Optimizing Pre-Training Data Mixtures with Mixtures of Data Expert Models

Lior Belenki[1], Alekh Agarwal[2], Tianze Shi[1], Kristina Toutanova[1]
[1]Google DeepMind, [2]Google Research,
Correspondence: belenkil@google.com, kristout@google.com

Abstract

We propose a method to optimize language model pre-training data mixtures through efficient approximation of the cross-entropy loss corresponding to each candidate mixture via a Mixture of Data Experts (MDE). We use this approximation as additional features in a regression model, trained from observations of model loss for a small number of mixtures.

Experiments with Transformer language models between 70M and 10B parameters on the SlimPajama dataset show that our method achieves significantly better performance than approaches that train regression models using only the mixture rates as input features. Combining our method with an objective that takes into account cross-entropy on end task data leads to superior performance on few-shot downstream evaluations. We also provide theoretical insights on why aggregation of data expert predictions can provide good approximations to model losses for data mixtures.

1 Introduction

Datasets used for pre-training language and multimodal models are often heterogeneous, with distinct sources having different quality, number of available documents, combination of modalities and styles, and relevance to end tasks of interest. Different data sources are often sampled at different rates during training, effectively up-weighting or down-weighting individual mixture components.

Prior work has shown that source sampling proportions have a large impact on model performance, both on cross-entropy of held-out examples from the training sources, and accuracy on downstream tasks (Hashimoto, 2021; Xie et al., 2023; Alayrac et al., 2022; Albalak et al., 2023, *inter alia*).

The sampling proportions of a data mixture with k source domains define $k-1$ real-valued hyper-parameters. It is infeasible to evaluate the performance of many mixtures for large language models trained on sequences of hundreds of billions of tokens and the largest models are typically trained only once with the best data mixture guess. The problem could be viewed as a bi-level optimization process which is known to be computationally challenging, both in the worst-case (Grüne and Wulf, 2024; Bolte et al., 2025), and in practice due to the difficulty of evaluating gradients, which require solving a non-convex minimization in the inner loop. In practice, most large-scale pre-training efforts rely on heuristics (Gao et al., 2020).

Approaches that optimize mixtures to improve generalization loss are based on proxy models, which are smaller in number of parameters and tokens seen than the target model of interest. Based on proxy models, data mixtures can be optimized through an online algorithm (Fan et al., 2024; Xie et al., 2023), or offline, through observing the generalization loss of multiple trained proxy models, and predicting the loss of other mixtures through regression models. Mixtures are optimized to minimize loss according to the trained regressors (Liu et al., 2025; Ye et al., 2024; Ge et al., 2024).

Regression models observe the generalization losses $s(\lambda_1), \ldots, s(\lambda_N)$ achieved by proxy language models $\theta_1, \ldots, \theta_N$, trained with the the corresponding data mixtures $\lambda_1, \ldots, \lambda_N$. Their goal is to predict the generalization loss for unseen mixtures λ, without training proxy models for those new mixtures. Regression-based methods are simpler to implement, as they do not require changes in the LM training algorithm. They also have the advantage that the same set of trained proxy language models can be used to optimize data mixtures for multiple loss criteria. On the other hand, these approaches require an up-front cost of training multiple (usually 30 to 500) proxy models θ_n.

We show how such regression models can be significantly improved through the use of a new Mixture of Data Experts approximation (MDE). MDE is a simple predictor which requires the training of only k proxy models, where k is the number of source domains. Each of these models (termed *data experts*) is trained on data from a single domain D_i. Using these expert models, for each candi-

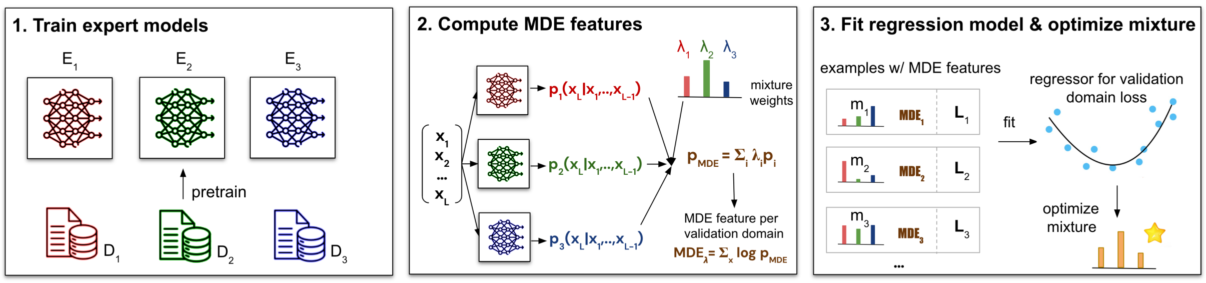


Figure 1: Illustration of our approach. Data experts E_i are trained from individual pre-training mixture domains D_i. The per-token p_{MDE} approximations are generated as a λ-weighted average of the probabilities predicted by the individual experts. Then, for each validation domain, the *MDE feature* is computed as the average of log-probability under p_{MDE} across its tokens. Lastly, the mixture weights λ and the MDE features are used to fit a regression model that maps λ to predicted validation losses. The optimal set of weights are found by optimizing an objective function based on the regression model.

date λ, we define the predictor $f_{\mathrm{MDE}}(\lambda)$ as the loss obtained by an ensemble model over the experts using mixture weights λ. Figure 1 illustrates the method. We define generalization losses for mixtures through aggregation of cross-entropy loss on multiple validation domains. MDE can be used on its own or as a source of features in regression models (one feature value for each validation domain).

A simple theoretical analysis justifies the aggregation of predictions from data experts to approximate the outcome of actually training a language model with data mixture weights λ and identifies directions to improve upon MDE (see Section 3.3).

Our results indicate that the MDE approximation leads to substantial improvement in mixture ranking quality across multiple regression models. We evaluate the contribution to linear models, gradient boosting machines (GBM), and multi-task Gaussian process models (MTGP). Ranking is improved across all regression models (e.g. Spearman's correlation improved from **0.65 to 0.95** for linear regressors, and **0.81 to 0.95** for GBM). MDE can also be used to optimize data mixtures on its own, thus requiring the training of only k proxy models to achieve performance comparable to regressors from prior work at **3x less computational cost**.

We experiment with Transformer decoder-only language models with 70M, 150M, 280M, 510M, 1B, and 10B parameters, using the SlimPajama (Soboleva et al., 2023) dataset, and training models for up to 100B tokens. We show that mixture rates selected based on a regression model trained from 25 examples of validation losses from 280M-sized proxy models trained to 5B tokens, lead to better generalization losses for 1B and 10B models trained on 100B tokens, compared to the mixture rates optimized for the same dataset by baselines including DoGE (Fan et al., 2024) and DoReMi (Xie et al., 2023).

We further define a generalization loss on SlimPajama validation domains and task-relevant validation examples and optimize mixture weights based on this loss, showing that the resulting mixtures outperform heuristic baselines and prior data mixture optimization methods on average few-shot downstream task accuracies over a suite of generation and ranking tasks for both 1B and 10B parameter models.

2 Related work

There is an extensive body of work on data selection and mixture optimization for pretraining language models. Albalak et al. (2024) offer a comprehensive recent survey. Approaches for data selection and cleaning consider different granularities of data, such as individual tokens, document samples, and groups of multiple documents, often derived from meta-data such as the web domain (like Wikipedia) or source collection (such as C4).

Closest to our focus is work selecting or sampling data at the level of large sample groups, often termed domains. Data mixture sampling rates can be static over the course of model training, or dynamic (Albalak et al., 2023; Piergiovanni et al., 2023), forming a curriculum over sampling rates which could for example facilitate faster progress through learning easier skills first. We focus on static mixtures.

Online optimization of domain mixture rates through proxies DoGE (Fan et al., 2024) presents an efficient method to optimize data mixture rates through a first-order bi-level optimization approach, with cost 2x the cost of training a single proxy model. Our approach is simpler to implement as it does not require changes in the LM training algorithm, and also offers the possibility to derive optimal weights for a set of different criteria while reusing the same proxy models. DoReMi (Xie et al., 2023) also proposes an online method with similar computational requirements to those of DoGE. We compare to these methods in Section 4.

Regression-based optimization through proxies Multiple methods that fit regression models to predict the performance of unseen mixtures have been devised. Some make predictions based on the domain mixture rates as features RegMix (Liu et al., 2025), while others additionally extrapolate across number of tokens BiMix (Ge et al., 2024), or both token and model size scaling parameters DML (Ye et al., 2024). Our work is most similar to RegMix, in that we approximate the rankings of full-sized models through extrapolation from smaller proxy model, assuming that data mixture rankings at different scales are sufficiently similar. We compare our methods to the regressors used in these works, showing the value of the MDE features across multiple regression model parametric families.

Generalization losses Xie et al. (2023) and Fan et al. (2024) optimized toward aggregate losses defined from training domains, while Liu et al. (2025) and Ge et al. (2024) optimize toward single domains. We propose to define the generalization loss to optimize as an aggregate over both training domain heldout data, and validation examples from end tasks. We analyze the correlation between different losses and downstream task generation/ranking performance.

Approximation to models trained on data mixtures Na et al. (2024) explored approximating models trained on combined datasets by averaging the parameters of independently trained models. This method was applied to continuously pretrained models with limited source configurations, leveraging parameter-averaging effectiveness within the same loss basin (Wortsman et al., 2022; Neyshabur et al., 2020). While effective for models fine-tuned from the same strong starting point, it presents challenges for pretraining from scratch, as our experiments in Appendix C.7 confirm.

3 Method

Task Definition We consider a data corpus consisting of data from k training domains $D_1, \ldots, D_k$. Data mixture proportions (weights) λ define a distribution over text sequences x:

$$D_\lambda(x) = \sum_{i=1}^{k} \lambda_i \mathbf{unif}(D_i)$$

This sampling distribution is used to train a language model. The training loss for mixture rates λ and parameters θ is $L(\theta, \lambda) = -E_{x \sim D_\lambda} \ln p(x|\theta)$. The trained parameters for weights λ approximate:

$$\theta^*_\lambda = \mathbf{argmin}_\theta L(\theta, \lambda)$$

At a high level, our task is to find mixture rates λ, for which the corresponding trained model θ^*_λ has optimal generalization performance. In this work we assume that we are given a set of validation datasets $V_1, \ldots, V_m$, and a validation set loss aggregator g, such that we define generalization performance as the score:

$$s(\theta) = g(L(\theta, V_1), \ldots, L(\theta, V_m)),$$

where the aggregator function takes as arguments the cross-entropy losses of model θ on all validation domains. g can be a simple unweighted average, or a more complex function. Our task is then:

Find λ, such that the estimated generalization loss $s(\lambda)$ defined as the loss of the trained parameters corresponding to these mixture proportions $s(\lambda) = s(\theta^*_\lambda)$, is minimized.

Note that in practice one might want to optimize model decoding performance rather than cross-entropy losses. While e.g. a sigmoid of cross-entropy would provide a better fit for decoding task accuracy (Llama3-Team (2024)), here we focus on simple weighted average loss aggregators.

Proxy language models We follow prior work and use proxy language models (Xie et al., 2023) to estimate the effect of different mixture proportions on LLM generalization performance. The proxy models can be significantly smaller than the full-scale size, and can be trained over a much shorter token horizon. Here we use LMs of size 280M trained for 10K steps (5B tokens) as proxies for learning to rank data mixtures for up to 10B-sized models trained for 200K steps (100B tokens). We consider additional proxy configurations for analysis. Appendix B.2 details the exact number of parameters of all proxy model configurations.

3.1 Mixture of Data Experts approximation

Our Mixture of Data Experts (MDE) approximation provides an estimate $\hat{s}(\lambda)$ at the cost of training only k (number of training domains) language models and computing the cross-entropy loss with these models on samples from each of the m validation domains. The k trained data expert models $\theta_1^*, \ldots, \theta_k^*$ are trained on the individual domains: $\theta_i^* = \mathbf{argmin}_\theta - E_{x \sim \mathbf{unif}(D_i)} \ln p(x|\theta)$.

Given these data experts, for every candidate mixture $\lambda = (\lambda_1, \ldots, \lambda_k)$, we form the following ensemble language model (termed MDE), with a next token distribution defined as:

$$P_{\mathbf{MDE}}(x_t|x_{1 \cdots t-1}, \lambda) := \sum_{i=1}^{k} \lambda_i P_{\theta_i^*}(x_t|x_{1 \cdots t-1})$$

Given this ensemble language model, we can compute cross-entropy losses on each of the validation domain datasets $L(P_{\mathbf{MDE}}(\lambda), V_j)$ and aggregate these estimates according to g. Here we omit the dependence on the trained data expert model parameters for brevity.

We then arrive to our MDE approximation estimate of the generalization performance corresponding to candidate mixture λ, $s_{\mathbf{MDE}}(\lambda)$, as: $g(L(P_{\mathbf{MDE}}(\lambda), V_1), \ldots, L(P_{\mathbf{MDE}}(\lambda), V_m))$.

Efficient implementation To compute the MDE generalization estimate for each mixture λ, we do not need to run neural network inference. Instead, we can pre-compute and cache the per-token probabilities for all tokens $x_j, j = 1, \ldots, |V_j|$ in datasets V_j, according to each of the experts θ_i^*. The probability of token x_j according to expert θ_i^* is $P(x_j|x_1, \ldots, x_{j-1}, \theta_i^*)$. We can then compute the MDE estimates for each λ on CPU, while performing only $O(k)$ operations over each token to compute a weighted sum and logarithm of the per-token probabilities. Since validation sets are usually much smaller than training sets and we don't require neural network inference, the cost is negligible in practice (also see Appendix B).

3.2 Regression models

The MDE approximation provides one estimate of the generalization losses for each mixture. We additionally build on prior work that learns estimates through regression models, based on observations of mixture weights and corresponding losses. To create training examples for the regression models, we sample mixtures λ_n, train corresponding proxy models θ_n, and obtain loss measurements for each of the validation domains through LM inference. Appendix B.4 details how mixtures were sampled.

For fixed model/data scale, prior work considers only the mixture rates λ as input features for such regressors. Here, we study the value of the MDE approximation as additional source of features. We consider linear models, gradient boosting, and multi-task Gaussian process (MTGP; Bonilla et al., 2007). For an arbitrary mixture, we predict validation losses by first computing the MDE per-domain loss approximations and then inputting them to the regression model to get the prediction $\hat{L}_j(\lambda)$ for the validation loss on domain V_j corresponding to data mixture λ:

$$L^j_{\mathbf{MDE}} = L(P_{\mathbf{MDE}}(\lambda), V_j), \forall j \in 1, \ldots, m$$
$$\hat{L}_j(\lambda) = \mathrm{M}_j(\underset{\substack{\swarrow \\ \text{features used in prior work}}}{\lambda};\ \underbrace{L^1_{\mathbf{MDE}}, \ldots, L^m_{\mathbf{MDE}}}_{\text{features introduced by this work}}),$$

where M_j denotes some regression model. Note that MDE features approximating the loss on other domains $V_{j'}$ are also used when predicting the loss on domain V_j.

In the experiments section, we evaluate the contribution of MDE features to multiple regressors.

Finding optimal mixtures To find the optimal mixture we first define the optimization objective $s(\lambda)$. For given λ, the value $s(\lambda)$ is computed through aggregating loss predictions on each of the validation domains V_j. We experiment with the average validation loss of pretraining domains as in Fan et al. (2024) and other variants that use end task validation domains. We use the Vizier framework (Song et al., 2024) to perform the optimization. We define the search space as k non-negative parameters corresponding to the mixture component weights and later normalize them to a valid probability distribution. The framework is general and does not require differentiability of the objective.

3.3 Theoretical justification of MDE

Let us assume that each example in the pre-training dataset contains a prefix x followed by token y. Thus, each component in pre-training data mixture can be described in terms of a distribution $D_{i,x}$ over the prefixes and $D_{i,y}$ over the following token.

We now give our main theoretical result relating the minimizer of $L(p, \lambda)$ with the MDE approximation.

Proposition 3.1. *For any λ in the $k-1$-simplex, let $p^\star_\lambda = \arg\min_{p \in \mathcal{P}} L(p, \lambda)$ be the minimizer of the λ-weighted loss over all probability distributions. Then we have for any (x, y):*

$$p^\star_\lambda(y|x) = \sum_{i=1}^{k} \lambda'_i(x) p^\star_i(y|x),$$

where we use the shorthand $p_i^\star$ for the minimizer of $L(p, D_i)$, the expected loss on domain i. The coefficients λ_i' satisfy: $\lambda_i'(x) \propto D_i(x)\lambda_i$. In particular, we have $\lambda_i' \propto \lambda_i p_i$, whenever $D_i(x) \equiv p_i$ for any x such that $D_i(x) > 0$, for each domain i.

We prove the proposition in Appendix A. In words, the result says that the distribution which minimizes the pre-training loss for the λ-weighted mixture can be expressed as a weighted combination of the data experts trained on the individual domains. In the simplest case where the domains only differ in the conditional distributions $D_i(y|x)$ and $D_i(x) = D_j(x)$ for all i, j, these coefficients are further equal to λ_i, since $p_i = p_j$ for all ij in this case. This matches our MDE approximation in the most ideal scenario. When the p_i are not all identical, but $D_i(x)$ is still uniform over its support, then the optimal mixture coefficients λ' are still independent of x, and hence can potentially be captured by the regression methods we use in this work. In a general setting, the coefficients in this linear combination have an x dependent relationship with respect to λ. This suggests that more flexible approximations that induce the mixture weights as a function of the token prefix might yield even better estimates of validation loss. We do not pursue these approaches here due to the relative simplicity and efficiency of the MDE approximation.

4 Experiments

We perform two groups of experiments to assess the contribution of MDE to the quality of data mixture loss prediction and ranking, and study the downstream task performance of data mixtures optimized according to different criteria.

4.1 Datasets

We overview the datasets used for language model training, validation domains for generalization loss estimations, and few-shot downstream tasks.

Language model training datasets We train Transformer language models on the SlimPajama dataset (Soboleva et al., 2023), treating the seven top-level domains as different sources for training dataset mixtures. We split the documents into segments of at most 1024 tokens according to the Gemma (Gemma-Team, 2024) text-only SentencePiece (Kudo and Richardson, 2018) tokenizer with a vocabulary size of 256,000 tokens.

Validation domain datasets We use samples from the development subsets of the SlimPajama dataset as one source of validation domains for generalization loss estimation. We term these SP validation domains. Additionally, we use ARC (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), and MultiRC (Khashabi et al., 2018), covering question answering, commonsense reasoning, and reading comprehension, as validation sets for generalization loss estimation. ARC has two subsets, Easy and Challenge, which we refer to as ARC-E and ARC-C respectively. We use separate downstream tasks for validation and final evaluation to prevent overfitting. There are a total of 11 validation domains from end tasks,[1] which we term ET (from end task) validation domains. The loss on each of these ET domains as defined through the next-token probabilities from the language model, considering the concatenation of each prompt and gold response as a single sequence. The number of tokens per domain is in Appendix B.7.

Downstream evaluation datasets and settings We evaluate models on a test suite of 10 downstream tasks. For *generation*, we use TriviaQA (Joshi et al., 2017), NaturalQuestions (NQ; Kwiatkowski et al., 2019), WebQuestions (WQ; Berant et al., 2013), SQuAD 2.0 (Rajpurkar et al., 2018), and LAMBADA (Paperno et al., 2016), covering question answering, reading comprehension and word prediction tasks. For *ranking* (multiple-choice question) tasks, we use COPA (Roemmele et al., 2011), PIQA (Bisk et al., 2020), WiC (Pilehvar and Camacho-Collados, 2019), WinoGrande (Sakaguchi et al., 2021), and HellaSwag (Zellers et al., 2019) spanning across question answering and commonsense reasoning. We prepare all the above tasks in 0-, 1-, and 5-shot formats, and report exact match (EM) accuracies for generation tasks and standard accuracies for ranking tasks.

4.2 Benchmarked regression models

We consider baselines and methods from prior work, including:

- **Empirical Mean (baseline)**: Average loss per domain for any mixture.
- **DML** (Ye et al., 2024): Predicts mixture loss with $L_i(\lambda_{1..k}) = c_i + k_i \exp(\sum_{j=1}^{k} t_{ij}\lambda_j)$.

[1]We prepare the 4 end tasks in 0-shot, 1-shot, and 5-shot formats and treat each task-format combination as a domain. We discard 5-shot MultiRC because texts are often too long to fit into the 1024 token segment size, resulting in 11 domains.

- **BiMix** (Ge et al., 2024): Models validation loss using data quantity and mixing weight, and given a fixed quantity the formula is $L_i(\lambda_i) = \frac{A_i}{\lambda_i^{\alpha_i}}$.
- **Gradient Boosting** (GBM-RegMix; Liu et al., 2025): Uses ensembles of regression trees to predict mixture losses.
- **Linear Model** (Liu et al., 2025): Predicts losses via regularized weighted sum of features.

and our models, including:

- **MDE**: Predicts losses directly with Mixture of Data Experts.
- **MTGP**: Uses Multi-task Gaussian Process regressors.
- **X-MDE**: Denotes any model *X* that uses mixture weights and MDE as features.

See Appendix B.6 for details on hyperparameters and software packages used.

4.3 Results on validation loss prediction

We begin with experiments predicting losses for new mixture proportions λ, given a training set of models θ_{λ_n} corresponding to a set of sampled mixture proportions $\lambda_1, \ldots, \lambda_N$. Appendix B.4 details how the mixture examples were sampled and Appendix B.2 reports on language model sizes and training configurations.

Extrapolation to mixtures of the same scale

In the first set of experiments, we aim to assess the ability of different methods to predict validation losses and loss aggregates for new mixtures λ, given a training set of measurements for models of the same size and number of training steps.

We look at per-validation domain performance, as well as the performance corresponding to multiple loss aggregators — AVG-SP: Average loss on the seven SlimPajama validation datasets,which has been a common optimization target used by baselines including DoGe and DML; AVG-ET: Average loss on the eleven validation end task domains detailed in Section 4.1; AVG-ET+SP: Average loss across all 18 validation domains – the union of SlimPajama and end task validation datasets.

We evaluate regression methods using squared error between predicted and true loss values, along with Spearman's rank correlation. A training set of 25 mixtures and a test set of 48 distinct mixtures, each with 280M-sized models trained for 10K steps (5B tokens), are used for comparison. Table 1 reports mean squared error and Spearman's rank correlation for AVG-SP and AVG-ET+SP aggregated losses. Figure 2 shows squared error for each individual SlimPajama validation domain. The reported results are averages from 5 training runs for each method, with a different sampled training set of mixtures for each run.

For the regressors using MDE features, we denote with e.g. MTGP-MDE-SP models that use the MDE features only from the 7 SP domains, and also predict the losses only on those domains. In Table 1, the regressors using MDE use only the SP domain features for the results in the first two columns, and all 18 MDE features for the results in the second two columns.

We note that: (*i*) As a standalone predictor, MDE performs no better than the empirical mean baseline in loss prediction for AVG-SP, while substantially outperforming that baseline for AVG-ET+SP. (*ii*) As a standalone ranker, MDE's performance is very respectable and close to that of the best regressors which use 3x more trained proxy models. (*iii*) MDE as a source of features brings large improvements in MSE and Spearman's, across multiple regression model families (Linear, MTGP, GBM), (e.g. improvement from 0.65 to 0.95 for linear regressors), substantially improving over prior state-of-the-art regressors, while using equivalent computational resources. Note that while we only report the mean and not the confidence intervals for each predictor in the table, we verified the gains are statistically significant. In Appendix C.7 we consider alternate ways to approximate data mixture losses using trained experts, showing MDE achieves superior performance.

Extrapolating to larger scale models

Our ultimate goal is to compare and optimize mixtures according to the performance corresponding to the largest models, trained for a maximum token budget (here 10B models and 100B tokens). While REGMIX (Liu et al., 2025) found that very small models trained over relatively few tokens (1M model size and 1B tokens) are sufficient as proxies for learning to rank much more scaled versions, we find that the approximation quality is dependent on the choice of generalization loss estimate we aim to optimize. To understand this, we train proxy models of different sizes corresponding to the same set of 55 data mixtures λ. The proxies are of sizes 70M, 150M, 280M, and 510M, and are trained to a token horizon of up to 50K steps (25B tokens). We then see whether the ranking of the mixtures at the largest configuration (510M, 50K

	MSE SP (↓)	ρ SP (↑)	MSE ET+SP (↓)	ρ ET+SP (↑)
Emp. mean	0.01151	N/A	0.01250	N/A
LINEAR	0.01637	0.23426	0.00655	0.64618
MTGP	0.00460	0.85829	0.00231	0.89911
BIMIX (Ge et al., 2024)	0.00327	0.86051	N/A	N/A
DML (Ye et al., 2024)	0.00296	0.91991	0.00116	0.89188
GBMREGMIX (Liu et al., 2025)	0.00242	0.92256	0.00431	0.81442
MDE (ours)	0.02809	0.91222	0.00391	0.88571
GBM+MDE (ours)	0.00140	0.94963	0.00089	**0.95462**
LINEAR+MDE (ours)	**0.00050**	0.97555	**0.00048**	0.95274
MTGP+MDE (ours)	0.00053	**0.98383**	0.00116	0.93469

Table 1: Mean squared error (MSE) and Spearman's rank correlation (ρ) on prediction of averaged loss over SlimPajama domains only (SP) and all (ET+SP) validation domains, using different regressors from prior work, and ones proposed in this work. Regressors are fitted using 25 train mixtures (except MDE that uses only 7 train mixtures), and evaluated with 48 held-out mixtures. MDE features bring large improvements across regressors.

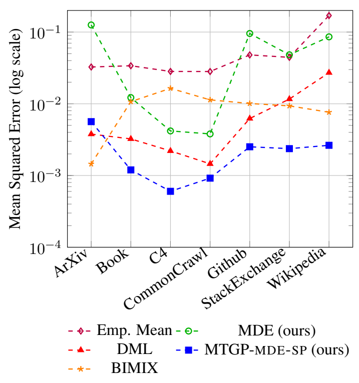


Figure 2: Per-domain mean loss squared error for SlimPajama validation domains.

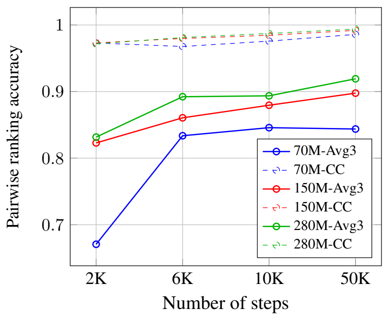


Figure 3: Pairwise ranking accuracy of 55 data mixtures (510M models trained to 50K steps) based on proxies of different size and number of training steps.

steps) can be predicted through the true losses of proxies of different scales for the same mixtures.[2]

Figure 3 shows that, as REGMIX observed, ranking according to a single training domain, SlimPajama CommonCrawl, is well predicted by all proxy models, with a small difference between 70M and 280M models and a small improvement with the number of training steps (dashed lines). On the other hand, for a harder ranking metric, which requires mixtures to be ordered correctly simultaneously according to the three aggregate losses AVG-SP, AVG-ET, and AVG-ET+SP, 70M models and ones trained to less than 6K steps are substantially less accurate proxies. We thus choose to use 280M models trained to 10K steps as proxies for optimizing scaled models, as a tradeoff between accuracy and efficiency.

Impact of number of training mixtures

We analyze how ranking performance scales with the number of training mixtures. Figure 4 illustrates the learning curve for Spearman's rank correlation of the average loss (AVG-SP). Sets of 280M-parameter proxy models, trained for 10K steps are used to predict the ranking order of the average domain loss for larger 510M-parameter models from unseen data mixtures, trained for 50K steps.

In the low-data regime, MDE consistently outperforms all other models. However, as more training examples become available, MTGP-MDE-SP and LINEAR-MDE-SP steadily improve, eventually surpassing MDE to achieve the best performance. We observe diminishing returns beyond 25 training examples, suggesting a saturation point in the benefits of additional data.

4.4 Correlation between validation loss and downstream task accuracy

Section 4.3 shows that our methods produce more accurate validation loss prediction results than prior methods. Can such improvement help guide us towards finding mixture weights that improve on downstream evaluations? In downstream tasks, models are usually evaluated based on generation or ranking accuracies instead of cross-entropy loss. Additionally, capable models should generalize be-

[2]Note that our model sizes are total parameters and the number of non-embedding ones is smaller, e.g. 2.6M for the 70M model and 85M for the 280M model, see Appendix B.2.

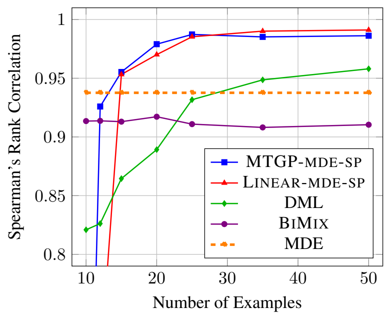


Figure 4: Spearman's rank correlation of SP validation domains as a function of number of training mixtures.

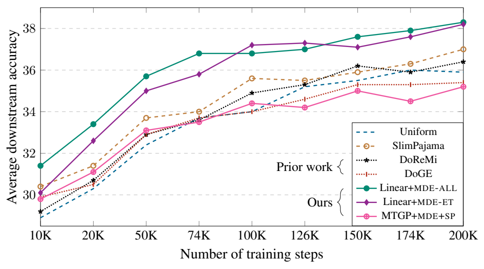


Figure 5: Downstream task accuracy (average over 0-shot,1-shot, and 5-shot formulations over a suite of generation and ranking tasks) for 1B models optimized through our methods using MDE versus prior work.

VAL. TASK	VAL. TASKS SELF	VAL. TASKS AVG.	DOWNSTREAM TASKS GEN.	DOWNSTREAM TASKS RANK.	DOWNSTREAM TASKS ALL
ARC-C	0.452	0.771	0.613	0.846	0.845
ARC-E	0.903	0.761	0.608	0.845	0.840
OPENBOOKQA	0.862	0.785	0.626	0.840	0.846
MULTIRC	0.245	0.698	0.653	0.728	0.796
AVERAGE-ET	—	0.772	0.630	0.833	0.844
AVERAGE-SP	—	0.320	0.189	0.330	0.282

Table 2: Spearman's rank correlation between validation tasks' loss and accuracy metrics, considering the same task (SELF), the average accuracy across all validation end tasks (AVG.), and metrics for downstream test tasks: average on the generation (GEN.), ranking (RANK.), and all test tasks (ALL).

yond tasks seen during development and should perform well on unseen tasks. To understand the potential impact of the choice of using AVG-ET as a mixture weight optimization objective, we conduct a study comparing end task validation loss and test accuracies using 510M parameter models trained up to 50K steps. As observed in Table 2, there is a strong correlation[3] between validation tasks' language modeling loss and model performance on the downstream test tasks. In contrast, the average SP domains' validation losses (last row) show much lower correlation with end task evaluation results.

4.5 Results with optimized data mixtures

Based on our study with models in the range of 70M to 510M parameters, we choose to optimize training mixtures using well-performing regressors for each objective, from 280M-sized proxies. We optimize mixtures for three different criteria: (*i*) AVG-SP the average loss on SlimPajama domains, (*ii*) AVG-ET, the average loss on end task validation domains, and (*iii*) AVG-SP + AVG-ET, also called AVG-ALL, the sum of the two averages. Much prior work has focused on optimizing AVG-SP or the loss on a single domain. Section 4.4 shows AVG-ET correlates better with downstream accuracy, though a small set of validation tasks may not be sufficient to cover all requisite skills for LM generalization. Thus, we consider the combination of the unsupervised loss (AVG-SP) with the end-task aware loss (AVG-ET).

To optimize the mixtures, we trained regressors using 25 mixture examples (including the 7 experts), each of size 280M trained to 10K steps. The optimized models for the three criteria are denoted as MTGP-MDE-SP, LINEAR-MDE-ET, and LINEAR-MDE-ALL. Their corresponding mixture weights are given in Appendix C.

Table 4 shows that the mixture optimized with MTGP-MDE for AVG-SP loss leads to a 1B model that achieves the best AVG-SP generalization loss compared to prior work that optimized the same loss (DOGE and DML), and other baselines. In that table and other comparisons in this section, we use the mixture weights optimized in prior work directly from the corresponding papers, and train 1B models with those weights for comparison. For DOGE and DOREMI, we used the mixture weights reported in Fan et al. (2024), optimized from their 124M proxies which are similar in scale to our 280M proxies in the number of non-embedding parameters. We note that differences in tokenization and other hyper-parameters could have results in different optimized weights if we had applied the prior work's methods on our data to derive the mixture weights.

In Table 5, we additionally include models optimized for the losses of the two other end-task related criteria.[4] Our approaches lead to successful optimization of the desired generalization losses for the 1B models. In appendix C.5 we see that

[3]Correlations are negative because a lower language modeling loss typically corresponds to better end task evaluation results. In Table 2, we show the absolute values for readability.

[4]These optimized weights included 0 values for some domains and we smoothed the solutions $\hat{\lambda} = .99\lambda_{\text{opt}} + .01\mathbf{uniform}$.

	Model	Generation Tasks					Ranking Tasks					Average (↑)
		WQ	NQ	SQuAD	TriviaQA	LAMBADA	COPA	PIQA	WIC	WinoGrande	HellaSwag	
Baselines	Uniform	4.4	2.4	35.8	10.9	21.9	70.0	67.9	49.1	54.2	42.3	35.9
	SlimPajama	**6.9**	**3.9**	37.0	15.7	18.9	71.3	67.2	49.5	54.7	45.3	37.0
	DoGE (124M)	5.2	2.3	33.5	10.0	19.9	70.0	68.4	48.1	54.0	43.1	35.4
	DoReMi (124M)	5.1	2.8	37.1	13.6	21.7	71.7	66.4	48.8	54.5	42.3	36.4
	DML	4.1	1.9	34.4	9.1	15.5	71.7	68.1	**50.9**	54.0	42.9	35.3
Ours	MTGP-mde-sp	4.0	2.4	34.2	9.4	19.6	68.7	67.6	50.4	52.8	43.0	35.2
	Linear-mde-et	6.4	3.5	34.7	**17.6**	22.1	**75.3**	69.3	49.3	**55.7**	47.6	38.2
	Linear-mde-all	6.1	3.1	**37.2**	14.7	**24.1**	73.3	**70.4**	50.4	**55.7**	**47.7**	**38.3**

Table 3: Downstream model performance on 5 prediction tasks and 5 ranking tasks. Results are averaged across 0-shot, 1-shot, and 5-shot performances. For generation tasks, we report exact match (EM) accuracies (%), and for ranking tasks, we report accuracies (%). All models are 1B parameter models trained for 200K steps.

	Uniform	SlimPajama	DoGE	DoReMi	DML	MTGP-mde-sp
ArXiv	4.90	5.41	5.06	5.45	4.64	5.13
Book	14.77	15.28	15.76	15.44	15.40	15.12
C4	17.62	15.72	16.41	17.08	17.00	16.93
CommCrawl	14.43	12.52	13.78	13.22	14.52	14.25
Github	2.59	2.89	2.71	2.77	2.58	2.64
StackExch.	5.33	6.12	5.26	5.65	5.33	5.33
Wikipedia	8.89	10.67	8.70	8.09	13.02	8.24
Average	8.085	8.449	8.072	8.156	8.482	**8.038**

Table 4: Generalization on validation SP domains for 1B parameter models trained for 100B tokens with mixtures optimized according to different methods over the SP domains. We compare Baselines (uniform and proportional to size), DoGE(124M), DoReMI (124M), to the mixture derived by MTGP-MDE-SP. Per-domain and average (exponentiatated average loss) perplexity.

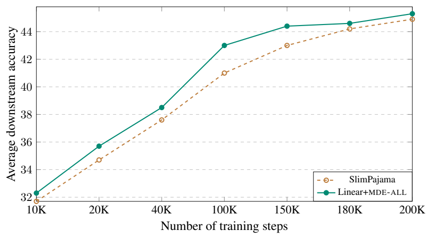


Figure 6: Downstream task accuracy (average over 0-shot,1-shot, and 5-shot formulations over a suite of generation and ranking tasks) for 10B models optimized through our method versus the strongest baseline.

	Uniform	SlimPJ	DoGE	DoReMI	DML	MTGP-mde-sp (ours)	Lin-mde-et (ours)	Lin-mde-all (ours)
ARC-c	19.93	17.81	19.45	19.37	19.54	20.02	**16.92**	17.72
ARC-e	20.86	18.57	20.52	20.33	20.66	21.17	**17.55**	18.45
ObQA	48.45	44.99	48.20	47.60	48.09	48.00	**43.78**	44.17
MultiRC	10.44	9.80	10.40	10.21	10.59	10.36	**9.56**	9.67
Avg. SP	8.08	8.45	8.07	8.16	8.48	**8.04**	10.44	9.26
Avg. ET	22.86	20.81	22.56	22.32	22.69	22.89	**19.96**	20.59

Table 5: Generalization on end task validation domains for 1B parameter models trained for 100B tokens. Our model mixtures are optimized based on different generalization criteria, AVG-SP, AVG-ET, and AVG-ALL.

	SlimPajama	Linear-mde-all
MMLU (5 shot multi-choice)	25.56	**26.51**
GSM8k (3-shot no COT)	2.9	**4.2**

Table 6: Performance of 10B model and baseline at 200K steps on MMLU and GSM8K.

the optimized mixture's advantage is maintained for 10B-sized models.

4.6 Downstream task few-shot prediction

We compare performance of 1B parameter models on downstream tasks in Table 3 with learning curves in Figure 5. We observe that the token-proportional SlimPajama baseline is a strong baseline as it outperforms the uniform baseline and other baseline from prior work including DoGE, DoReMi, and DML (Ye et al., 2024). While our model optimized for only the AVG-SP loss has relatively low average accuracy, the variants that are optimized taking into account validation end tasks LINEAR-MDE-ET and LINEAR-MDE-ALL outperform all baselines and models from prior work.

Figure 6 shows similar trends of downstream task performance of 10B models comparing the best baseline SlimPajama to our method LINEAR-MDE-ALL. We also evaluate the 10B models on two significantly harder benchmarks — MMLU (Hendrycks et al., 2021) and GSM8k (Cobbe et al., 2021), which evaluate a broad set of skills we did not optimize for, with results shown in Table 6. While the absolute performance is weak in comparison to same-sized models trained on orders of magnitude more data, it is comparable to those of Ibrahim et al. (2024) (10B model, 3x tokens) and Falcon 7B (Almazrouei et al., 2023) (7B model, 15x tokens), respectively. The optimized weights lead to higher performance.

5 Conclusion

This work introduced the Mixture of Data Experts approximation which advances pre-training data mixture optimization. By leveraging MDE as a predictive feature in regression models, we improve the mixture ranking quality, loss prediction accuracy, and sample efficiency of regression. Our findings emphasize the value of task-aware mixture optimization, showing that incorporating end-task validation signals leads to notable improvements on downstream tasks.

6 Limitations

While this study provides insights into using mixtures of data experts (MDE) to better predict validation loss and optimize mixtures, several limitations should be acknowledged.

First, we conducted experiments solely on the SlimPajama dataset, which consists of seven training domains with predominantly English text. We have not evaluated our method on datasets with a larger number of domains or multiple languages. Additionally, our experiments were limited to text datasets, and we did not explore multi-modal data. Furthermore, we assume that training domains are predefined and meaningful, without addressing how to construct such domains from raw data.

Second, we only experimented with models up to the size of 10B parameters and have not evaluated our method on larger models or models trained for more than 100B tokens. Assessing its effectiveness on larger models/datasets remains an important area for future research. When the token horizon allows for sources to be repeated many times, diminishing returns from data repetition need to be taken into account as well.

Third, although we evaluated mixture performance using 12 downstream generation and ranking tasks, expanding the evaluation to a broader and more diverse set of tasks would provide a more comprehensive picture. Additionally, we did not investigate safety and inclusion-related criteria, which are important considerations for deploying such methods in real-world scenarios.

Despite these limitations, our findings contribute to the existing literature by demonstrating that MDE features can significantly improve performance and design sample-efficient regression models that outperform previous approaches, offering a strong foundation for further research in this field.

Acknowledgements

We are grateful to Pete Shaw, Kenton Lee, Michael Boratko, Sagi Perel, Sebastian Borgeaud, Adam Fisch, Jennifer Brennan, Yuan Zhang, Ruppert Koch, Jacob Eisenstein, Anton Alexandrov, Boqing Gong, Andreea Gane, Kelvin Guu, Luheng He, Jason Riesa, Mohammed Saleh, Raphael Hoffmann, and Slav Petrov for discussion and feedback.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millicah, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: a visual language model for few-shot learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

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. 2024. A survey on data selection for language models. *Preprint*, arXiv:2402.16827.

Alon Albalak, Liangming Pan, Colin Raffel, and William Yang Wang. 2023. Efficient online data mixing for language model pre-training. *Preprint*, arXiv:2312.02406.

Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. 2023. The falcon series of open language models. *Preprint*, arXiv:2311.16867.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *Proceedings of The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, pages 7432–7439, New York, New York, USA. AAAI Press.

Jérôme Bolte, Quoc-Tung Le, Edouard Pauwels, and Samuel Vaiter. 2025. Geometric and computational hardness of bilevel programming. *Preprint*, arXiv:2407.12372.

Edwin V Bonilla, Kian Chai, and Christopher Williams. 2007. Multi-task gaussian process prediction.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

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

Simin Fan, Matteo Pagliardini, and Martin Jaggi. 2024. DOGE: Domain reweighting with generalization estimation. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 12895–12915. PMLR.

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. 2020. The pile: An 800gb dataset of diverse text for language modeling. *Preprint*, arXiv:2101.00027.

Ce Ge, Zhijian Ma, Daoyuan Chen, Yaliang Li, and Bolin Ding. 2024. Bimix: Bivariate data mixing law for language model pretraining. *Preprint*, arXiv:2405.14908.

Sara A Geer. 2000. *Empirical Processes in M-estimation*, volume 6. Cambridge university press.

Gemma-Team. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Christoph Grüne and Lasse Wulf. 2024. Completeness in the polynomial hierarchy for many natural problems in bilevel and robust optimization. *Preprint*, arXiv:2311.10540.

Tatsunori Hashimoto. 2021. Model performance scaling with multiple data sources. In *International Conference on Machine Learning*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats Leon Richter, Quentin Gregory Anthony, Eugene Belilovsky, Timothée Lesort, and Irina Rish. 2024. Simple and scalable strategies to continually pre-train large language models. *Transactions on Machine Learning Research*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 252–262, New Orleans, Louisiana. Association for Computational Linguistics.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

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

Llama3-Team. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Clara Na, Ian Magnusson, Ananya Harsh Jha, Tom Sherborne, Emma Strubell, Jesse Dodge, and Pradeep Dasigi. 2024. Scalable data ablation approximations for language models through modular training and merging. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21125–21141, Miami, Florida, USA. Association for Computational Linguistics.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. 2020. What is being transferred in transfer learning?

In *Advances in Neural Information Processing Systems*, volume 33, pages 512–523. Curran Associates, Inc.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in python.

AJ Piergiovanni, Weicheng Kuo, Wei Li, and Anelia Angelova. 2023. Dynamic pre-training of vision-language models. In *ICLR Workshop on Multimodal Representation Learning*.

Mohammad Taher Pilehvar and Jose Camacho-Collados. 2019. WiC: the word-in-context dataset for evaluating context-sensitive meaning representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1267–1273, Minneapolis, Minnesota. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Logical Formalizations of Commonsense Reasoning — Papers from the 2011 AAAI Spring Symposium (SS-11-06)*, Stanford, California, USA. Association for the Advancement of Artificial Intelligence.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. WinoGrande: an adversarial Winograd Schema Challenge at scale. *Communications of the ACM*, 64(9):99–106.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. 2023. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. `https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama`.

Xingyou Song, Qiuyi Zhang, Chansoo Lee, Emily Fertig, Tzu-Kuo Huang, Lior Belenki, Greg Kochanski, Setareh Ariafar, Srinivas Vasudevan, Sagi Perel, et al. 2024. The vizier gaussian process bandit algorithm.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 23965–23998. PMLR.

Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. 2023. Doremi: Optimizing data mixtures speeds up language model pretraining. In *Advances in Neural Information Processing Systems*, volume 36, pages 69798–69818. Curran Associates, Inc.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Tong Zhang. 2006. From ε-entropy to kl-entropy: Analysis of minimum information complexity density estimation. *The Annals of Statistics*, pages 2180–2210.

A Proof of Proposition 3.1

Proof. Standard analysis of maximum likelihood estimation suggests that whenever the class of distributions $\mathcal{P}$ that p is chosen from is expressive enough, then the optimal solution $p^\star_\lambda = \arg\min_{p\in\mathcal{P}} L_\lambda(p)$ satisfies:

$$\mathbb{E}_{x\sim D_x} \|p^\star_\lambda(\cdot|x) - \sum_i \lambda_i D_{i,y}(\cdot|x)\|_1 \leq \epsilon, \tag{1}$$

where ϵ is a parameter which depends on the sample size n and the statistical complexity of the function class $\mathcal{P}$ (Geer, 2000; Zhang, 2006). For example, the statistical complexity is equal to $\ln|\mathcal{P}|$ for finite classes, and can be replaced with a log-covering number more generally. The main takeaway from Equation 1 is that the optimal solution $p^\star_\lambda$ can be written as $p^\star_\lambda = \sum_i \lambda_i p^\star_{e_i}$ in this case, where e_i is the i_{th} basis vector with all zeros and one in the i_{th} position.

More generally, when the marginal distributions over x are different, we can write

$$\begin{aligned} p^\star_\lambda(y|x) &= \sum_i p(i|x) D_i(y|x) = \sum_i \frac{p(x|i)p(i)}{\sum_j p(x|j)p(j)} D_i(y|x) \\ &= \sum_i \frac{D_i(x)\lambda_i}{\sum_j \lambda_j D_j(x)} D_i(y|x) \\ &= \sum_i \lambda_i' p^\star_{e_i}, \end{aligned}$$

□

B Implementation Details

B.1 MDE Approximation Pseudo-code

Algorithm 1 MDE loss approximation

```
Input:
  • Cached Per-token probs $\mathbf{p_i^{(j)}}$ of experts $\theta_i^*$ for validation domain tokens $j = 1, \ldots, |V_j|$.
  • Mixture $\lambda$.
Output:
  • The MDE loss approximation of model trained with mixture $\lambda$.
Algorithm:
for all $j \in \{1, 2, \ldots, |V_j|\}$ do
    $\mathbf{p}_{\text{MDE}}{}^{(j)} = \sum_i (\lambda_i \mathbf{p_i^{(j)}})$
end for
$\text{Loss}_{\text{MDE}} = \frac{1}{|V_j|} \sum_{j=1}^{|V_j|} - \ln \mathbf{p}_{\text{MDE}}{}^{(j)}$
```

B.2 Model and training details

Table 7 specifies the model sizes used during the experiments. Note that, due to the large vocabulary size, the number of non-embedding parameters is much smaller than the number of total parameters. For example, our 280M proxy models have fewer non-embedding parameters than DoGE-124M.

Models of all sizes used batch size of 512 sequences of up to 1024 text tokens. The maximum number of steps 200K corresponds to about 100B tokens. All language models were optimized using Adafactor (Shazeer and Stern, 2018) with initial learning rate of 1e-3, weight decay of 1e-2, and gradient clipping to norm 1. We decay the learning rate exponentially until it reaches a minimum of 1e-4 at the end of training, with a linear warmup of 6% of the total training steps.

Table 8 details the Google Cloud TPU configurations used to train models of each size.

Model size	Model dim.	# Layers	Hidden dim.	# Attention heads	Non-embedding params
70M	256	3	1,024	4	2,625,984
150M	512	6	2,048	8	19,171,712
280M	768	12	3,072	12	85,312,768
510M	1,024	12	8,192	16	252,125,952
1B	2,048	16	8,192	32	805,993,472
10B	4,096	32	24,576	32	8,592,168,960

Table 7: Architecture details for models used in our experiments. All models use the same vocabulary with a size of 256,000.

Model size	TPU v3 chips
70M	16
150M	16
280M	32
510M	32
1B	64
10B	64

Table 8: Hardware used for each model size.

B.3 Expert mixtures as regression examples

When fitting regression model with MDE features we experimented with both using the expert mixtures as examples and not using them, as expert mixtures like $(1, 0, 0, ...)$ may exhibit significantly different behavior from near-corner mixtures such as $(1 - \epsilon, \epsilon, 0, ...)$. We evaluated whether including these corner mixtures enhances or degrades model generalization performance. For Linear-MDE and GBM-MDE, we found that adding expert mixtures degrades performance, whereas for MTGP-MDE, it improves performance. We speculate that MTGP offers greater flexibility in modeling behavior at the corners without compromising predictions at other points. Nonetheless, when reporting the number of training examples, we always account for the expert examples used to generate the MDE features, ensuring that expert mixtures are included in the training example count.

B.4 Generating training mixture examples

Our goal is to sample a diverse set of mixture examples to fit a regression model that generalizes well across the entire mixture search space while also accounting for training domain token frequency. (Liu et al., 2025) suggested sampling from a Dirichlet distribution and setting the concentration parameters based on token frequency of the training domains. We opted to emphasize less prior domain token counts, as lower value training domains may contain a higher number of tokens. Instead, we set the concentration parameters as a weighted average between domain frequency and a uniform distribution. Additionally, we sampled scaling factors between 0.5 and 2.0 and multiplied the concentration parameters with them to introduce varying levels of diversity.

B.5 Splitting Examples for Training and Testing

To assess model performance, we randomly split the mixture examples into training and test sets five times. For each metric, we computed the sample mean and 95% confidence interval, and verified results we highlighted as different did not have overlapping confidence intervals. The reported loss-related squared error and ranking metrics represent the mean across the five folds.

B.6 Fitting Regression Models

MTGP - We trained the multi-task Gaussian process with a separable kernel using the open-source Vizier framework (Song et al., 2024).

Gradient Boosting - We initially considered using the default LightGBM (Ke et al., 2017) setup from (Liu et al., 2025). However, its default minimum leaf size is 20, which is unsuitable to our low-data

regime of about 20 examples. Instead, we used Scikit-Learn's (Pedregosa et al., 2011) gradient boosting model and performed a 5-fold cross-validation hyper-parameter grid search over the number of estimators $\{10, 50, 100\}$, learning rate $\{0.01, 0.1\}$, and maximum tree depth $\{2, 3, 4\}$. For the rest of the hyper-parameters we used the default settings.
Linear - We trained a Scikit-Learn linear model with Ridge regularization and performed 5-fold cross-validation to tune the regularization factor.

B.7 Additional dataset details

All training and evaluation datasets are predominantly in English, with possible exceptions for some SlimPajama texts.

We list the number of tokens in each of the 18 validation domains we used for loss optimization in Table 9. For the end-task derived datasets, both the question and gold response are included in the token counts.

DOMAIN	NUMBER OF TOKENS
ARXIV	4,105,850
BOOK	4,188,414
C4	1,719,076
COMMONCRAWL	3,281,676
GITHUB	2,861,175
STACKEXCHANGE	2,265,251
WIKIPEDIA	2,131,781
ARC-C-0SHOT	43,413
ARC-C-1SHOT	87,117
ARC-C-5SHOT	258,026
ARC-E-0SHOT	74,902
ARC-E-1SHOT	152,276
ARC-E-5SHOT	455,940
MULTIRC-0SHOT	2,704,800
MULTIRC-1SHOT	3,581,458
OPENBOOKQA-0SHOT	12,743
OPENBOOKQA-1SHOT	26,175
OPENBOOKQA-5SHOT	82,318

Table 9: Number of tokens in validation domains used for loss prediction and optimization.

B.8 Use of AI Assistants

We have used Gemini models to understand tikz for drawings and to suggest ways to format equations and algorithms. We also used Gemini to suggest ways to shorten some sentences and took some suggestions with additional edits.

C Additional results and analysis

C.1 Comparing Optimized Mixtures Across Different Scales

To better understand the impact of model size and training steps on the optimized mixture, we compare the mixtures obtained using the LINEAR+MDE method for two different models: (i) 70M-parameter model trained for 10K steps, and (ii) 280M-parameter model trained for 50K steps.

The optimized mixture for the *70M, 10K-step* model is:

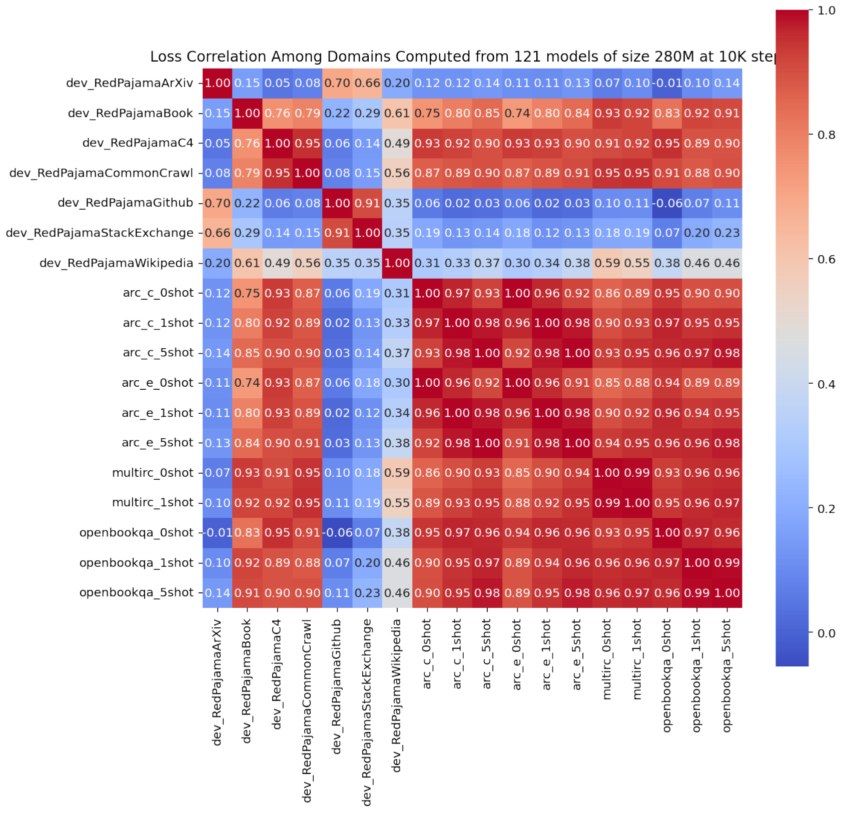
$$[0.078, 0.28, 0.411, 0.072, 0.0, 0.012, 0.148]$$

The optimized mixture for the *280M, 50K-step* model is:

$$[0.039, 0.287, 0.373, 0.259, 0.0, 0.0, 0.041]$$

Despite differences in model scale and token horizon, the mixture weights remain relatively similar, with a **cosine similarity of 91.32%**. This strong alignment further supports the validity of our proxy model mixture optimization approach.

C.2 Correlation among losses of different validation domains



	dev_RedPajamaArXiv	dev_RedPajamaBook	dev_RedPajamaC4	dev_RedPajamaCommonCrawl	dev_RedPajamaGithub	dev_RedPajamaStackExchange	dev_RedPajamaWikipedia	arc_c_0shot	arc_c_1shot	arc_c_5shot	arc_e_0shot	arc_e_1shot	arc_e_5shot	multirc_0shot	multirc_1shot	openbookqa_0shot	openbookqa_1shot	openbookqa_5shot
dev_RedPajamaArXiv	1.00	0.15	0.05	0.08	0.70	0.66	0.20	0.12	0.12	0.14	0.11	0.11	0.13	0.07	0.10	-0.01	0.10	0.14
dev_RedPajamaBook	0.15	1.00	0.76	0.79	0.22	0.29	0.61	0.75	0.80	0.85	0.74	0.80	0.84	0.93	0.92	0.83	0.92	0.91
dev_RedPajamaC4	0.05	0.76	1.00	0.95	0.06	0.14	0.49	0.93	0.92	0.90	0.93	0.93	0.90	0.91	0.92	0.95	0.89	0.90
dev_RedPajamaCommonCrawl	0.08	0.79	0.95	1.00	0.08	0.15	0.56	0.87	0.89	0.90	0.87	0.89	0.91	0.95	0.95	0.91	0.88	0.90
dev_RedPajamaGithub	0.70	0.22	0.06	0.08	1.00	0.91	0.35	0.06	0.02	0.03	0.06	0.02	0.03	0.10	0.11	-0.06	0.07	0.11
dev_RedPajamaStackExchange	0.66	0.29	0.14	0.15	0.91	1.00	0.35	0.19	0.13	0.14	0.18	0.12	0.13	0.18	0.19	0.07	0.20	0.23
dev_RedPajamaWikipedia	0.20	0.61	0.49	0.56	0.35	0.35	1.00	0.31	0.33	0.37	0.30	0.34	0.38	0.59	0.55	0.38	0.46	0.46
arc_c_0shot	0.12	0.75	0.93	0.87	0.06	0.19	0.31	1.00	0.97	0.93	1.00	0.96	0.92	0.86	0.89	0.95	0.90	0.90
arc_c_1shot	0.12	0.80	0.92	0.89	0.02	0.13	0.33	0.97	1.00	0.98	0.96	1.00	0.98	0.90	0.93	0.97	0.95	0.95
arc_c_5shot	0.14	0.85	0.90	0.90	0.03	0.14	0.37	0.93	0.98	1.00	0.92	0.98	1.00	0.93	0.95	0.96	0.97	0.98
arc_e_0shot	0.11	0.74	0.93	0.87	0.06	0.18	0.30	1.00	0.96	0.92	1.00	0.96	0.91	0.85	0.88	0.94	0.89	0.89
arc_e_1shot	0.11	0.80	0.93	0.89	0.02	0.12	0.34	0.96	1.00	0.98	0.96	1.00	0.98	0.90	0.92	0.96	0.94	0.95
arc_e_5shot	0.13	0.84	0.90	0.91	0.03	0.13	0.38	0.92	0.98	1.00	0.91	0.98	1.00	0.94	0.95	0.96	0.96	0.98
multirc_0shot	0.07	0.93	0.91	0.95	0.10	0.18	0.59	0.86	0.90	0.93	0.85	0.90	0.94	1.00	0.99	0.93	0.96	0.96
multirc_1shot	0.10	0.92	0.92	0.95	0.11	0.19	0.55	0.89	0.93	0.95	0.88	0.92	0.95	0.99	1.00	0.95	0.96	0.97
openbookqa_0shot	-0.01	0.83	0.95	0.91	-0.06	0.07	0.38	0.95	0.97	0.96	0.94	0.96	0.96	0.93	0.95	1.00	0.97	0.96
openbookqa_1shot	0.10	0.92	0.89	0.88	0.07	0.20	0.46	0.90	0.95	0.97	0.89	0.94	0.96	0.96	0.96	0.97	1.00	0.99
openbookqa_5shot	0.14	0.91	0.90	0.90	0.11	0.23	0.46	0.90	0.95	0.98	0.89	0.95	0.98	0.96	0.97	0.96	0.99	1.00

Figure 7: Correlation among model losses on different heldout training and end task domain datasets.

From Figure 7 we see that the SlimpPajama domains most correlated with validation end task domains are Book, C4, and CommonCrawl.

C.3 Mixture rates for SlimPajama from baselines and our work

In the experiments section, we reported losses and downstream from baseline mixtures, ones derived in prior work (in which case we copied the mixture rate values from the respective papers), and mixtures optimized in this work. Here we list the mixture proportion values λ for completeness in Tables 10 and 11.

Domain	Uniform	SlimPajama	DoGE-124M	DoReMi-124M	DML
Arxiv	0.1429	0.0458	0.0890	0.0434	0.2500
Book	0.1429	0.0420	0.0456	0.0546	0.0938
C4	0.1429	0.2660	0.2789	0.1127	0.2500
CommonCrawl	0.1429	0.5203	0.1968	0.3781	0.1250
Github	0.1429	0.0522	0.0714	0.0753	0.1406
StackExchange	0.1429	0.0337	0.1703	0.0919	0.1250
Wikipedia	0.1429	0.0399	0.1480	0.2440	0.0156

Table 10: SlimPajama data mixture rates derived through different approaches from prior work. DoGE and DoReMI weights are from the SlimPajama experiments of (Fan et al., 2024). DML weights are copied from (Ye et al., 2024).

C.4 Loss learning curve for 1B models

Figure 8 shows the average SP domain loss of 1B models with different data mixture proportions. We see that MTGP-MDE-SP achieves lower loss than other approaches.

Domain	MTGP-mde-sp	Linear-mde-et	Linear-mde-all	MDE-sp	MDE-et	MDE-all
Arxiv	0.0791	0.0014	0.0404	0.0666	0.0015	0.0372
Book	0.0931	0.2412	0.2859	0.1870	0.3251	0.2732
C4	0.2282	0.2952	0.3710	0.0837	0.3286	0.1868
CommonCrawl	0.1335	0.4578	0.2581	0.1602	0.2734	0.2249
Github	0.1047	0.0014	0.0014	0.1161	0.0000	0.0395
StackExchange	0.1454	0.0014	0.0014	0.2187	0.0715	0.1575
Wikipedia	0.2161	0.0014	0.0418	0.1678	0.0000	0.0810

Table 11: SlimPajama data mixture rates derived through optimizing AVG-SP, AVG-ET, and AVG-SP+AVG-ET with regressors using MDE or MDE on its own.

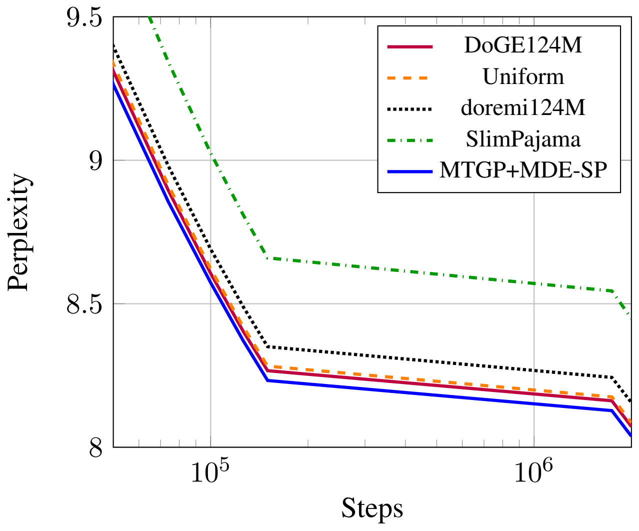


Figure 8: Convergence curve of training 1B parameter model for up to 200K steps for the different methods.

	SlimPajama	Lin-mde-all
ARC-c	13.95	**13.51**
ARC-e	14.97	**14.49**
ObQA	40.10	**39.33**
MultiRC	8.23	**8.07**
Avg.	17.23	**16.78**

Table 12: Generalization on end task validation domains for 10B parameter models trained for 100B tokens. Comparing baseline SlimPajama weights to our optimized ones. We report per-domain group and average perplexity.

C.5 End task validation domain losses of 10B models

In table 12, we report perplexity on the end task validation domains achieved at 100B tokens by our 10B parameter models trained with the baseline mixture and our optimized one. We observe that the mixture optimized at much smaller token and model size scale successfully improves upon the baseline at a significantly larger scale.

C.6 Optimizing mixtures with MDE only

We additionally optimize the mixtures for different criteria using the MDE approximation only, from 280M-sized models at 6K training steps. This requires training only seven proxy language models and no regression. In Table 13, we see that models optimized for AVG-SP based on MDE lead to worse but respectable AVG-SP loss than MTGP-MDE. Models optimized for the end task validation domains are best on those domains, and models optimized for average of SlimPajama and ET domains achieve slightly better tradeoff between those groups of domains than models optimized for ET domains only.

Domain	MTGP-mde-sp	Linear-mde-et	Linear-mde-all	MDE-sp	MDE-et	MDE-all
Avg. SP	**8.04**	10.44	9.26	8.110	10.501	8.228
Avg. task	22.89	**19.96**	20.59	23.246	20.439	21.800

Table 13: Generalization on SlimPajama and end task validation domains for 1B models trained for 100B tokens. Comparing MDE to MTGP-MDE and LINEAR-MDE optimized weights. We report average perplexities on SlimPajama and end task validation domains.

C.7 MDE vs related approximations through domain-specific expert models

To understand the performance of MDE in the context of related ideas from Na et al. (2024), which, as mentioned in Section 2, approximates the loss of a model trained on a union of datasets with the loss of a model which is a parameter average of expert models trained on the individual datasets, we analyze the importance of using model ensembles instead of parameter-averaged models. In addition, we evaluate MDE in comparison to a simpler and even faster to compute version, which interpolates per-dataset average probabilities instead of per-token ones.

We compute Spearman's rank correlation between the true domain losses versus the ones predicted by MDE and the two alternative methods, using 20 models of size 280M trained for 10K steps, corresponding to 20 different data mixtures. We report ρ across the seven SplimPajama domains, and also average across the full 18 domains (SlimPajama and end-task validation domains). Table 14 shows the results. Note that these metrics are averages of performance for predicting single domain losses, and are a bit higher than the metrics for predicting aggregated losses that we saw in Table 1. We can note that model merging, where for each candidate λ we first compute a weighted average of expert model parameters, and then run inference to compute losses on the validation domains, has very poor performance. This agrees with prior work which finds parameter averaging to work well only for models fine-tuned from a common initialization points. The per-domain interpolation approach does not use token-level probabilities from the experts, but only computes a weighted average (with λ) of their per-validation dataset average probabilities. We see that this approach does surprisingly well, but is still substantially weaker than MDE.

Based on these results we conclude that parameter averaging of expert models is not a useful approach for approximating losses for pre-training data mixtures. We also see that there is value in interpolating per-token probabilities, as in MDE, instead of interpolating per-dataset average probabilities. Per-dataset probability interrelation is a bit easier to implement and faster to compute, and could also be useful. Future work could also explore both MDE and per-domain interpolation as feature sources in the same regression model.

	MDE	Model Merging	Per-Domain Interpolation
SP Spearman	**0.951**	-0.139	0.898
All Spearman	**0.942**	-0.120	0.927

Table 14: MDE versus Model Merging and Per-Dataset Interpolation.