OPTIMAL SPLITTING OF LANGUAGE MODELS FROM MIXTURES TO SPECIALIZED DOMAINS

Skyler Seto[†] **,Pierre Ablin**[†]**, Anastasiia Filippova**[†]**, Jiayuan Ye**[†‡]**,**[*] **Louis Bethune**[†]**, Angelos Katharopoulos**[†]**, David Grangier**[†]
[†] Apple [‡]National University of Singapore

ABSTRACT

Language models achieve impressive performance on a variety of knowledge, language, and reasoning tasks due to the scale and diversity of pretraining data available. The standard training recipe is a two-stage paradigm: pretraining first on the full corpus of data followed by specialization on a subset of high quality, specialized data from the full corpus. In the multi-domain setting, this involves continued pretraining of multiple models on each specialized domain, referred to as split model training. We propose a method for pretraining multiple models independently over a general pretraining corpus, and determining the optimal compute allocation between pretraining and continued pretraining using scaling laws. Our approach accurately predicts the loss of a model of size N with D pretraining and D' specialization tokens, and extrapolates to larger model sizes and number of tokens. Applied to language model training, our approach improves performance consistently across common sense knowledge and reasoning benchmarks across different model sizes and compute budgets.

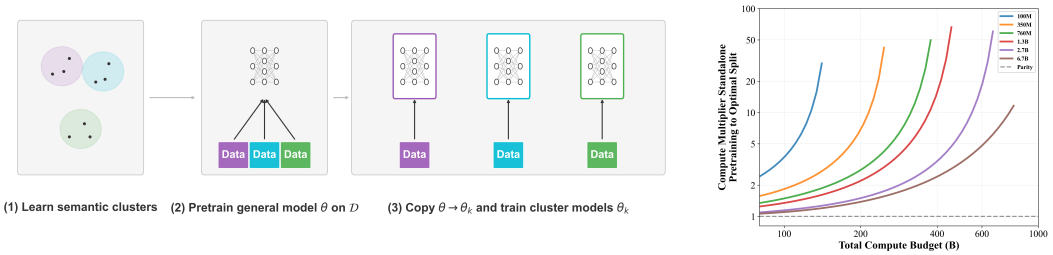
1 INTRODUCTION

Large language models (LLMs) have demonstrated remarkable performance across a wide range of language understanding and knowledge-intensive tasks (Achiam et al., 2023; Brown et al., 2020; Bubeck et al., 2023; Liu et al., 2024; Team et al., 2023). Their widespread success can be attributed to the scale of high quality pretraining data with large-scale training datasets exceeding trillions of tokens from the web (Hojel et al., 2025; Li et al., 2024). Many of the best performing models such as Qwen (Yang et al., 2025), Llama 3 (Grattafiori et al., 2024), Olmo (OLMo et al., 2024; Olmo et al., 2025), and SmolLM (Bakouch et al., 2025) are trained for 4-30 trillion tokens, even for the small model sizes ($<$ 7B parameters), greatly exceeding the compute-optimal training times recommended by the Chinchilla scaling law (Hoffmann et al., 2022).

The data used for pretraining (PT) language models is typically derived from large web crawls, and often spans multiple domains of interest; we denote K the number of domains. The goal is to train a model that performs well on average across all domains. In the setting where $K = 1$, the typical recipe is a two-stage training paradigm: initial pretraining on the full corpus followed by a specialization phase - continued pretraining (CPT) - on data from the target domain of interest. There are many specialized domains one can choose such as high quality math, code, reasoning, instruction, or multilinguality (Bakouch et al., 2025). In many cases, only a short amount of the training time (20%) is spent on specialization (Bakouch et al., 2025; Olmo et al., 2025).

When targeting $K > 1$ domains, this strategy can be extended by employing a shared generic pretraining phase followed by independent specialization for each domain. In that case, the allocation of the total computational budget between generic pretraining and domain-specific specialization presents a non-trivial optimization problem. Each step in the shared pretraining phase benefits all downstream domains when learning shared features - for example general language syntax and grammars. Conversely, each specialization step benefits only a single domain but may be more effective, as such a step adapts the parameters without the optimization constraints imposed by the other domains - for example when learning fine-grained vocabularies particular to certain field such

[*]Work done during internship at Apple.



(a) Split model training

(b) Compute multiplier

Figure 1: (a) Split model training: The pretraining corpus is clustered based on semantics, then a model is trained on all domains before being copied and subsequently trained independently on an individual cluster. (b) Compute multiplier: The multiplicative factor on the amount of data required for standard pretraining to match optimal split model training performance. At large compute budgets, standard pretraining requires up to 50$\times$ the amount of data to match the loss of optimal split training.

as science, mathematics, or code. This is unlike the $K = 1$ setting where one should train on the target domain for as long as data is available.

How much of the compute to dedicate to the shared pretraining phase vs. the specialization phase is understudied. Prior works that study CPT don't consider the cost of pretraining in the compute allocation and focus primarily on predicting the loss on the CPT data domain (Bethune et al., 2025; Que et al., 2024; Hernandez et al., 2021). Other works show that overtraining in the pretraining phase leads to worse results after finetuning (Springer et al., 2025), but don't aim to quantify the optimal allocation of compute between pretraining and finetuning, nor do they target multiple domains.

This work explores the question *what is the optimal allocation of compute for pretraining and continued pretraining over multiple data domains?* We study the scaling behaviors of split model training as a function of pretraining, and specialization data budgets with the goal of demonstrating that split model training with only a small amount of joint pretraining is sufficient. Our main contributions are

- providing a recipe for pretraining and continued pretraining over multiple domains summarized in Figure 1a,
- deriving a scaling law that predicts the loss of split model training as a function of compute and identifying the optimal splitting time,
- demonstrating that split model training at the optimal compute allocation determined by our scaling law improves over full pretraining in data efficiency in Figure 1b.

2 Related Work

Scaling Laws for Language Models: Scaling laws aim to predict a model's performance as a function of the total training compute. Early studies show that the loss of language models exhibit a power law scaling (Hoffmann et al., 2022; Kaplan et al., 2020). These laws have limitations as they consider only pretraining, a fixed data distribution, and dense models. Recent studies have expanded the scaling laws to tackle data mixture dependence including optimal data mixture weights (Shukor et al., 2025; Ye et al., 2024), selection ratios for repetitions of high quality data (Goyal et al., 2024) and repetitions of data more broadly (Muennighoff et al., 2023), and data quality (Chang et al., 2024). Other works study scaling laws for continued pretraining (Que et al., 2024), and finetuning (Bethune et al., 2025; Zhang et al., 2024), and have been used to measure the effective amount of data from pretraining for code and text tasks (Hernandez et al., 2021). Concurrent to our work, scaling laws for bootstrapped continued pretraining are proposed for continued pretraining on a new domain or model growth (Liew & Kato, 2025). Finally there is a set of work investigating scaling laws for sparsity (Abnar et al., 2025; Krajewski et al., 2024; Wang et al., 2024), and routing (Clark et al., 2022) in Mixture of Expert (MoE) models. However, these works do not consider training MoEs with CPT.

Split Model Training: Prior works have explored split training of models. One set of these approaches are based on a branch, train, and merge strategy where a base seed model is trained, and individual expert models are trained from the base seed model before merging (Li et al., 2022). Gururangan et al. (2023) cluster data into different domains and train individual models on each domain before merging, while others train an MoE architecture asynchronously (Filippova et al., 2025; Shi et al., 2025). These approaches focus on training a mixture and merging asynchronously rather than completely split and routed models, and don't consider pretraining cost as in this work. Alternatively, several works use several seed models and train an orchestrator that routes queries or parts of a query to different models with limited (Dann et al., 2025; Lee et al., 2024; Wang et al., 2025), or no additional training (Wu & Silwal, 2025), and sometimes to models of varying sizes (Chuang et al., 2024). In both settings, a majority of the training is done with a single model limiting efficiency gains from training multiple models, and other benefits of split model training.

Data Mixtures: Training on a mixture of data domains has become a standard for training language models. Early datasets such as the Pile (Gao et al., 2020), SlimPajama (Soboleva et al., 2023), and Dolma (Soldaini et al., 2024) were aggregated from multiple sources from the web, and recent models are also trained on mixtures of domains (Bakouch et al., 2025; Olmo et al., 2025). Other works discover domains through clustering the pretraining corpus (Gururangan et al., 2023; Diao et al., 2025) extending up to millions of domains (Grangier et al., 2025), or by synthetically generating different data mixtures (Maini et al., 2024; Su et al., 2025). Our clustering approach is similar to that of (Gururangan et al., 2023) but we focus on optimal pretraining compute allocation rather than learning to branch and merge split models from an already trained seed model.

3 Split Model Training

3.1 Learning Split Models with Clustering

Our approach works in three stages, considering both the pretraining (PT) and continued pretraining (CPT) cost, and is highlighted in Figure 1a.

First, we cluster all documents in the pretraining corpus $\mathcal{D}$ into K disjoint clusters $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K$ using document embeddings from a document embedding model. A router $\mathcal{R}(x; W)$ is learned from the clustering to map from documents to cluster assignments. Second, we perform a two stage training of θ. A seed model θ' is trained on D tokens from $\mathcal{D}$ to approximately minimize

$$\theta(D) = \arg\min_{\theta} \; \mathbb{E}_{x\sim\mathcal{D}}\big[\mathcal{L}(x;\theta)\big], \tag{1}$$

with the next-token prediction loss

$$\mathcal{L}(x;\theta) = -\sum_{s} \log p_\theta(x_s|x_{<s}). \tag{2}$$

After training $\theta(D)$, we create K copies $\theta_1, \theta_2, \ldots, \theta_K$ and train each on $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_K$ respectively with D' tokens

$$\theta_k = \arg\min_{\theta} \; \mathbb{E}_{x\sim\mathcal{D}_k}\left[\mathcal{L}(x;\theta)\right]. \tag{3}$$

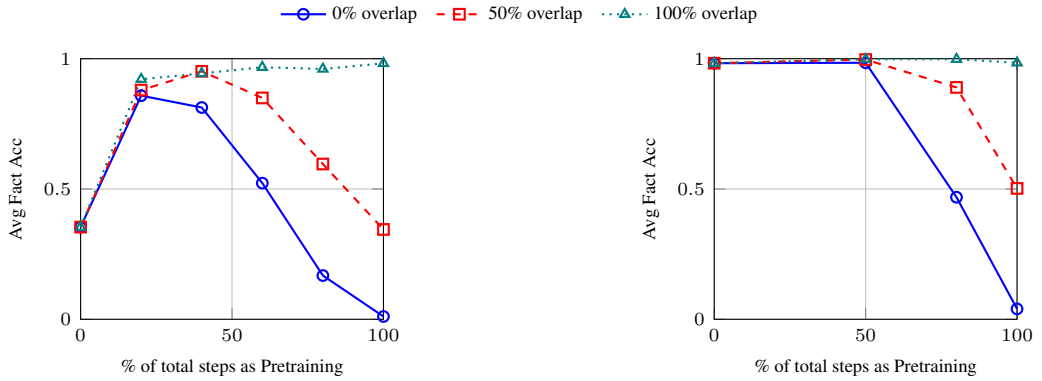
We refer to this as *split model training* as all of the data is split across K models which are trained completely independently from the seed model. The total training cost of split model training is a function of the total number of tokens $D + K \times D'$, where D is the total number of pretraining tokens, and D' is the number of tokens each expert model is trained on.

At inference, given a new document x, a prefix x_p (such as a question, starting sentence, or context passage) is selected, and the router selects the corresponding cluster index c as

$$c = \arg\min_{c} \mathcal{R}(x_p; W) = \|x_p - W_c\|. \tag{4}$$

The model θ_c is then used on the entire document x.

Prior works that train split models use a pretrained seed model and do not consider the amount of data for PT D' and CPT D' (Li et al., 2022; Gururangan et al., 2023). These works train on the order of hundreds of billions of tokens, but a large part ($> 50\%$) of the training is still done in the pretraining phase. They do not consider whether a short or no pretraining phase could yield similar or better performance. We start by demonstrating that this is an important consideration as pretraining for too short or too long can reduce performance under the same total training budget, and that the optimal time to split depends on the per-cluster data distributions and the total compute budget.



(a) Limited Total Budget (100k Steps)

(b) Sufficient Total Budget (200k Steps)

Figure 2: Average fact accuracy (over two clusters) vs. split step for synthetic phonebook fact clusters with different overlaps under different total training budget. We say the total budget is sufficient if split training on two cluster each using (Total Steps/2) from step 0 reaches 100% average fact accuracy, and otherwise say the total budget is limited.

3.2 Phonebook Memorization Experiments

Implementation Details: We first motivate that split model training has varying performance improvements compared with pretraining on all domains. We start with an experiment on fact memorization in transformers. For our experiments, we train a transformer model with 12.9M parameters (6 layers, 8 attention heads, and hidden dimension 256). We fix the learning rate at 0.0001, batch size as 640, and warmup steps as 20000. We additionally study perplexity improvements when training 1.3B parameter language models on three pairs of clusters from a clustered DCLM dataset in Appendix F.

Methodology: We follow (Jelassi et al., 2024) and use synthetically generated phonebook records to measure fact memorization. Each cluster contains 80000 randomly generated phonebook facts, where each fact is a (name, phone-number) tuple of the format

$$< 6 \text{ alphabet tokens} > \mid < 22 \text{ digit tokens} >,$$

where the name contains six randomly drawn alphabetical tokens from a to z, and the phone-number contains 22 randomly sampled digits from 0 to 9. In such setting, we define similarity between the two clusters as $\alpha = \frac{|\text{cluster}_1 \cap \text{cluster}_2|}{80000}$.

Findings: Results for 100K steps of training are in Figure 2a and 200K steps in Figure 2b. In the limited budget setting, the best performing model varies with the degree of overlap. In the 100% overlap setting, it is always better to pretrain for longer on both domains. In the 0% and 50% settings, the average fact accuracy peaks when allocating 20%-40% of the total compute budget to pretraining, then subsequently decreases zero as we allocate more compute budget to pretraining, as the amount of combined information in two clusters is too large for a single pretrained model to memorize accurately. In the sufficient budget setting, similar trends are shown, but here the beginning of the curve (early splitting) also achieves perfect memorization.

4 Scaling Laws for Split Model Training

In Section 3, we found that split model training improvements depend on the amount of PT, CPT, total compute budget, and domain similarity. Predicting the final model loss from independent model training as a function of the number of pretraining tokens D, and identifying optimal values of D, can lead to better-trained models with greater efficiency. We consider neural scaling laws for predicting the test loss of the model.

4.1 SETUP

Our prediction of $\mathcal{L}$ is a function of the learning algorithm which has parameters $D_1, \ldots, D_K$, the number of tokens for training from each subdomain, and N, the number of parameters in θ. Note that predicting the loss of all k domains is equivalent to predicting the loss of individual domains and taking the weighted average. Thus, for simplicity we consider estimation of the loss $\mathcal{L}(N, D, D_k)$ of domain k from which we can compute the average loss over all domains.

4.2 SCALING LAW FUNCTIONAL FORM

Neural scaling laws describe a model's loss as a function of the model size, amount of training data, and compute budget. The seminal Chinchilla scaling law for language models is an additive law of the form

$$\mathcal{L}(N, D) = E + A \cdot N^{-\alpha} + B \cdot D^{-\beta} \tag{5}$$

where E, A, α, B, β are learnable parameters dependent on the data, model architecture, and optimization procedure (Hoffmann et al., 2022).

A key difference in our work is that we consider a two-stage training with D and D_k, which leads to differences in the loss functional form. First, the model should be a function of both D and D_k, and in the absence of the other, both should follow a power law. That is, training only on D_k and only on D should independently follow a standard Chinchilla scaling law. Second, we note that as $\mathcal{D}_k \subset \mathcal{D}$, training on D should improve loss on D_k but at a slower rate. We have the following desiderata for a scaling law that models $\mathcal{L}(N, D, D_k)$: i) the limited capacity of the model prevents it from learning all of $\mathcal{D}$ and $\mathcal{D}_k$ from $\mathcal{D}$. Therefore, we would like the scaling law asymptotics to predict a different irreducible loss when pretraining with infinite compute on D compared to training only on D_k, ii) $\mathcal{L}(N, D, D_k)$ should recover a Chinchilla type-scaling when either D or $D_k = 0$, and iii) taking D_k to infinity should lead to the same irreducible loss, regardless of D. In order to satisfy i), we consider a bias E_Δ, which is a sigmoid function of D_k and N:

$$E_\Delta = E_p \cdot \frac{1}{1 + (N/N_s)^{\gamma_1}} \cdot \frac{1}{1 + (D_k/D_s)^{\gamma_2}}. \tag{6}$$

Our final law is

$$\mathcal{L}(N, D, D_k) = E_0 + E_\Delta + A \cdot (D_k^{\alpha_1} + cD^{\alpha_2})^{-1} + B \cdot N^{-\kappa} \tag{7}$$

where E_p, E_0, N_s, D_s, A, B, γ_1, γ_2, α_1, α_2, c and κ are all learnable parameters. Note that our scaling law handles the conditions stated above with an irreducible loss independent for both D and D_k.

5 EXPERIMENTS FOR SCALING LAWS

5.1 EXPERIMENTAL SETUP

We use decoder only transformer models in the parameter sizes of 100M, 350M, 760M, 1.3B, 2.7B parameters. All models are trained using the DCLM dataset with a batch size of 1M tokens[1] following the hyperparameters in Appendix A. The DCLM dataset is clustered using a balanced K-means clustering with 16 clusters following details in Section E. The scaling law is fitted using the basin-hopping algorithm and the methodology described in (Shukor et al., 2025) and in Appendix G.1.

5.2 SCALING LAW FITTING

We train the scaling law in (3) using data of the form $(D, D_k, N, \mathcal{L})$ obtained by training models across a range of values and estimating the loss $\mathcal{L}$ on a validation set of data from D_k with $k = 8$ and total number of domains $K = 16$. We consider three scenarios when testing the scaling law:

- Scenario 1: Train on small model size ($\leq$ 1.3B), and test on larger model size (2.7B)

[1]We use a context length of 1024 and 1024 samples per batch. This is approximately 1M tokens and we use the number of steps in the thousands interchangeably with billions of tokens.

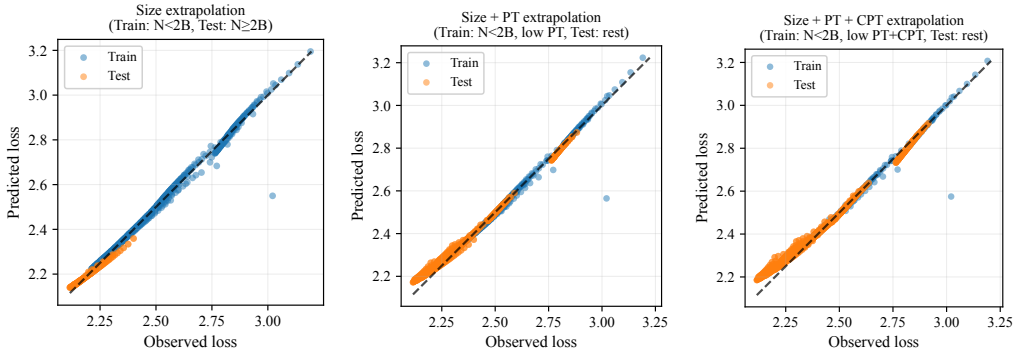


Figure 3: Predicted loss vs. observed loss across multiple scenarios. The scaling laws are fit on smaller models with varying amount of PT and CPT token budgets. Loss is estimated over domain 8.

- Scenario 2: Train on small model size ($\leq$ 1.3B) and lower half of the range of D, and test on rest
- Scenario 3: Train on small model size ($\leq$ 1.3B) lower half of the range of D and D_k, and test on rest.

Table 1 describes our results in terms of the mean absolute error (MAE) and R^2 coefficient. Both MAE is low from $0.012 - 0.023$, and the R^2 is high and close to 1 even for setting where we test on only a small handful of runs - 130 for scenario 3. When testing with similar number of runs but for smaller models we have overall low MAE and high R^2.

	Train Size	Test Size	MAE	R2
Scenario 1	652	144	0.012	0.934
Scenario 2	347	449	0.021	0.982
Scenario 3	156	640	0.023	0.979

Table 1: MAE and R^2 for different model size on varying held out test conditions.

Additionally, we plot the observed loss vs. predicted loss in Figure 3. We find that the predicted loss and observed loss are extremely close, and that both train and test points do not deviate from the fit. A comparison with (Liew & Kato, 2025) is available in Appendix H.

5.3 Optimal Number of Pretraining Tokens

We now use the previous scaling laws to predict the optimal number of pretraining tokens, after which it becomes worthwhile to specialize and split models.

Taking model size aside, define $\mathcal{L}_k(D, D')$ as the loss of a model pretrained for D tokens on $\mathcal{D}$ and then trained for D' tokens on $\mathcal{D}_k$. Since training is split, the total number of tokens used to train these K models is $D_T = D + K \times D'$. Therefore, assuming we are looking for the best models on average over all domains that minimize $L(D, D') := \sum_{k=1}^{K} \mathcal{L}_k(D, D')$, the question of the optimal number of pre-training tokens becomes that of optimal allocation:

$$\min_{D,D'} L(D, D') \text{ s.t. } D + K \times D' = D_T \tag{8}$$

We define t_S as the **optimal splitting point**, that is, the number of pre-training tokens that solves the previous problem. It depends on the total budget D_T. First, we may wonder if it is optimal to split training at all. Say that we have pretrained a model for D tokens, and that we have an additional infinitesimal budget of δ tokens. We can either a) use these δ tokens to keep on pretraining on D, which will give us a loss $L(D+\delta, 0) \simeq L(D, 0) + \frac{\partial L(D_{\text{split}}, 0)}{\partial D}\delta$, or b) we can split the models and train on each domain with δ/K tokens. This will give us a loss $L(D, \delta/K) \simeq L(D, 0) + \frac{1}{K}\frac{\partial L(D_{\text{split}}, 0)}{\partial D'}\delta$.

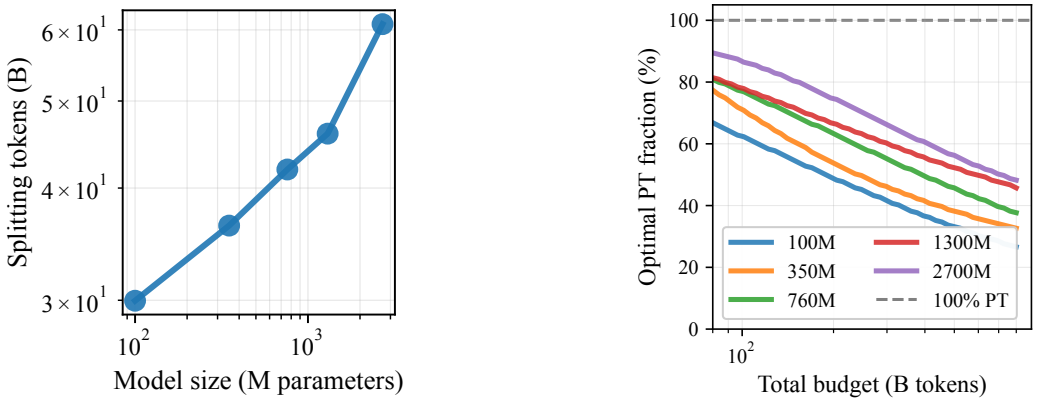


Figure 4: **Left** Number of pretraining tokens after which split training becomes beneficial D_{split}, as a function of model size. We compute it as the solution of equation 9. We see that it takes more tokens for larger models to start benefiting from split training. We only use the loss on one domain to approximate the average loss L. **Right** Optimal fraction of pretraining tokens as a function of the total training budget t_S/D_T: the fraction decays with budget, but the optimal number of pretraining tokens increases with budget.

The **minimal splitting point** D_{split} is such that b) becomes more advantageous than a), which gives

$$K\frac{\partial L}{\partial D}(D_{\text{split}}, 0) = \frac{\partial L}{\partial D'}(D_{\text{split}}, 0) \tag{9}$$

If the total budget is such that $D_T \leq D_{\text{split}}$, splitting is useless and the optimal allocation is to put $t_s = D_T$. Figure 4 shows this splitting point as a function of model size.

Findings: First, we note that as model size increases, the minimal splitting point D_{split} increases. For small model sizes, the curves appears to be linear in model size. Second, we empirically remark greedy splitting is not optimal and that the solution to equation 8 is *not* $t_S = D_{\text{split}}$, i.e., to pretrain for D_{split} and then do continued pretraining: if we have a larger budget, it might be worthwhile to pretrain for longer than D_{split}, and $t_s \geq D_{\text{split}}$. We fit the scaling law in equation 7, and then we report the optimal number of pretraining tokens for each budget in Figure 4. This number is not constant past the splitting point D_{split}. A greedy splitting solution that splits a model as soon as the budget goes above D_{split} is therefore not optimal; optimal splitting point must be aware of the training horizon.

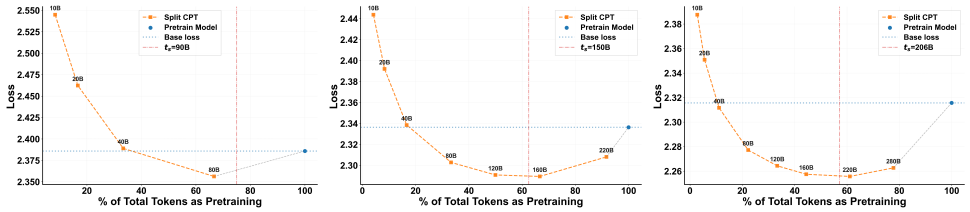
6 Language Model Performance

We compare scaling law results with LLM downstream performance on zero-shot question answer performance. We report the average accuracy across eight benchmarks: ARC-Easy, ARC-Challenge, HellaSwag, PIQA, SciQ, BoolQ, MMLU, and Winogrande. For additional details see Appendix C.2. We show that QA performance improves with split model training, and that performance matches models with larger inference costs. Next, we apply our optimal split point to train 1.3B and 2.7B parameter models that outperform models with larger inference budget. Finally, we consider some ablations on the number of models, and architecture trained.

6.1 Optimal Model Split Point for Varying Budget

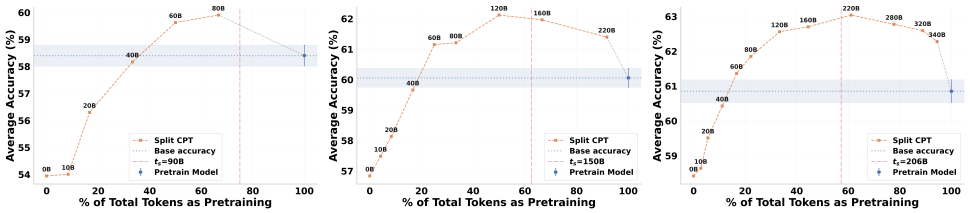
Methodology: We train 1.3B parameter models at varying amounts of compute in approximately {120, 240, 360} billion total tokens. We train several models with varying amount of pretraining in {0, 10, 20, 60, 80, 120, 160, 220, 280, 320, and 340} billion tokens, and additionally report the optimal t_s as reported by the scaling law.

Findings: We present results in Figure 5 for loss and Figure 6 for zero-shot QA benchmarks. For all of the compute scales considered, CPT from scratch performs much worse than pretraining on the full



(a) 120B (b) 240B (c) 360B

Figure 5: Loss vs. fraction of pretraining tokens for a 1.3B parameter models at 120, 240, and 360B total training tokens.



(a) 120B (b) 240B (c) 360B

Figure 6: Zero shot QA accuracy vs. fraction of pretraining tokens for a 1.3B parameter models at 120, 240, and 360B total training tokens.

data, or split training. Split training early at around 40B tokens performs similarly to pretraining over the full data. Although we do not train a model that matches exactly the optimal number of tokens, we see that on either side of the optimal t_s, the accuracy is increasing then decreasing indicating that the optimal t_s is a good indication of performance on language modeling and QA benchmarks. We report results for 125M models in Section J where we find better performance from split training with zero pretraining matching our results in Section 3 as the 125M models are sufficiently overtrained even at the earliest split training checkpoint.

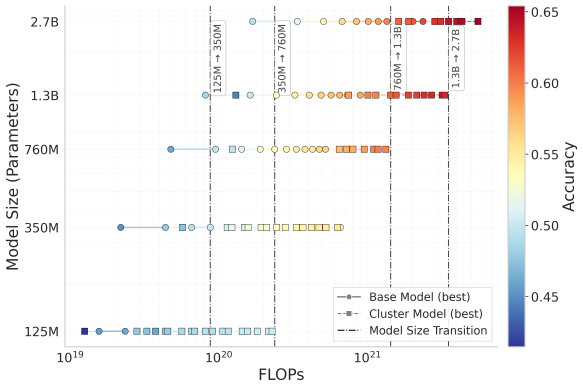
6.2 Optimal Model Split Point for Varying Sizes

Methodology: We investigate split model training performance as a function of model size. We repeat the analysis above, but for several model sizes from the GPT family of models. Details of the architectures are in Appendix A. We compare pretrained models and split models at various scales throughout training.

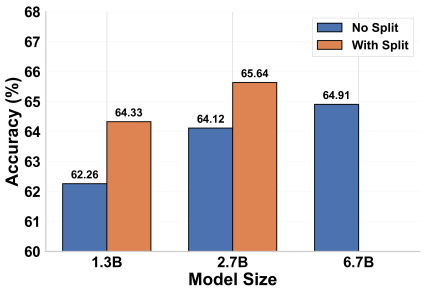
Findings: Results are summarized in Figure 7a. First, we observe that for a fixed model size (fixed line along the y-axis), the best performing model is a seed model (denoted by a circle), and changes to a split model (square) after which point split models always outperform the full pretrained model. Second, we note that for several model sizes, such as 125M, 760M, and 1.3B, the split model can outperform the larger model size as the same amount of compute. This is denoted by the switch point (vertical dashed lines) where the split models at the given size ar better than full pretrained models of the size higher. Following this observation, we note that in most cases, training a single pretrained model is rarely the best. The only instance where the optimal model switches to a fully pretrained model at the next model size is from 350M to 760M at which point the best configuration is pretraining a single model. At all other points of compute, split models are competitive or better.

6.3 Optimal Pretraining Tokens Performance

We verify that using our scaling law estimates optimal t_s in language modeling over specialized domains, and on zero-shot QA benchmarks.



(a) Optimal model size and splitting at varying flops.



(b) Average zero shot accuracy for pretrained and split models at the optimal allocation.

Figure 7: Performance comparison across model sizes for split training. (a) Zero-shot QA accuracy for standalone (base) and split (cluster) models at various model sizes. Dashed lines indicate where the best performing model changes in size as a function of the total compute (FlOPs). (b) Zero-shot QA accuracy for various standalone and split training models at 1.3B and 2.7B scale. All models are trained for a total compute budget equivalent to roughly 140B tokens of the 6.7B model, however 1.3B and 2.7B models have lower inference costs.

6.3.1 Pile Language Modeling

Methodology: We evaluate 1.3B models pretrained for the entire compute budget and split model training with $K = 16$ domains from clustering the DCLM dataset. We train for a compute budget of 720B tokens of the base pretrained model. The split models are first pretrained for $t_s = 340$B tokens before continued pretraining for the remainder. We evaluate perplexity across five specialized domains in the Pile[2]: ArXiv, DM Math, FreeLaw, Github, and PubMed Central (Gao et al., 2020). Details on the cluster distribution for the specialized domains are in Appendix E.2.

Findings: For split models, we compute the perplexity for each domain using the model routed to most frequently for data from that domain. We compare with base pretraining only in Table 3 in Appendix E.2. Split models perform better than the pretrained model with an average relative improvement in perplexity of 9.33%. The largest gains are from FreeLaw and ArXiv.

6.3.2 QA Benchmarks

Methodology: We compare models using our scaling law to determine the optimal t_s for training models, and train for a compute budget equivalent to $1\times$ Chinchilla for the 6.7B model equating to roughly 140B tokens as in (Li et al., 2024). We compare pretraining 1.3B and 2.7B parameter models, and split model training with the same size models. Note that while the number of parameters is larger, the number of active parameters is much smaller than that of a 6.7B model.

Findings: We use our scaling law to estimate the optimal allocation t_s. Results for 1.3B, 2.7B, and a pretrained 6.7B model are in Figure 7b. For the 1.3B parameter model this equated to 340B tokens for pretraining, less than half of the total pretraining tokens, and 220B tokens for the 2.7B parameter model. At this scale, split models perform better than full pretraining by $1.5 - 2\%$. The 1.3B split models perform worse than the 6.7B model, however, the 2.7B split models perform better by 0.6%.

6.4 Cluster Ablations

Prior sections split train models with a fixed number of domains $K = 16$ and use all models at evaluation. Naturally there are several important questions to consider. In this section, we provide brief discussion on whether all models are needed for inference, and what the optimal number of domains is. We defer to Section N for extended results of the ablations.

[2] https://huggingface.co/datasets/monology/pile-uncopyrighted

6.4.1 TWO CLUSTER ROUTING SPECIALIZATION

Methodology: Prior results train one model for each domain and route to all models at inference time. In some settings it may be infeasible to maintain a model for each domain in memory. We study specialization of domains at inference time where only two split models are used including based on the top-2 MMLU, and the most frequent models per task based on routing the validation set.

Findings: Our findings in Table 10 show that task specialization at evaluation reduces performance by around 1% compared to evaluation and training with the full clusters. The MMLU subset reduces performance more by up to 3%. This indicates that performing well across domains necessitates all models.

6.4.2 PERFORMANCE WITH VARYING NUMBER OF DOMAINS

Methodology: Our main results use 16 domains, however data may come with any number of domains. We compare performance of split model training across number of domains. We train a separate clustering for each number of clusters.

Findings: Results for training up to 200B tokens for the 1.3B parameter model are summarized in Figure 17. We find that the 16 domain model maintains a good balance in accuracy across PT tokens. At small the 4 domain models perform the best. At larger t_s, the 4 domain models are matched by the 16 domain models. Results for the 2.7B parameter model training up to 300B tokens are summarized in Figure 16.

7 CONCLUSION

Strong language models are being trained in two stages through general pretraining on a large corpus of data before continuing on high quality specialized data for downstream applications. This work discusses split model training as a recipe for training language models when there are multiple specialized data domains in the pretraining corpus. We propose a scaling law that directly addresses the question of how to allocate compute budget between pretraining on the entire dataset and continued pretraining on individual domains, and demonstrate that this is closely connected to downstream performance on zero-shot evaluation tasks.

The gain of training independent expert models lies in both paradigms: (i) improved performance from reduced training on extraneous domains, and (ii) greater training efficiency from reduced hardware constraints. Empirically, our scaling law accurately predicts performance of large models beyond the size trained and with additional tokens, and training language models with the optimal splitting point yields 2% improved performance across different compute budgets and model sizes in evaluations leading to greater performance at the same compute budget as models with double the inference cost. For the latter, specialized hardware (synchronous compute and large numbers of GPUs) are typically needed in order to train a single model, and there is a limit to the number of GPUs that can be used, especially for training smaller models due to limits in the batch size used for training. Split model training for larger compute budgets requires less training on specialized hardware as in some settings less than 50% of the compute budget is allocated to pretraining. For the remainder of training, models can be trained asynchronously and over less specialized hardware.

Additionally, there are other benefits to split model training over data domains beyond what we study here, for example in safe data handling. In this setting, it may be undesirable to share data from particular domains, or have a model trained on a mix of data from different sources as data attribution becomes challenging. Split training on a portion of shareable data during the PT stage and the private data during the CPT stage can make this feasible. In summary, it is important for researchers and practitioners to consider how to allocate compute towards training the next large language models. In many instances, pretraining and continued pretraining are considered independently, however greater improvements come from considering them jointly.

ACKNOWLEDGEMENTS

We are grateful to Zak Aldeneh, Natalie Schluter, Anastasiia Sedova, and Maartje ter Hoeve for their helpful discussions, comments, and thoughtful feedback in reviewing this work.

REFERENCES

Samira Abnar, Harshay Shah, Dan Busbridge, Alaaeldin Mohamed Elnouby Ali, Josh Susskind, and Vimal Thilak. Parameters vs flops: Scaling laws for optimal sparsity for mixture-of-experts language models. *arXiv preprint arXiv:2501.12370*, 2025.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Elie Bakouch, Loubna Ben Allal, Anton Lozhkov, Nouamane Tazi, Lewis Tunstall, Carlos Miguel Patiño, Edward Beeching, Aymeric Roucher, Aksel Joonas Reedi, Quentin Gallouédec, Kashif Rasul, Nathan Habib, Clémentine Fourrier, Hynek Kydlicek, Guilherme Penedo, Hugo Larcher, Mathieu Morlon, Vaibhav Srivastav, Joshua Lochner, Xuan-Son Nguyen, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM3: smol, multilingual, long-context reasoner. `https://huggingface.co/blog/smollm3`, 2025.

Louis Bethune, David Grangier, Dan Busbridge, Eleonora Gualdoni, Marco Cuturi, and Pierre Ablin. Scaling laws for forgetting during finetuning with pretraining data injection. *arXiv preprint arXiv:2502.06042*, 2025.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*, 2023.

Ernie Chang, Matteo Paltenghi, Yang Li, Pin-Jie Lin, Changsheng Zhao, Patrick Huber, Zechun Liu, Rastislav Rabatin, Yangyang Shi, and Vikas Chandra. Scaling parameter-constrained language models with quality data. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 80–97, 2024.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route llms with confidence tokens. *arXiv preprint arXiv:2410.13284*, 2024.

Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International conference on machine learning*, pp. 4057–4086. PMLR, 2022.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. 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)*, pp. 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Christoph Dann, Yishay Mansour, Teodor Vanislavov Marinov, and Mehryar Mohri. Principled model routing for unknown mixtures of source domains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, et al. Nemotron-climb: Clustering-based iterative data mixture bootstrapping for language model pre-training. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

Anastasiia Filippova, Angelos Katharopoulos, David Grangier, and Ronan Collobert. No need to talk: Asynchronous mixture of language models. In *ICLR*, 2025.

Trevor Gale, Deepak Narayanan, Cliff Young, and Matei Zaharia. Megablocks: Efficient sparse training with mixture-of-experts. *Proceedings of Machine Learning and Systems*, 5:288–304, 2023.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Sachin Goyal, Pratyush Maini, Zachary C Lipton, Aditi Raghunathan, and J Zico Kolter. Scaling laws for data filtering–data curation cannot be compute agnostic. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22702–22711, 2024.

David Grangier, Simin Fan, Skyler Seto, and Pierre Ablin. Task-adaptive pretrained language models via clustered-importance sampling. In *ICLR*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Suchin Gururangan, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Scaling expert language models with unsupervised domain discovery. *arXiv preprint arXiv:2303.14177*, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 30016–30030, 2022.

Andrew Hojel, Michael Pust, Tim Romanski, Yash Vanjani, Ritvik Kapila, Mohit Parmar, Adarsh Chaluvaraju, Alok Tripathy, Anil Thomas, Ashish Tanwer, et al. Essential-web v1. 0: 24t tokens of organized web data. *arXiv preprint arXiv:2506.14111*, 2025.

Samy Jelassi, Clara Mohri, David Brandfonbrener, Alex Gu, Nikhil Vyas, Nikhil Anand, David Alvarez-Melis, Yuanzhi Li, Sham M Kakade, and Eran Malach. Mixture of parrots: Experts improve memorization more than reasoning. *arXiv preprint arXiv:2410.19034*, 2024.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Jakub Krajewski, Jan Ludziejewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, et al. Scaling laws for fine-grained mixture of experts. *arXiv preprint arXiv:2402.07871*, 2024.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. Orchestrallm: Efficient orchestration of language models for dialogue state tracking. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1434–1445, 2024.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*, 2022.

Seng Pei Liew and Takuya Kato. From acceleration to saturation: Scaling behavior of bootstrapped language model pretraining. *arXiv preprint arXiv:2510.06548*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Pratyush Maini, Skyler Seto, Richard Bai, David Grangier, Yizhe Zhang, and Navdeep Jaitly. Rephrasing the web: A recipe for compute and data-efficient language modeling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14044–14072, 2024.

Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36:50358–50376, 2023.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025.

Hadi Pouransari, David Grangier, C Thomas, Michael Kirchhof, and Oncel Tuzel. Pretraining with hierarchical memories: separating long-tail and common knowledge. *arXiv preprint arXiv:2510.02375*, 2025.

Haoran Que, Jiaheng Liu, Ge Zhang, Chenchen Zhang, Xingwei Qu, Yinghao Ma, Feiyu Duan, Zhiqi Bai, Jiakai Wang, Yuanxing Zhang, et al. D-cpt law: Domain-specific continual pre-training scaling law for large language models. *Advances in Neural Information Processing Systems*, 37: 90318–90354, 2024.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Weijia Shi, Akshita Bhagia, Kevin Farhat, Niklas Muennighoff, Jacob Morrison, Evan Pete Walsh, Dustin Schwenk, Shayne Longpre, Jake Poznanski, Allyson Ettinger, et al. Flexolmo: Open language models for flexible data use. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Mustafa Shukor, Louis Bethune, Dan Busbridge, David Grangier, Enrico Fini, Alaaeldin El-Nouby, and Pierre Ablin. Scaling laws for optimal data mixtures. *arXiv preprint arXiv:2507.09404*, 2025.

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

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, et al. Dolma: An open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 15725–15788, 2024.

Jacob Mitchell Springer, Sachin Goyal, Kaiyue Wen, Tanishq Kumar, Xiang Yue, Sadhika Malladi, Graham Neubig, and Aditi Raghunathan. Overtrained language models are harder to fine-tune. In *Forty-second International Conference on Machine Learning*, 2025.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2459–2475, 2025.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Qwen Team. Qwen1.5-moe: Matching 7b model performance with 1/3 activated parameters", February 2024. URL `https://qwenlm.github.io/blog/qwen-moe/`.

Siqi Wang, Zhengyu Chen, Bei Li, Keqing He, Min Zhang, and Jingang Wang. Scaling laws across model architectures: A comparative analysis of dense and moe models in large language models. In *EMNLP*, 2024.

Xinyuan Wang, Yanchi Liu, Wei Cheng, Xujiang Zhao, Zhengzhang Chen, Wenchao Yu, Yanjie Fu, and Haifeng Chen. Mixllm: Dynamic routing in mixed large language models. *arXiv preprint arXiv:2502.18482*, 2025.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. *arXiv preprint arXiv:1707.06209*, 2017.

Fangzhou Wu and Sandeep Silwal. Efficient training-free online routing for high-volume multi-llm serving. *arXiv preprint arXiv:2509.02718*, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When scaling meets llm finetuning: The effect of data, model and finetuning method. In *ICLR*, 2024.

A TRAINING AND MODEL HYPERPARAMETERS

The 100M parameter model consists of 24 layers, 8 attention heads, and a hidden dimension of 512. It is trained with a learning rate of 0.0003, weight decay of 0.01, and gradient clipping of 0.1. The 125M parameter model consists of 12 layers, 12 attention heads, and a hidden dimension of 768. It is trained with a learning rate of 0.001, weight decay of 0.0, and gradient clipping of 0.1. The 350M parameter model consisting of 24 layers, 16 attention heads, and a hidden dimension size of 1024. It is trained with a learning rate of 0.0003, weight decay of 0.01, and gradient clipping of 0.1. The 1.3B parameter model consists of 24 layers, 16 attention heads, and a hidden dimension size of 2048. It is trained with a learning rate of 0.0003, weight decay of 0.01, and gradient clipping of 0.1. The 2.7B parameter model consists of 32 layers with 2560 hidden dimension and 32 attention heads. It is trained with a learning rate of 0.00016, weight decay of 0.01, and gradient clipping of 1.0. The 6.7B parameter model consists of 32 layers, 32 attention heads, and a hidden dimension fo 4096. It is trained with a learning rate of 0.00012, weight decay of 0.01, and gradient clipping of 1.0.

For our results in Section 5.2 and 6, all pretrained-only models are trained with a warmup of 10000 steps with fixed learning rate. Split models are trained with a fixed learning rate starting from the learning rate of the base model with the optimizer states reset to mirror CPT from a given pretrained model. If splitting from the start of training, we use a warmup of 3000 steps. For models trained in Section 3, we use a warmup set to 1% of the training time. All models are trained with the AdamW optimizer using $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Unless otherwise stated, models use the GPT2 tokenizer, with a total vocabulary size of 50K tokens using BPE.

All dense models are trained using NVIDIA's Megatron-LM[3] repository for pretraining language models.

For the MoE models, we train with AdamW (weight decay 0.1) using a learning-rate schedule with 5% linear warm-up followed by decay over the final 10% of training, for a total of 260B tokens. To obtain compute-matched clustered runs as in experiments with dense models, we start from the baseline checkpoint at step 150,000 and train $K = 16$ cluster models. Each cluster model is then continued for 6,250 steps so that the aggregate continuation across all clusters matches the baseline continuation from 150,000 to 250,000 steps (and thus matches total tokens and training FLOPs under the same per-step batch size and sequence length). We use the megablocks framework for training (Gale et al., 2023).

B TRAINING COST

A majority of the costs for running our experiments are for 1.3B and 2.7B models. The total training time for 1.3B models on roughly 100B tokens is around 1000 GPUh on Nvidia H100 GPUs. For a 2.7B model trained on 100B tokens, the total time is around 2200 GPUh on Nvidia H100.

For experiments in Figure 7b, this would equate to 7000+ GPUh for each model. For a model which is fully pretrained, this would involve training on 42 GPUs for a full week, which can be prohibitively expensive for many practitioners or researchers training language models. Training split models instead means that 47% of the total budget will be spent on pretraining. The remaining time can be spent training independently on non-specialized hardware as each split model individually needs less compute.

C DATASETS

C.1 TRAINING DATASETS

We pretrain all models using the DCLM-Baseline dataset (Li et al., 2024). We train on a random shuffled subset of around 400B tokens of data, which are split into up to 2049 sized chunks for clustering following Grangier et al. (2025); Pouransari et al. (2025). Except for Figure 7b where the 1.3B parameter model is trained on < 2 full repetitions of the data. At the 1.3B model scale, we do not suspect this leads to a decrease in performance. For cluster models, we cluster the full

[3] https://github.com/NVIDIA/Megatron-LM

corpus, but typically only train on a small portion of the data not exceeding 100B tokens in any of our experiments.

C.2 Evaluation Datasets

C.2.1 Validation Loss

We hold out a set of 10,000 documents from each cluster and evaluate the trained models on each document. We ensure that these documents are also not seen in the pretraining data. We use the validation set from Pile uncopyrighted.

C.2.2 Zero Shot Evaluations

- **SciQ**: A dataset of science exam questions for evaluating the ability of NLP models in understanding and reasoning within the science domain (Welbl et al., 2017).
- **ARC Challenge (ARC-C)**:Part of the AI2 Reasoning Challenge (ARC) (Clark et al., 2018), containing science exam questions from grades 3 to 9. The ARC Challenge set includes more difficult questions that necessitate higher-order reasoning.
- **ARC Easy (ARC-E)**: The Easy set of the AI2 Reasoning Challenge (Clark et al., 2018) features questions from the same source as ARC-C but are considered less challenging.
- **Winogrande (WG)**: This dataset challenges models on common sense reasoning in a language context, focusing on pronoun disambiguation tasks (Sakaguchi et al., 2021).
- **PIQA**: Physical Interaction Question Answering tests the understanding of everyday physical processes (Bisk et al., 2020).
- **HellaSwag (HS)**: Evaluates a model's ability to complete scenarios in a contextually and logically coherent manner (Zellers et al., 2019).
- **BoolQ**: A set of Yes/No questions from Google Search Queries. Samples contain a passage, and question (Clark et al., 2019).
- **MMLU**: Multi-domain question answering, MMLU assesses the model's expertise over a wide range of specialized subjects, from professional domains to academia (Hendrycks et al., 2020).

D Evaluation Details

We use the lm-eval-harness repository[4] for zero-shot accuracy on QA tasks[5]. All datasets are collected from the Huggingface datasets library. For all tasks, we use the continuation (cloze) style formatting for evaluation.

For split model selection over the evaluation set, we route using either the question, or passage. Table 2 summarizes the routing query for downstream evaluation tasks. For validation set evaluaiton, we evaluate using the first 32 tokens of the document, which would correspond to around 24 words using an estimate of 0.75 words per token.

E Cluster Details

E.1 Cluster Implementation Details

We cluster the training set using a balanced K-means clustering algorithm with a tree of depth 1 as we train with a maximum of 64 clusters following (Grangier et al., 2025). Before training the clustering, the dataset is segmented into non-overlapping 2,048 token windows and compute sentence-BERT embedding for every windows following (Pouransari et al., 2025). Embedding are built from the sentence-BERT MiniLM-L6-v2 model (Reimers & Gurevych, 2019).

[4] `https://github.com/EleutherAI/lm-evaluation-harness`

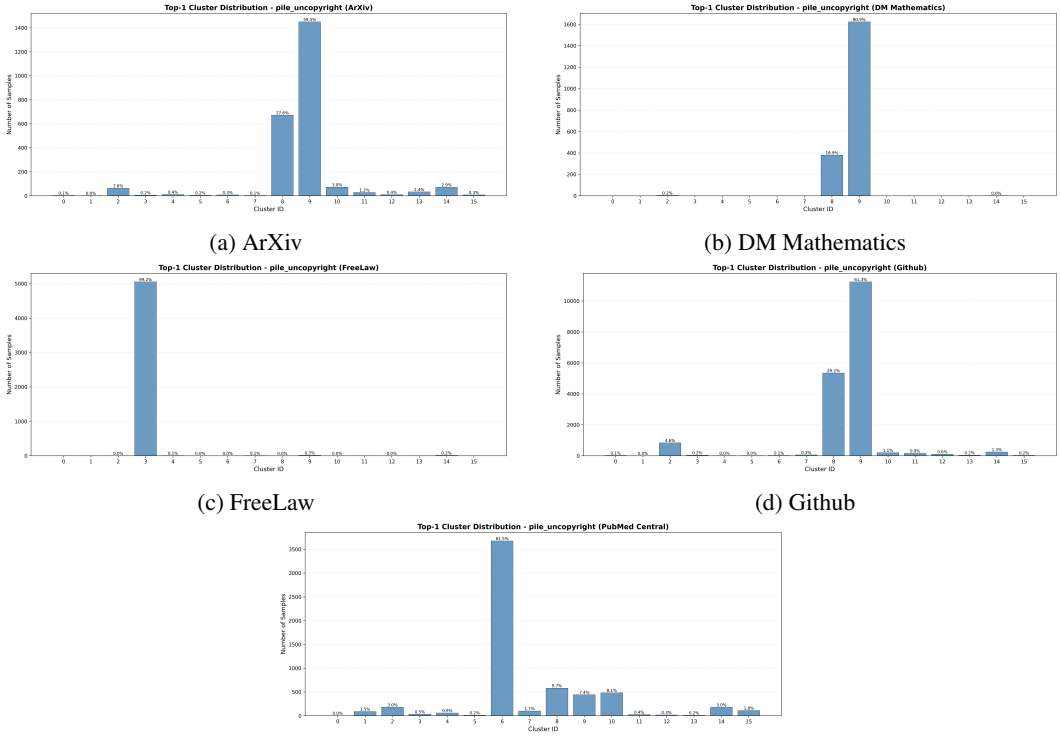
[5] We use the commit `03c44adc0586f88bb343a74da1a1c602103536dd`.

Task	Routing Text
ARC	question
SciQ	question
HellaSwag	ctx
PIQA	goal
BoolQ	question
Winogrande	sentence
MMLU	question

Table 2: Routing Text Field by Task

E.2 Pile Distribution and Loss Over Clusters

We first verify that the clustering produces specialized datasets and as a result specialized split models. To do so, we measure distribution of samples from different domains of the Pile. We consider ArXiv, DM Math, FreeLaw, Github, and PubMed Central for their diversity and relevance towards in many CPT settings such as training code, math, and medical language models. The distribution is shown in Figure 8.



(a) ArXiv

(b) DM Mathematics

(c) FreeLaw

(d) Github

(e) PubMed Central

Figure 8: Distribution of samples from Pile domains within the DCLM clustering.

Next, we compute the loss for each split model over the five domains in Figure 9. To compute the loss we take a subset of 1000 samples from the validation set of each domain.

Finally, we compare the perplexity of a base pretrained-only model trained for 720B tokens, and split models first pretrained for $t_s = 340$B tokens. We compute perplexity as

$$PPL(X) = exp\left(-\frac{1}{L}\sum_{t=1}^{L} p(x_t|x_{<t})\right)$$

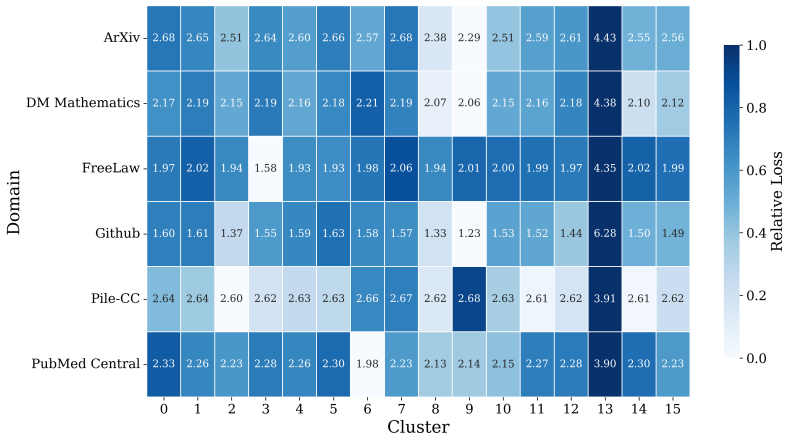


Figure 9: Loss for five domains of the pile and common crawl data for each split model. The models are 1.3B models pretrained for 340B tokens before CPT.

Domain	Pretrained	Split Model
ArXiv	11.01	9.92
DM Math	8.14	7.81
FreeLaw	5.76	4.85
Github	3.74	3.42
PubMed Central	7.90	7.25

Table 3: PPL for different domains of the Pile for a standalone pretrainged model only, and split models. The standalone model with pretraining only is trained for a total of 720B tokens. The split models are 1.3B parameters and trained for the same total compute budget with $t_s = 340$B.

We route all sequences from a domain to a single model. For some domains, better performance could be achieved as different samples are mapped to different clusters. Nonetheless, even routing samples only to the most common cluster, we still see relative improvements across these five domains.

E.3 Prefix Routing Accuracy

At inference time in settings where the domain is unknown, routing is done using a prefix. For QA tasks, this can be the entire query and context, or a portion of the query such as the question only. For general language modeling, a small portion at the start of the document is used. The loss is then computed over the remaining portion of the document. In Table 4, we evaluate sensitivity of the clustering to different prefix lengths by measuring retrieval accuracy of the full document given the prefix. Our findings show that retrieval accuracy is high starting at 16 tokens and diminishes in gain of R@5 starting from 32 tokens. We choose 32 tokens as it is a good tradeoff and should be representative of the context length of downstream tasks equating to roughly 1-2 sentences.

F Two Cluster DCLM Experiments

Methodology: We train language models with pairs of clusters from the clustered DCLM dataset. We create three pairs of clusters by computing the distance of the centroids from all clusters to cluster 0. We then take the two farthest clusters (denoted 0% similarity), the median cluster distance (denoted 50% similarity), and the two closest clusters (denoted 100% similarity). We compare split training on both clusters and training a single language model on both clusters combined.

Findings: Results are shown in Figure 10 for training at 90K steps (roughly 3× Chinchilla). For the 0% and 50% similarity, the loss is mostly increasing with proportion of budget for pretraining. For the 100% similarity setting, we find the reverse where the loss decreases as the proportion of

Prefix Length (tokens)	R@1	R@5
1	0.0128	0.0362
2	0.0955	0.1753
4	0.2664	0.3765
8	0.4802	0.5884
16	0.7112	0.8078
32	0.8775	0.9330
64	0.9663	0.9870
128	0.9972	0.9995
256	0.9998	1.0000

Table 4: Retrieval accuracy by prefix length (N=10000)

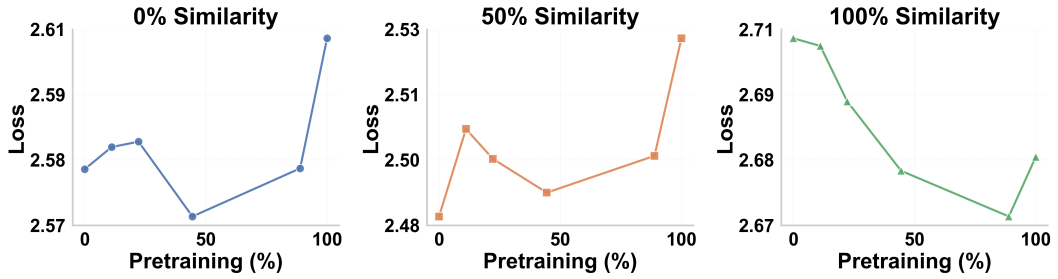


Figure 10: Average loss vs. fraction of compute for pretraining for different pairs of clusters in the DCLM-base dataset. Total compute budget is 90000 steps or roughly 90B tokens.

pretraining increases. This matches the findings from memorization of phonebooks in the previous section.

G Scaling Law Estimation

G.1 Scaling Law Optimization

We fit on data from for base model training for 5B-200B tokens for the pretrained model, and between 5B-30B tokens for continued pretraining depending on the amount of training done in pretraining. We remove outliers from the data (> 4.0 and < 0.5). We do not remove any steps and predict 0 steps for either pretraining or continued pretraining.

We fit the scaling laws using the same method as in (Shukor et al., 2025). Given a sequence of parameters N^i, D^i, D^i_k as well as the corresponding loss values L^i, for $i = 1 \dots n$, we find the optimal scaling law parameters Ω by solving

$$\min_{\Omega} \sum_{i=1}^{n} \text{Huber}(\mathcal{L}(N^i, D^i, D^i_k; \Omega) - L^i)$$

where Huber is the Huber loss with a shape parameter of $\delta = 10^{-3}$. In order to solve that problem, we resort to the Basin-Hopping algorithm with a grid of random initializations, which we found more efficient that the widely used L-BFGS approach.

Several parameters in the scaling law share overlapping functional roles, which can lead to identifiability issues during optimization. For example, E_p and E_0 jointly determine the asymptotic loss, while α_1, α_2, and c both modulate the effective contribution of pretraining tokens. Without constraints, the optimizer can find similar fits with very different parameter values. To regularize the optimization and ensure similar fits across clusters, data, and model sizes, we bound each parameter to a plausible range. Specifically, we use softplus for parameters that must be strictly positive, a scaled sigmoid for parameters confined to a known interval $[l, u]$, and the standard sigmoid for powers. The parameter bounds are summarized in Table 5.

Parameter	Transform	Effective range
E_p	softplus	$(0, \infty)$
E_0	scaled sigmoid	$[1.0,\ 3.0]$
A	softplus	$(0, \infty)$
α_1	sigmoid	$[0.1,\ 1.0]$
α_2	sigmoid	$[0.1,\ 1.0]$
c	sigmoid	$[0.5,\ 4.0]$
D_s	scaled sigmoid	$[500,\ 700]$
γ	sigmoid	$[0.1,\ 1.0]$

Table 5: Parameter constraints for the single-size CPT scaling law. Transformations are applied to unconstrained optimizer variables to enforce bounded, physically meaningful parameter values.

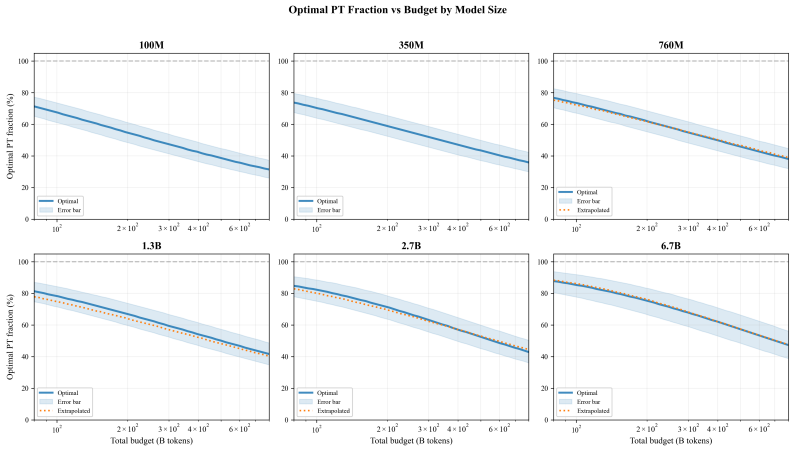


Figure 11: Estimated optimal fraction of pretraining tokens as a function of total compute budget, estimated from per-size scaling laws (solid lines) and extrapolation via power-law fit across smaller sizes (dotted lines). Shaded intervals represent the range of allocations that achieve loss within 0.005 of the optimum, capturing the flatness of the loss surface near the optimal split.

G.2 Optimal PT Extrapolation

We additionally present a simple extrapolation procedure that leverages scaling law fits from smaller models. For each model size N_i, we first fit a single-size scaling law $\hat{L}(D_{\text{pt}}, D_{\text{cpt}}; N_i)$ on observed loss data. Given a total compute budget B expressed in effective tokens, where the budget constraint is $D_{\text{pt}} + K \cdot D_{\text{cpt}} = D_T$ (with $K = 16$ reflecting the higher per-token cost of continued pretraining), we sweep over feasible allocations and identify the optimal pretraining fraction $f_s = t_s/D_T$ that minimizes predicted loss. For any fixed budget B, the optimal pretraining fraction varies smoothly as a function of model size (Figure 11), motivating a power-law fit $f_s(N) = a \cdot N^b$ across model sizes.

To extrapolate to a target model size, we fit this power law using data from all smaller model sizes in the dataset. For example, using scaling laws fit on 100M, 350M, 760M, and 1.3B models, we extrapolate the optimal pretraining fraction to 2.7B. We compare the extrapolated prediction with the ground-truth optimal f_s obtained by fitting a scaling law directly on data from the target model size in Figure 11. The extrapolation closely matches the per-size ground truth, suggesting that practitioners can estimate the optimal PT-CPT allocation for a larger model from a small number of cheaper, smaller-scale experiments, without requiring a joint scaling law that spans all model sizes.

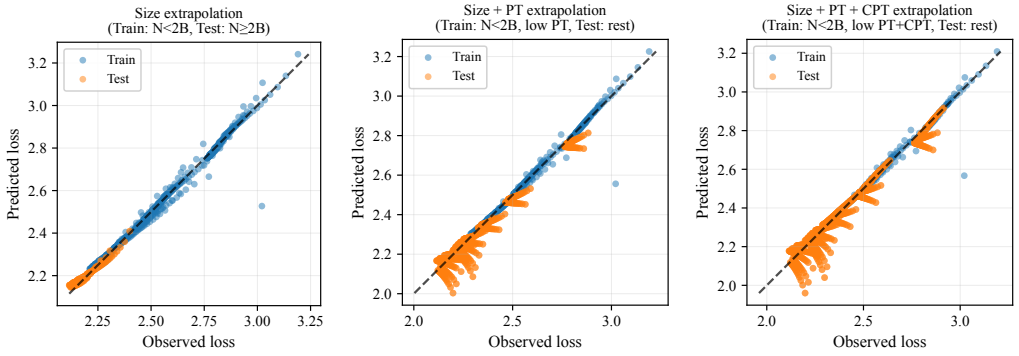


Figure 12: Estimated fits with the scaling law from (Liew & Kato, 2025)

H Comparison with other scaling laws for continued pre-training

Liew & Kato (2025) also tackle the problem of forecasting the loss of a model of size N, pretrained with D tokens and then trained with D' more token. They propose a scaling law of the form

$$L = E + AD'^{-\alpha_1} D^{-\alpha_2 + \alpha_3 \log D'} + BN^{-\beta}. \tag{10}$$

We fit that scaling law to our data using the procedure described above. We report the comparison of extrapolation capabilities in Table 6, and we observe that our law has a better predictive power. We report the predicted vs actual loss plots in Figure 12. We also note that the scaling law in Equation (10) does not verify the desiderata mentioned in Section 4.2.

	Train Size	Test Size	R2 (ours)	R2 (Liew & Kato, 2025)
Scenario 1	649	142	**0.934**	0.910
Scenario 2	344	447	**0.982**	0.946
Scenario 3	156	640	**0.979**	0.953

Table 6: Comparison of the test R2 scores with our proposed scaling law and that of (Liew & Kato, 2025).

I Scaling Law Results for Other Clusters

I.1 Extended Results for Domain 0

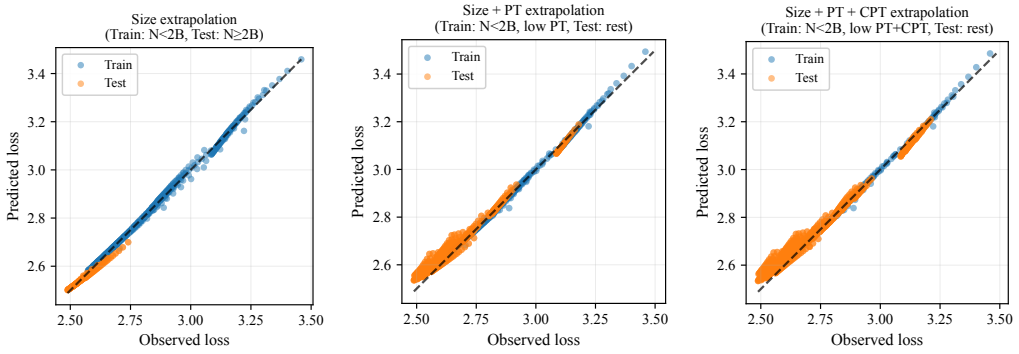


Figure 13: Predicted loss vs. observed loss across multiple scenarios. The scaling laws are fit on smaller models with varying amount of PT and CPT token budgets. Loss is estimated over domain 0.

We train the scaling law in (3) using data from domain $k = 0$ in addition to results in Section 5.2. We consider the same three scenarios for testing:

- Scenario 1: Train on small model size (< 2.7B), and test on larger model size (2.7B)
- Scenario 2: Train on small model size (< 2.7B) and low D, test on rest
- Scenario 3: Train on small model size (< 2.7B) and low D and low D_k, and test on rest.

Table 7 describes our results in terms of the mean absolute error (MAE) and R^2 coefficient. Both MAE is low from $0.6 - 1.5$, and the R^2 is high and close to 1 even for setting where we test on only a small handful of runs - 130 for scenario 3. When testing with similar number of runs but for smaller models we have overall low MAE and high R^2. We also plot the observed loss vs. predicted loss in Figure 13.

	Train Size	Test Size	MAE	R2
Scenario 1	649	142	0.009	0.934
Scenario 2	344	447	0.025	0.960
Scenario 3	156	640	0.022	0.973

Table 7: Fitting the scaling laws on domain 0 with multiple held out test conditions.

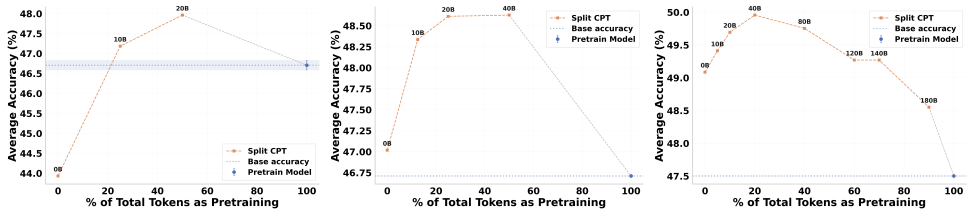
Our findings are similar to results for domain $k = 8$. We see that the scaling law fits the data well with low MAE and MRE, and high R^2.

I.2 Aggregated Results Across All Domains

We train a scaling law for each individual domain k. We report the average correlation, MAE, and MRE across domains for the 1.3B parameter models. Data from each of the models starts with PT in $\{10, 20, 40, 80, 120, 160\}$ thousand steps. Results are summarized in Table 8. We see that all clusters have low MAE, MRE, and high correlation indicating a good fit for each cluster.

J Zero-shot QA Performance with Large Token Budget

Our results in Section 6 use 1.3B parameter models at up to 420B tokens (roughly $14\times$ Chinchilla). However, we expect the performance to vary with the length of training as a function of both model size and total training budget. In this section, we show performance when scaling the amount of data to large compute budget. In order to do so, we consider a smaller model size with 125M parameters. We train for up to 200B tokens, which corresponds to a $100\times$ Chinchilla multiplier. Although these models are trained on less total data, the relative scale of both model size and data makes them more overtrained, and we expect to see results closer to the overtrained small models in Section 3. Note further that we expect to see similar trends when scaling to larger compute budgets for 1.3B models but is sufficient to test in the smaller setting.



(a) 40B (b) 80B (c) 200B

Figure 14: Performance across eight zero-shot QA benchmarks for a 125M model at the same training compute with varying amount of pretraining steps. Each subfigure has a different total number of training tokens.

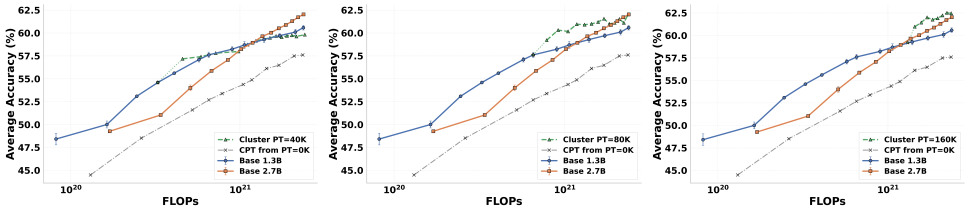
Results are presented in Figure 14. We see similar trends as in Figure 2a where at larger budgets, even the full split training performs better than full pretraining.

			Train			Test		
Domain	n_{train}	n_{test}	MAE	MRE	r	MAE	MRE	r
0	113	60	0.0021	0.0008	0.9993	0.0014	0.0005	0.9959
1	114	60	0.0023	0.0008	0.9945	0.0024	0.0009	0.9988
2	113	60	0.0020	0.0008	0.9995	0.0022	0.0009	0.9992
3	113	60	0.0026	0.0010	0.9989	0.0032	0.0013	0.9965
4	112	59	0.0020	0.0007	0.9993	0.0014	0.0005	0.9995
5	114	60	0.0015	0.0005	0.9994	0.0006	0.0002	0.9999
6	113	60	0.0020	0.0008	0.9995	0.0025	0.0011	0.9996
7	112	60	0.0016	0.0006	0.9997	0.0013	0.0006	0.9996
8	113	60	0.0053	0.0019	0.9393	0.0017	0.0008	0.9994
9	114	59	0.0031	0.0013	0.9954	0.0015	0.0007	0.9993
10	94	50	0.0020	0.0008	0.9996	0.0017	0.0007	0.9997
11	112	59	0.0014	0.0005	0.9998	0.0028	0.0010	0.9991
12	113	51	0.0010	0.0004	0.9999	0.0019	0.0008	0.9988
13	110	60	0.0016	0.0006	0.9996	0.0027	0.0011	0.9965
14	111	60	0.0018	0.0006	0.9994	0.0014	0.0005	0.9996
15	105	49	0.0086	0.0028	0.8697	0.0018	0.0007	0.9985
Avg			0.0026	0.0009	0.9871	0.0019	0.0008	0.9987

Table 8: Per-domain scaling law fit metrics for the 1.3B model. Scaling laws are trained with the CPT steps $<$ 20B tokens, and tested with CPT steps $\geq$ 20B tokens.

K 1.3B Model Size Zero-shot QA Training Curves

Prior experiments focus on performance at the end of training. However, it is unclear whether the training dynamics of split model training is similar to that of full pretraining. We compare three sets of models: a pretrained 1.3B model, sixteen 1.3B models with split training on DCLM clusters, and a larger 2.7B model which has twice the number of inference flops. We choose the three split points: $t_s = 40000$, where the model performs similarly to the fully pretrained model, $t_s = 80000$ corresponding to the compute allocation where the model performs similarly to the 2.7B parameter model, and $t_s = 160000$ our closest run to the estimated optimal allocation from our scaling law. We see that across all values of t_s the model has similar behavior. There is a sharp increase for the first few thousand steps of each split model, followed by a similar trend as the fully pretrained model afterwards.



(a) 40B PT (b) 80B PT (c) 160B PT

Figure 15: Performance of 1.3B models at the same training compute with different amount of pretraining steps. We compare split training with pretraining from scratch, split training with no pretraining, and finetuning (split training for only 1B tokens).

L Performance with Varying Number of Domains for 2.7B Models

We extend the experiments for the 1.3B parameter models wiht varying number of domains to the 2.7B setting. Results for the 2.7B parameter model training up to 300B tokens are summarized in Figure 16. We see that the 64 domain models perform the best at all split training points, although the performance is close to the 4 domain at low t_s and 16 domain at higher t_s. We also see that the performance improvement is relatively flat at later t_s for both the 1.3B and 2.7B indicating that only

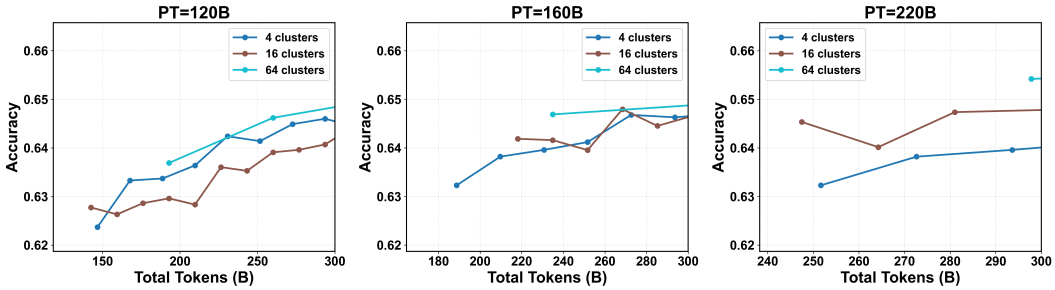


Figure 16: Accuracy vs. number of training tokens for varying number of domains and varying number of PT steps. All domain models are 2.7B parameter models. The total number of parameters varies with the number of domains, but the number of active parameters remains constant at a single domain model.

a small number of steps may be needed for more specialized domains. In many settings, it can also be expensive to route and maintain 64 domain models. Thus we find 16 domains to be the best tradeoff of performance at all scales, and number of parameters for practical settings.

M Individual Accuracies for Optimal LM Pretraining Allocation

We report individual task accuracies in Table 9.

Benchmark	Pretrain 1.3B	Split 1.3B	Pretrain 2.7B	Split 2.7B	Pretrain 7B
ARC-Challenge	39.25	**42.75**	40.70	**45.82**	42.06
ARC-Easy	68.06	**72.31**	70.62	**73.74**	72.94
BoolQ	**64.01**	64.19	65.78	**66.21**	63.12
HellaSwag	63.91	**67.30**	67.60	**69.37**	69.39
MMLU	35.59	**37.04**	37.42	**38.54**	38.74
PIQA	75.35	**78.29**	78.24	**78.29**	77.75
SciQ	90.00	**90.60**	89.20	**89.90**	91.40
WinoGrande	**61.88**	62.19	**63.85**	63.22	65.19

Table 9: QA benchmark accuracy for split model training with optimal token budget allocation.

N Cluster Ablations

Prior sections split train models with a fixed number of domains $K = 16$ and use all models at evaluation. Naturally there are several important questions to consider. In this section, we discuss whether all models are needed for inference, what the optimal number of domains is.

N.0.1 Two Cluster Routing Specialization

Methodology: Prior results train one model for each domain and route to all models at inference time. In some settings it may be infeasible to maintain a model for each domain in memory. In this section, we study specialization of domains at inference time where only two split models are used. This reduces the number of total parameters greatly. To select the domain models, we cluster the training sets of each of the QA tasks. We take the top two clusters from each dataset for a total of seven unique clusters. We refer to this as Task Subset. Alternatively, we could select a subset based on a single task which may be representative of a broader range of tasks. To this end, we also consider an MMLU subset formed from the top two clusters from the MMLU train set.

Findings: Our findings in Table 10 show that task specialization at evaluation reduces performance by around 1% compared to evaluation and training with the full clusters. The MMLU subset reduces performance more by up to 3%. This indicates that performing well across domains necessitates

	Subset Routing		Full
PT Tokens	MMLU Subset	Task Subset	Full
10B	53.85	55.85	56.94
20B	55.68	57.05	58.11
40B	57.47	59.08	59.53
80B	59.41	60.70	61.26
120B	59.62	60.97	61.53

Table 10: Routing comparisons for split training of a 1.3B model at 150B tokens. Subset routing uses only two models per task at inference time whereas full routing uses all domain models.

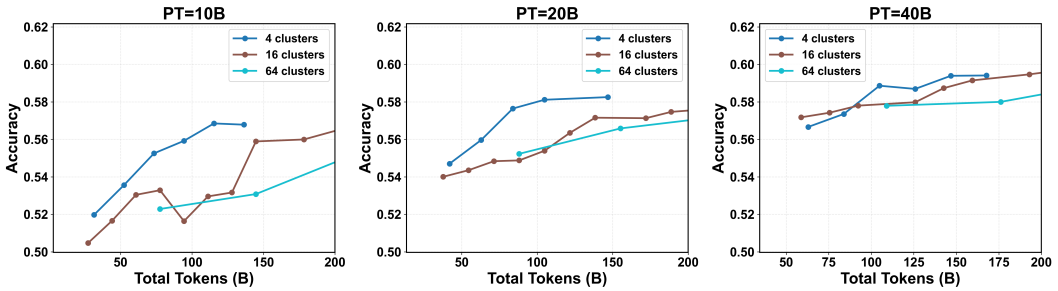


Figure 17: Accuracy vs. number of training tokens for varying number of domains and varying number of PT steps. All domain models are 1.3B parameter models. The total number of parameters varies with the number of domains, but the number of active parameters remains constant at a single domain model.

all models, however some task specialization at inference time still improves over a base model in situations where the total number of parameters may be limited.

N.0.2 Performance with Varying Number of Domains

Methodology: Our main results use 16 domains, however data may come with any number of domains. When the number of domains is small, the similarity may be large between the clusters, but the multiplicative factor to the number of steps will be reduced compared to more specific clusters. In contrast when the number of domains are large, each individual step on a domain improves the domain, but comes at a significant compute cost. We compare performance of split model training across number of domains. We train a separate clustering for each number of clusters.

Findings: Results for training up to 200B tokens for the 1.3B parameter model are summarized in Figure 17. We find that the 16 domain model maintains a good balance in accuracy at small and large number of PT tokens. At small number of PT tokens, the 16, and 64 domain models achieve low performance, and the 4 domain models perform the best. At larger t_s, the 4 domain models are matched by the 16 domain models. Results for the 2.7B parameter model training up to 300B tokens are summarized in Figure 16.

O Mixture of Experts Model Perplexity

We test whether our clustering-based split training can be extended with a Mixture-of-Experts (MoE) decoder-only Transformer. We train an MoE model that has 3.8B total parameters and 0.7B active parameters per token, and follows a scaled-down Qwen1.5-style MoE architecture (Team, 2024). Table 11 reports per-cluster perplexities, where the split MoE models are better in every cluster, and have an average relative improvement of 2.2%.

We report perplexity for each individual cluster for the split Mixture of Experts (MoE) model compared with a single MoE. We find that improvements from split model training are complementary as training with MoE improves over training a single model.

Cluster ID	Base PPL	Split MoE PPL
0	20.190	19.819
1	18.075	17.731
2	18.600	18.354
3	16.980	16.903
4	21.036	18.414
5	19.639	19.550
6	15.551	15.097
7	16.774	15.972
8	15.046	14.960
9	15.724	15.405
10	18.052	17.783
11	22.305	22.002
12	17.818	17.731
13	19.544	19.461
14	20.784	20.656
15	18.390	18.294
Average	**18.407**	**18.008**

Table 11: Per-cluster perplexity (PPL) comparing split training of Mixture of Expert (MoE) models with a single model pretrained on all clusters.