ADSO: ADAPTIVE DATA MIXTURE & SCALE OPTIMIZATION A MULTI-SCALE MULTI-FIDELITY BAYESIAN OPTIMIZATION APPROACH

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM pre-training requires careful curation of data sources, a process that currently relies heavily on intuition or costly trial-and-error. Since existing ad hoc approaches are unlikely to transfer across domains or data types, we present a unifying framework for data mixture optimization where (mixtures, model scale, training steps) are chosen to balance cost and potential information gain. Going beyond the canonical deterministic extrapolation in scaling laws, we present a sequential decision-making framework where uncertainty in outcomes is explicitly modeled and sharpened as more measurements are gathered. In particular, we formulate a multi-scale, multi-fidelity Bayesian Optimization (BO) problem where information from smaller-scale experiments can systematically inform larger-scale training decisions. We design an adaptive algorithm that takes into account different measurement fidelities provided by model scale and training steps and empirically demonstrate it on a predictor built on 472 pre-training runs with varying data compositions. Compared to standard BO baselines, instantiating our approach with even simple kernels and acquisition functions can allow principled decisions across training models from 20M to 1B parameters and achieve **2.7x** and **6x** speedups compared to multi-fidelity BO and random search baselines in finding the best data mixture for downstream performance under fixed compute budgets. In sum, our adaptive framework underscores potential efficiency gains achievable by developing principled and transferrable data mixture optimization methods. Our code is publicly available at `https://github.com/anonWAEWA/ADSO`.

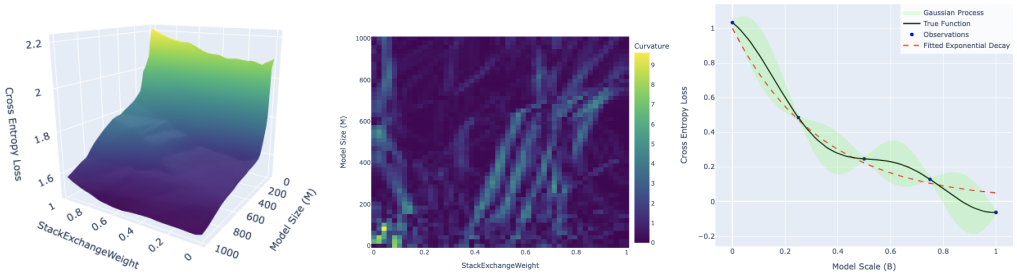

Figure 1: **Left:** The predicted loss as a function of data mixing coefficient and model sizes from a data-driven predictor on 472 runs with high R^2. Notice the highly non-smooth geometry. **Middle:** The curvature (2nd Derivative) at these points shows there are points of high irregularities. **Right:** A demonstration showing how fitted functional forms like exponential decay would demonstrate a high predictive error if fitted on smaller model points. In contrast, a Gaussian Process would capture uncertainty over the points.

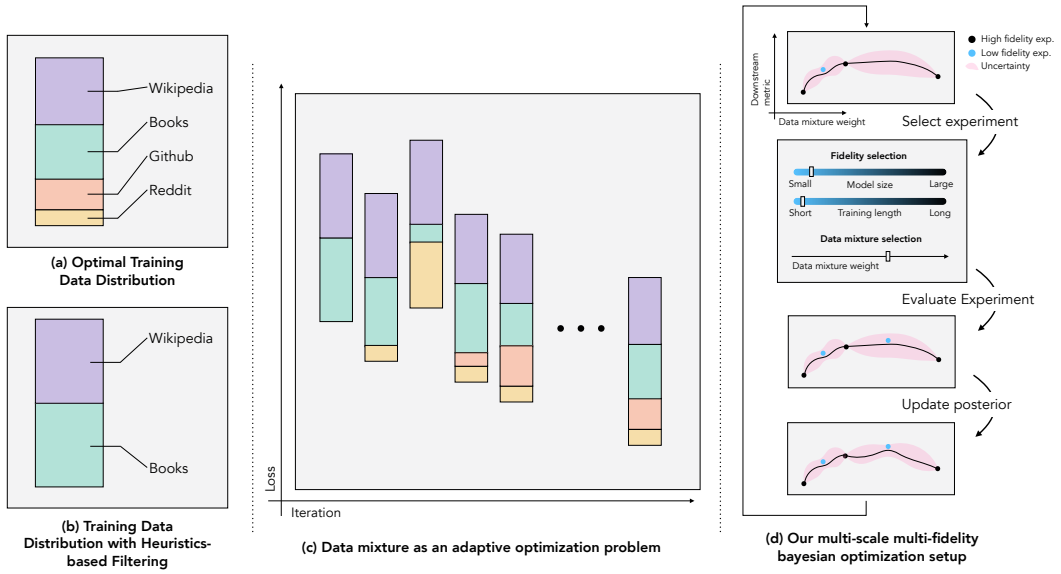

Figure 2: Our multi-scale multi-fidelity Bayesian optimization framework. (a) Given an unknown optimal training data distribution that we have to find, (b) present methods use heuristics-based filtering that guesses and checks. (c) Our algorithm treats data mixture optimization as a Bayesian Optimization problem. (d) Under cost constraint, we explore data mixtures while being cost-aware, where our costs are determined by the fidelity (model size and training steps) that we evaluate. The evaluated result then updates our probabilistic belief over the data mixture, model size, and training steps space, which guides subsequent parameters.

1 Introduction

Data is the foundational infrastructure that all AI systems build on. Scaling data has been a key driver of progress in machine learning, particularly in language model training (Deng et al., 2009; Hoffmann et al., 2022a; Gadre et al., 2024). While this data-centric approach has yielded impressive performance gains, it incurs substantial computational and financial costs in training state-of-the-art language models (Hoffmann et al., 2022a; Luccioni et al., 2023). Beyond raw scale, the composition of training data has emerged as a critical factor: when working with heterogeneous data sources, the choice of training mixture has been shown to significantly impact model performance (Albalak et al., 2023a; Goyal et al., 2024a). This recognition has motivated extensive effort in optimizing data mixing strategies. Some institutions have developed proprietary data mixtures based on domain expertise and empirical observations (Radford et al., 2021; Jiang et al., 2023; OpenAI, 2024), while others have proposed systematic heuristics ranging from Wikipedia upsampling to perplexity-guided data selection (Thrush et al., 2024; Blakeney et al., 2024). However, these approaches are unlikely to transfer across domains and data types. For instance, when organizations in specialized sectors such as healthcare or finance seek to train custom language models on proprietary datasets, it remains unclear whether heuristics developed for public datasets are still effective. Given the substantial resources required for training high-performance language models, there is a pressing need for a principled framework to address data mixture optimization.

Recent works have proposed frameworks that attempt to model the relationship between data mixing coefficients and model performance (Ye et al., 2024; Ge et al., 2025a). However, these approaches make strong assumptions that warrant careful examination. The functional relationship between mixing coefficients and performance is likely context-dependent, varying across different data settings and objectives. Moreover, the assumption in these frameworks that the functional relationship is independent of model scales remains untested across different model sizes. In our empirical study, we trained a data-driven predictor on results from 472 random large language model pretraining runs at various scales, which suggests non-trivial relationships in the performance landscape (see Figure 1).

Instead, we propose viewing the problem of curating the optimal data mixture as an adaptive optimization problem where practitioners iteratively refine their mixing decisions based on empirical observations from previous experiments. This framework leverages the intuition that model performance exhibits local consistency across similar mixtures and model sizes/scales, while avoiding rigid assumptions about the global structure of the performance landscape.

Sequential optimization of data mixtures necessitates comprehending which data compositions suffer the highest uncertainty and sharpening beliefs on performance as more observations are gathered. In particular, good adaptive policies must distinguish between aleatoric and epistemic uncertainty: epistemic uncertainty can be reduced with more data, while aleatoric is irreducible. Measurements must be planned to maximally reduce epistemic uncertainty on future runs by balancing exploration and exploitation.

We formulate this sequential optimization framework as a Bayesian optimization problem: we maintain probabilistic beliefs on the performance of various data mixtures and model scales, and we use these beliefs to choose the next model scale to train, on what data mixture, and for how long. Once we fit and evaluate this new model, we use its performance to update our beliefs (Hutter et al., 2011; Falkner et al., 2018; Frazier, 2018).

In traditional BO, the cost of each new observation is the same, and we aim to optimize an objective while observing the smallest number of points possible. Our setting is more complicated – the cost of training a new model and observing its performance is affected by (1) the number of steps for which the model is trained and (2) the scale of the model (the number of parameters therein).

The number of steps for which a model is trained affects the *quality* of the observation – the more steps we use to train the model, the more accurately the results will reflect the utility of training on the data mixture in question. Previous work has handled this conundrum using so-called multi-fidelity Bayesian optimization, in which evaluations are 'stopped early' during the training process if it becomes clear the information revealed during additional training steps will not be worth the expense (Swersky et al., 2014; Domhan et al., 2015; Kandasamy et al., 2017; Li et al., 2018a).

Our setting is distinguished by the second factor above – we also want to use data gathered on smaller model scales to guide our search over parameters for larger models. Importantly, varying model scale differs fundamentally from traditional fidelity dimensions like training steps. When training for z steps, we naturally obtain observations for all intermediate steps up to z. In contrast, evaluating a model of size m provides no inherent information about the performance of smaller or larger architectures. This raises interesting questions about how to appropriately treat and exploit the this structure, opening new methodological directions for investigations.

Luckily, in contrast to conventional hyperparameters like learning rate or momentum, where optimal configurations exhibit complex scaling behavior across model sizes (Yang et al., 2022), recent empirical evidence suggests that optimal data mixture compositions enjoy greater transferability from smaller to larger model architectures (Ye et al., 2024; Ge et al., 2025a). This transferability property enables the strategic use of smaller-scale evaluations to identify optimal data mixture configurations that remain effective at target model scales, substantially reducing the computational cost of the optimization process.

The main contributions of the paper are as follows:

- *We propose Multi-Fidelity Multi-Scale Bayesian Optimization settings, combining the works in optimizing hyperparameters and scaling laws under one intellectual framework. Our Bayesian Optimization approach better explores different data mixtures and model scales to deliver the best terminal model 2.7x faster in achieving optimal downstream task performance. (Section 4)*
- *We show how smaller model sizes affect the predictive utility of larger runs (e.g. how much does training runs below 500M help predict the losses on 1B) by ablating predictors over training runs of different model scales. (Section 3.3)*
- *We show how earlier training steps improve the predictive utility of others (e.g. given constant FLOPs for hyperparameter search, we show it's better to have 5 full train runs and 10 half train runs vs 10 full train runs) by ablating predictors of different train steps. (Section 3.4)*

2 Problem Formulation: Multi-Scale Multi-Fidelity Optimization

We have access to a set of n datasets $\mathcal{D} = \{d_1, d_2, \ldots, d_n\}$, and wish to train a model comprising m^* parameters for z^* training steps using T datapoints. We study the problem of finding the optimal fraction of our data budget T to draw from each of our n datasets. Let $w_i T$ denote the number of points we draw from dataset i, with $\boldsymbol{w} = \{w_1, w_2, ..., w_n\} \in \Delta^n$, where Δ^n is the n-dimensional probability simplex.

Let $\mu(\boldsymbol{w}, m, z)$ denote the performance of a model comprising m parameters trained for z training steps with dataset proportions $\boldsymbol{w}$ on some downstream task of interest. We seek to solve the following optimization problem

$$\arg\max_{\boldsymbol{w}} \mu(\boldsymbol{w}, m^*, z^*) \tag{1}$$

We have a budget B with which we can experiment with different values of $\boldsymbol{w}$, m, and z. Each evaluation of $\mu(\boldsymbol{w}, m, z)$ incurs a cost $c(m, z)$.

Using m^* parameters and z^* steps every time we evaluate a new set of weights would quickly exhaust our budget. Instead, therefore, we might probe a particular set of weights on a smaller model with m parameters, or with $z < z^*$ training steps – while the resulting observation $\mu(\boldsymbol{w}, m, z)$ would be less informative than $\mu(\boldsymbol{w}, m^*, z^*)$, it would be considerably cheaper and still provide a valuable update to our posterior. This technique is called multi-fidelity Bayesian optimization.

Traditional approaches to fidelity-aware Bayesian optimization primarily address scenarios in which the model architecture m remains fixed and only the number of training steps z is varied (Swersky et al., 2014; Domhan et al., 2015; Kandasamy et al., 2017; Li et al., 2018a). We add a layer of complexity by also considering model scale. It is interesting to note that a fundamental distinction between model scale and number of training steps is that in the course of evaluating a model trained for z training steps, we must also evaluate that model for all steps $z' < z$. No such hierarchical relationship exists for evaluations across different model scales. This structural difference suggests promising avenues for novel methodological developments in multi-fidelity optimization theory, though such extensions lie beyond the scope of our present work.

In this paper, we propose a novel way to take advantage of this additional degree of freedom. To reduce the computational burden of evaluating our technique, we test it on a 'predictor' comprising a surrogate model trained on 472 pretraining runs across diverse data mixture coefficients and model scales. This predictor can accurately predict training loss trajectories for any given model scale and set of mixture coefficients. We then evaluate various Bayesian optimization methods using this surrogate model, assessing their efficacy in both optimizing the predicted utility functions and identifying optimal data mixtures among the sampled configurations at our target scale of 1B parameters.

3 Predictors over Data Mixture and Scale

In this section, we discuss the training of the predictor we will use to test our optimization methods. This predictor serves two purposes in our paper (1) as we mentioned above, it allows us to reduce the computational burden required to test our optimization method by evaluating its performance on *predicted* losses (2) it allows us to develop high-level intuition about the way runs involving smaller models or fewer training steps can inform larger runs. The training of such a predictor is *not* required to use the optimization technique we develop in this paper but serves as a justification for the overall validity of our multi-scale, multi-fidelity formulation.

3.1 Pretraining - Collecting Predictor's Data

We pretrained 472 language models using the OLMo 2 package (OLMo et al., 2024) and data from SlimPajama (Shen et al., 2024), a deduplicated version of RedPajama (Weber et al., 2024). Slimpajama contains seven data categories – *Wikipedia*, *StackExchange*, *Github*, *ArXiv*, *Book*, *CommonCrawl*, and *C4*. We used only data from the first five categories to train the language models while holding out the data from *CommonCrawl* and *C4* to simulate data mixture optimization in

out-of-distribution settings. For each run, we randomly sample the data mixture proportions from a Dirichlet distribution to uniformly sample from the probability simplex and train the models for 196 training steps. Under this setup, we trained models ranging from 20M to 1B parameters. Additional details on the pretraining setup are provided in Appendix A.

3.2 Predictor Training

Our predictor is a multilayer perceptron (MLP) comprising 5,000 parameters that for a given model, predicts its cross-entropy loss over eleven different datasets: the model's training data, each category in RedPajama listed above, and two additional datasets: *CommonCrawl* and *C4*. In addition, the model is evaluated on three downstream tasks: *hellaswag*, *piqa*, *arc_easy*.

The model takes the following inputs (1) the model size (2) the number of steps the model is trained for (3) the proportion of each of the five dataset categories mentioned above used in training. All runs are trained with the same number of tokens. The model is trained to minimize the R^2 between predicted and true values.

We now turn to various insights that can be obtained from this predictor.

3.3 Small Models Help Predict Larger Models Outcomes

	Train	Test
E_1	half of 1B runs	remaining 1B runs
E_2	half of 1B runs	remaining 1B runs + 700M runs
E_3	half of 1B runs	remaining 1B runs + all smaller runs
E_4	half of 700M runs	remaining 700M runs
E_5	half of 700M runs	remaining 700M runs + 500 runs

Table 1: Model size experiments

Dataset	E1	E2	E3	E4	E5
wikipedia	0.75	**0.96**	0.94	0.73	**0.88**
arxiv	0.68	**0.92**	0.93	0.59	**0.82**
github	0.66	0.95	**0.95**	0.62	**0.87**
book	0.83	**0.97**	**0.97**	0.79	**0.92**
stackexchange	0.73	**0.95**	**0.95**	0.68	**0.90**
commoncrawl	0.84	**0.98**	**0.98**	0.81	**0.94**
c4	0.86	**0.99**	0.98	0.82	**0.95**
arceasy	0.92	**0.94**	**0.94**	0.88	**0.90**
hellaswag	0.97	**0.98**	0.97	0.94	**0.96**
piqa	0.94	**0.96**	**0.96**	0.90	**0.93**

Table 2: Results of the experiments listed in table 1, averaged over 3 random seeds. Notice that $E_2 > E_1$ and $E_5 > E_4$ – our ability to predict the performance of larger models is considerably enhanced by insights from smaller models. Note also that $E_3 \approx E_2$; adding information about *much* smaller models does not seem to help.

We begin by investigating the extent to which smaller model runs can inform the dynamics of larger ones. Table 1 details these experiments, and Table 2 lists the results of the experiment.

We note that – as expected – information garnered from training runs on *smaller* models seems to considerably increase the accuracy of our predictions on *larger* models, motivating our hope that a carefully crafted optimization algorithm can exploit the relationship.

Unsurprisingly, we note that the closer in scale the smaller models are to the larger model about which we wish to make a prediction, the more useful the information is. We, therefore, expect our optimization algorithm to 'step through' model scales, starting with small and cheap models to

identify promising data mixtures, and then progressing to larger and larger models, all the while refining the data mixtures it considers optimal.

3.4 Earlier Training Steps Help Predict Later Training Steps

The second central premise of our approach is that given a fixed computer budget, it is better to attempt many runs for fewer training steps than fewer runs for a larger number of training steps.

To test this hypothesis, we carry out three additional experiments. In each of these experiments, we attempt to predict the final losses in 30% of our model runs (evenly distributed across model sizes). The MLP for each of these experiments is trained on (1) a set of complete runs, one for each model size (2) a set of 'truncated' runs, evenly distributed across model sizes. In E_6, we use 16 runs truncated at 196 training steps, in E_7, we use 22 runs truncated at 130 training steps, and in E_8, we use 32 runs truncated at 85 steps; thus, these experiments are trained on numbers generated with the *same* FLOPS budget.

Dataset	E_6	E_7	E_8
R^2	0.69	0.77	**0.82**
$R^2(log)$	0.74	0.82	**0.85**

Table 3: Notice the predictive power of our MLP is strongest when it is trained on many runs for fewer steps. Results averaged over 3 runs. This validates that given a fixed compute budget, it is better to have more runs with fewer training steps than fewer runs for a large number of training steps.

4 Bayesian Optimization Methods

In this section, we describe the Bayesian optimization methods we employ to solve the data mixture problem and evaluate the efficacy of our proposed framework.

4.1 Multi-fidelity Multi-scale Gaussian Process (MFMS-GP)

Algorithm 1 Multi-fidelity Multi-scale Gaussian Process (MFMS-GP)

Require: Probability space Δ^n, model-scale space $\mathcal{M}$, training-step space $\mathcal{Z}$, and cost function $c(\cdot, \cdot)$

1: Initialize Gaussian Process (GP) surrogate model with three RBF kernels over Δ^n, $\mathcal{M}$, and $\mathcal{Z}$ and a linear mean function
2: Randomly sample points from Δ^n, $\mathcal{M}$, and $\mathcal{Z}$ to initialize hyperparameters of GP.
3: Initialize history $\mathcal{H}$ with the randomly sampled points
4: **for** each optimization iteration **do**
5: **for** each $(m, z) \in \mathcal{M} \times \mathcal{Z}$ **do**
6: Optimize EI within (m, z) using gradient descent
7: **end for**
8: Select next configuration $\lambda_{\text{next}} = (\boldsymbol{w}_{\text{next}}, m_{\text{next}}, z_{\text{next}})$ using Expected Improvement per Unit (EIpu):
 $\text{EI}_{\text{pu}}(\lambda) = \frac{\text{EI}(\lambda)}{c(m,z)}$ ▷ EI per unit cost
9: Evaluate $\mu(\lambda_{\text{next}})$
10: Store results in $\mathcal{H}$
11: Update posterior of GP with $\mathcal{H}$
12: **end for**
13: **return** best configuration $\lambda^* = \arg\max_{\lambda \in \mathcal{H}} \mu(\lambda)$

We implement a Gaussian Process (GP) surrogate model for our multi-fidelity multi-scale setting. The kernel of the GP is a product of three separate RBF kernels for the data proportion, the model scale, and the training step dimensions. To enable learning the positive correlation between model performance and both model scales and training steps, we use a linear mean function.

For the acquisition function, we use Expected Improvement (EI). EI aims to quantify the expected gain over the current best-observed function value, $\text{EI}(\mathbf{x}) := \mathbb{E}\left[\max(y^* - f(\mathbf{x}), 0)\right]$, where the expectation is taken over the posterior distribution predicted by the surrogate models, and y^* represents the current best-observed function value, given by $y^* := f(\mathbf{x}_{\min})$ (Frazier, 2018). The EI function quantifies the expected improvement in the objective value compared to the current best, thereby encouraging the selection of points that are likely to yield better performance.

Equipped with EI, the usual Bayesian optimization approach proceeds by optimizing EI over the parameter space to find the most promising point to evaluate, using gradient-based methods such as L-BFGS-B (Zhu et al., 1997). However, motivated by the fact that the parameter space is discrete over parameter counts (m) and training steps (z), we optimize EI over each unique tuple (m, z). Then, to account for the fact that evaluation for each tuple incurs varying costs $(c(m, z))$, we chose to evaluate the point that has the greatest EI per unit cost (EIpu) (Lee et al., 2020).

4.2 Baselines: Multi-fidelity Bayesian Optimization

Algorithm 2 Hyperband with Random Forest, EI

Require: Probability space Δ^n, training-step space $\mathcal{Z}$, target model scale m^*, and cost function $c(m^*, \cdot)$

```
Initialize random forest surrogate model RF
Set initial design with Random Sampling
Initialize history H = ∅
for each Hyperband iteration do
    Split the total computation budget into s brackets
    for each bracket s_i do
        Generate initial configurations w_1, ..., w_n at lowest fidelity z = 1
        for each fidelity z from 1 to z* do
            Evaluate configurations w_i at fidelity z
            Store results in H
            Fit RF on H
            Select next λ_next using Expected Improvement
            Update H with new evaluations
        end for
    end for
end for
return best configuration λ* = arg max_{λ∈H} μ(λ, m*, z*)
```

As a baseline for multi-fidelity Bayesian optimization, we use Hyperband implemented by SMAC: Sequential Model-Based Optimization for General Algorithm Configuration (Lindauer et al., 2022). This multi-fidelity Bayesian optimization uses a random forest as a surrogate model, expected improvement as the acquisition function, and uses Hyperband (Li et al., 2018b), which is an early stopping technique that focuses on efficiently evaluating multiple parameter configurations by progressively eliminating poorly performing candidates, and exploring many combinations with fewer resources. Since the multi-fideltiy framework does not offer an straightforward way to incorporate the additional dimension of model scale, throughout the optimization, we fix the number of model's parameters to the target model scale m^*.

4.3 Baselines: Random and Grid Search

Random and Grid Search selects hyperparameters that are uniformly drawn from our data proportion space. We then run it against the largest model size and training steps.

5 Results

To initiate the hyperparameters of MFMS-GP, we randomly select 20 configurations up to training step $z = 9$ to fit the kernel and mean functions' parameters using the Adam optimizer (Kingma and Ba, 2014). The cost of evaluating these configurations are accounted for. Additionally, since it is

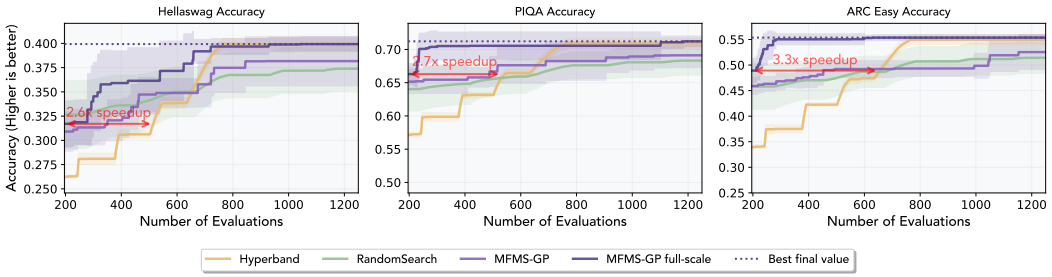

Figure 3: On maximizing accuracy in the downstream tasks, our multi-scale multi-fidelity approach achieves more than 2.7x speedup and finds the best configuration the fastest.

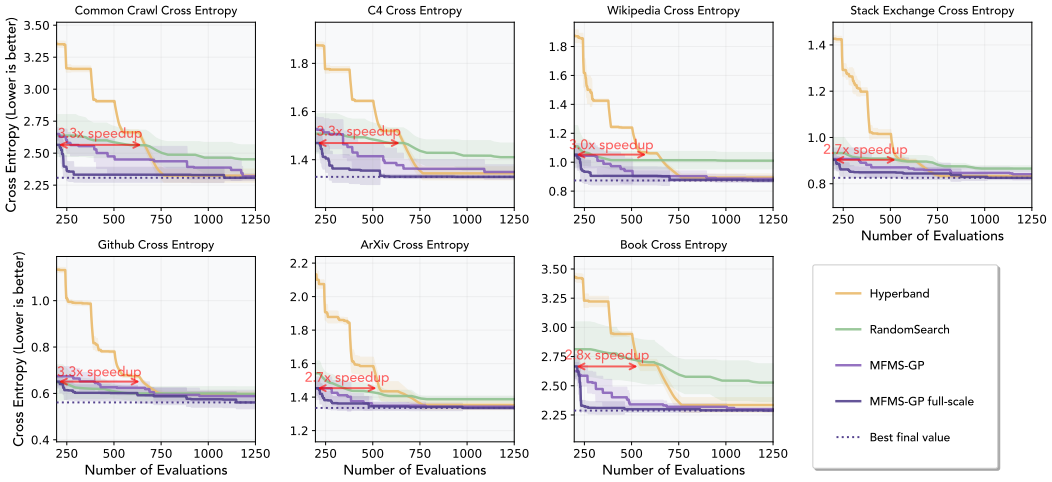

Figure 4: On minimizing the validation cross-entropy losses, our multi-scale multi-fidelity approach achieves more than 2.7x speedup and finds the best configuration the fastest.

prohibitively expensive to optimize EI for each of 196 training steps, for multi-fidelity multi-scale GP, we limit the space of training steps to be $\mathcal{Z} = \{60, 120, 197\}$. Additional details of the experiments are available in Appendix B

All experiments are run over 5 seeds, and the plots show a 1 standard deviation bound. The number of evaluations (the x-axis) are in terms of training FLOPS needed to train one 1B model at 100 training steps. As an example, for Random Search, the 1200 evaluation budget would allow sampling 6 full runs.

Since MFMS-GP relies on GP posterior and potentially noisy EI optimizations to select model scales and training steps, it may take a while to sample points at the target scale and fidelity. Therefore, we add an additional plot, MFMS-GP full-scale, that shows the performance one would have gotten if one takes the best configuration MFMS-GP has observed, and simply set the model scale and training steps to the target m^* and z^*.

In Figures 3 & 4, we see that both of the plots for our MFMS-GP algorithm have a 2.6 to 3.3x speedup in finding the configuration that achieves the highest accuracy. The advantage of the MFMS-GP method speaks to the tremendous potential of considering our framework for large scale language model training.

6 Related Works

Data Mixtures Several approaches aim to move beyond heuristic methods for data mixture by leveraging algorithmic techniques. Albalak et al. (2023b) propose an online data mixing strategy

using a non-stochastic bandit algorithm to dynamically adjust data proportions during training, maximizing perplexity. DoReMi Xie et al. (2023) focuses on identifying and emphasizing the "hardest" datasets for a base model through distributionally robust language modeling to improve training efficiency. Ge et al. (2025b) models a joint scaling behavior of domain proportions and training steps, we push this further through modeling the model scale. Goyal et al. (2024b) delve into the quality-quantity tradeoff in data, exploring how data filtering and repetition affect model performance and introducing scaling laws that account for data utility decay. These works highlight the increasing interest in principled and adaptive methods for data mixture optimization, yet often focus on fixed model scales, contrasting with our multi-scale approach.

Scaling Laws Scaling laws provide crucial insights into the relationship between model size, training compute, and performance in large language models (Kaplan et al., 2020). Hoffmann et al. (2022b) established foundational scaling laws demonstrating predictable performance improvements with increased compute, model parameters, and training data. Muennighoff et al. (2023) investigate the impact of data repetition in data-constrained scenarios, showing diminishing returns beyond a certain repetition threshold. Ruan et al. (2024) propose observational scaling laws based on "principal capabilities" to explain and predict language model performance across diverse models and benchmarks. These scaling law studies inform our framework by providing a theoretical basis for understanding performance variations across model scales and data mixtures, allowing us to integrate these insights into a multi-fidelity multi-scale Bayesian optimization approach.

Bayesian Optimization Data mixture optimization, like hyperparameter tuning, benefits from efficient search strategies. Approaches range from full configuration selection with methods like Bayesian Optimization (BO) to configuration evaluation which employs early termination of unpromising runs. Early BO methods Hutter et al. (2011) used Gaussian Processes (GPs) to model the relationship between hyperparameters and model performance, subsequent works explored random forests (Lindauer et al., 2022) and Parzen estimators (Bergstra et al., 2011) as surrogate models.

Early stopping techniques like Hyperband (Li et al., 2018b) focus on efficiently evaluating multiple parameter configurations by progressively eliminating poorly performing candidates, and exploring many combinations with fewer resources. More recent methods like BOHB (Falkner et al., 2018) combine these ideas, leveraging the BO exploration of Parzen estimators with the multi-fidelity benefits of Hyperband. Our work, ADSO, is the first to explore a multi-scale multi-fidelity approach for data mixture optimization.

7 Conclusion and Future Work

This work introduces a principled framework, multi-fidelity multi-scale Bayesian optimization, for optimizing data mixture compositions in large language model training, a critical challenge in modern AI system development. Our framework unifies recent advances in predicting optimal data mixtures across scales with classical multi-fidelity Bayesian optimization techniques. Based on this unified framework, we implemented the Gaussian process using the RBF kernels and expected-improvement-per-unit acquisition function to balance the information gain and the cost of exploring new points in the functional landscape. We find that the method achieves optimal downstream task performance 2.7 times faster than traditional multi-fidelity approaches by strategically exploring the joint space of data mixtures and model scales.

In addition, we empirically demonstrate two key insights that inform future efficient optimization of data mixtures. First, our analysis reveals that training runs on smaller models (below 500M parameters) provide valuable predictive signals for optimizing larger architectures (1B parameters). Second, we establish that partial training runs can effectively inform full-scale training decisions. Specifically, our results show that a combination of full and partial training runs (e.g. 5 complete and 10 half-length runs) yields better predictive utility than an equal-compute allocation of full training runs alone (e.g. 10 complete runs).

Several promising directions emerge for future research. First, extending our framework to more settings such as language model fine-tuning, data filtering, and more diverse collections of datasets would validate its generalizability across different data mixing scenarios. From a methodological perspective, incorporating domain knowledge about the positive correlation between model performance and both parameter count and training duration could enhance the Gaussian process kernel

design. Additionally, the fundamental differences between model scale and training steps as fidelity dimensions call for deeper methodological investigation in their appropriate treatment in the framework. Finally, exploring alternative acquisition functions, such as knowledge gradient (Poloczek et al., 2016; Wu et al., 2019), could further improve the framework's efficiency in navigating the optimization landscape.

IMPACT STATEMENT

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

REFERENCES

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. Efficient online data mixing for language model pre-training, 2023a. URL https://arxiv.org/abs/2312.02406.

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. Efficient online data mixing for language model pre-training, 2023b. URL https://arxiv.org/abs/2312.02406.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'11, page 2546–2554, Red Hook, NY, USA, 2011. Curran Associates Inc. ISBN 9781618395993.

Cody Blakeney, Mansheej Paul, Brett W. Larsen, Sean Owen, and Jonathan Frankle. Does your data spark joy? performance gains from domain upsampling at the end of training, 2024. URL https://arxiv.org/abs/2406.03476.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 3460–3468. AAAI Press, 2015. ISBN 9781577357384.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: robust and efficient hyperparameter optimization at scale. *CoRR*, abs/1807.01774, 2018. URL http://arxiv.org/abs/1807.01774.

Peter I. Frazier. A tutorial on bayesian optimization, 2018. URL https://arxiv.org/abs/1807.02811.

Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, Rui Xin, Marianna Nezhurina, Igor Vasiljevic, Jenia Jitsev, Luca Soldaini, Alexandros G. Dimakis, Gabriel Ilharco, Pang Wei Koh, Shuran Song, Thomas Kollar, Yair Carmon, Achal Dave, Reinhard Heckel, Niklas Muennighoff, and Ludwig Schmidt. Language models scale reliably with over-training and on downstream tasks, 2024. URL https://arxiv.org/abs/2403.08540.

Ce Ge, Zhijian Ma, Daoyuan Chen, Yaliang Li, and Bolin Ding. Bimix: A bivariate data mixing law for language model pretraining, 2025a. URL https://arxiv.org/abs/2405.14908.

Ce Ge, Zhijian Ma, Daoyuan Chen, Yaliang Li, and Bolin Ding. Bimix: A bivariate data mixing law for language model pretraining, 2025b. URL https://arxiv.org/abs/2405.14908.

Sachin Goyal, Pratyush Maini, Zachary C. Lipton, Aditi Raghunathan, and J. Zico Kolter. Scaling laws for data filtering—data curation cannot be compute agnostic. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22702–22711, 2024a. doi: 10.1109/CVPR52733.2024.02142.

Sachin Goyal, Pratyush Maini, Zachary C. Lipton, Aditi Raghunathan, and J. Zico Kolter. Scaling laws for data filtering – data curation cannot be compute agnostic, 2024b. URL https://arxiv.org/abs/2404.07177.

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

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

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Proceedings of the 5th International Conference on Learning and Intelligent Optimization*, LION'05, page 507–523, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 9783642255656. doi: 10.1007/978-3-642-25566-3_40. URL https://doi.org/10.1007/978-3-642-25566-3_40.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabás Póczos. Multi-fidelity Bayesian optimisation with continuous approximations. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1799–1808. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/kandasamy17a.html.

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, 2020. URL https://arxiv.org/abs/2001.08361.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL https://api.semanticscholar.org/CorpusID:6628106.

Eric Hans Lee, Valerio Perrone, Cédric Archambeau, and Matthias W. Seeger. Cost-aware bayesian optimization. *CoRR*, abs/2003.10870, 2020. URL https://arxiv.org/abs/2003.10870.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018a. URL http://jmlr.org/papers/v18/16-558.html.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization, 2018b. URL https://arxiv.org/abs/1603.06560.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhkopf, René Sass, and Frank Hutter. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23(54):1–9, 2022. URL http://jmlr.org/papers/v23/21-0888.html.

Alexandra Sasha Luccioni, Sylvain Viguier, and Anne-Laure Ligozat. Estimating the carbon footprint of bloom, a 176b parameter language model. *Journal of Machine Learning Research*, 24(253):1–15, 2023. URL http://jmlr.org/papers/v24/23-0069.html.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models, 2023. URL https://arxiv.org/abs/2305.16264.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 olmo 2 furious, 2024. URL https://arxiv.org/abs/2501.00656.

OpenAI. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Matthias Poloczek, Jialei Wang, and Peter I. Frazier. Multi-information source optimization, 2016. URL https://arxiv.org/abs/1603.00389.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL https://arxiv.org/abs/2103.00020.

Yangjun Ruan, Chris J. Maddison, and Tatsunori Hashimoto. Observational scaling laws and the predictability of language model performance, 2024. URL https://arxiv.org/abs/2405.10938.

Zhiqiang Shen, Tianhua Tao, Liqun Ma, Willie Neiswanger, Zhengzhong Liu, Hongyi Wang, Bowen Tan, Joel Hestness, Natalia Vassilieva, Daria Soboleva, and Eric Xing. Slimpajama-dc: Understanding data combinations for llm training, 2024. URL https://arxiv.org/abs/2309.10818.

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

Tristan Thrush, Christopher Potts, and Tatsunori Hashimoto. Improving pretraining data using perplexity correlations, 2024. URL https://arxiv.org/abs/2409.05816.

Maurice Weber, Daniel Fu, Quentin Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Ré, Irina Rish, and Ce Zhang. Redpajama: an open dataset for training large language models, 2024. URL https://arxiv.org/abs/2411.12372.

Jian Wu, Saul Toscano-Palmerin, Peter I. Frazier, and Andrew Gordon Wilson. Practical multi-fidelity bayesian optimization for hyperparameter tuning. *CoRR*, abs/1903.04703, 2019. URL http://arxiv.org/abs/1903.04703.

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, 2023. URL https://arxiv.org/abs/2305.10429.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer, 2022. URL https://arxiv.org/abs/2203.03466.

Jiasheng Ye, Peiju Liu, Tianxiang Sun, Yunhua Zhou, Jun Zhan, and Xipeng Qiu. Data mixing laws: Optimizing data mixtures by predicting language modeling performance, 2024. URL https://arxiv.org/abs/2403.16952.

Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4): 550–560, December 1997. ISSN 0098-3500. doi: 10.1145/279232.279236. URL https://doi.org/10.1145/279232.279236.

A PRETRAINING RUNS DETAILS

We use the OLMo 2 OLMo et al. (2024) package for training our language models. The model configurations are

Group	d_model	n_heads	n_layers	Runs
20M	256	8	8	115
60M	512	8	8	71
150M	768	12	12	53
300M	1024	16	16	74
500M	1280	16	16	39
700M	1536	16	16	52
1B	2048	16	16	68

Table 4: Model Architecture Details by Group with Number of Runs

The training configurations (learning rate, momentum etc.) are directly taken from OLMo's configuration files (e.g. 700M) We study the compute optimal regime Hoffmann et al. (2022a): for each 1B model runs, we used 20B tokens in total for training. In the interest of collecting more runs, all other model scales are trained on 10B tokens.

B BAYESIAN OPTIMIZATION DETAILS

For MFMS-GP, the cost of evaluating a run at a particular model scale is taken from the number of FLOPS the corresponding model scale costs during the pretraining runs. The costs are scaled appropriately such that a unit of cost corresponds FLOPS required to train 1B model for 1 training step.

The GP hyperparameters are trained using the Adam optimizer with 0.1 learning rate for 50 iterations.

To search for optimal EI within each (m, z) tuple, we initiate 5 random probability weights and perform a gradient search over the probability simplex.

Occasionally, the GP would be too certain of its posterior prediction such that the optimized EI are all small in magnitude. Therefore, when the optimal EI is below a certain threshold, we lower the length scales of the RBF kernels to encourage more exploration. The threshold is set to be $1e^{-4}$, and the length scales would be lowered to 95% of their original values.

As a measure to encourage selecting higher cost evaluations later in the optimization cycle, instead of using $\text{EI}_{\text{pu}}(\lambda) = \frac{\text{EI}(\lambda)}{c(m,z)}$, we introduce an additional parameter α that controls the importance of cost, and pick the configuration that maximizes $\frac{\text{EI}(\lambda)}{c(m,z)^\alpha}$. Initially $\alpha = 1$, and it decays by 1% for every step of the Bayesian optimization. You may include other additional sections here.