ESTIMATING OPTIMAL HYPERPARAMETER SCALING LAWS EFFICIENTLY THROUGH POWER-LAW ENTROPY SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimal hyperparameter scaling laws describe how the best hyperparameters for large language model (LLM) training change with model and data scale, enabling practitioners to predict optimal configurations at production scales without expensive large-scale tuning. However, estimating these scaling laws conventionally requires exhaustive grid searches over thousands of training runs, consuming enormous computational resources. We introduce Power-Law Entropy Search (**PLES**), a computational cost-aware acquisition function built on multi-fidelity Bayesian optimization that efficiently estimates optimal hyperparameter scaling laws through adaptive experimentation. A key innovation in **PLES** is that it searches for candidates that reduce the overall uncertainty of a scaling law estimate, instead of optimizing a single objective function. At each iteration, **PLES** selects the candidate configuration that maximally reduces the uncertainty of the scaling law estimates per unit computational cost, naturally favoring informative small-scale experiments. We evaluate **PLES** on synthetic benchmarks, surrogate models fitted to real LLM training data, and actual LLM pre-training runs. Across all settings, **PLES** converges to accurate optimal hyperparameter scaling laws using 10% of the computational budget required by conventional grid search and other baselines.

1 INTRODUCTION

Optimal hyperparameter scaling laws (Li et al., 2025) describe how optimal hyperparameters (e.g., learning rate, batch size, weight decay, etc) change w.r.t. increasing model and data sizes. These scaling laws are critical tools to guide the development of large language models (LLMs) pretraining because they allow practitioners to find optimal hyperparameters for production-level model and data scales without actually performing expensive hyperparameter optimization (HPO) (Feurer & Hutter, 2019; Akiba et al., 2019) at the largest scale.

It is costly to derive optimal hyperparameter scaling laws. Unlike conventional scaling laws Kaplan et al. (2020); Lin et al. (2026); Hoffmann et al. (2022) which merely describe how LLM loss changes with scale, optimal hyperparameter scaling laws describe how the *optimal* hyperparameter changes with scale. A typical approach to find optimal hyperparameter scaling laws is to perform hyperparameter sweep at every scale. This is unfortunately computationally expensive. For instance, Li et al. (2025) "*trained over 3,700 LLMs from scratch across 100 trillion tokens, consuming nearly one million NVIDIA H800 GPU hours*" to establish an optimal scaling law for learning rate and batch size. In addition, scaling laws become outdated with newer model architectures and settings, requiring us to often repeat such expensive grid-search approaches.

We show that it is not necessary to perform hyperparameter sweep (or optimization) at every scale to learn the optimal hyperparameter scaling law. Instead, we can share information across experiments at different model and data scales to recover the scaling law efficiently. We introduce a new adaptive experimentation method that leverages principles from cost-aware *multi-fidelity Bayesian optimization* (BO) (Kandasamy et al., 2017; Wu et al., 2020) to balance the computational cost of running experiments at different scales against the information gained from each experiment. Our method employs a Gaussian Process (GP) (Williams & Rasmussen, 2006) as a surrogate model to share

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

information across experiments at different scales and capture how the LLM loss varies with respect to hyperparameters and scale. Concretely, our contributions are as follows:

1. We introduce a new acquisition function, called Power-Law Entropy Search (**PLES**), that proposes experimentation candidates to efficiently estimate optimal hyperparameter scaling laws. Unlike conventional optimization problems that optimize a single objective function, **PLES** exploits information from experiments at multiple scales to accurately estimate the entire functional form of a scaling law.
2. We show that acquisition optimization in **PLES** can be solved by drawing Thompson samples from the GP, and the sampled solution converges to the true solution asymptotically.
3. We empirically demonstrate that, under the same computational budget, **PLES** produces significantly more accurate optimal hyperparameter scaling laws than conventional baselines in a variety of problems, including hyperparameter optimization for LLM pretraining. In particular, the cumulative computational cost of running small-scale experiments with candidates suggested by **PLES** is significantly lower than if we had performed hyperparameter optimization directly at the largest scale.

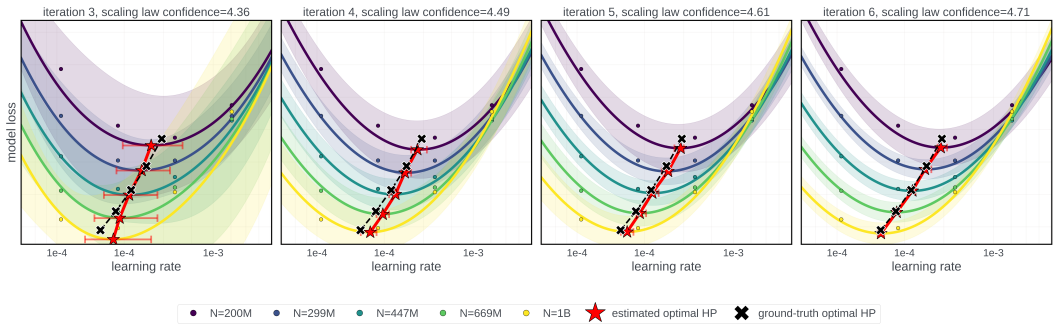


Figure 1: Through small-scale experiments that share information across different model and data scales, **PLES** produces an optimal hyperparameter scaling law estimate (red line) that converges to the ground-truth optimal hyperparameter scaling law (black dotted line). In addition, **PLES** produces uncertainty intervals (red horizontal bars) that indicate our confidence of the scaling law estimate.

2 Related Work

LLM Scaling Laws. Conventional LLM Scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022) describe how LLM validation loss varies with changing model and data scales. Optimal hyperparameter scaling laws (Yang et al., 2021; McCandlish et al., 2018; Li et al., 2025) offer a nuanced distinction from conventional scaling laws because they describe how the *optimal* hyperparameters (conditioned on a given model and data scale) change with increasing model and data scales. Our aim is to learn optimal hyperparameter scaling laws through a series of experimentation at small model and data scales to infer the optimal hyperparameters at larger, production-level model and data scales.

Experimental design. Bayesian Experimental Design (BED) Rainforth et al. (2023); Huang et al. (2025) offers a principled statistical framework for selecting experiments that maximize the information gained about a latent quantity of interest, typically quantified via information gain (Chaloner & Verdinelli, 1995). Our work leverages some of the same principles found in BED (more details covered in Section 5) and uses information gain as a latent quality to drive *where* we perform small-scale experiments. As far as we know, (Li et al., 2026) is the only piece of work that leverages similar principles in designing experiments to learn scaling laws. However, they did not focus on the specific problem of *optimal hyperparameters* scaling across model and data scales, which, as we have explained earlier, contains a subtle difference from conventional scaling laws. In addition, their approach centers around a Gaussian mixture model formulation in modeling the scaling law parameters while our paper adopts a multi-fidelity BO approach. Last but not least, Li et al. (2026)'s empirical results are produced from hypothetical experiments from an existing scaling law benchmark, whereas we verified our method on actual LLM training runs (see Section 6).

Gaussian Process (GP). A GP (Williams & Rasmussen, 2006) defines a distribution over functions, $f \sim \mathcal{GP}(\mu, \kappa)$, meaning that any random draw from the process yields a continuous function over the input domain. A GP is fully specified by its *prior* mean function $\mu(x)$ and a covariance kernel function $\kappa(x, x')$ for all $x, x' \in \mathbb{R}^d$, where the kernel characterizes the correlation strength between any two input variables. Conditioned on a set of past observations, $\mathcal{D}_t = \{(x_\tau, y_\tau)\}_{\tau=1}^t$, a GP's posterior belief of any new point x' is a Gaussian distribution, whose mean and variance are:

$$\mu_t(x') \triangleq \kappa_t^\top(x')(K_t + \zeta I)^{-1}\boldsymbol{y}_t \tag{1}$$

$$\sigma_t^2(x') \triangleq \kappa(x', x') - \kappa_t^\top(x')(K_t + \zeta I)^{-1}\kappa_t(x') \tag{2}$$

where $\kappa_t(x') \triangleq [\kappa(x', x_1), \ldots, \kappa(x', x_t)]^\top$, $K_t \triangleq [\kappa(x_\tau, x_{\tau'})]_{\tau,\tau' \in \{1,\ldots,t\}}$ is a $t \times t$ covariance matrix, and $\zeta > 0$ is a regularization hyperparameter (Chowdhury & Gopalan, 2017) and also describes the observation noise under the GP model.

3 Problem Setup

Let $N \in \mathbb{R}$ denote model size, $D \in \mathbb{R}$ data size, and $\theta \in \mathbb{R}$ a hyperparameter of interest (e.g., learning rate, batch size, weight decay). To ease our exposition, we assume that there is only one hyperparameter, although in Section 5.3 we will extend our formulation to more than one hyperparameters (and explain the subtleties involved). Training a model with configuration (N, D, θ) yields an evaluation loss $\ell(N, D, \theta)$ and consumes some amount of training budget (e.g., GPU hours, FLOPs), often approximated with $6ND$ (Kaplan et al., 2020). At each model and data scale, the hyperparameter that minimizes the evaluation loss is referred to as the *optimal hyperparameter* at that scale (every scale, there is a different optimal hyperparameter). An example of how the optimal learning rate (red star) scales with increasing model and data sizes for Llama-3 pretraining is shown in Figure 2.

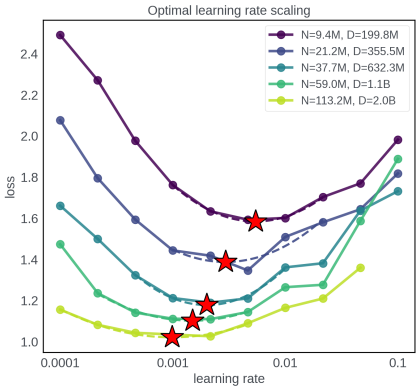


Figure 2: An example of how the optimal learning rate of Llama-3 (Touvron et al., 2023) changes with increasing model parameters and training data tokens. The change in optimal learning rate (red stars) is described by an optimal hyperparameter scaling law, which we aim to estimate efficiently.

Optimal hyperparameter scaling law. The optimal hyperparameter scaling law describes how the optimal hyperparameter changes with scale. More formally, an optimal hyperparameter scaling law is a predictive function Π that maps a set of "scale" parameters to optimal hyperparameters. In our paper, we specifically focus on LLM model scale N (i.e., model parameters) and data scale D (i.e., tokens), and Π maps these scales to the optimal hyperparameter θ^* that minimizes the model evaluation loss at that scale. This relationship is described with the following expression:

$$\theta^* = \Pi(N, D)$$

where $\Pi : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}$ is a parametric function with learnable coefficients. We are interested in finding the optimal hyperparameter θ^*_{max} at the largest target model scale N_{max} and data scale D_{max} which minimizes the evaluation loss, i.e., $\theta^*_{\text{max}} = \arg\min_\theta \ell(N_{\text{max}}, D_{\text{max}}, \theta)$. From hereon, we refer to the largest scale as the *held-out scale*. Assuming the scaling law is known (more specifically, if its coefficients are estimated accurately), the optimal hyperparameter at the held-out scale can be inferred directly from the scaling law:

$$\theta^*_{\text{max}} = \Pi(N_{\text{max}}, D_{\text{max}}).$$

Power-law as functional form of scaling law. In our paper, as with (Li et al., 2025), we assume that the optimal hyperparameter scaling law Π is a power-law function of both model scale N and data scale D (as we show in Figures 6 and 7, this assumption is realistic):

$$\theta^* = cN^\alpha D^\beta. \tag{3}$$

As such, finding the power-law coefficients is equivalent to finding the optimal hyperparameter scaling law. Our aim is to estimate the power-law coefficients c, α, β efficiently through small-scale experimentation.

4 CONVENTIONAL APPROACHES

How would practitioners typically find the optimal hyperparameters at the held-out scale (e.g., the largest model to serve users)? The simplest way is to perform Bayesian optimization (Frazier, 2018; Srinivas et al., 2010; Chen et al., 2024; 2026) at the largest scale. However, this is often infeasible in the context of LLMs because we do not have enough resources to perform multiple rounds of pretraining at the largest held-out scale. Another alternative is shown in Fig. 3, where we perform multiple independent hyperparameter optimization (HPO) at a few small scales and perform extrapolation (using scaling law) to find the optimal hyperparameter at the held-out scale. The shortfall of this method is that information is not shared between different scales, leading to resource wastage.

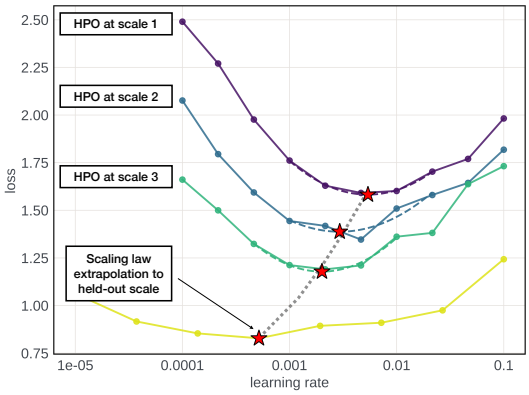


Figure 3: A simple strategy of performing independent HPO at a few small scales and using scaling laws to extrapolate to the held-out scale. This strategy works but does not share information obtained across different scales with each other.

5 METHOD

Our approach circumvents the shortfalls of existing methods by explicitly sharing information across experiments run on different scales under the Bayesian Optimization (BO) framework. At each iteration, (a) we maintain a GP surrogate whose inputs are previously experimented configurations (N, D, θ) (model size, data size, and hyperparameter) and output is observed model loss. We use Thompson samples from the GP to maintain an estimate of the optimal hyperparameter scaling law. (b) Using a novel cost-aware acquisition function called **PLES**, we find the candidate model scale, data scale, and hyperparameter configuration to experiment next that *maximally* reduces the uncertainty of our scaling law estimate. Asymptotically, our estimate converges to the true optimal hyperparameter scaling law.

5.1 ESTIMATING POWER-LAW COEFFICIENTS FROM A GP

Given a fitted Gaussian Process (GP) that takes in input configurations (N, D, θ), we first introduce a procedure to recover the scaling law's power law form and quantify our uncertainty about its coefficients at any point. To achieve this, we rewrite the power-law in Equation (3) into a linear form:

$$\log \theta^* = \log c + \alpha \log N + \beta \log D. \tag{4}$$

We view this as a Bayesian linear regression problem that learns the power-law coefficients $\mathbf{w} = (\log c, \alpha, \beta)^\top$ and their uncertainty. If we are certain about the coefficients, it implies we are certain about the power-law.

Finding θ^* for one particular model N_i and data scale D_i. For a single, fixed model and data scale (N_i, D_i), we perform Thompson sampling (Chapelle & Li, 2011) over the current GP ($\mathcal{GP}$) fitted from observed data to empirically estimate the expectation of the optimal hyperparameter θ_i^* at scale i:

$$\theta_i^* = \mathbb{E}_{f\sim\mathcal{GP}}[\arg\min_\theta f(N_i, D_i, \theta)] \approx \frac{1}{K}\sum_{k=1}^{K} \arg\min_\theta f_k(N_i, D_i, \theta) \tag{5}$$

where f_k is one of K Thompson samples from the GP. We also observe that sufficient Thompson samples asymptotically recover the true solution. We repeat this procedure across m model $[N_1, \cdots, N_m]$ and data $[D_1, \cdots, D_m]$ scales to produce an estimate of optimal hyperparameters across different scales $[\theta_1^*, \cdots, \theta_m^*]$, along with their empirical standard deviations $[\sigma(\theta_1^*), \cdots, \sigma(\theta_m^*)]$.

Using Bayesian linear regression to estimate power-law coefficients $\mathbf{w}$. We aggregate the evaluations across all m scales to obtain target vector $\mathbf{y}$, the design matrix $\mathbf{X}$, and the diagonal noise

covariance matrix $\mathbf{\Lambda}$:

$$\mathbf{y} = [\log \theta_1^*, \cdots, \log \theta_m^*]^\top, \quad \mathbf{X} = \begin{bmatrix} 1 & \log N_1 & \log D_1 \\ & \vdots & \\ 1 & \log N_m & \log D_m \end{bmatrix}, \quad \mathbf{\Lambda} = \text{diag}(\sigma^2(\log \theta_1^*), \ldots, \sigma^2(\log \theta_m^*)), \tag{6}$$

where $\sigma(\log \theta_i^*) \approx \sigma(\theta_i^*)/\theta_i^*$ (Casella & Berger, 2024). The power-law is fitted based on $\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Lambda})$. Given a Gaussian prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_0)$ [1], our estimate of the power-law coefficients is $p(\mathbf{w} \mid \mathbf{y}, \mathbf{X}) = \mathcal{N}(\boldsymbol{\mu}_\mathbf{w}, \mathbf{\Sigma}_\mathbf{w})$, with mean and variance:

$$\mathbf{\Sigma}_\mathbf{w} = \left(\mathbf{\Sigma}_0^{-1} + \mathbf{X}^\top \mathbf{\Lambda}^{-1} \mathbf{X}\right)^{-1} \tag{7}$$

$$\boldsymbol{\mu}_\mathbf{w} = \mathbf{\Sigma}_\mathbf{w} \mathbf{X}^\top \mathbf{\Lambda}^{-1} \mathbf{y} \tag{8}$$

From hereon, we use $\mathbf{\Sigma}_\mathbf{w}(\mathcal{GP}(\mathcal{D}))$ and $\boldsymbol{\mu}_\mathbf{w}(\mathcal{GP}(\mathcal{D}))$ to denote the covariance and mean of power-law coefficients estimated from the GP fitted on observed data $\mathcal{D}$.

Uncertainty of power-law coefficients. The differential entropy $\mathbf{logdet}\mathbf{\Sigma}_{\boldsymbol{w}}(\mathcal{GP}(\mathcal{D}))$ indicates the amount of uncertainty surrounding the current power-law coefficients estimated from observed data $\mathcal{D}$ (Chaloner & Verdinelli, 1995). This expression represents the amount of uncertainty associated with the power-law estimate at any given iteration.

5.2 Power-Law Entropy Search (PLES)

Using the procedure in Section 5.1, we introduce a computational cost-aware acquisition function, **PLES**, that searches for candidates that have the largest uncertainty reduction per unit-cost of the estimated power-law coefficients. The overall pseudocode for **PLES** is provided in Section A. At iteration t, the acquisition function of a candidate hyperparameter and scale configuration $x = (N, D, \theta)$ given historical observed data $\mathcal{D}_t$ is:

$$\textbf{PLES}(x) = \max_x \frac{\mathbf{logdet}\mathbf{\Sigma}_\mathbf{w}(\mathcal{GP}(\mathcal{D})) - \mathbf{logdet}\mathbf{\Sigma}_\mathbf{w}(\mathcal{GP}(\mathcal{D} \cup x))}{(ND)^d}, \tag{9}$$

where d is a computational cost-cooling factor that discourages experimentation at larger model and data scales (Kandasamy et al., 2017). If $d = 1$ is chosen, it reduces the acquisition function to search for the largest information gain per unit-cost of experimentation. Choosing a larger d favors experimentation at smaller scales, and vice versa. Note that to obtain an updated $\mathcal{GP}(\mathcal{D} \cup x)$, we draw a fantasy observation for x from the current GP. The reduction of the shaded region in Figure 4 describes how an informative candidate reduces the empirical uncertainty of the optimal hyperparameter at a given scale.

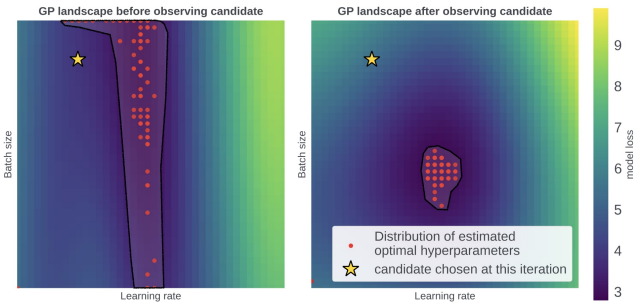


Figure 4: PLES identifies an informative candidate that greatly reduces the empirical uncertainty of the estimated optimal hyperparameters from the GP. This leads to lower variance about the estimated power-law coefficients, which is represented by the expression $\mathbf{logdet}\mathbf{\Sigma}_\mathbf{w}(\mathcal{GP}(\mathcal{D} \cup x))$ in Equation (9).

[1] We can use a uniform prior if no prior knowledge of the power-law coefficients is known. Otherwise, we can consider using more informative priors.

5.3 Extension to more than one hyperparameters

Often, there is more than one hyperparameter of interest when practitioners train LLMs. Our GP modeling process in Section 3 can be extended straightforwardly to these cases, by assuming θ has a cardinality of more than one. However, the scaling laws of different hyperparameters of LLMs (e.g., learning rate, batch size) are governed by independent power-laws with different coefficients (Li et al., 2025). To infer the optimal hyperparameters at the largest scale, we will need to maintain an estimate of different, independent power-laws (Equation (3)) during the BO process. Then, we need to modify our acquisition function in Equation (9) to search for candidates that reduce the uncertainty of multiple scaling laws *simultaneously*, by reducing a weighted-sum of differential entropy (mathematical formulation provided in Section B).

Interestingly, we observed in our experiments that explicitly maintaining multiple scaling laws may not always be necessary. For example, in the case of two hyperparameters (e.g., learning rate and batch size), we find that performing acquisition solely to reduce uncertainty in the scaling law for learning rate can, as a byproduct, also recover an accurate scaling law for batch size. We hypothesize that this occurs because **PLES**, in identifying configurations that are informative about the optimal learning rate, naturally also reveals regions with the optimal batch size. We analyze these results in greater detail in Section 6.5.

5.4 Stopping criterion

A key advantage of maintaining a probabilistic estimate of the power-law coefficients is that it provides a natural stopping criterion for experimentation. At each iteration t, the empirical uncertainty of the optimal hyperparameter at the held-out scale is: $\mathbf{x}_{\text{max}}^\top \boldsymbol{\Sigma}_{\boldsymbol{w}} \mathbf{x}_{\text{max}}$, where $\mathbf{x}_{\text{max}} = (1, \log N_{\text{max}}, \log D_{\text{max}})^\top$ (see Section 5.1). Essentially, this is the uncertainty of the optimal hyperparameters (in the power-law) at the held-out scale. When this value is small, it suggests that our power-law estimate has become accurate. This is shown in column 3 and 4 of Figure 6, where the uncertainty at the held-out scale is small and further experimentation does not improve the estimated power-law significantly.

6 Experiments

6.1 Setup

We evaluate our method on a few settings, including *real LLM training runs*. **First**, we consider a synthetic function that describes how model loss varies with respect to varying learning rate, batch size, model scale N and data scale D. The function's formula is provided in Section C. **Second**, we retrieved past LLM training runs from (Li et al., 2025; Lin et al., 2026), which trained models of up to 6.1B parameters. The training runs contain a grid search of how model loss varies w.r.t. learning rate, batch size, model and data scales. We fit a GP to the results of these training runs, use the GP as an oracle function and apply our method on a continuous set of configuration choices at every acquisition step. Section D shows that the GP has a good fit over the past LLM training run data, serving as a good surrogate ground-truth. **Third**, we evaluate our method on *real LLM training runs*, whose training setup we cover next.

Setup for real LLM training runs. For simplicity, we use one dimension of scaling and only vary the model size N. For each model size chosen, we use a data size of 20 Tokens-Per-Parameter (TPP), which follows the Chinchilla Law (Hoffmann et al., 2022). We train a Llama-8B model. To adjust the model size, we vary the number of layers and hidden layer dimensions in the intermediate layers simultaneously, while keeping the model's aspect ratio roughly the same. The range of configurations used is as follows: $N \in [9.4M, 1B], D \in [199.8M, 20B], \text{LR} \in [0.0001, 0.1]$. For every configuration, we pre-train the LLM using a subset of the Llama training data (Touvron et al., 2023) for one epoch and record the final validation loss on a held-out set from the training data. We use a batch size of 2048 and the sequence length of each batch is 2048 tokens.

Evaluation methods. We evaluate the effectiveness of our method by evaluating how good the optimal hyperparameter at the held-out scale is. There are two ways to do so. For the **synthetic** and **surrogate** experiments, we know the ground-truth optimal θ at the held-out scale. For the **Real LLM** experiment, we performed grid search and interpolation at the held-out scale to empirically estimate the optimal θ. We then measure the percentage error of the optimal θ inferred from the estimated

power-law at the held-out scale. The held-out scale is indicated by the figure subtitles in Figure 5. The second evaluation metric is to evaluate the model loss of the estimated optimal θ at the held-out scale at the end of every acquisition step.

6.2 BASELINES

We compare our method against three baselines. All baselines operate under the same total compute budget (measured as the cumulative ND spent across all evaluations) and, where applicable, share the same Sobol-initialized starting configurations to ensure a fair comparison. For every method, we repeat our experiments with 10 seeds, which influence the initial starting configurations and also the randomness involved in every method.

Grid search. This is the conventional method used to fit scaling laws in prior work (Hoffmann et al., 2022; Li et al., 2025). For each model/data scale (N, D), we evaluate the loss on a dense grid over the hyperparameters θ, take the per-scale grid minimizer as the estimated optimal hyperparameter $\theta^\star(N, D)$, and then fit the scaling law. The budget is allocated such that every scale contains approximately the same number of training runs. There are many ways to allocate compute resources for grid search, but we find this approach more effective than others. We provide a more thorough analysis of budget allocation of grid search in Section E.

Ladder BO. This baseline performs independent Bayesian optimization with the Expected Improvement (Ament et al., 2023) to estimate the optimal hyperparameter θ at each scale. To achieve this, independent GPs are fitted to the results at each scale, and computational resources are allocated such that we have an equal amount of training runs at each scale. The optimal hyperparameter estimated at each scale is then used to fit the scaling law, allowing us to estimate the optimal hyperparameter at the held-out scale.

Sobol. This baseline draws configurations (θ, N, D) from a quasi-random, space-filling Sobol sequence until the budget is exhausted, then fits a GP over the hyperparameter θ and scale (N, D). The scaling law is estimated from the random configurations using the same procedure as in Section 5.1.

6.3 MAIN RESULTS

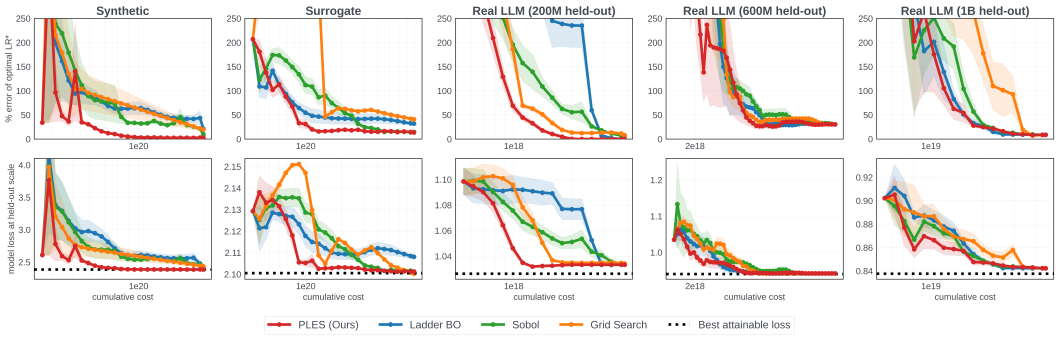


Figure 5: Results on three settings: a **Synthetic function**, a **Surrogate** GP model that is fitted on real LLM training runs in (Li et al., 2025), and **Real LLM** pretraining runs on the Llama-3 (Touvron et al., 2023) model. The first row shows how accurate the optimal hyperparameter estimated by each method is w.r.t. increasing experimentation computational cost. The second row shows the model loss attained at the estimated optimal hyperparameter at the held-out scale.

Our results in Figure 5 show that **PLES** (red) consistently achieves a smaller percentage error in the estimated optimal hyperparameter than other baselines in the held-out scale. In addition, the optimal hyperparameter estimated by **PLES** converges much more quickly than other methods, reaching the best attainable model loss at the held-out scale in *less than one-tenth* of the computational budget as compared to canonical baselines.

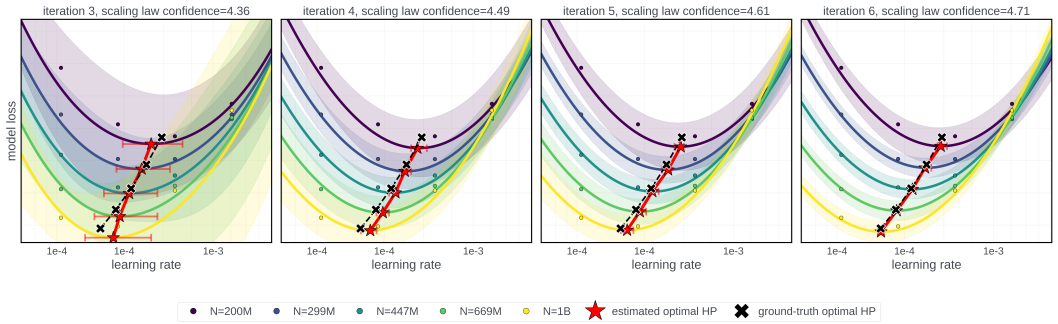


Figure 6: The fitted GP and estimated optimal learning rate scaling law at each iteration of the **surrogate** experiment. The variance of the GP reduces and the estimated scaling law converges to the ground-truth scaling law with more iteration. At iteration 6 (3rd column), the uncertainty at the held-out scale is sufficiently small, serving as a stopping criterion for the experiment. Further experimentation at the 4th column does not improve the estimated scaling law significantly.

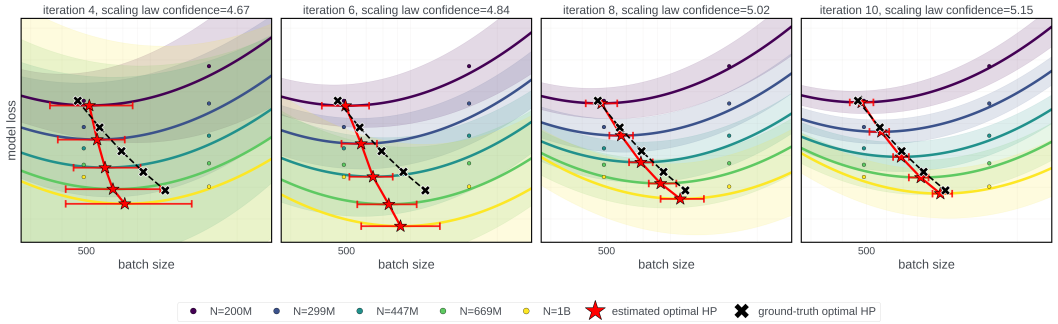


Figure 7: Although **PLES** does not explicitly model the optimal batch size scaling law for the **surrogate** experiment, it still converges to the ground-truth optimum.

6.4 Analysis of scaling law uncertainty with increasing computational cost

In this section, we analyze how the confidence of our estimated scaling law changes as we incur more experimentation budget in the **surrogate** experiment. The size of the horizontal standard deviation bar in Figure 6 represents our uncertainty of the learning rate scaling law fit $\mathbf{\Lambda}$ (see Equation (6)) at each iteration. We observe that the uncertainty decreases with more experimentation budget. In addition, the size of the horizontal standard deviation at the largest held-out scale (in this case, at N=1B) can also be used as a stopping criterion (see Section 5.4) in PLES. At iteration 6 (3rd column), the uncertainty of the estimated scaling law at the held-out scale is small, suggesting that our scaling law estimate has (rightfully so) already converged to the ground-truth scaling law. As expected, performing experiments for more iterations (from 3rd to 4th column) does not improve the scaling law estimate significantly.

6.5 Results on more than one hyperparameters (learning rate and batch size)

In Section 5.3, we mentioned that PLES can recover multiple optimal hyperparameter scaling laws even when it is designed to minimize the uncertainty about one scaling law. In the **surrogate** experiment, our GP takes in both learning rate and batch size but we designed **PLES** to specifically acquire candidates that reduce the uncertainty of the learning rate scaling law. Despite so, the result in Figure 7 shows that PLES surprisingly recovers the batch size scaling law despite only being designed to estimate the learning rate scaling law. One possible explanation for this phenomenon is that PLES uncovers regions that contain the optima of multiple hyperparameters simultaneously; this is seen in Figure 4, where an acquired candidate (yellow star) that reduces the uncertainty of one hyperparameter (e.g., learning rate) also reduces the uncertainty of another hyperparameter (e.g., batch size).

6.6 Analysis of the scale of candidates acquired by **PLES**

Figure 8 shows that most of the candidate scales proposed by acquisition function **PLES** are concentrated at small scales that are less than 2% of the computational cost of running the experiments at the expensive held-out scale. This is because **PLES** is computational cost-aware and uses a cost-cooling factor in Equation (9) to favor candidates at cheaper scales. As a result, the computational cost of running all the experiments with the candidates suggested by **PLES** is significantly smaller than the computational cost of performing hyperparameter optimization at the held-out scale directly. For instance, for the one hyperparameter setting (on the right) for pretraining LLMs, PLES requires a cumulative cost less than 0.5x of a held-out scale as compared to 5 held-out scale runs if we had simply performed HPO at the held-out scale. As such, PLES consumes less than 10% of the computational budget as compared to performing HPO at the largest scale.

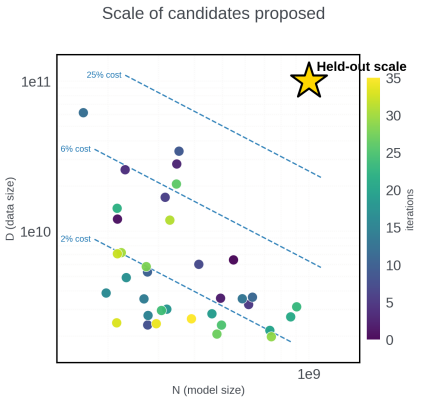


Figure 8: The candidate scales that **PLES** proposes throughout the course of learning the optimal hyperparameter scaling law. Most candidate scales are less than 2% of the computational cost of the held-out scale.

6.7 Computational complexity

PLES allows us to recover the optimal hyperparameter scaling law in an efficient amount of computational budget. Some computational complexity is necessary to evaluate the acquisition value of each candidate configuration and solve the optimization problem in Equation (9). The per-iteration complexity is $\mathcal{O}(C \cdot K \cdot Q)$, where C is the number of candidate configurations we consider for the optimization problem, K is the number of Thompson samples used to estimate θ^* in Section 5.1, and Q is the computational cost of solving the inner $\arg\min_\theta$ problem (Equation (5)) for each sample. Leveraging batched pathwise sampling and vectorized operations in BoTorch (Balandat et al., 2020), the Thompson sampling loop and inner minimizations can be fully parallelized across number of fantasies K and number of candidates considered at each acquisition step C. Hence, the computational overhead in **PLES** is negligible compared to the reduction in the amount of model training computational cost saved.

7 Conclusion and Limitations

We introduced Power-Law Entropy Search (**PLES**), a cost-aware acquisition function that efficiently estimates optimal hyperparameter scaling laws for large language model pretraining through efficient experimentation that shares information across training scales.

One limitation is that our method assumes that the optimal hyperparameter scaling law follows a known functional form (Equation (3)). Although this assumption is well-supported empirically for the hyperparameters and architectures studied in this paper (Figures 6 and 7) and in prior work (Li et al., 2025), it may not hold universally. It is possible that in new models, the scaling law of optimal hyperparameters deviates from a simple power-law. In such cases, the Bayesian linear regression step in Section 5.1 would produce biased coefficient estimates and degrade the extrapolation accuracy at the held-out scale. That said, extending PLES to accommodate more flexible functional representations while retaining tractable uncertainty quantification is a promising direction for future work.

Our experiments across synthetic benchmarks, surrogate models fitted to real LLM training data, and actual LLM pre-training runs demonstrate that **PLES** converges to accurate optimal hyperparameter scaling laws using less than one-tenth of the computational budget required by conventional grid search and other baselines. Furthermore, **PLES** provides a natural stopping criterion based on the posterior uncertainty at the target scale, allowing practitioners to terminate experimentation once sufficient confidence is achieved. We also showed that **PLES** can recover scaling laws for multiple hyperparameters simultaneously, even when the acquisition function is designed to reduce uncertainty for only one.

REFERENCES

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proc. of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Sebastian Ament, Sam Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected improvements to expected improvement for bayesian optimization. In *Proc. NeurIPS*, 2023.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization, 2020. URL https://arxiv.org/abs/1910.06403.

George Casella and Roger Berger. *Statistical inference*. Chapman and Hall/CRC, 2024.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical science*, pp. 273–304, 1995.

Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Proc. NeurIPS*, 2011.

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, 2026. URL https://arxiv.org/abs/2502.00270.

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

Matthias Feurer and Frank Hutter. Hyperparameter optimization. In *Automated machine learning: Methods, systems, challenges*, pp. 3–33. Springer, 2019.

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

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, 2022. URL https://arxiv.org/abs/2203.15556.

Daolang Huang, Yujia Guo, Luigi Acerbi, and Samuel Kaski. Amortized bayesian experimental design for decision-making, 2025. URL https://arxiv.org/abs/2411.02064.

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity bayesian optimisation with continuous approximations. In *Proc. ICML*, 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 preprint arXiv:2001.08361*, 2020.

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Zhenyu Ding, Haoying Wang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part i, step law – optimal hyperparameter scaling law in large language model pretraining, 2025. URL https://arxiv.org/abs/2503.04715.

Sijie Li, Shanda Li, Haowei Lin, Weiwei Sun, Ameet Talwalkar, and Yiming Yang. Spend less, fit better: Budget-efficient scaling law fitting via active experiment selection. *arXiv preprint arXiv:2604.22753*, 2026.

Haowei Lin, Haotian Ye, Wenzheng Feng, Quzhe Huang, Yujun Li, Hubert Lim, Zhengrui Li, Xiangyu Wang, Jianzhu Ma, Yitao Liang, and James Zou. Can language models discover scaling laws?, 2026. URL https://arxiv.org/abs/2507.21184.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Tom Rainforth, Adam Foster, Desi R Ivanova, and Freddie Bickford Smith. Modern bayesian experimental design, 2023. URL https://arxiv.org/abs/2302.14545.

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.

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. URL https://arxiv.org/abs/2302.13971.

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

Jian Wu, Saul Toscano-Palmerin, Peter I. Frazier, and Andrew Gordon Wilson. Practical multi-fidelity Bayesian optimization for hyperparameter tuning. In *Proc. UAI*, 2020.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In *Proc. NeurIPS*, 2021.

A PSEUDOCODE FOR **PLES**

Input: Current $\mathcal{GP}$ model fitted on observations prior to iteration t; a set of candidate configurations each with model scale N, data scale D, cost-cooling factor d
Output: Acquisition value for each candidate.

Compute current power-law uncertainty $= \mathbf{logdet}\boldsymbol{\Sigma}_{\boldsymbol{w}}(\mathcal{GP}(\mathcal{D}))$
foreach *candidate* **do**
1. Draw a fantasy observation and update GP model: $\mathcal{GP}(\mathcal{D} \cup x)$.
2. Compute updated power-law uncertainty $\mathbf{logdet}\boldsymbol{\Sigma}_{\boldsymbol{w}}(\mathcal{GP}(\mathcal{D} \cup x))$ (Section 5.1).
3. Acquisition value of candidate: $\frac{\mathbf{logdet}\boldsymbol{\Sigma}_{\mathbf{w}}(\mathcal{GP}(\mathcal{D})) - \mathbf{logdet}\boldsymbol{\Sigma}_{\mathbf{w}}(\mathcal{GP}(\mathcal{D} \cup x))}{(ND)^d}$.

end

Algorithm 1: Pseudo code of Power-Law Entropy Search (**PLES**) at iteration t that computes the acquisition value of each candidate. The candidate with the highest acquisition value is selected.

B PLES FOR MULTIPLE HYPERPARAMETERS

Without loss of generality, we illustrate the extension of PLES to two hyperparameters $\theta^{(1)}$ and $\theta^{(2)}$ (e.g., learning rate and batch size), each with its own independent power-law:

$$\log \theta^{(1)*} = \log c_1 + \alpha_1 \log N + \beta_1 \log D, \tag{10}$$

$$\log \theta^{(2)*} = \log c_2 + \alpha_2 \log N + \beta_2 \log D. \tag{11}$$

Following the procedure in Section 5.1, we estimate each set of power-law coefficients independently:

$$\mathbf{w}_1 = (\log c_1,\ \alpha_1,\ \beta_1)^\top, \quad \mathbf{w}_2 = (\log c_2,\ \alpha_2,\ \beta_2)^\top, \tag{12}$$

each with its own Bayesian linear regression posterior:

$$p(\mathbf{w}_j \mid \mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{w}_j},\ \boldsymbol{\Sigma}_{\mathbf{w}_j}), \quad j \in \{1, 2\}. \tag{13}$$

These two regression problems share the same design matrix $\mathbf{X}$ (since the (N_i, D_i) scales are common) but have different target vectors $\mathbf{y}_1, \mathbf{y}_2$ and noise matrices $\boldsymbol{\Lambda}_1, \boldsymbol{\Lambda}_2$, estimated in the same manner as Section 5.1. The modified **PLES** acquisition function searches for candidates to reduce the uncertainty of both scaling laws *simultaneously* by maximizing the sum of entropy reductions:

$$\mathbf{PLES}(x) = \max_x \frac{\Delta_1(x) \ + \ \Delta_2(x)}{(ND)^d}, \tag{14}$$

where

$$\Delta_j(x) = \mathbf{logdet}\,\boldsymbol{\Sigma}_{\mathbf{w}_j}(\mathcal{GP}(\mathcal{D})) \ - \ \mathbf{logdet}\,\boldsymbol{\Sigma}_{\mathbf{w}_j}(\mathcal{GP}(\mathcal{D} \cup x)) \tag{15}$$

is the reduction in differential entropy of the j-th scaling law's coefficients after observing candidate x. Without any loss of generality, this formulation here can be extended to as many hyperparameters as desired.

C DETAILS OF SYNTHETIC FUNCTION

To evaluate our method in a controlled setting with a known ground-truth optimum, we construct a four-dimensional synthetic benchmark that mimics the loss landscape of neural language-model training. The function takes two optimization variables — the learning rate η and the batch size B (measured in tokens) — together with two fidelity variables, the model size N (parameters) and the data size D (tokens). All four inputs are parameterized in $\log_{10}$ space, and the computational cost of an evaluation is defined as the compute proxy $\text{cost}(N, D) = N \cdot D$, so that low-fidelity (small-N, small-D) queries are cheap and high-fidelity queries are expensive.

The benchmark is built so that the loss-minimizing hyperparameters follow prescribed scaling laws. The optimal learning rate obeys a joint power law in model and data size, $\eta(N, D) = A, N^b D^c$

with $(A, b, c) = (0.1896, , -0.734, , 0.342)$, so that the optimal learning rate decreases with model size and increases mildly with data; the optimal batch size follows a power law in the data size alone, $B(D) = G, D^d$ with $(G, d) = (14.9624, , 0.5)$. Around this optimum the loss is modeled as a smooth quadratic "bowl" in log-space,

$$\mathcal{L}(\eta, B, N, D) = L_{\text{floor}}(N, D) + \alpha_\eta \big(\log \eta - \log \eta\big)^2 + \alpha_B \big(\log B - \log B\big)^2,$$

with curvature coefficients $\alpha_\eta = 0.40$ and $\alpha_B = 0.15$ (deviations measured in natural log; the optional η–B cross-term is set to zero, giving an axis-aligned bowl). The additive term L_{floor} is a Chinchilla-style irreducible loss that improves with scale,

$$L_{\text{floor}}(N, D) = L_\infty + \frac{A_N}{N^{\beta_N}} + \frac{A_D}{D^{\beta_D}}, \qquad L_\infty = 1.69, ; (A_N, \beta_N) = (406.4, 0.34), ; (A_D, \beta_D) = (410.7, 0.28).$$

D Using GP as Oracle Ground-Truth

In our **surrogate** experiments, we fitted a surrogate GP over the data from training runs in (Li et al., 2025) and use it as the ground-truth to validate our algorithm, **PLES**. Figure 9 shows that the surrogate GP is well-fitted over the data from the training runs. Specifically, the optimal learning rate and batch size at each model and data scale from the training runs (yellow star) are close to the predicted optimal hyperparameters from the GP (red star). In addition, the RMSE are all smaller than 0.01, suggesting that the GP surrogate serves as a good fit to represent the data from real training runs.

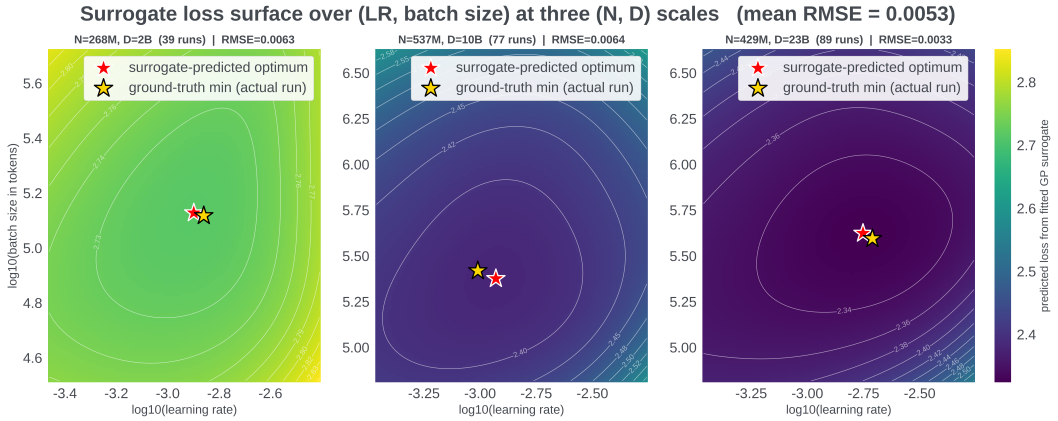


Figure 9: The surrogate loss surface shows that the GP serves as a fit over the data from training runs in (Li et al., 2025). This implies it can be used as an accurate ground-truth function to validate the **PLES** algorithm in our **Surrogate** experiments.

E Budget Allocation for Grid Search

When fitting an optimal hyperparameter scaling law using grid search, a natural instinct is to spend the entire tuning budget on small models, since they are cheap and one can afford many runs to pin down the optimum at small scale precisely. This turns out to be a poor use of compute: predicting the best hyperparameter at a much larger scale is similar to extrapolation in a regression problem, and spreading the budget across more scales is more effective. Points crowded together at small scales pin down the line's slope poorly, and even a tiny slope error would propagate to larger uncertainties at larger scales. Spreading experiments across a wide range of scales gives a better estimate, even though each individual measurement is coarser (Figure 10). Therefore, in our grid search baseline in Section 6.2, we chose to allocate experimental runs evenly across all scales to make it a stronger baseline.

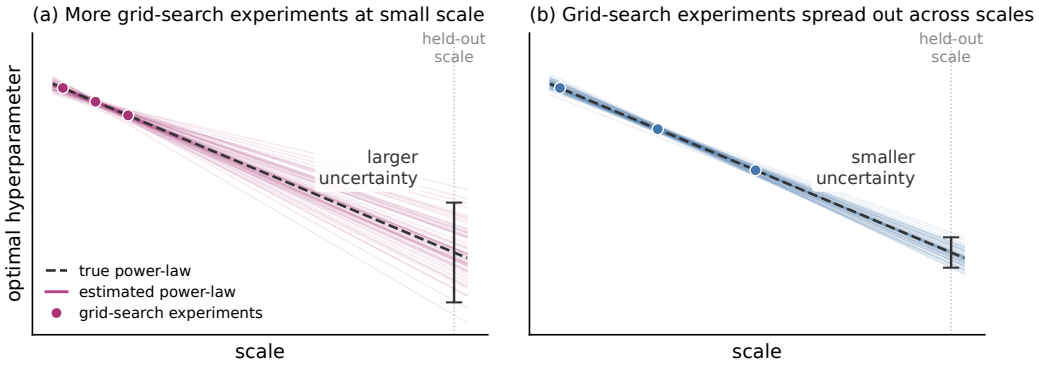


Figure 10: In grid search, we allocate budget uniformly across all scales because it is a stronger baseline than allocating budget focused at a small scale.