (viewed as batch size in this setting). We first present the well-known results on the convergence of the loss function under the mini-batch SGD setting in the proposition below.

Proposition A.1. *Let $b > 1$ be the training data size (out of a larger full training dataset of size $n > b$) and m be the number of model parameters. Let T be the training steps budget allocated for our model with parameters θ_m. Assume θ_m^* are the optimal model parameters for the full training dataset and let $f(\theta, \mathcal{X})$ be a convex loss function with respect to model parameters θ and input examples $\mathcal{X}$. Define the gradient noise as $\sigma_f^* \triangleq Var[\nabla f(\theta_m^*, x)]$ for a randomly sampled datapoint x from the full training data set. Let L_m be the lipschitz constant of the loss function f of a model with m parameters. Lastly, assume $||\theta_m - \theta_m^*||^2 \leq K$ for some constant K and any m.*

If we perform minibatch stochastic gradient descent on f with a randomly sampled data batch of size b (from the full training dataset) on model parameters θ_m with constant step size $(4L_m)^{-1}$ for T iterations, then

$$\mathbb{E}[f(\theta_m^T) - f(\theta_m^*)] \leq \frac{4L_m K}{\sqrt{T}} + \frac{2(n-b)\sigma_f^*}{4L_m b(n-1)\sqrt{T}} \tag{4}$$

where θ_m^T is the model parameters after T SGD steps.

The above proposition tells us that if the training data is sampled randomly from the full training dataset, the deviation between the optimal loss (over the full training dataset) and the loss w.r.t. learnt parameters θ_m^T is upper-bounded by the right term in Eq. 4. We can see that the loss w.r.t. learnt parameters θ_m^T will eventually converge to the optimal loss $f(\theta_m^*)$ w.r.t. increasing training steps.

Interestingly, we observe that the choice of m, b, and T is constrained by the given training time budget. We make two assumptions about the relationship between m, b and T.

1. **Assumption 1.** The lipschitz constant of loss function is governed by $L_m = c_1/m$ for some positive constant c_1. This implies the larger the model size, the smaller the lipschitz constant of the loss function (and faster the model learns).
2. **Assumption 2.** $T = bm/c_2$ for some positive constant c_2. This implies that model and training data size both increases the training budget required to train the model. Given a fixed T, we cannot choose a large model size m and training data size b.

In the next Theorem, we show that given a training budget T, there exists a particular model size m and training size b that will minimize the upper bound in Eq. 4. The Theorem is established by substituting **Assumption 1** and **Assumption 2** into Eq. 4 and noticing that a) for positive T, the feasible region exists and b) the objective function is bounded (since denominators are non-zero). We would like to emphasize that even though a particular choice of m, b could lead to smaller upper bound, it does not necessarily guarantee that the deviation in Eq. 4 is actually smaller. However, our theorem provides theoretical insights to possibly explain why training with certain choices of b, m could yield better model performance. We have also provided empirical evidence to show that certain training component configurations can produce better-performing LLMs in Sec. 4.

Theorem A.2. *Under the same setting as Proposition A.1 and given **Assumptions 1** and **2**, for a fixed training budget of $T > 0$, the upper bound from Proposition A.1 is minimized by solving the following constrained optimization problem:*

$$\begin{aligned} &\min_{m,b} \frac{4c_1 K}{m} + \frac{2(n-b)m\sigma_f^*}{4c_1 b(n-1)} \\ &\textit{s.t.} \quad mb = c_2 T \ . \end{aligned} \tag{5}$$

Therefore, an optimal m, b would minimize the constrained optimization problem and minimize the error bounds in Eq. 4. The proof is explained in the paragraph above.

B MORE BACKGROUND ON BAYESIAN OPTIMIZATION

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 DISCUSSION RELATED TO OPTIMIZATION COST

D 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 $1 - \delta$:*

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

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

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 prediction (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 what the LLM would have performed 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. 3). As such, this formulation is in-lined with the BO algorithmic framework introduced in Sec. 5.1, where we make 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 D.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 $t \leq T$:*

$$|\mu_t(x) - f(x)| \leq \left(\mathcal{B} + R\sqrt{2(\gamma_t + 1 + \ln(1/\delta)}\right)\sigma_t(x) \tag{6}$$

where γ_t is the maximum information gain after t observations and $\mu_t(x), \sigma_t^2(x)$ are mean and variance of posteror 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. D.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 $1-\delta$,

$$
\begin{aligned}
|f(x^*) - f(x_t)| &\overset{(1)}{\leq} \beta_t \sigma_{t-1}(x_t) + \mu_{t-1}(x_t) - f(x_t) \\
&\overset{(2)}{\leq} \beta_t \sigma_{t-1}(x_t) + \mu_{t-1}(x_t) + (\beta_t \sigma_{t-1}(x_t) - \mu_{t-1}(x_t)) \\
&\leq 2\beta_t \sigma_{t-1}(x_t)
\end{aligned} \tag{7}
$$

where $\overset{(1)}{\leq}$ uses the fact that via Lemma D.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 D.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. 7, we can bound the cumulative regret by

$$
\sum_{t=1}^{T} r_t = \sum_{t=1}^{T} (f(x^*) - f(x_t)) \leq 2 \sum_{t=1}^{T} \beta_t \sigma_{t-1}(x_t) \; . \tag{8}
$$

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

$$
\begin{aligned}
\mathcal{R}_T &= \sum_{t=1}^{T} r_t && (9) \\
&\leq 2 \sum_{t=1}^{T} \beta_t \sigma_{t-1}(x_t) && (10) \\
&\leq 2\mathcal{O}(\sqrt{T\gamma_T})(\mathcal{B} + R\sqrt{2(\gamma_T + 1 + \ln(1/\delta))}) && (11) \\
&= \mathcal{O}\left(\mathcal{B}\sqrt{T\gamma_T} + R\sqrt{T}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right) && (12) \\
&\overset{(1)}{=} \mathcal{O}\left(\mathcal{B}\sqrt{\left\lfloor \frac{C-NB}{B_{\text{small}}} \right\rfloor \gamma_T} + R\sqrt{\left\lfloor \frac{C-NB}{B_{\text{small}}} \right\rfloor}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right) && (13) \\
&\overset{(2)}{=} \mathcal{O}\left(\mathcal{B}\sqrt{\left\lfloor \frac{C-NB}{B_{\text{small}}} \right\rfloor \gamma_T} + \frac{k}{\sqrt{N}}\sqrt{\left\lfloor \frac{C-NB}{B_{\text{small}}} \right\rfloor}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right), && (14)
\end{aligned}
$$

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:

$$
\begin{aligned}
\frac{\mathcal{R}_T}{T} &\overset{(1)}{=} \mathcal{O}\left(\mathcal{B}\sqrt{\frac{\gamma_T B_{\text{small}}}{C - NB - B_{\text{small}}}} + \frac{k}{\sqrt{N}}\sqrt{\frac{B_{\text{small}}}{C - NB - B_{\text{small}}}}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right) && (15) \\
&= \mathcal{O}\left(\left(\sqrt{\frac{B_{\text{small}}}{C - NB - B_{\text{small}}}}\right)\left(\mathcal{B}\sqrt{\gamma_T} + \frac{k}{\sqrt{N}}\sqrt{\gamma_T^2 + \gamma_T \ln(1/\delta)}\right)\right) && (16)
\end{aligned}
$$

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 $a \neq 0$. □

E MORE EXPERIMENTAL DETAILS

Here, we provide details of how we ran our experiments for JoBS. Our data configuration consists of 10 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 in which we only optimize data configuration, we will use the default model configuration above. In all our main results (Tables 1, 6, and 5), we used 8-shot prompting with CoT. We used 100 BO iterations with a shortened training time of $B_{\text{small}} = 50$ seconds at each iteration. 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, their partial and full fine-tuning performance, before training a densely-connected, 64-width, 3 layers neural network $\mathcal{F}$ to predict the full performance. This random sequence is also added to our initial GP model to warm-start BO in JoBS. We used a deep 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 the deep kernel. We normalize and rescale all our LLM training configuration parameters to be between 0 and 1 when fitting our GP. For binary or integer decisions, we use continuous parameterization (Daulton et al., 2022) to project them into the same continuous space as other variables.

Throughout our experiments, we used a single L40 GPU to fine-tune our LLM.

F COMPUTATIONAL COST OF JoBS VS. OTHER BASELINES

Qualitative comparison. We can actually concisely summarize the computational cost of JoBS. We used 30 observations from fully fine-tuning an LLM with random LLM training configurations for 1000 seconds (to learn our performance scaling law predictor and forming the first 30 observations of our trials). Then, we run JoBS for 70 iterations, each taking 50 seconds of fine-tuning time. This means JoBS uses 33500 seconds (9+ hours) of fine-tuning time. This is faster or comparable to many state-of-the-art data selection algorithms; refer to the next section for a more precise quantitative comparison. For instance, computing the Influence Function (IF) scores (Koh & Liang, 2020) of all data points took a few days. In addition, JoBS is an *anytime* algorithm, meaning if computational cost is an issue, we can terminate it at any step to obtain a sub-optimal (but still reasonable good) solution.

Quantitative comparison of wall-clock hours. 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. We ensured they run for 33500 seconds. Hence, they have equal computation time (JoBS achieves better performance, as seen in Table 5). For data selection methods (LESS, DoReMi, IF, Diversity), we recorded their wall-clock runtime in Table 2. In general, we found data selection methods to be very computationally expensive, taking as much or more time than JoBS. One of the key reasons that JoBS runs faster is due to our scaling law predictor (Sec. 5.2), which drastically reduces the fine-tuning time needed at each BO iteration.

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	**9.3**

G QUALITATIVE COMPARISON OF OPTIMAL LLM TRAINING CONFIGURATIONS FOUND BY JoBS VS. OTHER BASELINES

Here, we display 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. Notice the training domain does not contain **GSM8K** because all our evaluation is done in a much harder out-of-domain setting.

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

	CQA	HQA	PQA	SciQ	TrivQA	TruthQA	Wiki	MMLU	ARC
JoBS	0.12	0	0	0.10	0.19	0	0.28	0.31	0
DoReMi	0.08	0.11	0.18	0.05	0.08	0.14	0.04	0.16	0.13

On particular interest is that JoBS optimizes the data mixture by placing more weights into some data domains based on the 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 SciQ, TriviaQA, Wikipedia, and MMLU contains some math information, and thus chooses them in the optimized data mixture.

On the other hand, DoReMi is a distributionally robust data mixing approach, and results in a more uniform data mixing ratio. This means the data mixture is not tailored specifically to the evaluation task GSM8K, and hence does not perform as well.

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

Next, we examine the optimal model configurations found JoBS. We noticed that JoBS prefers a higher LoRA rank and layer (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 $= 1$).

Table 5: Comparison of baselines with JoBS. Results are shown w.r.t. different evaluation tasks and LLMs (Higher is better), averaged over 5 trials. We choose to present a few better performing baselines (combining data and model selection methods in an alternating manner). Default fine-tuning means we just fine-tuned the model with standard data and model configurations (App. E).

Model	Task	Default fine-tuning	LESS + AutoLoRA	DoReMi + DARTS	JoBS
Llama-3-8B-Instruct	GSM8K	$68.1_{\pm 2.1}$	$74.8_{\pm 0.9}$	$73.2_{\pm 1.4}$	$80.4_{\pm 1.9}$
	TruthfulQA	$55.4_{\pm 1.6}$	$66.2_{\pm 0.7}$	$68.9_{\pm 1.2}$	$75.8_{\pm 1.3}$
	CommonsenseQA	$76.3_{\pm 1.0}$	$80.5_{\pm 1.4}$	$79.9_{\pm 1.0}$	$84.3_{\pm 2.4}$
	HeadQA	$47.0_{\pm 0.9}$	$46.5_{\pm 1.5}$	$54.0_{\pm 1.8}$	$55.8_{\pm 1.5}$
	MMLU	$61.2_{\pm 1.3}$	$67.6_{\pm 2.9}$	$64.1_{\pm 1.1}$	$69.5_{\pm 0.8}$
	ARC	$54.7_{\pm 1.3}$	$66.3_{\pm 1.6}$	$62.5_{\pm 0.7}$	$70.4_{\pm 1.3}$
	TriviaQA	$61.3_{\pm 2.4}$	$70.4_{\pm 3.6}$	$71.3_{\pm 1.4}$	$76.2_{\pm 1.2}$
Qwen2.5-7B-Instruct	GSM8K	$70.2_{\pm 0.6}$	$73.7_{\pm 0.9}$	$71.1_{\pm 1.4}$	$81.3_{\pm 1.4}$
	TruthfulQA	$56.4_{\pm 0.7}$	$67.2_{\pm 1.3}$	$68.3_{\pm 0.9}$	$74.8_{\pm 1.7}$
	CommonsenseQA	$77.6_{\pm 0.4}$	$82.1_{\pm 0.3}$	$80.2_{\pm 0.6}$	$81.7_{\pm 0.6}$
	HeadQA	$52.5_{\pm 0.3}$	$51.3_{\pm 1.4}$	$50.8_{\pm 0.9}$	$58.6_{\pm 0.9}$
	MMLU	$72.5_{\pm 1.4}$	$73.9_{\pm 1.6}$	$72.8_{\pm 0.3}$	$78.4_{\pm 1.2}$
	ARC	$64.6_{\pm 0.8}$	$69.1_{\pm 3.1}$	$71.5_{\pm 3.2}$	$75.0_{\pm 0.3}$
	TriviaQA	$55.3_{\pm 0.8}$	$65.1_{\pm 2.0}$	$64.8_{\pm 1.2}$	$68.5_{\pm 1.3}$
Mistral-7b-Instruct-v0.3	GSM8K	$52.2_{\pm 0.8}$	$58.7_{\pm 0.6}$	$63.0_{\pm 1.1}$	$66.4_{\pm 0.5}$
	TruthfulQA	$56.4_{\pm 0.7}$	$59.8_{\pm 1.7}$	$62.2_{\pm 0.6}$	$73.5_{\pm 0.6}$
	CommonsenseQA	$77.6_{\pm 0.4}$	$78.3_{\pm 1.1}$	$77.9_{\pm 1.2}$	$83.5_{\pm 0.8}$
	HeadQA	$57.8_{\pm 0.3}$	$56.3_{\pm 0.9}$	$57.9_{\pm 1.2}$	$59.2_{\pm 1.1}$
	MMLU	$63.6_{\pm 0.5}$	$71.8_{\pm 0.9}$	$71.6_{\pm 1.3}$	$73.8_{\pm 0.9}$
	ARC	$66.3_{\pm 0.8}$	$70.2_{\pm 2.0}$	$72.9_{\pm 1.0}$	$74.7_{\pm 0.6}$
	TriviaQA	$58.2_{\pm 0.3}$	$57.8_{\pm 1.8}$	$60.5_{\pm 0.5}$	$66.3_{\pm 1.1}$

H MORE EXPERIMENTAL RESULTS AND DISCUSSION

In Tables 6, 7, 8, 9, 10, and 11, we repeated the experimental set-up as those in Table 1 and mixed and matched different model and data selection methods over another 5 evaluation tasks (CommonsenseQA, HeadQA, MMLU, ARC, and TriviaQA). The results show that JoBS outperforms all combinations of data and model selection works. This suggests that jointly adjusting 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.

H.1 OTHER NAIVE BASELINES

In Table 12, we jointly optimized LLM training configurations using several other naive approaches in our experiments. We tried 3 naive approaches: (a) **Random** randomly picking 100 different LLM training configurations, fine-tune them for 50 seconds 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.

Table 6: Evaluation task: TruthfulQA (Lin et al., 2022).

↓ **Model | Data** →	Default	LESS	DoReMi	IF	Diversity	BO	JoBS
Default	$55.4_{\pm 1.6}$	$56.4_{\pm 0.8}$	$58.2_{\pm 2.4}$	$57.3_{\pm 1.1}$	$59.8_{\pm 1.0}$	$70.2_{\pm 0.8}$	-
DARTS	$56.7_{\pm 1.1}$	$57.0_{\pm 0.4}$	$62.8_{\pm 1.1}$	$59.1_{\pm 0.3}$	$59.6_{\pm 1.0}$	$72.4_{\pm 0.8}$	-
AutoLoRA	$56.0_{\pm 0.8}$	$62.6_{\pm 1.0}$	$58.8_{\pm 0.9}$	$59.6_{\pm 1.0}$	$60.8_{\pm 0.4}$	$68.4_{\pm 0.3}$	-
RoBoT	$59.1_{\pm 0.4}$	$60.2_{\pm 0.5}$	$53.4_{\pm 1.1}$	$52.4_{\pm 0.8}$	$60.9_{\pm 0.4}$	$69.6_{\pm 1.1}$	-
BO	$61.0_{\pm 1.0}$	$62.0_{\pm 0.3}$	$64.0_{\pm 0.7}$	$64.8_{\pm 0.8}$	$60.3_{\pm 1.2}$	$71.7_{\pm 1.8}$	-
JoBS	-	-	-	-	-	-	$\mathbf{75.8}_{\pm 1.9}$

Table 7: 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 8: HeadQA (Vilares & Gómez-Rodríguez, 2019).

↓ **Model** | **Data** →	Default	LESS	DoReMi	IF	Diversity	BO	JoBS
Default	$47.0_{\pm 0.9}$	$46.4_{\pm 1.0}$	$46.3_{\pm 0.8}$	$46.1_{\pm 0.7}$	$45.8_{\pm 1.2}$	$49.2_{\pm 0.6}$	-
DARTS	$43.6_{\pm 0.2}$	$46.7_{\pm 1.3}$	$53.0_{\pm 2.4}$	$40.7_{\pm 1.5}$	$47.3_{\pm 0.7}$	$48.9_{\pm 1.2}$	-
AutoLoRA	$49.1_{\pm 1.4}$	$49.4_{\pm 0.4}$	$50.3_{\pm 0.9}$	$47.7_{\pm 1.1}$	$48.4_{\pm 1.0}$	$51.3_{\pm 0.3}$	-
RoBoT	$49.5_{\pm 1.2}$	$48.0_{\pm 1.0}$	$48.7_{\pm 1.7}$	$49.2_{\pm 1.2}$	$50.6_{\pm 0.8}$	$50.8_{\pm 0.6}$	-
BO	$49.6_{\pm 0.8}$	$51.3_{\pm 1.0}$	$52.0_{\pm 0.6}$	$52.6_{\pm 0.3}$	$50.3_{\pm 0.7}$	$48.2_{\pm 0.6}$	-
JoBS	-	-	-	-	-	-	$\mathbf{55.8}_{\pm 1.5}$

Table 9: 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 10: 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 11: 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	-	-	-	-	-	-	$\mathbf{76.2} \pm 1.9$

Table 12: 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**). **Random** means that we randomly selected LLM training configurations.

Model	Task	**Random**	**Random Data**	**Random Model**	JoBS
Llama-3-8B-Instruct	GSM8K	$66.5_{\pm 2.4}$	$67.3_{\pm 1.6}$	$71.5_{\pm 0.9}$	$80.4_{\pm 1.9}$
	TruthfulQA	$59.1_{\pm 1.9}$	$59.8_{\pm 1.5}$	$64.2_{\pm 1.4}$	$75.8_{\pm 1.3}$
	CommonsenseQA	$78.8_{\pm 3.2}$	$76.4_{\pm 1.2}$	$76.3_{\pm 1.2}$	$84.3_{\pm 2.4}$
	HeadQA	$51.5_{\pm 2.1}$	$51.3_{\pm 2.1}$	$53.2_{\pm 1.2}$	$55.8_{\pm 1.5}$
	MMLU	$67.6_{\pm 2.9}$	$66.4_{\pm 0.7}$	$63.1_{\pm 1.1}$	$69.5_{\pm 0.8}$
	ARC	$60.5_{\pm 3.2}$	$65.2_{\pm 1.7}$	$64.6_{\pm 0.6}$	$70.4_{\pm 1.3}$
	TriviaQA	$58.2_{\pm 3.6}$	$61.7_{\pm 2.4}$	$63.2_{\pm 1.5}$	$76.2_{\pm 1.2}$