Bayesian Optimization for Mixture Optimization in Large Lanuage Models

1st Tim Lindenau
Technical University of Munich
Munich, Germany
tim.lindenau@outlook.de

***Abstract*—To achieve broad knowledge across many tasks, language models (LM) are pre-trained with text from a wide range of different datasets/domains. The proportions of these datasets play a crucial role in determining LM performance. In this work, we introduce a novel approach to data-mixture optimization, formulating it as a sequential black-box optimization problem, solved using Bayesian Optimization (BO). BO is employed to leverage its sample efficiency, enabling effective exploration of the data mixture space while minimizing the number of training runs required. Instead of working on a full production scale, our method identifies optimal data mixtures on a down-scaled proxy model, which is more effective to train than the full-scale model, assuming that optimal mixture distributions remain consistent across model scales.**

Through extensive experiments, we demonstrate the effectiveness of our approach in identifying promising proxy-model mixtures, outperforming random search, several popular mixtures from the literature, and state-of-the-art (SOTA) mixture optimization methods. However, we also identify limitations in our approach, the biggest lying in the assumption of scale invariance, showing that transferring optimal mixtures from proxy models to larger models poses challenges, as some mixtures that perform poorly at small scales outperform our proxy mixtures when scaled up. This highlights the need for further research into the scaling behavior of data mixtures to better harness the insights gained from small proxy models.

I. Introduction

To equip language models (LMs) with broad knowledge and capabilities, these models are pre-trained on a large corpus of diverse text data. Diversity is achieved as this dataset consists of many sub-sources, we call them domains, each revolving around different tasks and different text origins. For example, the recently published DOLMA dataset [27] consists of data from 15 domains like web data, code, or scientific papers. Naturally, the proportions of these datasets used during training significantly affect the achievable LM performance [33, 2, 15] to different downstream tasks and therefore carefully selecting these mixtures is important for generating well-performing, general models.

Up to now, domain weights were mostly decided based on empirical intuition about the quanlity of different datasets [11] or by random searching about the mixture space, potentially requiring training thousands of models [5]. Recently, the research has shifted towards identifying more principled approaches for identifying optimal mixture distributions, while requiring as few additional compute as possible.

Online methods [1] optimize the mixture while training the model based on e.g., the per-domain perplexity. However, they do not provide any interpretable insight about optimal mixtures but only about the next-best sample to choose at each time step.

Offline methods, on the other hand, identify promising mixtures before training, typically based on smaller proxy models, assuming that insights scale from proxy to production model [33, 12, 34]. As, even when utilizing proxy models, compute constraints can quickly become a limiting factor, a key requirement of these offline methods is to achieve sample efficiency in the number of proxy models that are required to train.

Multiple interesting approaches towards offline mixture optimization have already been proposed. The earliest method, DoReMi [33], draws inspiration from robust optimization [24] and aims to minimize the worst-case loss compared to a second reference model. Other works, such as those by Ye et al. [34] and Ge et al. [12], focus on identifying different functional forms of scaling laws to predict optimal mixtures for large models by fitting these laws to proxy model evaluations. However, none of these methods have demonstrated sufficient effectiveness to achieve widespread real-world adoption with their weights either failing to achieve improvements at all or not scaling well across e.g., different model architectures. For instance, DoReMi has been shown to struggle with scaling when faced with changes in model architecture and tokenizer [12], which limits its practical utility and highlights the need for further research in this area.

Our work investigates mixture-search by phrasing it as a sequential black-box optimization problem which we solve using Bayesian Optimization (BO). While BO for sequentioal optimization is widely studies with numereous real-world applcations [] we are, to the best of our knowledge, the first to apply it to the problem of finding optimal data mixtures,

Based on a number of initial proxy-model training runs, our approach builds a probabilistic model, predicting language modelng performance as a function of mixture weights. In the subsequent optimization runs, this probabilisit model is then uses to propose the next training mixture, thereby trading off exploiting promising regions and exploring unknown regions.

For the smaller proxy models, we demonstrate that the Bayesian formulation is an effective approach for identifying promising regions of the search space achieving higher

sample-efficiency than vanilla random search. This is shown as in all experiments, Bayesian Optimization (BO) consistently found superior mixtures when normalizing to the same number of function evaluations. Additionally, our method outperforms several baselines based on mixing laws, as well as the commonly referenced weights from LLaMA [28] and Dolma. While effective in identifying weights for the small-proxy model architecture, we also show that simply transferring the optimal proxy-model weights to larger model architectures does not work well as mixtures inferior at the proxy-model scale can achieve better performance when scaling up the architecture by a factor of five. Furthermore, we show limitations in scaling up BO to many domains, as well as practical issues of the commonly used optimization criterion of average domain performace, risking the creation of expert models that do not generalize well. All code is available at GitHub.

II. Bayesian Optimization For Data Mixing

We investigate BO for effectively identifying optimal training data mixtures for large language models. The overall goal is that given a production-grade model architecture f (generally, very large and expensive to train) and a dataset $\mathcal{D}$ consisting of k domains $\mathcal{D}_i$, we aim to find the best mixture proportions $\alpha \in \Delta^k$ that minimizes a loss criterion R of a model $f(\alpha)$ trained on this data:

$$\alpha^* = \arg\min_{\alpha \in \Delta^k} R(f(\alpha)). \tag{1}$$

Solving this optimization problem is challenging as $f(\alpha)$ behaves like a black-box with the only information available for optimization being the function evaluations at some mixture α_0. As introduced more specifically in Appendix B multiple approaches have recently benn proposed to tackle this challenge [33, 12, 34, 17]. What they all have in common is, is that instead of utilizing the large and expensive to train model f, they all employ a smaller proxy model f' for identifying optimal mixture weights. The optimal mixture of f is then found either based on assuming is equal across model and dataset scales [33] or by postulating some scaling law of optimal mixture across model-size that is fitted by optimizing mixtures for different proxy scales (i.e., f', f'', ...) [17]).

We propose to utilize BO as a compute-efficient way for identifying the optimal mixture distribution α' of f'. The choice of BO is natural, as it is one of the most sample-efficient approaches for solving black-box optimization problems and has previously already demonstrated its effectiveness in domains such as experiment design [13] and hyperparameter optimization [26]. A detailed introduction into the theory behind BO is given in Appendix A, with a shorter introduction within the context of mixture-optimization provided in the following. The general idea of BO for data mixing is to sequentially build a probabilistic model $g(r, \alpha|\mathcal{B}_t)$ of the LM performance as measured by the criterion R over a sequence of training trials $\mathcal{B}_t = \{\alpha_t, R(f'(\alpha_t))\}_{t=0}^t$. More specifically,

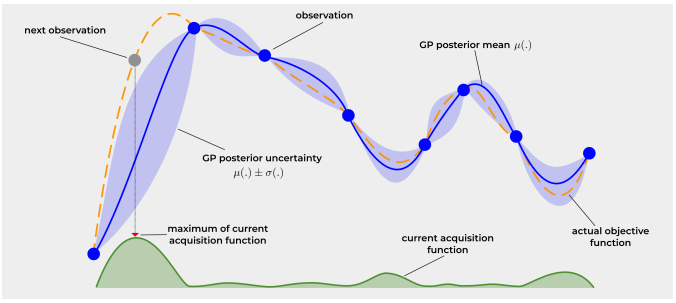


Fig. 1: Visualization Bayesian Optimization Algorithm. The blue curve shows how a probabilistic model of the underlying objective function is built based on the previous function evaluations $\mathcal{B}_t = \{\boldsymbol{x}_i, y_i\}_{i=1}^t$. Green illustrates the acquisition function estimating the value of each potential next sampling location. Orange is the updated predicted mean after incorporating the next sample. Visualization: [8]

.

given some mixture α, the probabilistic model g gives a probability of the expected performance

$$g(r_0, \alpha_0|\mathcal{B}_t) = \Pr\left(R(f'(\alpha)) = r_0 | \alpha = \alpha_0, \mathcal{B}_t\right). \tag{2}$$

Based on this probabilistic model, the next mixture-proposal is chosen by optimizing a so-called acquisition function that measures the expected utility of each possible mixture-proposal based on its expected performance value and uncertainty under g. It is exactly this smart way of proposing the next sampling location that makes BO so sample-effective and therefore a well-motivated approach towards our goal of data mixing. Figure 1 illustrates the BO process on a simple 1D example.

We build the probabilistic model g based on the model of a Gaussian Process (GP), which is the most common probabilistic model used in BO. The choice of acquisition function is a very important design decision as it decides how to value the utility of the next sampling position, trading of exploration and exploitation. We decide to employ the acquisition function of expected improvement due to its widespread adoption and generally good performance. Let r_{best} be the best function evaluation so far, then the expected improvement is defined as

$$\text{EI}(\alpha) = E_r\left[\max\left(0, r_{\text{best}} - g(r, \alpha|\mathcal{B})\right)\right]. \tag{3}$$

Maximizing this function gives the next sampling location as

$$\alpha_{t+1} = \arg\max EI(\alpha). \tag{4}$$

To simplify optimization of the acquisition function we utilize a logarithmic version of EI instead of the vanilla method, as this has been shown to improve performance without incurring any downside cost [3].

Based on the theoretic background, each experiment follows the same process. We first initialize it by randomly sampling n_{init} initialization mixtures and corresponding function evaluations. We employ a sobol sampling strategy to sample the available mixture space most uniformly, thereby providing

maximal information gain in this initial stage. After initialization, the process is then followed by n_{optim} optimization stages where at each time step t, the new sampling location is proposed based on all previous evaluations $\mathcal{B}_t$.

III. Experiments

A. Training Setup

We employ the OpenLM [14] training framework for training and evaluating all models. Unless stated otherwise the default model size is 25M parameters. All models are trained *Chinchilla optimal*, meaning the number of training tokens equals 20 times the parameter count [15]. All other hyperparameters are selected according to Gadre et al. [10], which have empirically been proven to be suitable across different model sizes.

We are using DOLMA [27] as the source for all training data, which is a curation of different language datasets that were post-processed for the highest quality by focusing on e.g. filtering and deduplication. Specifically, we employ DOLMA version *v1.6* consisting of the seven domains *Common Crawl (CC), The Stack (Stack), C4, Reddit, PeS2o, Project Gutenberg (Books), and Wikipedia (Wiki)*. This is a diverse collection providing text types from general web pages to encyclopedias to code. From all sources, we sample enough tokens such that the training is never limited by token count and all possible domain mixtures are achievable without duplication.

The decision of how to measure the performance of a given mixture is important as well. Ideally, this indicator should directly measure downstream task performance for example by reporting the results of a collection of benchmarks. As, however, models of our size do not produce any usable results of common benchmark suites, we instead utilize log-perplexity as a proxy criterion. More specifically, we first measure each model's performance on a held-out split of each domain. Averaging the results across all domains then gives the final performance score.

We are employing a diverse number of baselines. First, random search using a sobol sampling strategy (Sobol) to achieve the highest possible sample efficiency. Further baselines are uniform weights (Uniform), as well as the default weights of Dolma and LLaMA, always appropriately scaled to the number of domains used in each experiment. Finally, the performance of the mixing laws of Ge et al. [12] (BiMix) and Touvron et al. [28] (MixLaws) are employed, based on our own re-implementation of their code.

B. Data Mixing in the Two Domain Setting

We start our investigation by focusing on the setting of two domains to validate our algorithm and show that BO is capable of finding good mixtures with few samples. For the mixture of C4 and Books, Figure 2a shows the average log-perplexity depending on the ratio of books (left plot), as well as for each domain, how its log-perplexity behaves as a function of its domain weight (center- and right plot). All values are collected over different checkpoints, to verify whether trends are consistent over different data scales.

Previous work on mixture laws already identified an exponential relationship between domain-loss and mixing weights [12, 34] of the form

$$L_i(w_i) = c_i + k_i \exp\left(t_i r_i\right), \tag{5}$$

with c_i, k_i, and r_i being learnable parameters. Our results reproduce this finding as for both domains, performance on a log-y scale follows a linear relationship over mixing weight (center- and right plot). However, unlike the other works, we note that this relationship only holds once a minimum amount of domain weight is reached, with the perplexity increasing steeply before this point.

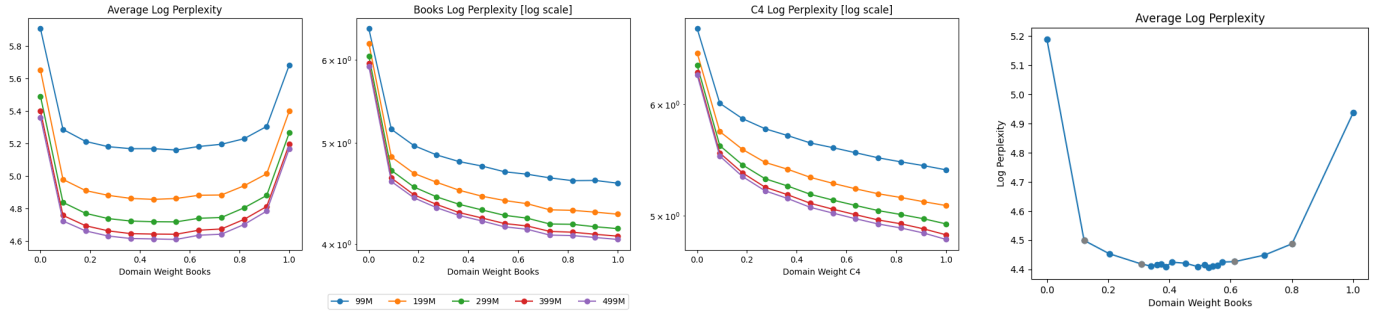
Based on the insight into the linear relationship, it makes sense that for the two-domain setting, there is a large flat region with near-optimal values as it is shown on the left. The specific location on the optimum is dependent on the sensitivity of both domains in relation to different domain weights. The optimum is in the center if both domains show similar sensitivity and shifts to the right for unequal sensitivity values. For example for the shown mixture of C4 and Books, the optimum is located approximately in the center, while in a mixture of CC and Stack, the latter shows a much higher sensitivity leading to an optimal ratio close to 30/70.

To provide a deeper understanding of the Bayesian Optimization (BO) process, Figure 2b illustrates its key mechanisms, highlighting how BO achieves its high sample efficiency. After an initial random phase, where a few points are sampled uniformly across the search space (grey), BO quickly identifies promising regions. In our specific example, it zeroes in on areas slightly left and right of the center, where the likely optimal values are located. Subsequently, BO focuses the majority of future evaluations in these regions. This behavior of BO aligns well with our prior analysis of the function's sensitivity, which predicted the maximum to be near the center, with slight shifts to the left and right based on actual sensitivities.

To conclude, this experiment confirms that our implementation of BO is functioning as expected, effectively identifying and exploiting optimal regions. With this validation, the next section will focus on scaling up BO to additional domains, assessing its performance across a more complex setting and larger search spaces.

C. Bayesian Optimization Successfully Identifies Effective Mixtures of Five Domains

Based on the promising insight from the two domain setting, we shown that BO is capable of identifying optimal-mixture distributions while being more sample efficient than random search and also outperforming the state of the art of mixing laws. Table I summarizes the main results of mixing the five domains of CC, Books, Stack, Pes2O, and Reddit. The baselines are the ones introduced previously. To gain comparability, Sobol, BiMix, Data Mixing, and BO all share the same number of 64 total function evaluations. For BO we split them up into 32 initialization and 32 optimizations.



(a) Performance measurements across data mixtures and training checkpoints show the optimal mixture near the center, slightly offset based on per-domain sensitivity (left). Perplexities follow a linear scaling relationship when plotted in a log-y plot (center, right).

(b) BO successfully identifies desirable points near the center, leading to concentrated sampling in that region (grey: initialization, blue: BO).

Fig. 2: Visualization Bayesian Optimizaiton in two domains.

Method	Sobol	LLaMa	Dolma	Uniform	BiMix	MixLaws	BO
Avg. Perp	4.4109	4.6826	4.8835	4.46588	4.4375	4.6547	4.3688
Books	0.160	0.049	0.002	0.2	0.154	0.265	0.158
CC	0.242	0.740	0.800	0.2	0.088	0.00	0.207
Stack	0.312	0.071	0.144	0.2	0.427	0.442	0.256
Pes20	0.0490	0.028	0.0245	0.2	0.243	0.294	0.271
Reddit	0.237	0.110	0.0312	0.2	0.088	0	0.108

TABLE I: Average validation perplexity on the mixture of five domains across different optimal-mixture identification methods. All values reported in Log-Perplexity.

Starting with the baseline of random search (Sobol) we can see that it is generally performing well, achieving the second-best performance of all methods. The Dolma baseline does not seem to function at all achieving by far the worst results. We believe this could be due to its large focus on common crawl. As discussed in Kang et al. [17], unstructured data such as common crawl improves performance only for very large models trained on appropriate amounts of data but hinders performance for small models with fewer data; exactly what we see in our experiment. The LLaMa mixture performs slightly better but also suffers from its large dependence on CC. Uniform provides a decent baseline, especially considering that it does not require any function evaluations during optimization. Switching over to the methods that require multiple function evaluations and focusing on both SOTA mixing law methods, we can see that both cannot reproduce the promised performances stated in the original papers, only achieving results worse than random search, indicating that their proposed laws cannot faithfully predict the true function shape. This hypothesis is further strengthened as our testing shows that the predicted performance of both optimal mixtures is significantly better than the performance achieved when training said mixture on the true model. Finally, BO achieves the best perplexity out of all methods. Compared to the second-best approach (Sobol) it improves performance by 0.042 in absolute- and $0.9\,\%$ in relative terms. This improvement over random search speaks to the sample efficiency of BO, which it achieves by focusing mostly on promising regions, omitting unpromising regions that random search spends a lot of its evaluation budget on.

D. The Importance of Selecting an Appropriate Number of Total Trials and Initializations

This section gives more insight into the process of BO, and the importance of selecting an appropriate number of total evaluations and initialization. To this end, Table II provides further experimental results, exchanging the number of total trials, initialization trials, as well as the quality of initialization.

Name	No Trials	No Init	Best Init	Perplexity
Short - Good Init	32	16	4.4109	4.3927
Short - Bad Init	32	16	4.4393	3.4385
Long - Bad Init	64	16	4.4393	3.3823
Long - Many Init	64	32	4.4109	4.3688

TABLE II: A deeper look into the behavior of BO dependent on the number of trials and the quality of initialization.

First, focusing on the total number of trials, it must naturally be a function of the dimensionality of the search space, as the number of dimensions significantly increases the size of possible values to sample from. The first two rows of Tab. II show that while in our five-domain experiment, it is possible to gain decent performance with only 16 initialization and 32 optimizations, generally the information acquired with this few evaluations is not enough to give BO a decent overview of the optimization space and instead of being the result of effective optimization, the achieved performance depends more on the quality of initialization. This explains why *Short – Good Init*, where the initialization samples are in a favorable region, BO still performs well because it can exploit the knowledge from the good initialization. When, however, the initialization occurs in less favorable regions, as in *Short – Bad*

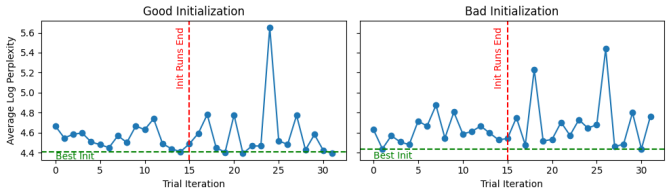


Fig. 3: Effect of initialization quality on BO with insufficient number of total trials.

Init, performance drops substantially because BO is incapable of identifying a new promising region to sample without being able to exploit the initialization. Figure 4 illustrates the same behavior by visualizing the experiment history of *Short – Good Init* and *Short – Bad Init*. What one can see is that in both cases BO does not have enough information to explore completely new and novel regions, leading to no significant performance improvement over the initialization. Rather both just find slight optimizations around the area of the initialization, leading to only small improvements unable to overcome the burden of a weak initialization.

If on the other hand, the number of trials is large enough, the luck in the initialization becomes less important and the model *Long – Bad Inint* becomes capable of reasoning about the space search effectively, identifying promising regions by itself and significantly improving overall performance. Figure 4 illustrates the capability of identifying new promising regions that outperform the original initialization well. Unlike the case of *Short Init*, BO now achieve significant performance improvements over the initialization starting from around iteration 35. Importantly these new identified mixtures diverge significantly from the original initialization samples, as is illustrated with the colored points, where each color means one cluster of close mixing weights. No initialization points belong to any of the best-performing clusters, while on the other hand, many of the well-performing mixtures share few clusters, illustrating the property of exploiting.

Given that Bayesian Optimization (BO) can effectively explore the search space with a sufficient number of trials, a natural question arises: why use initialization at all? Prior work [26] has demonstrated that sufficient initialization samples – particularly when acquired through uniform sampling of the search space – play a valuable role in achieving best performance when constrained by the number of function evaluations. In the early stages of optimization, uniformly sampled initial points provide more diverse coverage of the search space, enabling faster information gain compared to relying solely on the acquisition function to guide exploration. For instance, in our testing, we observed that for five domains BO with 32 initialization was capable of finding a slightly better optimum than with only 16 initialization. The difference was, however, close and not necessarily significant enough to make a clear statement about the optimal number of initializations one should utilize for best results in our five-domain experiment.

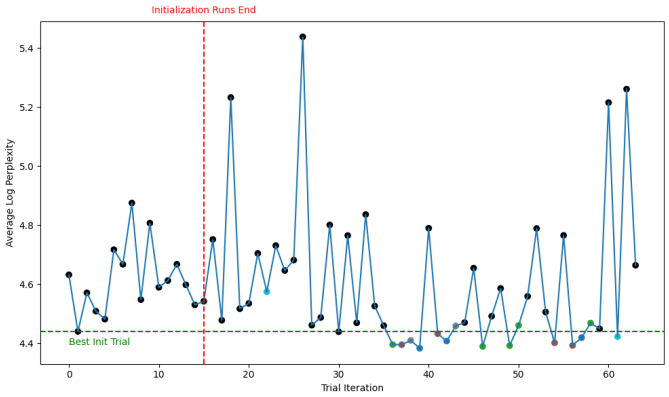


Fig. 4: BO with sufficient trials identifies maximum clearly outperforming initialization.

E. Relevany of the Results

Name	Trial 0	Trial 1	Trial 2	Trial 3	Avg. Perp.	Std
Sobol	4.4109	4.4281	4.4531	4.4552	4.4368	0.0183
BO	4.3688	4.3781	4.4123	4.3842	4.3858	0.0162

TABLE III: Variance analysis. Repetition of best BO and Sobol run using different random initializations.

The experiments (e.g., Tab I) have shown that BO is effective given sufficient iterations, outperforming the random search and mixture laws when utilizing the same number of trials. The result that BO is more sample efficient than random search aligns with previous work of Turner et al. [29]. They analyzed all results of a black-box optimization challenge held at *NeurIPS 2020* and showed that in this challenge BO always outperformed random search, achieving a maximum sample efficiency improvement of more than 100 %. In this section, we provide a more systematic review of our results, critically asking whether our improvements are sufficient enough to validate the effectiveness of our method.

On the variance of the results: The first variance experiment asks how much variance training a LM gives as naturally our results are only meaningful if the improvement outperforms the variance that we naturally would expect between different repetitions of the same data mixture. Tab. III summarizes the results of repeating training with the best Sobol and BO mixture for a total of 4 trials, each utilizing a different random seed. Three insights can be taken. First, while there is variance in all experiments, the variance values of $\sigma < 0.02$ are smaller than the improvement of 0.042, indicating that our reported improvements are significant. Second, out of all repeated trials, the original one is always the best, indicating that our optimization approach exploits the uncertainty in the evaluation, especially by sampling many trials with similar mixture distributions as can be seen well in the clusters in Fig. 4. It is therefore to be expected that the result for the initial train (Trial 0) is always better than the average results that can be expected over different repetitions that switch up the random seed. Third, even after averaging the results,

BO remains better by a margin of 0.051. This improvement exceeds the originally reported margin based on the Trial 0 evaluations, demonstrating that both BO and random search exploit randomness to a similar extent and strengthening the evidence that our improvements are significant.

Name	Trial 1	Trial 2	Trial 3	Trial 4	Avg. Perplexity
Sobol	5.4745	5.4591	5.4601	5.4553	5.4622
BO	5.4478	5.4387	5.4412	5.4409	5.4422

TABLE IV: Relative performance comparison BO vs. Sobol. Each run is executed for 32 trials using a 15M parameter model on the four domains of Books, CC, Stack, and Pes2O.

Reproducibility in repeated execution: The second experiment investigates the reproducibility of our results, specifically addressing whether repeated BO restarts consistently outperform random search with the same number of iterations. Table IV presents the results of optimizing the mixture of a 15M model across four domains: Books, CC, Stack, and Pes2O, over 32 trials. Notably, even in the worst case, BO outperforms the best random search run, reinforcing BO's effectiveness in identifying better mixtures. However, the average improvement observed (0.02) is less substantial compared to the 0.05 improvement seen in the main experiment and only marginally exceeds the expected variance reported before.

Interpretation of this last result is therefore challenging. On one hand, the consistent superiority of BO over random search across individual runs suggests that BO's performance is reproducible. On the other hand, the small magnitude of improvement raises concerns about the significance of these improvements. Two interpretations emerge: (a) with the four domains and the given model, the random search may already be consistently finding near-optimal distributions, leaving little room for BO to provide further improvement, or (b) BO may not be as effective in this specific experimental context. Further investigations are required to determine which hypothesis holds, and if (b) is the case, to better understand when BO fails and when it succeeds.

Limitation of the average perplexity optimization criterion: Within our and most similar works, the average perplexity across domains is typically employed as the optimization criterion. However, this approach risks undesirable outcomes as it might be more effective to just significantly improve performance on one domain while decreasing performance on all others, as long as it would boost average performance. Such behavior would be a problem because it would create a model that despite high performance would actually generalize badly in the real-world. To assess this risk, we compare the per-domain perplexity of our best model against the best random search model. Figure 5 shows that the hypothesized risk is indeed a problem: despite BO achieving lower average perplexity, it underperforms on four out of the five domains, significantly outperforming the Sobol mixture only on the Pes20 dataset. To conclude, optimizing for the lowest average domain perplexity may not be the ideal criterion. Alternative metrics, such as loss relative to a reference model as used in DoReMi, might be more effective in developing models that excel across all domains.

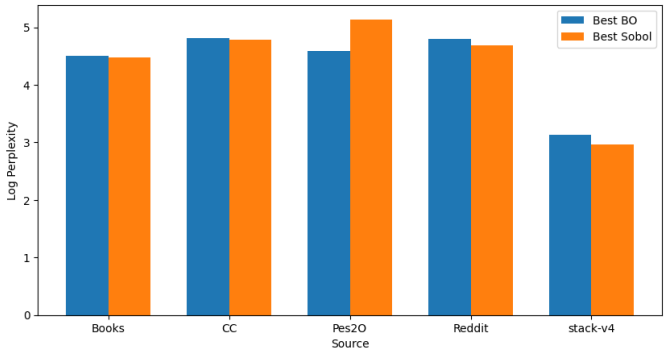


Fig. 5: Per domain performance comparison between the best sobol mixture and the best BO mixture.

F. Scaling Up Mixtures in Nontrivial. Re-Using Small-Scale Optimal Mixtures Does Not Generalize to Larger Models.

Model Size	Sobol	Uniform	BO
25M	4.4109	4.4659	4.3688
154M	3.0587	3.0066	3.0085

TABLE V: Scaling behavior of mixtures identified at a small 25M parameters model to a five times larger 154M parameters model.

The experiments thus far demonstrate that BO is an effective method for identifying well-performing mixture distributions for a given model architecture and dataset. However, because BO operates on a significantly smaller proxy model f' rather than the true model f, the next critical question is whether the mixtures identified on f' translate well to f when scaled up. Ye et al. [34] give some justification for this assumption, showing that when the number of training samples is held constant, the relative performance order achieved by a mixture on a small- and large model mostly remains the same. Table V makes a similar evaluation for our specific examples, by comparing the performance of Uniform-, Sobol-, and BO-optimized mixtures on a small model (25M parameters) and a large model (154M parameters). Unlike [34] we however, chosse the number of training samples at each model size Chinchilla optimal. While the BO mixture continues to outperform the Sobol mixture on the 154M model – even improving its relative advantage from 0.9 % to 1.6 % – the uniform mixture shows a surprising shift. Initially being the worst-performing mixture on the small model, the uniform approach becomes the best-performing one on the larger model.

These results suggest that the assumption of size-invariance, employed not only by us but also in approaches like DoReMi, does not universally hold. This means that a mixture that performs well on a small model does not necessarily maintain its performance advantage when scaled up to a larger model which is particularly noteworthy given that our experiments involved a relatively modest six-fold increase in model size.

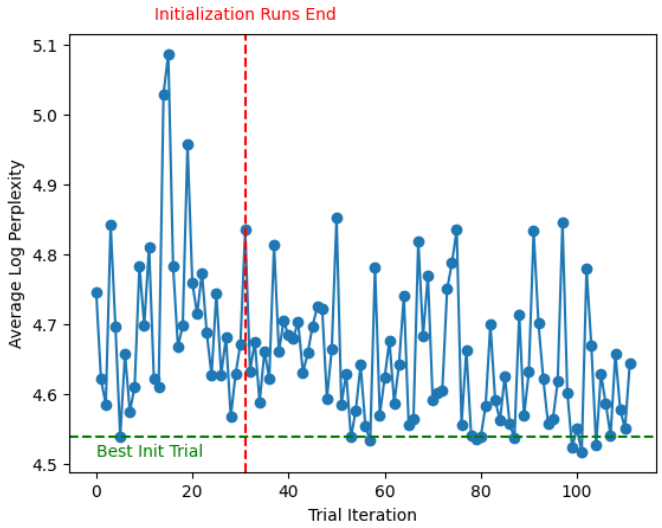


Fig. 6: BO with seven domains struggles to propose well-performing sampling locations for many trials, only identifying locations outperforming initialization after iteration 100.

In real-world, industry-grade settings, where model size can be several orders of magnitude larger, the discrepancy between small and large model performance is likely to be even more pronounced further amplifying the identified issue.

Some previous works on mixture laws have already identified this problem and proposed methods for addressing it. Importantly unlike the mixture laws employed in this work, these laws do not model how performance behaves as a function of mixture weight fixed to the same LM architecture. Rather, they model how the optimal mixture identified at one model-scale changes as a function of scale difference. Especially the work of Kang et al. [17] is interesting in this regard. They propose a mixture law for predicting the optimal mixture weights for a similar problem, i.e., the scaling of the optimal mixture over different sizes of datasets while, however, keeping the model size the same. They show that by fitting a law to multiple optimal mixtures at a small dataset scale, a scaling law can be formulated that well predicts the optimal mixture distribution for much larger datasets.

We believe that identifying a similar law to model simultaneous changes in model architectures and scale is crucial for further advances in data mixing based on proxy models and will be the key to effectively transferring insights gained from proxy-scale to large production models.

G. Scaling Up BO to more domains

While our experiments demonstrate the effectiveness of BO for five domains, real-world datasets commonly consist of more subdomains. Scaling up BO to more domains is non-trivial as every extra dimension exponentially increases the size of the search space. A large collection of work has therefore already investigated the challenge of *high-dimensional BO*. These works typically aim to minimize the problem of a growing search space by relying on simplifications, thereby reducing its effective size. TurBo [6], for example, improves scalability by focusing sampling only on a smaller trust region around the optimum identified in initialization This approach reduces complexity because instead of searching the whole space, only the trust region is searched for new optima. Of course, it is not guaranteed that the initialization-optimum and thus the trust region is located in a promising region. Therefore exploration is achieved by restarting the optimization multiple times, each focusing on a different initialization. Wang et al. [31], on the other hand, assumes that the search space can be reduced to a lower dimensionality via projection, thus reducing the complexity of the overall problem, i.e., $f(\boldsymbol{x}) = f(\boldsymbol{A}\boldsymbol{z}), z \in \mathbb{R}^d, d \ll D$). Finally a third line of works (e.g., Kandasamy et al. [16]) reduces complexity by assuming additive decomposability of the objective $f(\boldsymbol{x}) = f_1(\boldsymbol{x}_1) + f_1(\boldsymbol{x}_2)$, where each objective is fitted individually and $\boldsymbol{x}_1, \boldsymbol{x}_2$ are assumed to only depend on a subset of the original dimensions.

We demonstrate the challenges of scaling up vanilla BO to more domains in an experiment that utilizes seven domains, 32 initialization, and 112. trials in total. As shown in Fig. 6, BO struggles in outperforming the initialization, not being able to noticeably outperform the initialization up to the 100th trial. The best improvement is achieved at trial 102, reducing the perplexity from 4.5175 to 4.5398. While this improvement of ≈ 0.02 is larger than the variance bound identified before, it remains quite small, especially considering the fact that a significant number of iterations was required to achieve it.

Given the challenges associated with scaling up the number of domains, future work could focus on enhancing performance in high-domain settings by exploring specific high-dimensional optimization methods such as TurBo. These approaches have the potential to improve performance while minimizing the number of required samples, offering a more efficient solution for managing large, complex domain spaces in Bayesian data-mixing optimization.

IV. Conclusion

Our work investigated Bayesian optimization as an effective approach for identifying optimal data-mixture distributions on small proxy models. We demonstrated that, for a fixed model and data scale, our method outperforms several existing baselines while using the same number of training runs. However, we identified some limitations in our approach. While using average perplexity as the optimization criterion is straightforward, it can incentivize the model to prioritize performance on one domain at the expense of others, ultimately hindering the development of a well-generalizing model. Furthermore, scaling our method to accommodate many domains proved challenging due to the exponentially increasing search space and the known limitations of high-dimensional BO. While we did not explore high-dimensional BO methods such as TurBo [6], we believe this could be a promising direction for future research[1]. Additionally, we identified limitations in our

[1]TuRbo is already implemented in our testing framework, we just did not have the time to run sufficient experiments.

assumption that optimal mixtures are invariant to model size. Unfortunately, this limitation currently hinders the application of our approach to optimizing pre-training mixtures of very large models. Further work into the scaling behavior of optimal mixtures is required to fully leverage the insight won from proxy models to production grade systems.

REFERENCES

[1] Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. Efficient Online Data Mixing For Language Model Pre-Training, December 2023.

[2] Alon Albalak, Yanai Elazar, Sang Michael Xie, Shayne Longpre, Nathan Lambert, Xinyi Wang, Niklas Muennighoff, Bairu Hou, Liangming Pan, Haewon Jeong, Colin Raffel, Shiyu Chang, Tatsunori Hashimoto, and William Yang Wang. A Survey on Data Selection for Language Models, March 2024.

[3] Sebastian Ament, Samuel Daulton, David Eriksson, Maximilian Balandat, and Eytan Bakshy. Unexpected Improvements to Expected Improvement for Bayesian Optimization. *Advances in Neural Information Processing Systems*, 36:20577–20612, December 2023.

[4] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

[5] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. PaLM: Scaling Language Modeling with Pathways, October 2022.

[6] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable Global Optimization via Local Bayesian Optimization. volume 32. Curran Associates, Inc., 2019.

[7] Simin Fan, Matteo Pagliardini, and Martin Jaggi. DoGE: Domain Reweighting with Generalization Estimation, February 2024.

[8] Firas. Finding the Optimal Learning Rate using Bayesian Optimization - Tutorial — firasalhafez.com. https://firasalhafez.com/2021/05/12/finding-the-optimal-learning-rate-using-bayesian-optimization/. [Accessed 03-10-2024].

[9] Peter I. Frazier. A Tutorial on Bayesian Optimization, July 2018.

[10] 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, Alexandros G. Dimakis, Gabriel Ilharco, 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, March 2024.

[11] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The Pile: An 800GB Dataset of Diverse Text for Language Modeling, December 2020.

[12] Ce Ge, Zhijian Ma, Daoyuan Chen, Yaliang Li, and Bolin Ding. Data Mixing Made Efficient: A Bivariate Scaling Law for Language Model Pretraining, May 2024.

[13] Stewart Greenhill, Santu Rana, Sunil Gupta, Pratibha Vellanki, and Svetha Venkatesh. Bayesian optimization for adaptive experimental design: A review. *IEEE access*, 8:13937–13948, 2020.

[14] Suchin Gururangan, Mitchell Wortsman, Samir Yitzhak Gadre, Achal Dave, Maciej Kilian, Weijia Shi, Jean Mercat, Georgios Smyrnis, Gabriel Ilharco, Matt Jordan, Reinhard Heckel, Alex Dimakis, Ali Farhadi, Vaishaal Shankar, and Ludwig Schmidt. open_lm: a minimal but performative language modeling (lm) repository. https://github.com/mlfoundations/open_lm, 2023. GitHub repository.

[15] 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, March 2022.

[16] Kirthevasan Kandasamy, Jeff Schneider, and Barnabas Poczos. High Dimensional Bayesian Optimisation and Bandits via Additive Models. https://arxiv.org/abs/1503.01673v3, March 2015.

[17] Feiyang Kang, Yifan Sun, Bingbing Wen, Si Chen, Dawn Song, Rafid Mahmood, and Ruoxi Jia. AutoScale: Automatic Prediction of Compute-optimal Data Composition for Training LLMs, July 2024.

[18] 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, January 2020.

[19] Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, et al. The stack: 3 tb of permissively licensed source code. *arXiv preprint arXiv:2211.15533*, 2022.

[20] Jeffrey Larson, Matt Menickelly, and Stefan M. Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, May 2019. ISSN 0962-4929, 1474-0508. doi: 10.1017/S0962492919000060.

[21] Sijia Liu, Pin-Yu Chen, Bhavya Kailkhura, Gaoyuan Zhang, Alfred O. Hero III, and Pramod K. Varshney. A Primer on Zeroth-Order Optimization in Signal Processing and Machine Learning: Principals, Recent Advances, and Applications. *IEEE Signal Processing Magazine*, 37(5):43–54, September 2020. ISSN 1558-0792. doi: 10.1109/MSP.2020.3003837.

[22] David JC MacKay et al. Introduction to gaussian processes. *NATO ASI series F computer and systems sciences*, 168:133–166, 1998.

[23] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[24] Yonatan Oren, Shiori Sagawa, Tatsunori B. Hashimoto, and Percy Liang. Distributionally Robust Language Modeling, September 2019.

[25] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer school on machine learning*, pages 63–71. Springer, 2003.

[26] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian Optimization of Machine Learning Algorithms, August 2012.

[27] Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research, January 2024.

[28] 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, February 2023.

[29] Ryan Turner, David Eriksson, Michael McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian Optimization is Superior to Random Search for Machine Learning Hyperparameter Tuning: Analysis of the Black-Box Optimization Challenge 2020. In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, pages 3–26. PMLR, August 2021.

[30] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

[31] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. Bayesian Optimization in a Billion Dimensions via Random Embeddings, January 2016.

[32] Christopher Williams and Carl Rasmussen. Gaussian Processes for Regression. In *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995.

[33] 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, November 2023.

[34] Jiasheng Ye, Peiju Liu, Tianxiang Sun, Yunhua Zhou, Jun Zhan, and Xipeng Qiu. Data Mixing Laws: Optimizing Data Mixtures by Predicting Language Modeling Performance, March 2024.

APPENDIX

A. Bayesian Optimization

Optimizing functions of the form

$$\max_{\boldsymbol{x}\in\mathcal{A}} f(\boldsymbol{x}), \tag{6}$$

where $\boldsymbol{x}$ is the input to optimize contrained to a set $\mathcal{A}$, are an important problem in many scientific disciplines. In solving these problems it often helps that $f(\boldsymbol{x})$ is either analytically known or that at least gradient information are available to employ standard first- or second-order optimization algorithms (e.g., gradient descent).

The field of *zeroth-order optimization* asks the question of how to effectively optimize a black box function when neither analytical expressions nor gradient information are available and the only way to gain information about the objective function is by sampling it at different arguments $\boldsymbol{x}$ and observing (potentially noisy) observations

$$y_t = f\left(\boldsymbol{x_t}\right) + z_t,\, z \in \mathcal{N}\left(0, \sigma_n^2\right). \tag{7}$$

Grid search and random search are some of the simplest approaches towards zeroth-order optimization, however, suffer from being inefficient in the number of samples required to find a decently well optimum [29]. A second popular approach is to approximate the gradient of f via function evaluations and then apply typical first-order methods like SGD on top of these estimations [20, 21]. Again, estimating the gradient requires many function evaluations, and therefore this approach also only works well if evaluating the function of interest is cheap.

In many problems of interest, function evaluations are, however, not cheap. For example, when optimizing for the

optimal mixture of an LLM, each function evaluation consists of training a new model from scratch which quickly becomes very expensive, requiring a focus on sample efficiency within the optimization approach.

Bayesian optimization (BO) is one of the most sample-effective strategies for optimizing costly to evaluate, black box objective functions [29]. Sample efficiency is achieved by iterating building a probabilistic surrogate model of the function of interest based on previous function evaluations (e.g., using a Gaussian Process), and proposing new sampling locations based on an acquisition function that quantifies the expected value gain of sampling a specific location. The next section introduces how to build a probabilistic model based on the Gaussian Process and how to identify the next sampling location based on a acquisition function. The content of the next section is based on Frazier [9] and Snoek et al. [26], who both provide a deep introduction to BO and are highly recommended to the interested reader.

1) Gaussian Process as Probabilistic Surrogate Model: The goal of the surrogate model $\hat{f}$ in BO is to build a probabilistic representation of the function of interest f based on the knowledge gained from only observing previous sampling locations $\mathcal{B}_t = \{y_i, \boldsymbol{x}_i\}_{i=1}^t$. While different model classes could potentially fulfill this requirement (e.g., Parzen Tree Estimator [4]), the Gaussian process (GP) [22] has become the model of choice in most applications of BO.

Textbooks define the Gaussian Process as a "collection of random variables, any finite number of which have consistent Gaussian distributions" [25]. Simpler to understand, the GP provides a distribution over functions $\hat{f}_t \sim \mathcal{GP}_t(\boldsymbol{m}, \boldsymbol{K}_t)$ that allows predicting a gaussian posterior probability $p(f(\boldsymbol{x_0}) \mid \mathcal{B}_t) \sim \mathcal{N}(\mu_t(\boldsymbol{x}_0), \sigma_t^2(\boldsymbol{x}_0))$ of the function values $f(\boldsymbol{x}_0)$ at each $\boldsymbol{x_0} \in \mathcal{A}$.

m is the mean function of the GP that is often assumed to be 0. $\boldsymbol{K}_t$ is the covariance kernel matrix measuring the similarity between all combinations of $\boldsymbol{x}, \boldsymbol{x'} \in \mathcal{B}_t$ based on some underlying kernel function such as the squared exponential kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}) = \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^T \mathrm{diag}(\boldsymbol{\theta}^2)^{-1}(\boldsymbol{x} - \boldsymbol{x}')\right), \tag{8}$$

with the lenghtscale hyperparameter $\boldsymbol{\theta}$.

Finally, by defining $\boldsymbol{y}_t$ as the vector of all observations $y_i \in \mathcal{B}_t$ and $\boldsymbol{k}_t$ as the vector of all kernel value between $\boldsymbol{x}$ and $\boldsymbol{x}_i \in \mathcal{B}_t$, the closed form solution is tractable as:

$$\mu_t(\boldsymbol{x}) = \boldsymbol{k}_t^T(\boldsymbol{x})(\boldsymbol{K}_t + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{y}_t \tag{9}$$

$$\sigma_t(\boldsymbol{x})^2 = k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}_t^T(\boldsymbol{K}_t + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{k}_t(\boldsymbol{x}). \tag{10}$$

The hyperparameters such as $\boldsymbol{\theta}$ are typically decided by maximizing the likelihood of the observed data under these parameters (MLE), a process known as automatic relevancy determination [32]. Sometimes a prior on the parameter values is utilized as well (MAP) to incorporate any further knowledge.

2) Acquisition Function: Given the current surrogate model, the goal of the acquisition function is to propose the next sampling location that maximizes a utility value derived from the previous function evaluations according to some utility objective defined within the acquisition function. This process naturally is a trade-off between exploiting high-value regions and exploring regions with high uncertainty.

Expected Improvement (EI) is the most widely used acquisition function in BO. The fundamental idea behind EI is to balance exploration and exploitation by selecting the next sample point that maximizes the expected improvement over the best-observed value so far.

Mathematically, the expected for the current function $\hat{f}_t$ estimate and a proposal point $\boldsymbol{x}$ is:

$$EI(\hat{f}_t, \boldsymbol{x}) = \boldsymbol{E}_{y \sim \hat{f}_t(\boldsymbol{x})}\left[\max(0, y - y_t^+)\right] \tag{11}$$

where y_t^+ is the best function value observed so far.

Maximizing the acquisition function finally gives the next sampling location:

$$\boldsymbol{x}_{t+1} = \arg\max_{\boldsymbol{x}} EI(\hat{f}_t, \boldsymbol{x}). \tag{12}$$

It is important to acknowledge that maximizing the acquisition function itself can be challenging due to its potential non-convexity and the presence of multiple local optima. However, this problem is generally considered to be significantly simpler than the original optimization problem due to knowledge about the function shape and cheap sampling. In practice, the acquisition function is either maximized analytically – if possible – or using Monte Carlo [23] sampling.

B. Data Mixing

To achieve diverse knowledge, language models are typically pre-trained on datasets sourced from various domains, including general web data (e.g., Common Crawl), academic literature (e.g., arXiv), and programming code repositories (e.g., GitHub repositories). Due to limited computational resources, it is not feasible to utilize all available training data equally. Instead, careful decisions must be made to strategically design a data mixture that maximizes overall model performance. The Pile dataset [11], a common dataset with explicitly optimized mixtures, for example, consists of 24 % web data, 9 % Wikipedia, 4 % Code, etc. [19]. The importance of optimizing the data mixture has been demonstrated in numerous prior studies [33, 11, 34] and therefore approaches are necessary for optimizing data mixtures on first principles instead of relying on human-assumptions about the quality of each dataset for the learning process. This section provides a comprehensive overview of the field of data mixture optimization, the challenges, and different solution approaches.

1) Challenges of Data Mixing: The most important consideration in data mixing is ensuring that it does not significantly increase the computing requirements of training the final – typically very large – model that is intended to be used in practice. A simple way to guarantee compute efficiency is optimizing the mixtures *online* in parallel to training the

production model. *Offline* approaches, on the other hand, aim to identify optimal mixture distributions before training the production model. These offline methods typically achieve compute efficiency by utilizing smaller proxy models and making the assumption that mixtures performing well at a small model scale will generalize to the larger production model.

A second challenge lies in the difficulty of accurately measuring downstream task performance. Ideally, one would rely on established downstream benchmark suites (e.g., SuperGLUE [30]) to assess performance. However, the performance of small proxy models on these benchmarks is generally not good enough to gain sufficient insight. Therefore, perplexity on a held-out dataset is commonly employed as the optimization objective, assuming that language-modeling improvements will translate to downstream task improvements.

2) Offline Data Mixing: DoReMi [33] is one of the first principled approaches towards offline data mixing. DoReMi aims to achieve low loss across all domains by posing the mixture problem as one of worst-case robustness. However, as optimizing worst-case domain performance suffers from focusing too much on improving hard-to-learn domains, they instead optimize for the worst-case excess loss compared to a pre-trained reference model. The reference model (proxy size) is first trained on a pre-determined data mixture (e.g., uniform). Then the DoReMi model (also proxy size) is trained by updating the domain weights for the next train iteration based on the previously observed excess loss. After training the optimal domain mixtures are calculated as the average of all mixtures utilized throughout training.

Based on the objective criterion it is expected that DoReMi should be especially effective at guaranteeing all domains perform well with respect to the reference model, however, might not necessarily improve final average performance significantly averaged across domains. Experiments validated the first assumption and furthermore showed that by employing worst-case optimization even average performance was increased significantly. However, there are also points of criticism about DoReMi. First, while DoReMi comes at the advantage of only requiring the training of two proxy models, much fewer than the majority of approaches discussed in the following, it is limited by its dependence on the reference model, where manual decisions about the mixtures are still required. Additionally, recent work has shown that DoReMi weights do not generalize well and significantly degrade when changing model components such as the tokenizer [12]. Inspired by DoReMi, similar approaches have been developed recently that aim to overcome some of its limitations. Most notably DoGe [7] overcomes the dependence on the reference model by optimizing for gradient alignment instead of excess loss.

In addition to DoReMi-like models, a second family of mixture-law-based offline models has emerged recently. Inspired by the success of neural scaling laws [15, 18], these approaches seek to identify data-mixing laws that model perplexity as a function of the data mixture. The optimal data mixture is then identified by fitting the proposed laws to multiple evaluations of the proxy model and subsequently predicting the optimal mixture by minimizing the resulting approximation. *Data Mixing Laws* [34] is the seminal work in this field. Starting from two-domains, Ye et al. [34] first realized that performance on the i-th domain follows an exponential function dependent on the i-th domain weight:

$$L_i(w_i) = c_i + k_i \exp\left(t_i r_i\right), \tag{13}$$

with c_i, k_i, and r_i being learnable parameters that must be identified from data. They then further show that a similar law can be extended to M domains, with the per-domain validation loss written as

$$L_i(w_1, ..., w_M) = c_i + k_i \exp\left(\sum_{j=1}^{M} t_{ij} w_j\right). \tag{14}$$

Both c_i and t_{ij} can be understood to have practical meaning with c_i modeling irreducible loss of domain i and t_{ij} modeling the interaction between both domains. In experiments, they show that by fitting their mixture-law on a large number of small-scale model evaluations the true function shape can be evaluated well. Furthermore, they show that they are able to predict a mixture that compared to a baseline-mixtures enables the large model to reach the same perplexity in only 73 % of the steps.

An extension of data mixing laws is BiMiX [12]. Ge et al. [12] criticize that the law of Ye et al. [34] requires too many coefficients to fit, therefore necessitating training many proxy models. They instead propose a much simpler law predicting the i-th domain perplexity as

$$L_i(w_i) = \frac{C_i}{w_i^{\beta}}. \tag{15}$$

In total their proposed law reduces the number of parameters from $M^2 + 2M$ to just $2M$, thus significantly simplifying model fitting. However, while in the reported experiments BiMix performed well and was even capable of outperforming [34], its assumption of independent domains without modeling any interactions and synergies is likely to oversimplify real-world behavior.

All approaches so far are based on the assumption that optimal mixtures transfer from small to large model sizes, thus justifying the use of proxy models. Ye et al. [34] provide empirical support for this assumption, showing that the relative performance rankings among data mixtures remain consistent when scaling up model size while maintaining a fixed token count. In practice, however, when scaling model size, the token count typically increases proportionally (e.g., following the scaling laws proposed by [18, 15]). Addressing this limitation, Kang et al. [17] explore the impact of data scale on model performance, demonstrating that optimal mixtures are closely tied to the scale of the data. For smaller data scales, easily learnable domains with clear structure, such as Wikipedia, are advantageous, but as the token count increases, the optimal mixtures shift toward more information-rich, general domains

like Common Crawl (CC). Based on this insight, Kang et al. [17] ask, whether this behavior could be modeled by a new form of scaling law – one that predicts not perplexity as a function of mixture weight, but rather the optimal mixture distribution based on an initial optimal distribution and the difference in data scale. In their work, they demonstrate that such a law can indeed be formulated, allowing them to effectively model the shift in optimal mixtures across multiple magnitudes of scale.

The existence of such an identified law is crucial because it indicates that small proxy models can still be effectively utilized even when there are substantial differences in data scale between proxy and production model. The identified mixtures must just be scaled depending on the data-scale difference between training and inference.

3) Online Data Mixing: In addition to offline data mixing methods, there are also online methods that aim to dynamically adapt mixture distributions during the training of large production models. The motivation behind these methods is straightforward: they eliminate the need to rely on small proxy models, which, as previously discussed, may struggle to accurately predict optimal mixtures at larger scales.

Efficient Online Data Mixing [1] is among the most prominent works in the field of online data mixing. It addresses the challenge of optimizing data mixtures by framing it as a multi-armed bandit problem, where the decision of which domain to sample next is guided by the per-domain training loss as the reward function. Experiments validate the effectivity of the approach, showing that it is capable of outperforming DoReMi, as well the default weights of the Pile dataset, while only incurring negligible overhead at train time.

However, a key limitation of all online approaches, including this one, is that they do not produce a full mixture distribution over the dataset. Instead, they provide insight only into the next best domain to sample at each time step, which is less useful for gaining a comprehensive understanding of the overall contributions of the different datasets.

C. Neural Scaling Laws

Neural scaling laws provide a foundational understanding of how model performance improves as a function of increased resources, such as the amount of training data, model size, or compute power. These laws have been empirically validated across various domains of machine learning, particularly in the context of LLMs. By establishing predictable relationships between model performance and the scale of resources used, scaling laws serve as a crucial tool for guiding decisions about resource allocation during model development.

One of the earliest demonstrations of scaling laws in LLMs was provided by Kaplan et al. [18] which was later improved and extended by Hoffmann et al. [15]. Both works show that language modeling performance scales predictably with the number of parameters, the amount of data, and the compute used. Specifically, they found that for sufficiently large models, increasing the model size or the data size results in a logarithmic decrease in loss.

The relevance of scaling laws to this work lies in their implication that performance improvements observed in smaller proxy models can, in theory, generalize to larger models thereby motivating the idea of utilizing small proxy models for identifying promising mixtures which then must be just extrapolated properly.