DoGE: Domain Reweighting with Generalization Estimation

Simin Fan [1] **Matteo Pagliardini** [1] **Martin Jaggi** [1]

Abstract

The coverage and composition of the pretraining data significantly impacts the generalization ability of Large Language Models (LLMs). Despite its importance, recent LLMs still rely on heuristics and trial and error to increase or reduce the influence of data-domains. We propose DOmain reweighting with Generalization Estimation (DoGE), which optimizes the probability of sampling from each domain (domain weights) in a principled way. Our approach is a two-stage process consisting of (i) training a *proxy model* to obtain domain weights using a bi-level optimization algorithm; (ii) training a larger *base model* by sampling training domains according to the learned domain weights. In our experiments, we extensively show how DoGE improves the generalization of the base model to any target data mixture. On the SlimPajama dataset, our base model gets better perplexity and few-shot reasoning accuracies across 6 tasks compared to baseline methods. Moreover, aiming to generalize to out-of-domain target tasks, which is unseen in the pretraining corpus (OOD domain), DoGE can effectively identify inter-domain dependencies, and consistently achieves better test perplexity on the target domain. We provide the codebase at https://github.com/Olivia-fsm/doge.

1. Introduction

Pretrained Large Language Models (LLMs) demonstrate impressive generalization abilities, making them the workhorse of today's NLP research and many practical use cases (Devlin et al., 2019; Brown et al., 2020; Chowdhery et al., 2022; Touvron et al., 2023a;b). They are trained on very large text corpora collected from various source domains to obtain a generalization ability, which enables an efficient adaptation to specific downstream tasks by fine-tuning. The composition of the pretraining corpus often depend on the accessibility of each data sources. For example, 72.6% of RedPajama (Together Computer, 2023) are sampled from CommonCrawl, while only 1.7% from Stackexchange. While recent research has demonstrated the significance of the quantity and quality of the pretraining corpus (Kaplan et al., 2020; Hoffmann et al., 2022; Longpre et al., 2023), there are few explorations into how its composition from various source domains could contribute to the generalization ability of the language model (Lee et al., 2023; Hashimoto, 2021; Xie et al., 2023a). The *domain weights* adopted by current state-of-the-art LLMs are mostly determined by heuristics (Gao et al., 2020) or tuned according to a series of downstream tasks (Du et al., 2022), which can be sub-optimal and costly.

Recently, Xie et al. (2023a) proposed a learnability-based domain reweighting framework DoReMi, which settles *domain weights* using two small-scale auxiliary models: first, a reference model is "well-trained" using uniform domain weights; next, a second auxiliary model—referred to as proxy model—is trained from scratch with the objective to find domain weights that minimize the worst-case *excess loss*, i.e. the per-domain loss gap between the proxy model and the well-trained reference model. The *excess loss* is interpreted as an estimation for the remaining learnability of a given domain at each training step—a large gap indicating the proxy model can further learn to model the associated domain. Despite the encouraging empirical results of DoReMi, minimizing the worst-case loss gap (i) creates a strong dependency on the well-trained model whose capacity can strongly influence the overall accuracy and requires appropriate tuning, and (ii) creates a dissonance between the ideal goal of minimizing the average validation loss across domains and the employed objective which seeks to simply mimic the well-trained model. Moreover, this approach cannot be used when the target domains are different from the training domains.

To mitigate these issues, we propose DOmain reweighting with Generalization Estimation (DoGE), which finds optimal *domain weight* distributions by explicitly optimizing for best generalization to a given set of domains. We follow the two-stage process of DoReMi which consists of first

[1]EPFL, Switzerland. Correspondence to: Simin Fan <simin.fan@epfl.ch>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

obtaining optimized domain weights by training a small-scale proxy model, and, in the second stage, training a final larger model on data sampled according to those weights. In contrast to DoReMi, DoGE only requires the training of one proxy model. Moreover, we found DoGE to be less dependent on the capacity of this proxy model (see § C.4).

When training the proxy model, at timestep t, we re-weight the gradient from each source domain to greedily minimize the average target domain loss at the next step $t+1$. Our derivation in § 2 shows that the resulting algorithm up-weights training domains with a large gradient alignment (inner-product) with the target domains, which reflects the principle:

> *A data domain should receive a large weight if it contributes to the learning of target domains.*

Similarly to DoReMi, the final domain weights are obtained by averaging the domain weights over the training of the proxy model. The base model is then trained by sampling its training data according to the final domain weights. A visual overview of the DoGE method is shown in Fig. 1.

Contributions. We summarize our contributions as follows:

- We introduce and rigorously derive DoGE, an efficient and effective domain reweighting framework, which explicitly aims to generalize to a specific set of target domains (§ 2);
- We empirically show that our method outperforms strong baselines including DoReMi in terms of (i) average perplexity, and (ii) few-shot reasoning capabilities across 6 tasks (§ 3.1);
- We show how DoGE can handle cases where the target domains are different from the training domains, and consistently outperforms the baseline with uniform domain weights (§ 3.2).

2. Domain Reweighting with Generalization Estimation

In this section, we motivate and derive DoGE, for the goal of re-weighting training domains $D_{train} \triangleq \{D_1, \dots, D_k\}$ to improve the model's generalization to a given set of target domains. We distinguish two scenarios for generalization: (1) **Universal generalization**, where the target objective is to minimize the validation loss across all source domains $\{D_1, \dots, D_k\}$; as well as (2) **Out-of-domain generalization** where we aim at minimizing the validation loss on a specific target domain (D_{ood}), while $D_{ood} \notin D_{train}$. The first case applies in most of the scenarios for LLM pretraining, where no specific downstream target has been set. The later case is especially relevant when considering generalization to specific target domain datasets (e.g. science, low-resource languages) which are too small to have a significant impact when used during pretraining.

Setup & notation. Let $D_{train} \triangleq \{D_1, \dots, D_k\}$ be a large corpus split into k domains according to meta-attributes (e.g. source, topic). We aim to find ***domain weights*** over the probability simplex $\boldsymbol{\alpha} \in \Delta^k \subset \mathbb{R}^k$. The final data mixture used to train the full-size language model is constructed by first sampling a domain according to the domain-wise distribution $\boldsymbol{\alpha}$, followed by uniformly sampling a batch B from that domain ($B \sim \text{UNIF}(D_i)$). Overall, this leads to the instance-wise distribution $P_{\boldsymbol{\alpha}} \triangleq \sum_{i=1}^{k} \alpha_i \cdot \text{UNIF}(D_i)$. In the following, we will describe how to optimize $\boldsymbol{\alpha}$ guided by training a proxy model of parameters $\boldsymbol{\theta}$ on D_{train}. We denote by $l_i(\boldsymbol{\theta})$ the expected next token prediction loss of the proxy model on domain D_i. Let $\bar{l}(\boldsymbol{\theta}) \triangleq \frac{1}{k}\sum_{i\in[k]} l_i(\boldsymbol{\theta})$ be the average loss across all k domains. Let $|D|$ refer to the number of samples in D.

Universal generalization. In the case of universal generalization, our goal is to minimize $\bar{l}(\boldsymbol{\theta})$. This posit that all k given training domains have the same importance. As a point of comparison, note that the classical loss used to train large language models is $\tilde{l}(\boldsymbol{\theta}) = \sum_{i\in[k]} \frac{|D_i|}{|D_{train}|} l_i(\boldsymbol{\theta})$ which could severely bias to domains with larger scale. One naive approach could consist in re-weighting samples by the inverse of the sampling probability: $\tilde{l}(\boldsymbol{\theta}) = \sum_{i\in[k]} \tilde{\alpha}_i \frac{|D_i|}{|D_{train}|} l_i(\boldsymbol{\theta})$ with $\tilde{\alpha}_i = \frac{|D_{train}|}{|D_i|}$, however, this approach ignores everything of the complex intra-domain interactions considering the nature of the textual corpus which (i) have inevitable lexical, syntactic or semantic overlaps, and (ii) can be more or less challenging to learn. In practice, this naive uniform sampling approach provides a strong baseline but often hinders the generalization compared to other methods (see § 3).

We instead propose to optimize domain weights $\boldsymbol{\alpha} \in \Delta^k$ along the training of the proxy model $\boldsymbol{\theta}$, as a stochastic bi-level optimization problem:

$$\boldsymbol{\alpha} \in \arg\min_{\boldsymbol{\alpha}\in\Delta^k} \sum_{i\in[k]} l_i(\boldsymbol{\theta}^\star(\boldsymbol{\alpha}))$$
$$s.t.\ \boldsymbol{\theta}^\star(\boldsymbol{\alpha}) \in \arg\min_{\boldsymbol{\theta}} \sum_{i\in[k]} \alpha_i l_i(\boldsymbol{\theta})$$

In the inner loop (1), the proxy model $\boldsymbol{\theta}(\boldsymbol{\alpha})$ is updated using the rescaling factor $\boldsymbol{\alpha}$; in the outer loop (2), we update $\boldsymbol{\alpha}$ to adapt to the target given the updated model status. To avoid complicated multi-step gradient unrolling, we only update $\boldsymbol{\theta}$ in the inner optimization problem over a single stochastic step:

$$\boldsymbol{\theta}^{(t+1)} \triangleq \boldsymbol{\theta}^{(t)} - \eta^{(t)} \sum_{i\in[k]} \alpha_i^{(t)} \nabla l_i(\boldsymbol{\theta}^{(t)}) \qquad (1)$$

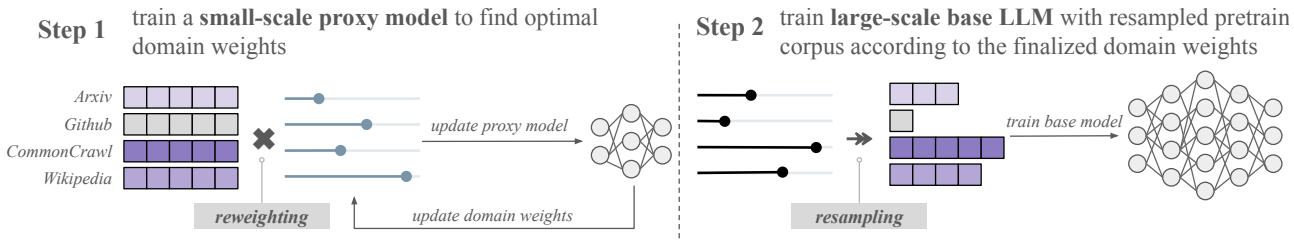


Figure 1. **Summary of DoGE .** Our method consists of two steps. In **Step 1**, we learn domain weights which maximize the generalization of the proxy model to the target domains. The finalized domain weights are then used in **Step 2** to resample data from each domain to build a new pretraining data mixture for a larger-scale base model.

where $\boldsymbol{\alpha}^{(t)} \in \Delta^k$ is used to re-weight the loss from each domain at time-step t, $\eta^{(t)}$ is the step size, and $\nabla l_i(\boldsymbol{\theta}^{(t)})$ is a stochastic gradient for samples of D_i. The outer-loop in bi-level optimization techniques usually requires second-order derivatives (Grangier et al., 2023; Zhou et al., 2023), which could introduce huge computation costs. Instead, we update $\boldsymbol{\alpha}$ by a simpler fully first-order rule, which allows to reuse the gradients from the inner-loop.

Specifically, the update rule of the domain weights $\boldsymbol{\alpha}$ can be derived as follows. Denote $\Delta\boldsymbol{\theta}^{(t)} = \boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}$, at step t, we aim to find the optimal $\boldsymbol{\alpha}^{(t)}$ to minimize the original unweighted domain loss at the next step:

$$\begin{aligned}
\boldsymbol{\alpha}_\star^{(t)} &= \arg\min_{\boldsymbol{\alpha}\in\Delta^k} \bar{l}(\boldsymbol{\theta}^{(t+1)}) \\
&= \arg\min_{\boldsymbol{\alpha}\in\Delta^k} \sum_{i\in[k]} [l_i(\boldsymbol{\theta}^{(t+1)}) - l_i(\boldsymbol{\theta}^{(t)})] \qquad (2)\\
&= \arg\min_{\boldsymbol{\alpha}\in\Delta^k} \sum_{i\in[k]} \langle \nabla l_i(\boldsymbol{\theta}^{(t)}), \Delta\boldsymbol{\theta}^{(t)} \rangle + o(\|\Delta\boldsymbol{\theta}^{(t)}\|) \\
&= \arg\min_{\boldsymbol{\alpha}\in\Delta^k} \sum_{i\in[k]} \Big\langle \nabla l_i(\boldsymbol{\theta}^{(t)}), -\eta^{(t)} \sum_{j\in[k]} \alpha_j \nabla l_j(\boldsymbol{\theta}^{(t)}) \Big\rangle + \boldsymbol{\varepsilon}^{(t)},
\end{aligned}$$

where $\boldsymbol{\varepsilon}^{(t)} = o(\|\Delta\boldsymbol{\theta}^{(t)}\|) = o(\|\boldsymbol{\alpha}^{(t)}\|)$ (1), which is a high-order remainder in the Taylor expansion. Let $W_j^{(t)} \triangleq \langle \nabla l_j(\boldsymbol{\theta}^{(t)}), \sum_{i\in[k]} \nabla l_i(\boldsymbol{\theta}^{(t)}) \rangle$ be the stochastic *generalization estimation function* on the j^{th} domain. Intuitively, this quantity measures the alignment of the learning tasks across domains: a high $W_j^{(t)}$ means learning D_j will also contribute to learning other domains. We write $\mathcal{W}^{(t)} = [W_1^{(t)}, \ldots, W_k^{(t)}] \in \mathbb{R}^k$ for the vectorized generalization estimation scores across all domains. We can rewrite the outer loop update (2) simply as:

$$\boldsymbol{\alpha}_\star^{(t)} = \arg\min_{\boldsymbol{\alpha}\in\Delta^k} -\eta^{(t)} \boldsymbol{\alpha}^\top \mathcal{W}^{(t)} + \boldsymbol{\varepsilon}^{(t)} \,. \qquad (3)$$

We solve (3) by estimating $\boldsymbol{\varepsilon}^{(t)}$ as the Bregman divergence $D_\Psi(\boldsymbol{\alpha}\|\boldsymbol{\alpha}^{(t-1)})$ with $\Psi(\boldsymbol{\alpha}) = \sum_i \alpha_i \log(\alpha_i)$, which is a common technique in mirror descent (Nemirovski & Yudin, 1983; Beck & Teboulle, 2003) :

$$\boldsymbol{\alpha}^{(t)} = \arg\min_{\boldsymbol{\alpha}\in\Delta^k} -\eta^{(t)} \boldsymbol{\alpha}^\top \mathcal{W}^{(t)} + \mu D_\Psi(\boldsymbol{\alpha}\|\boldsymbol{\alpha}^{(t-1)}), \quad (4)$$

with μ as a hyperparameter controls the strength of regularization. This yields the following multiplicative weights update rule, see e.g. (Beck & Teboulle, 2003):

$$\boldsymbol{\alpha}^{(t)} = \frac{\hat{\boldsymbol{\alpha}}^{(t)}}{\sum_{i\in[k]} \hat{\alpha}_i^{(t)}}, \qquad (5)$$

with $\hat{\boldsymbol{\alpha}}^{(t)} = \boldsymbol{\alpha}^{(t-1)} \odot \exp\Big(\frac{\eta^{(t)} \mathcal{W}^{(t)}}{\mu}\Big)$. We estimate the average domain loss $\sum_{i\in[k]} \nabla l_i(\boldsymbol{\theta}^{(t)})$ by sampling another batch consisting of instances uniformly sampled from all domains. At each time-step t, we alternatively update $\boldsymbol{\alpha}^{(t)}$ and $\boldsymbol{\theta}^{(t)}$. The final algorithm is summarized in Alg. 1. The detailed derivation is presented in Appendix (§ B).

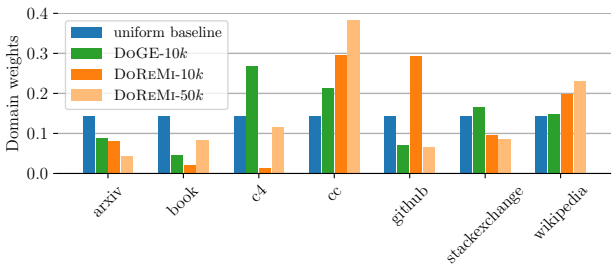
Out-of-domain generalization. In the out-of-domain generalization scenario we want to generalize to a D_{ood} domain that is not part of D_{train}. The above derivation still holds only with minor modifications: (i) we are now considering our objective to be $l_{ood}(\boldsymbol{\theta})$ instead of $\bar{l}(\boldsymbol{\theta})$, and (ii) we now have $W_j^{(t)} \triangleq \langle \nabla l_j(\boldsymbol{\theta}^{(t)}), \nabla l_{ood}(\boldsymbol{\theta}^{(t)}) \rangle$, for clarity we call $\mathcal{W}_{ood}^{(t)} = [W_1^{(t)}, \ldots, W_k^{(t)}]$. The update of $\boldsymbol{\alpha}^{(t)}$ is the same as in (5) replacing $\mathcal{W}^{(t)}$ with $\mathcal{W}_{ood}^{(t)}$. The associated algorithm can be seen in App. B (See Alg. 2), where all differences with universal generalization (Alg. 1) are highlighted in blue.

Link between $\mathcal{W}^{(t)}$ and influence functions. Following (Pruthi et al., 2020), given samples from a source and target domain $B_s \sim D_s$ and $B_t \sim D_t$, the influence of D_s on D_t can be estimated by $\mathcal{I}(B_s, B_t) = \langle \nabla l_s(\boldsymbol{\theta}), \nabla l_t(\boldsymbol{\theta}) \rangle$. Considering the definition of $W_j^{(t)}$:

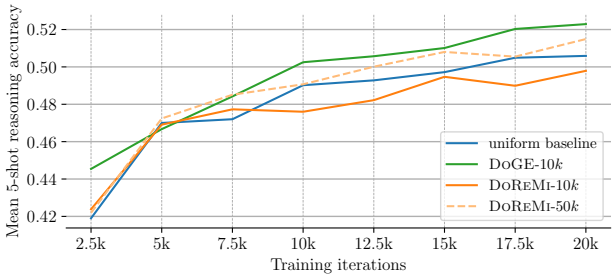
$$\begin{aligned}
W_j^{(t)} &\triangleq \langle \nabla l_j(\boldsymbol{\theta}^{(t)}), \sum_{i\in[k]} \nabla l_i(\boldsymbol{\theta}^{(t)}) \rangle \qquad (6)\\
&= \underbrace{\langle \nabla l_j(\boldsymbol{\theta}^{(t)}), \sum_{i\in[k], i\neq j} \nabla l_i(\boldsymbol{\theta}^{(t)}) \rangle}_{\text{out-of-domain influence}} + \underbrace{\|\nabla l_j(\boldsymbol{\theta}^{(t)})\|_2^2}_{\text{domain difficulty}}
\end{aligned}$$

The first term in (6) estimates the sum of influences from all the other $k-1$ domains on the j^{th} domain, while the second term denotes the magnitude of the gradient from domain D_j. Intuitively, a domain should be up-weighted when (i) it contributes to the learning of other domains (high out-of-domain influence), or (ii)—in the universal generalization case—when the domain itself has not been learnt enough (high magnitude of gradient for this domain). Those two mechanisms are precisely what Equ. (3) expresses.

Training the base model. Given the final domain weights $\bar{\boldsymbol{\alpha}}$, we train the full size model by sampling D_{train} according to $P_{\bar{\boldsymbol{\alpha}}} \triangleq \sum_{i=1}^{k} \bar{\alpha}_i \cdot \text{UNIF}(D_i)$.



(a) Domain weights



(b) Average reasoning accuracy

Figure 2. **Universal generalization results on SlimPajama.** In **(a)** we compare several domain weight distributions obtained by DoGE and DoReMi. We show two distributions for DoReMi obtained by training the auxiliary models for $10k$ and $50k$ iterations. DoGE's proxy model has been trained for $10k$ iterations. In **(b)** we plot the average 5-shot accuracy during the base model training. DoGE acquires few-shot reasoning ability faster than all other baseline methods and improves the final average accuracy by a large margin.

3. DOGE Improves Generalization

In this section, we show how DoGE is reweighting the source domains to improves the model's performances in both universal generalization and out-of-domain generalization settings.

3.1. Universal Generalization

In the case of universal generalization, we aim to improve the model's generalization across all domains present in the training set. We measure the average perplexity across all domains and 5-shot reasoning ability across a series of reasoning tasks, covering diverse knowledge fields including physics, social science, logic inference etc.: COPA (Gordon et al., 2012), SciQ (Welbl et al., 2017), PIQA (Bisk et al., 2019), LogiQA (Liu et al., 2020), WiC (Pilehvar

Algorithm 1 DoGE Domain Reweighting (for Universal Generalization).

Input: Domain data splits $D_1, \ldots, D_k$, Proxy model weights $\boldsymbol{\theta}^{(0)}$, Hyperparameters: number of training steps T, batch size b, step size $\eta^{(t)}$, Bregman coefficient μ.
Initialize proxy weights $\boldsymbol{\theta}^{(0)}$
Initialize proxy domain weights $\boldsymbol{\alpha}^{(0)} = \frac{1}{k}\mathbf{1}$
for $t \in [T]$ **do**
  Uniformly sample batch $B^{(t)} = \{B_1^{(t)}, \ldots, B_k^{(t)}\}$
  Obtain $\nabla l_i(\boldsymbol{\theta}^{(t)}, B_i^{(t)})$ for $i \in [k]$
  Compute $\mathcal{W}^{(t)}$
  Update domain weights according to Eq. (5):
    $\hat{\boldsymbol{\alpha}}^{(t)} \leftarrow \boldsymbol{\alpha}^{(t-1)} \odot \exp(\eta^{(t)}\mathcal{W}^{(t)}/\mu)$
    $\boldsymbol{\alpha}^{(t)} \leftarrow \hat{\boldsymbol{\alpha}}^{(t)} / \sum_{i=1}^{k} \hat{\alpha}_i^{(t)}$
  Update $\boldsymbol{\theta}^{(t)}$:
    $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta^{(t)} \sum_{i\in[k]} \alpha_i^{(t)} \nabla l_i(\boldsymbol{\theta}^{(t)}, B_i^{(t)})$
end for
Return Domain weights $\bar{\boldsymbol{\alpha}} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\alpha}^{(t)}$

Algorithm 2 DoGE Domain Reweighting (for Out-of-domain Generalization).

Input: Training domain data splits $D_1, \ldots, D_k$, OoD domain D_{ood}, Proxy model weights $\boldsymbol{\theta}^{(0)}$, Hyperparameters: number of training steps T, batch size b, step size $\eta^{(t)}$, Bregman coefficient μ.
Initialize proxy weights $\boldsymbol{\theta}^{(0)}$
Initialize proxy domain weights $\boldsymbol{\alpha}^{(0)} = \frac{1}{k}\mathbf{1}$
for $t \in [T]$ **do**
  Uniformly draw $B^{(t)} = \{B_1^{(t)}, \ldots, B_k^{(t)}\} \cup \{B_{ood}^{(t)}\}$
  Obtain $\nabla l_i(\boldsymbol{\theta}^{(t)}, B_i^{(t)})$ for $i \in [k]$
  Obtain $\nabla l_{ood}(\boldsymbol{\theta}^{(t)}, B_{ood}^{(t)})$
  Compute $\mathcal{W}_{ood}^{(t)}$
  Update domain weights according to Eq. (5):
    $\hat{\boldsymbol{\alpha}}^{(t)} \leftarrow \boldsymbol{\alpha}^{(t-1)} \odot \exp(\eta^{(t)}\mathcal{W}_{ood}^{(t)}/\mu)$
    $\boldsymbol{\alpha}^{(t)} \leftarrow \hat{\boldsymbol{\alpha}}^{(t)} / \sum_{i=1}^{k} \hat{\alpha}_i^{(t)}$
  Update $\boldsymbol{\theta}^{(t)}$:
    $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta^{(t)} \sum_{i\in[k]} \alpha_i^{(t)} \nabla l_i(\boldsymbol{\theta}^{(t)}, B_i^{(t)})$
end for
Return Domain weights $\bar{\boldsymbol{\alpha}} = \frac{1}{T}\sum_{t=1}^{T} \boldsymbol{\alpha}^{(t)}$

& Camacho-Collados, 2019) and WinoGrande (Sakaguchi et al., 2019). We use *LM-eval Harness* (Gao et al., 2021) to assess the few-shot reasoning performance.

Training setup. We experiment on SlimPajama (Soboleva et al., 2023), which is a deduplicated version of RedPajama consisting of data from 7 domains. We train a small 82M decoder-only transformer (Vaswani et al., 2023) as the *proxy model* for domain reweighting. Auxiliary models for both DOGE and DOREMI are trained for $10k$ iterations. We also experiment with training the auxiliary models of DOREMI for $50k$ steps, giving that baseline a strong advantage. The final domain weights are used to train larger *base models* (124M, 210M, 684M). We refer to those three methods as DOGE-$10k$, DOREMI-$10k$ and DOREMI-$50k$. We also compare to the BASELINE with uniform domain weights, which is the best heuristic for universal generalization without prior knowledge on inter-domain relatedness. We report domain weights from DOGE-$10k$ as the average of three random seeds. All models are trained from scratch with batch size of 128, and sequence length of 512. The vocabulary size of the tokenizer is 50304. Details on model architectures are provided in App. A.

Evaluation on language modeling ability. We measure the per-domain perplexity on held-out validation sets for the largest scale base model (684M). Results for other model sizes (124M and 210M) are provided in App. C. According to Tab. 1, DOGE-$10k$ outperforms BASELINE and DOREMI-$10k$ in 5 out of 7 domains, by a large margin. Notably, DOGE-$10k$ outperforms all the other baseline methods in terms of average perplexity, given a great advantage in the number of iterations to train DOREMI-$50k$.

Evaluation on few-shot reasoning accuracy. We test the 5-shot reasoning accuracy across 6 tasks for our largest (684M) models. According to Tab. 2 and Fig. 2.(b), DOGE-$10k$ improves few-shot reasoning ability of the *base model*, especially at the early training stage. Our method outperforms the BASELINE and DOREMI-$10k$ on all 6 tasks. In contrast, DOREMI-$10k$ slightly hurts the reasoning accuracy. With $40k$ more training iterations, DOREMI-$50k$ outperforms uniform BASELINE on most of the tasks, while still left behind by DOGE-$10k$ on 5 out of 6 reasoning tasks. On average, DOGE-$10k$ improves the 5-shot reasoning ability by 1.7 accuracy points over uniform BASELINE, which outperforms all the other methods.

Evolution of domain weights. Fig. 3 shows the step-wise **(Bottom)** and average **(Top)** domain weights evolution during the training of the proxy model. The step-wise domain weights can be interpreted as the online contributions from each domains, while the final domain weights $\bar{\alpha}$ are given by the average. According to Fig. 3.a and Fig. 3.d, DOGE shows a clear phase transition, with different stages of training, as in a curriculum: in an early stage, DOGE up-weights Arxiv and Stackexchange while gradually up-weighting C4, CC and Wikipedia, which contain a more diverse lexical coverage and complicated semantics. The domain-weights for the other two domains (GitHub and Book) are kept low. We hypothesise that Github—with its emphasis on code—has limited vocabulary and semantic knowledge, and the complexity of Book might be covered by C4 and CC, which are the two most up-weighted domains. In comparison, the step-wise dynamic of DOREMI-$10k$ in Fig. 3.e and DOREMI-$50k$ in Fig. 3.f oscillate greatly during training. Despite the additional training steps for both auxiliary models in DOREMI-$50k$, the final average domain weights differ greatly with the ones from DOREMI-$10k$ (by a mean absolute difference of 0.08), which indicates the strong dependency of DOREMI on the capacity of the reference model and training iterations. We present the final domain weights adopted by different methods in Fig. 2.a.

Robustness to the scale of proxy model. To further explore the how the scale of the proxy model could impact the final domain weights, we run the ablation experiments on three different scales (60M, 82M, 124M). Notably, DOGE's final domain weights are consistent across various scale of proxy model. The mean absolute difference of domain weights between 60M (resp. 124M) and 82M proxy models is less than 0.015 (resp. 0.005) across all 7 domains, which demonstrates the robustness of our method. Compared to DOREMI, DOGE has less dependencies on the capacity on the auxiliary model(s) which requires less efforts and costs to tune the size of the proxy model and choose the number of iterations. The details of the ablation experiments are presented in App. C.4.

Comparison of Computation Overhead. Our experiments show DOGE to be more memory, time, and data efficient than DOREMI. Indeed, DOREMI requires two auxiliary models of the same scale, while DOGE only requires a single proxy model. Moreover, while $10k$ steps were sufficient for DOGE to improve the perplexity and few-shot reasoning accuracies over the uniform baseline, DOREMI required $5\times$ more tokens and $10\times$ more floating point operations.

3.2. Out-of-Domain Generalization

In the case of Out-of-Domain (OoD) generalization, we aim to improve the model's generalization to a target domain which is not part of the training mixture D_{train}. Given the target domain is missing from D_{train}, we expect DOGE to up-weigh the helpful domains among D_{train} while sampling less from distinct ones. We consider two dataset: SlimPajama and Wiki40b. Since DOREMI does not support this use-case, we only compare DOGE with BASELINE with uniform domain weights. The ORACLE baseline also enables access to the target domain, which shares the same sampling weight as other source domains in D_{train}. We

Table 1. **Per-domain Perplexity for universal generalization with 684M parameter models.** We compare DoGE-10k with DoReMi-10k, DoReMi-50k and a Baseline with uniform domain weights. We measure the perplexities on validation sets for all of the 7 domains of SlimPajama. DoGE outperforms the uniform baseline on 5 out of 7 domains and achieves the best average perplexity over all baseline methods. Scores outperforming the baseline are in **Bold**. The average perplexity is calculated as the exponential of the average loss across all domains instead of the average of all domain perplexities.

Domain	Uniform baseline	DoReMi-10k	DoGE-10k	DoReMi-50k
Arxiv	**8.105**	8.698	8.207	9.378
Book	44.990	50.594	**44.574**	42.557
C4	49.066	56.116	**42.558**	41.388
CommonCrawl	45.903	46.459	**40.432**	41.067
Github	3.944	**3.739**	4.107	4.301
Stackexchange	8.628	9.022	**8.332**	9.235
Wikipedia	12.047	**11.380**	11.443	10.519
Average	16.526	17.172	**15.806**	16.124
Worst-case	49.066	56.116	**44.574**	42.557
# domains outperform Baseline	/	2	5	4

Table 2. **Exact-match accuracies(%) for 5-shot reasoning tasks.** In 5 out of 6 tasks, DoGE reaches the best accuracy compared to other baseline methods. Only DoReMi-50k slightly outperforms DoGE on PIQA, using 40k more steps to train the auxiliary models.

Task	Uniform baseline	DoReMi-10k	DoGE-10k	DoReMi-50k
COPA	58.00	59.00	**62.00**	61.00
SciQ	61.80	60.30	**65.00**	64.50
LogiQA	23.20	24.58	**25.50**	23.81
PIQA	59.85	56.86	60.34	**60.94**
WiC	**49.69**	48.59	**49.69**	49.53
WinoGrande	50.99	49.41	**51.22**	49.17
Average	50.59	49.79	**52.29**	51.49

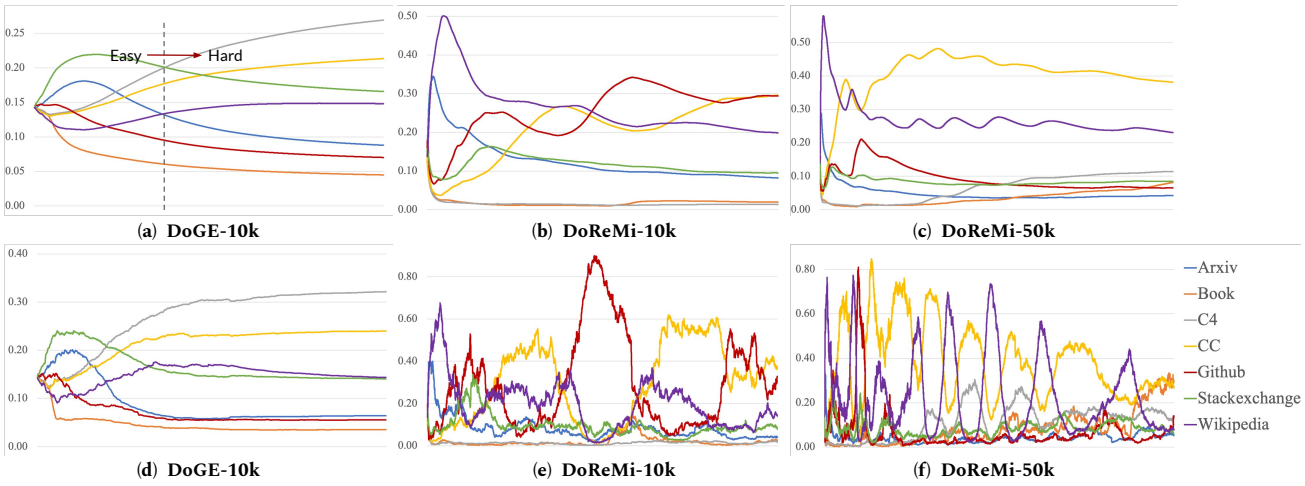
Table 3. **Out-of-Domain generalization results on SlimPajama.** DoGE significantly outperforms the uniform averaging baseline in all domains. This demonstrates DoGE's ability to select helpful sources of data for the given target. Domain weights for each target domain are present in Fig. 4.c. Even when finetuning the pretrained models on the target domain, DoGE models reach a better perplexity. Performance better than the baseline are highlighted in **bold**.

	Baseline (w/o target)	DoGE	Baseline (w/o target)+fine-tuning	DoGE+fine-tuning	Oracle (with target)
Arxiv	18.92±0.14	**16.70**±0.08	10.47±0.01	**10.20**±0.01	9.78±0.01
Book	82.57±0.05	**63.89**±0.18	65.73±0.06	**56.94**±0.24	66.43±0.19
C4	89.56±0.38	**63.96**±0.11	71.24±0.09	**56.91**±0.17	70.69±0.14
CommonCrawl	81.65±0.47	**57.77**±0.56	65.75±0.01	**51.173**±0.04	67.06±0.15
Github	6.675±0.00	**5.091**±0.03	4.99±0.01	**4.26**±0.01	4.97±0.01
StackExchange	16.941±0.02	**14.77**±0.01	11.24±0.004	**10.98**±0.002	11.26±0.03
Wikipedia	58.04±0.32	**53.87**±0.35	18.38±0.02	**17.71**±0.05	17.61±0.02

assess the target domain perplexity on the held-out test set and report average results over two seeds.

Wiki40b setup. We test the OOD-generalization capabilities of DoGE in a multilingual setting, aiming to facilitate low-resource language learning from mainstream language corpus. We use the Wiki40b dataset (Guo et al., 2020), which consists in a collection of Wikipedia articles in 40+ languages. We set English, German, Spanish, French and Russian as source domains in D_{train}. The target domain is set to Catalan or Dutch, which are considered as low-resource languages. We train the proxy model (82M) for 10k steps to obtain the domain weights and then train the base model (124M) for 10k steps.

SlimPajama-OoD setup. For out-of-domain generalization, we set each of the domains in SlimPajama as the target domain, with 0.05B tokens accessible. The remaining 6 domains are used as source domains D_{train}, each with 2B tokens accessible. We run the proxy model (82M) for 10k steps to obtain the domain weights and then train the base model (124M) for 10k steps. We continually fine-tunine the pretrained checkpoints for 1000 steps on the target domain.



(a) DoGE-10k (b) DoReMi-10k (c) DoReMi-50k

(d) DoGE-10k (e) DoReMi-10k (f) DoReMi-50k

Figure 3. **Average (Top row) and step-wise (Bottom row) domain weights evolution.** We train the auxiliary models of DoReMi for $10k$ or $50k$ steps, which yields DoReMi-$10k$ and DoReMi-$50k$. In **DoGE (a, d)**, we observe a clear two-phase curriculum from *easy* to *hard* domains. In contrast, the step-wise domain weights from both **DoReMi-$10k$ (e)** and **DoReMi-$50k$ (f)** oscillate greatly during training. The final average domain weights differ a lot between **DoReMi-$10k$ (b)** and **DoReMi-$50k$ (c)**, which reveals a strong dependency on the capacity of auxiliary models and the training iterations.

Perplexity on the target domain. In Tab. 3, we show how DoGE consistently outperforms the uniform baseline across all seven domains in SlimPajama. On C4 and CommonCrawl, DoGE achieves a better performance than the oracle without further fine-tuning. This demonstrates that irrelevant data sources can deteriorate the adaptation to the target domain, and DoGE can help to select helpful source domains. After finetuning the pretrained checkpoints on the target domain, DoGE pretrained models still outperform the finetuned baseline. In Fig. 4.a and Fig. 4.b, we show the test perplexity on Catalan and Dutch when training the base models. DoGE models show a significant improvement over the uniform baseline by learning from related mainstream languages.

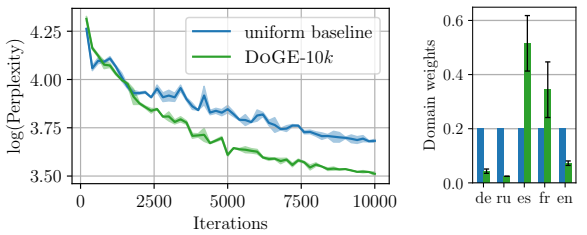
Automatically detected inter-domain affinities. For the SlimPajama experiments, we presents the auto-detected inter-domain affinities obtained by DoGE in Fig. 4.c. There is a clear inter-dependency between C4 and CommonCrawl, which are both web-crawled data sources; meanwhile, the strong affinity between Stackexchange and Github is also detected, which both contain code-related knowledge. Similarly, the domain weights obtained in multilingual experiments reflects the languages relatedness in etymology (Cole & Siebert-Cole, 2022), where Catalan is close to French and Spanish in Italic family, while Dutch is close to German in Germanic family. (Fig. 4.a and Fig. 4.b).

4. Discussion and Limitations

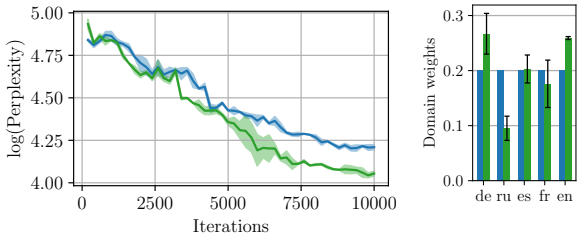
Stage-wise domain weights is no better than global average. Following the success of curriculum learning (Hacohen & Weinshall, 2019; Xu et al., 2020; Fan & Jaggi, 2023) in multiple fields, we explore the potential of applying stage-wise time-varying domain weights during the training of the base model. We manually divided the training process of the proxy model into $K = 2, 3, 10$ stages and average the step-wise domain weights respectively to be the stage-wise domain weights (Fig. 10). By applying stage-wise domain weights, the total amount of samples from each domain are the same as the global domain weights. As shown in App. E, none of the time-varying strategies show clear improvement over the global averaged domain weights in average perplexity. However, with $K = 2, 3$, the stage-wise domain weights help the model learn *hard* domains (Wikipedia, CC, C4) better, which aligns with the principle of curriculum learning. With $K = 10$, the domain weights updates every 1000 steps, while the performance of the base model is much worse than applying static domain weights.

The proxy model performs worse than a same-scale base model. With the step-wise dynamic adaptation, it is expected that the proxy model with the rescaled gradient could outperform the base model trained with the learned domain weights. However, compared with a base model with the same scale (82M), we find that the proxy model consistently performs worse in validation perplexity (see App. C.5). A similar behavior is also mentioned by Xie et al. (2023a), where both auxiliary models (the reference and the proxy) in DoReMi cannot reach comparable performance to the same-scale base model with resampling.

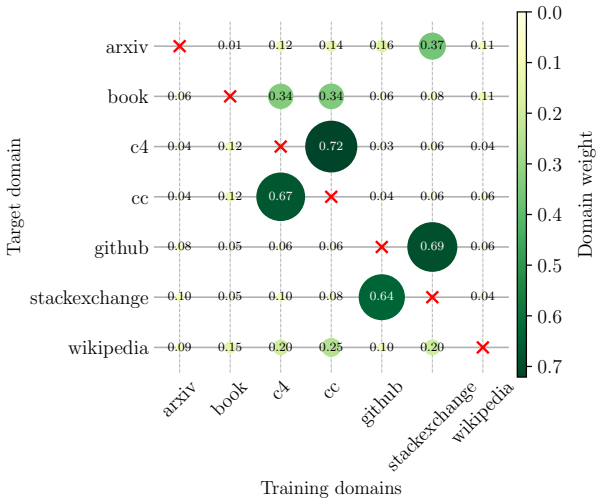
Better efficiency using parameter selection. The computation budget for generalization estimation $\mathcal{W}$ is quadratic to the scale of model. Thus, we explore the potential of pa-



(a) OoD generalization to Catalan



(b) OoD generalization to Dutch



(c) OoD generalization weights on SlimPajama

Figure 4. **Out-of-Domain generalization results.** In **(a)** (resp. **(b)**), we compare DoGE with the uniform domain weights baseline when attempting to generalize to Catalan (resp. Dutch) from a mixture of German, Russian, Spanish, French and English wikipedia articles from the Wiki40b dataset. The target languages are absent from the training mixture. The histograms show DoGE up-weights training languages having some similarity with the target. As a result, DoGE's loss on the target is decreasing faster than the baseline. In **(c)** we show DoGE domain weights for OoD generalization on the SlimPajama dataset. Those domain weights result from training mixtures consisting of all the training domains except for one which is used as the target (D_{ood}) domain. Each row represents a distribution returned by Alg. 2. The target domain is not used during training and hence is marked by a red cross. The weight distributions look very coherent, e.g. to generalize to GitHub, DoGE upweights stackexchange which contains a significant fraction of code. Similarly, to generalize to cc, the c4 domain—which also consists in web data—is up-weighted.

rameter selection based on cancellation effect following the empirical success of (Yeh et al., 2022). Specifically, we rank all parameter modules (i.e. transformer blocks or embedding layers) of model weights by the cancellation effect and only use gradients of the selected modules when compute $\mathcal{W}$. Among the five parameter selection strategies, selecting 30 modules with highest cancellation effect achieve the comparable average perplexity with only 2.5% computation costs for generalization estimation $\mathcal{W}$. We provide the details of parameter selection in Appendix (§ F).

5. Related Work

Data Selection for Language Modeling. Many works show how a rigorously selected training corpus can effectively improve downstream performance with fewer training tokens. Longpre et al. (2023) discover a trade-off between a model's toxic generalization behavior and its generalization ability by applying quality control with various thresholds. Gunasekar et al. (2023) and Li et al. (2023) trained a 1.3B model PHI-1 using 7B *text-book quality* code data, outperforming previous larger models trained on larger dataset, illustrating the potential of high-quality data.

However, due to scalability issues, most conventional data selection methods fail to be applicable for LLM pretraining. Data attribution methods based on influence function rank the entire set of data samples at a fixed time step and often require second-order computations, which can be intractable on gigantic pretraining corpus (Koh & Liang, 2020; Agarwal et al., 2017; Guo et al., 2021; Kwon et al., 2024). Classifier-based data filtering techniques are commonly used to construct a pretraining corpus (Gao et al., 2020; Penedo et al., 2023). Everaert & Potts (2023) propose GIO to select a subset that minimizes the KL-divergence to the target distribution, yet incurs high computation complexity. Xie et al. (2023b) present a scalable importance resampling strategy by reducing dimensionality into an n-gram-featured subspace, which risks from a weak representation for sophisticated semantics. Engstrom et al. (2024) train a linear datamodel first to predict a mapping from training dataset to downstream loss, then select a subset to minimize the approximated loss.

Data Reweighting for LLM Pretraining. Instead of selecting a subset, data reweighting remain the full access to the whole dataset while re-scale the contribution of each instance under various target tasks. Grangier et al. (2023) train an extra weighting network to re-weight the loss from each data point using bilevel optimization algorithms. Thakkar et al. (2023) measure self-influence as the sample importance during pretraining. Compared to instance-wise strategies, domain reweighting aims to reweigh or resample from various data groups, which offers better scalability for language model pretraining. DoReMi (Xie et al., 2023a) ap-

plies Group DRO on the loss gap between two auxiliary models to optimize the domain sampling weights. Chen et al. (2023) propose to build an online resampling curriculum by exploiting the dependency relationship among skills represented by a directed skill graph. While the computation cost for constructing the skill graph limits its applicability to general language model pretraining.

6. Conclusion

We introduced DOGE, an effective and efficient domain reweighting framework based on generalization estimation, which finds the optimal domain weights tailored to various generalization objectives. With the pretraining corpus with reweighted domain sampling weights, our experiments on SlimPajama show an improvment on LLM's universal generalization on langauge modelling and downstream few-shot reasoning ability. With out-of-domain generalization objective, DOGE efficiently accelerates the learning of the target domains and low-resource language by selectively learning from related data sources. Notably, DOGE gives robust domain reweighting results across various scales of proxy models, which demonstrates a great capacity to utilize small-scale proxy model to accelerate the training of larger models. Scaling-up experiments with larger models and datasets is an important future direction.

Impact Statement

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

References

Agarwal, N., Bullins, B., and Hazan, E. Second-order stochastic optimization for machine learning in linear time, 2017.

Beck, A. and Teboulle, M. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Oper. Res. Lett.*, 31(3):167–175, 2003.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language, 2019.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Chen, M. F., Roberts, N., Bhatia, K., Wang, J., Zhang, C., Sala, F., and Ré, C. Skill-it! a data-driven skills framework for understanding and training language models, 2023.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. Palm: Scaling language modeling with pathways, 2022.

Cole, T. and Siebert-Cole, E. Family tree of languages – part i: Indo-european (2022), 03 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., Zoph, B., Fedus, L., Bosma, M., Zhou, Z., Wang, T., Wang, Y. E., Webster, K., Pellat, M., Robinson, K., Meier-Hellstern, K., Duke, T., Dixon, L., Zhang, K., Le, Q. V., Wu, Y., Chen, Z., and Cui, C. Glam: Efficient scaling of language models with mixture-of-experts, 2022.

Engstrom, L., Feldmann, A., and Madry, A. Dsdm: Model-aware dataset selection with datamodels, 2024.

Everaert, D. and Potts, C. Gio: Gradient information optimization for training dataset selection, 2023.

Fan, S. and Jaggi, M. Irreducible curriculum for language model pretraining, 2023.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The pile: An 800gb dataset of diverse text for language modeling, 2020.

Gao, L., Tow, J., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., McDonell, K., Muennighoff, N., Phang, J., Reynolds, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. A framework for few-shot language model evaluation, September 2021. URL https://doi.org/10.5281/zenodo.5371628.

Gordon, A., Kozareva, Z., and Roemmele, M. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pp. 394–398, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics. URL https://aclanthology.org/S12-1052.

Grangier, D., Ablin, P., and Hannun, A. Adaptive training distributions with scalable online bilevel optimization, 2023.

Gunasekar, S., Zhang, Y., Aneja, J., Mendes, C. C. T., Giorno, A. D., Gopi, S., Javaheripi, M., Kauffmann, P., de Rosa, G., Saarikivi, O., Salim, A., Shah, S., Behl, H. S., Wang, X., Bubeck, S., Eldan, R., Kalai, A. T., Lee, Y. T., and Li, Y. Textbooks are all you need, 2023.

Guo, H., Rajani, N. F., Hase, P., Bansal, M., and Xiong, C. Fastif: Scalable influence functions for efficient model interpretation and debugging, 2021.

Guo, M., Dai, Z., Vrandečić, D., and Al-Rfou, R. Wiki-40B: Multilingual language model dataset. In Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Moreno, A., Odijk, J., and Piperidis, S. (eds.), *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 2440–2452, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL https://aclanthology.org/2020.lrec-1.297.

Hacohen, G. and Weinshall, D. On the power of curriculum learning in training deep networks, 2019.

Hashimoto, T. Model performance scaling with multiple data sources. In *International Conference on Machine Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:235826265.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions, 2020.

Kwon, Y., Wu, E., Wu, K., and Zou, J. Datainf: Efficiently estimating data influence in lora-tuned llms and diffusion models, 2024.

Lee, A., Miranda, B., and Koyejo, S. Beyond scale: the diversity coefficient as a data quality metric demonstrates llms are pre-trained on formally diverse data, 2023.

Li, Y., Bubeck, S., Eldan, R., Giorno, A. D., Gunasekar, S., and Lee, Y. T. Textbooks are all you need ii: phi-1.5 technical report, 2023.

Liu, J., Cui, L., Liu, H., Huang, D., Wang, Y., and Zhang, Y. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning, 2020.

Longpre, S., Yauney, G., Reif, E., Lee, K., Roberts, A., Zoph, B., Zhou, D., Wei, J., Robinson, K., Mimno, D., and Ippolito, D. A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, toxicity, 2023.

Nemirovski, A. and Yudin, D. *Problem complexity and Method Efficiency in Optimization*, volume 1. Wiley, New York, 1983.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. The refinedweb dataset for falcon llm: outperforming curated corpora with web data, and web data only. *arXiv preprint arXiv:2306.01116*, 2023.

Pilehvar, M. T. and Camacho-Collados, J. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations, 2019.

Pruthi, G., Liu, F., Sundararajan, M., and Kale, S. Estimating training data influence by tracing gradient descent, 2020.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale, 2019.

Soboleva, D., Al-Khateeb, F., Myers, R., Steeves, J. R., Hestness, J., and Dey, N. 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, 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Thakkar, M., Bolukbasi, T., Ganapathy, S., Vashishth, S., Chandar, S., and Talukdar, P. Self-influence guided data reweighting for language model pre-training, 2023.

Together Computer. Redpajama: An open source recipe to reproduce llama training dataset, 2023. URL https://github.com/togethercomputer/RedPajama-Data.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. Llama: Open and efficient foundation language models, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023b.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2023.

Welbl, J., Liu, N. F., and Gardner, M. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4413. URL https://aclanthology.org/W17-4413.

Xie, S. M., Pham, H., Dong, X., Du, N., Liu, H., Lu, Y., Liang, P., Le, Q. V., Ma, T., and Yu, A. W. Doremi: Optimizing data mixtures speeds up language model pretraining, 2023a.

Xie, S. M., Santurkar, S., Ma, T., and Liang, P. Data selection for language models via importance resampling, 2023b.

Xu, B., Zhang, L., Mao, Z., Wang, Q., Xie, H., and Zhang, Y. Curriculum learning for natural language understanding. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 6095–6104, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.542. URL https://aclanthology.org/2020.acl-main.542.

Yeh, C.-K., Taly, A., Sundararajan, M., Liu, F., and Ravikumar, P. First is better than last for language data influence, 2022.

Zhou, X., Pi, R., Zhang, W., Lin, Y., and Zhang, T. Probabilistic bilevel coreset selection, 2023.

A. Model Architectures

The maximal (min.) learning rate applied to train the largest model (684M) is 1.5×10^{-4} (5×10^{-5}), while others apply 5×10^{-4} (1×10^{-4}), with a cosine scheduler. The weight decay for all models is set as 0.01, the gradient clip is set as 1.0.

Table 4. Architecture hyperparameters for various model scales used in the paper. All models are vanilla Transformer decoder-only models.

	Layers	Attention heads	Embed dim	Hidden dim	Max. learning rate (min.)
60M	3	6	768	3072	5×10^{-4} (1×10^{-4})
82M	6	12	768	3072	5×10^{-4} (1×10^{-4})
124M	12	12	768	3072	5×10^{-4} (1×10^{-4})
210M	24	16	768	3072	5×10^{-4} (1×10^{-4})
684M	36	24	1200	4800	1.5×10^{-4} (5×10^{-5})

B. Derivation of Domain Weights Update Rule

To realize the optimal *universal generalization* performance within $\mathcal{T}$ steps, we optimize α_t at each training step t, which minimizes averaged cross-entropy loss $\bar{L}(\boldsymbol{\theta}_T)$ across all k domains at the final stage. Denote $l(\boldsymbol{\theta})$ as the next-token prediction (cross-entropy) loss of model parameterized by $\boldsymbol{\theta}$, $l_i(\boldsymbol{\theta})$ as the loss of the i^{th} domain D_i, our final objective can be written as:

$$\min_{\alpha_1,\ldots,\alpha_T \in \Delta^k} \bar{L}(\boldsymbol{\theta}^{(T)}) = \min_{\alpha_1,\ldots,\alpha_T \in \Delta^k} \sum_{i \in [k]} l_i(\boldsymbol{\theta}^{(T)}) \tag{7}$$

With a greedy approximation of (7), we search for the optimal domain weights α_t to minimize the average loss over k domains at step (t+1):

$$\begin{aligned} \arg\min_{\alpha_t \in \Delta^k} \bar{l}(\boldsymbol{\theta}^{(t+1)}) &= \arg\min_{\alpha_t \in \Delta^k} \sum_{i \in [k]} l_i(\boldsymbol{\theta}^{(t+1)}) \\ &= \arg\min_{\alpha_t \in \Delta^k} \sum_{i \in [k]} [l_i(\boldsymbol{\theta}^{(t+1)}) - l_i(\boldsymbol{\theta}^{(t)})] \end{aligned} \tag{8}$$

Take the first-order approximation, we estimate the loss for i^{th} domain as:

$$\begin{aligned} l_i(\boldsymbol{\theta}^{(t+1)}) &= l_i(\boldsymbol{\theta}^{(t)}) + \nabla l_i(\boldsymbol{\theta}^{(t)}) \cdot (\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}) + o(\|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\|) \\ &= l_i(\boldsymbol{\theta}^{(t)}) + \nabla l_i(\boldsymbol{\theta}^{(t)}) \cdot \left[-\eta_t \cdot \sum_{j \in [k]} \alpha_T^j \nabla l_j(\boldsymbol{\theta}^{(t)}) \right] + \boldsymbol{\varepsilon}^{(t)}, \end{aligned}$$

where $\boldsymbol{\varepsilon}^{(t)} = o(\|\boldsymbol{\theta}^{(t+1)} - \boldsymbol{\theta}^{(t)}\|) = o(\|\boldsymbol{\alpha}\|)$ as the high-order remainder from Taylor expansion. Denote $\mathcal{G}_t^i := \nabla l_i(\boldsymbol{\theta}^{(t)})$, $W_j^{(t)} \triangleq \langle \nabla l_j(\boldsymbol{\theta}^{(t)}), \sum_{i \in [k]} \nabla l_i(\boldsymbol{\theta}^{(t)}) \rangle$. We write $\mathcal{W}^{(t)} = [W_1^{(t)}, \ldots, W_k^{(t)}] \in \mathbb{R}^k$ for the score vector regrouping general-

ization estimations across all domains. Equ. (8) can be written as:

$$\begin{aligned}
\min_{\alpha_t \in \Delta^k} \bar{l}(\boldsymbol{\theta}^{(t+1)}) &= \min_{\alpha_t \in \Delta^k} \eta_t \cdot \sum_{i \in [k]} l_i(\boldsymbol{\theta}^{(t+1)}) \\
&\approx \min_{\alpha_t \in \Delta^k} -\eta_t \cdot \sum_{i \in [k]} \mathcal{G}_t^i \left(\sum_{j \in [k]} \alpha_t^j \mathcal{G}_t^j \right) \\
&= \min_{\alpha_t \in \Delta^k} -\eta_t \cdot \sum_{i \in [k]} \alpha_t^i \left(\mathcal{G}_t^i \sum_{j \in [k]} \mathcal{G}_t^j \right) \\
&= \min_{\alpha_t \in \Delta^k} -\eta_t \cdot \sum_{i \in [k]} \alpha_t^i W_i^{(t)} \\
&= \min_{\alpha_t \in \Delta^k} -\eta_t \cdot \langle \alpha_t, \mathcal{W}^{(t)} \rangle
\end{aligned} \tag{9}$$

We estimate $\boldsymbol{\varepsilon}^{(t)}$ by introducing a regularization term via Bregman divergence $D_h(\alpha || \alpha_{t-1}) = h(\alpha) - h(\alpha_{t-1}) - \langle \nabla h(\alpha_{t-1}), \alpha - \alpha_{t-1} \rangle$, $h(\alpha) = \sum_i \alpha_i \ln \alpha_i$. Adding this to (9), our optimization problem is:

$$\begin{aligned}
\alpha_t &:= \arg\min_{\alpha \in \Delta^k} \bar{l}(\boldsymbol{\theta}^{(t+1)}) \\
&\approx \arg\min_{\alpha \in \Delta^k} -\eta_t \cdot \langle \alpha, \mathcal{W}^{(t)} \rangle + \mu \cdot D_h(\alpha || \alpha_{t-1}) \\
&= \arg\min_{\alpha \in \Delta^k} -\eta_t \cdot \langle \alpha, \mathcal{W}^{(t)} \rangle + \mu (h(\alpha) - \langle \nabla h(\alpha_{t-1}), \alpha \rangle)
\end{aligned} \tag{10}$$

With $\nabla h(\alpha) = [\ln \alpha^i + 1]_i$, we take derivative of (10):

$$\begin{aligned}
\nabla(\cdot) &= \nabla \left(-\eta_t \cdot \langle \alpha, \mathcal{W}^{(t)} \rangle + \mu (h(\alpha) - \langle \nabla h(\alpha_{t-1}), \alpha \rangle) \right) \\
&= -\eta_t \cdot \mathcal{W}^{(t)} + \mu \cdot [\ln \alpha + 1]_i - \mu \cdot [\ln \alpha_{t-1} + 1]_i = 0
\end{aligned} \tag{11}$$

$$\Rightarrow \ln \alpha_t := \ln \alpha_* = \ln \alpha_{t-1} + \frac{\eta_t \mathcal{W}^{(t)}}{\mu} \tag{12}$$

Out-of-domain Generalization. ($D_{ood} \notin D_{train}$) Here we derive the update rule with the objective to generalize to a target domain, which is not included in the pretraining data sources, i.e. $D_{ood} \notin D_{train}$:

$$\min_{\alpha_1, \ldots, \alpha_T \in \Delta^k} \bar{L}(\boldsymbol{\theta}^{(T)}) = \min_{\alpha_1, \ldots, \alpha_T \in \Delta^k} l_{ood}(\boldsymbol{\theta}^{(T)})$$

Denote $\mathcal{G}_t^i := \nabla l_i(\boldsymbol{\theta}^{(t)})$, we search for the optimal domain weights α_t to minimize the average loss on D_{ood} at step (t+1):

$$\begin{aligned}
\arg\min_{\alpha_t \in \Delta^k} l_{ood}(\boldsymbol{\theta}^{(t+1)}) &= \arg\min_{\alpha_t \in \Delta^k} l_{ood}(\boldsymbol{\theta}^{(t+1)}) \\
&= \arg\min_{\alpha_t \in \Delta^k} [l_{ood}(\boldsymbol{\theta}^{(t+1)}) - l_{ood}(\boldsymbol{\theta}^{(t)})] \\
&= \arg\min_{\alpha_t \in \Delta^k} -\eta_t \cdot \sum_{i \in [k]} \alpha_t^i \left(\mathcal{G}_t^i \mathcal{G}_t^{ood} \right) + \boldsymbol{\varepsilon}^{(t)}
\end{aligned} \tag{13}$$

Alternatively, we define the generalization gain of i^{th} domain for the targeted domain D_{ood} as $\mathcal{W}_{ood}^{(t)} := \alpha_t^i \nabla l_i(\boldsymbol{\theta}^{(t)}) \cdot \nabla l_{ood}(\boldsymbol{\theta}^{(t)})$. Therefore, the optimization problem can be written as:

$$\begin{aligned}
\arg\min_{\alpha_t \in \Delta^k} l_{ood}(\boldsymbol{\theta}^{(t+1)}) &= \arg\min_{\alpha_t \in \Delta^k} -\eta_t \cdot \sum_{i \in [k]} \alpha_t^i \left(\mathcal{G}_t^i \mathcal{G}_t^{ood} \right) \\
&= \arg\min_{\alpha_t \in \Delta^k} -\eta_t \cdot \sum_{i \in [k]} \alpha_t^i \mathcal{W}_{ood}^{(t)} \\
&= \min_{\alpha_t \in \Delta^k} -\eta_t \cdot \langle \alpha_t, \mathcal{W}_{ood}^{(t)} \rangle
\end{aligned} \tag{14}$$

With Bregman divergence $D_h(\alpha||\alpha_{t-1}) = h(\alpha) - h(\alpha_{t-1}) - \langle \nabla h(\alpha_{t-1}), \alpha - \alpha_{t-1} \rangle$, $h(\alpha) = \sum_i \alpha_i \ln \alpha_i$, we can derive the update rule 15

$$\begin{aligned}
\alpha_t &:= \arg\min_{\alpha \in \Delta^k} l_{ood}(\boldsymbol{\theta}^{(t+1)}) \\
&\approx \arg\min_{\alpha \in \Delta^k} -\eta_t \cdot \langle \alpha, \mathcal{W}_{ood}^{(t)} \rangle + \mu \cdot D_h(\alpha||\alpha_{t-1}) \\
&= \arg\min_{\alpha \in \Delta^k} -\eta_t \cdot \langle \alpha, \mathcal{W}_{ood}^{(t)} \rangle + \mu(h(\alpha) - \langle \nabla h(\alpha_{t-1}), \alpha \rangle) \\
&\Rightarrow \ln \alpha_t^i := \ln \alpha_*^i = \ln \alpha_{t-1}^i + \frac{\eta_t \mathcal{W}_{ood}^{(t)}}{\mu}
\end{aligned} \tag{15}$$

C. Universal Generalization Evaluation

C.1. Domain Weights on SlimPajama.

Table 5. **Domain weights from DoGE with 82M proxy models.** The results are averaged by three random seeds with standard error.

Arxiv	Book	C4	CommonCrawl	Github	Stackexchange	Wikipedia
0.088±0.0008	0.045±0.0006	0.269±0.0047	0.214±0.0101	0.070±0.0037	0.166±0.0023	0.148±0.0061

C.2. Evaluation on Various Scales of Base Model.

We provide the detailed evaluation results on various scale of base model trained on the reweighted pretraining data corpus here (Table 6, Fig. 5, Fig. 6). According to the average perplexity, DoGE consistently outperforms all the other baseline methods. Besides, DoReMi-50k outperforms uniform baseline with both 124M and 210M base models, while DoReMi-10k fails to get the baseline performance, which suggests DoReMi has a great dependency on the capacity of the auxiliary models.

Table 6. **Perdomain perplexity results on 124/210M base model.** DoGE consistently achieves the best average perplexity over all baseline methods. Performance better than `Baseline` are in **Bold**.

	124**M**				210**M**			
Domain	Baseline	DoReMi-10k	DoReMi-50k	DoGE	Baseline	DoReMi-10k	DoReMi-50k	DoGE
Arxiv	**8.672**	9.353	10.149	8.954	**8.247**	9.041	9.637	8.456
Book	51.535	57.685	**49.038**	51.564	47.060	54.393	**45.192**	**46.940**
C4	56.424	63.968	**48.494**	**49.937**	51.862	60.781	**44.799**	**45.588**
CommonCrawl	52.661	53.347	**47.898**	**47.297**	48.319	50.456	**44.176**	**43.193**
Github	4.266	**4.008**	4.770	4.533	4.032	**3.871**	4.510	4.234
Stackexchange	9.555	9.898	10.392	**9.365**	8.948	9.477	9.760	**8.723**
Wikipedia	14.043	**13.208**	**12.246**	**13.567**	12.784	**12.351**	**11.358**	**12.348**
Average	18.566	19.208	18.355	**18.066**	17.218	18.285	17.119	**16.661**

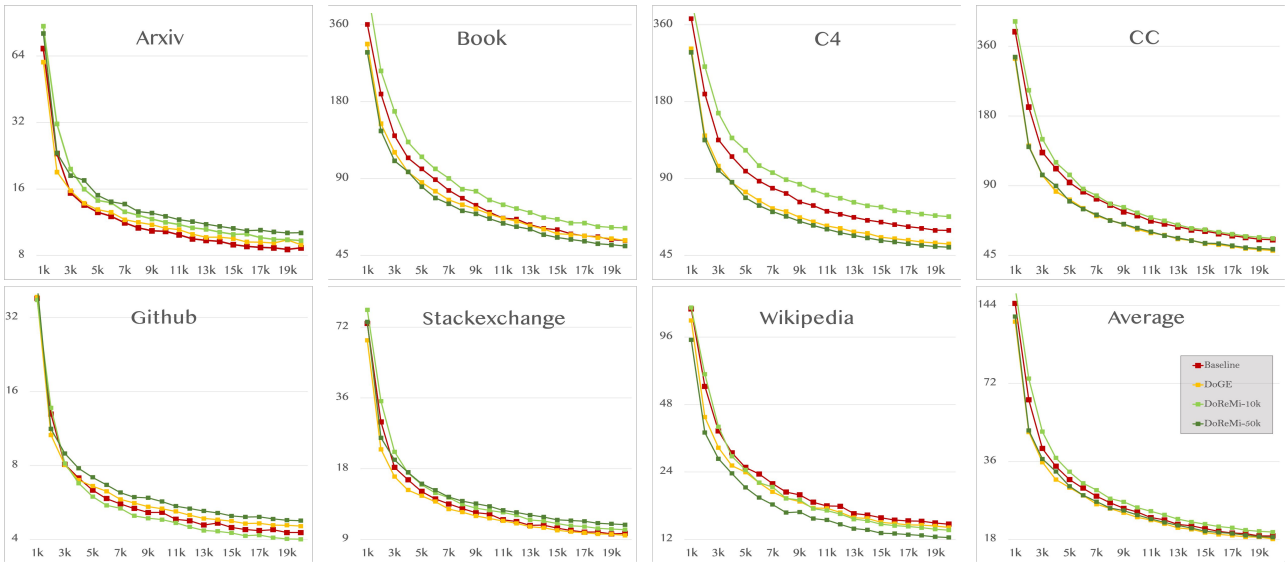


Figure 5. **Perdomain perplexity of 124M base Model.**

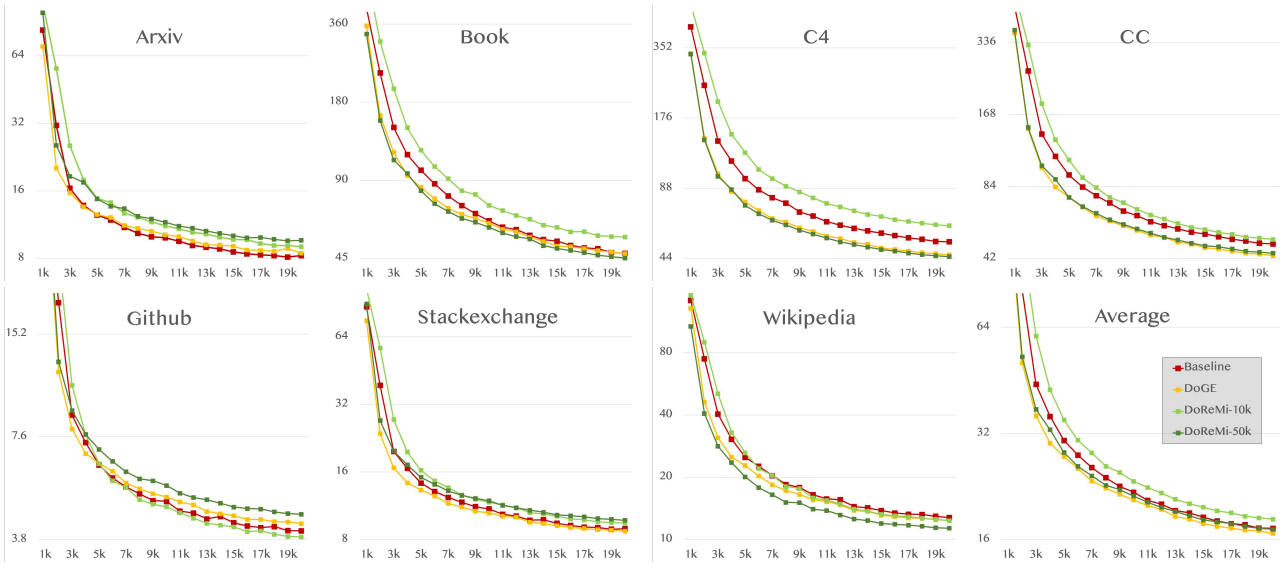


Figure 6. **Perdomain perplexity of 210M base Model.**

C.3. Early-stage Training Acceleration.

We have observed that DoGE reweighted pretraining corpus is able to accelerate the learning process, especially in the early training stage. Fig. (7) zooms in into the first 2500 training steps of the base model, where the validation perplexity from DoGE drops faster than all the other baseline models on each of all 7 domains, including those are downweighed with less tokens seen. It indicates DoGE facilitates the learning of general knowledge, which is shared across domains.

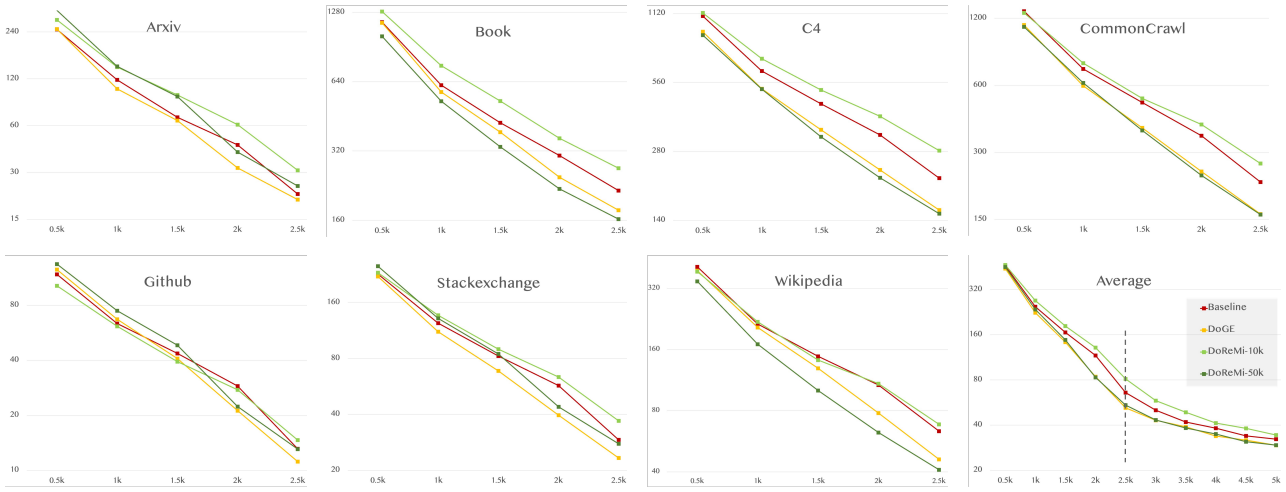


Figure 7. **Perdomain perplexity in early-stage of training (first 2500 steps).** DoGE outperforms baseline across all the domains no matter it is up-weighed or down-weighed.

C.4. Ablation on the Scale of Proxy Model.

To examine how robust DoGE is with various scale of the proxy model, we train DoGE with three model scales: 60M, 82M and 124M (Table 7). All proxy models are trained on the same dataset (Slimpajama) by 10k steps, with the same training hyperparameters. Notably, three proxy models with various scales results in consistent domain weights, with only 1.45% and 0.04% MAE across 7 domains. Since the difference between three sets of domain weights are negligible, we did not re-train the base model.

Table 7. **Domain weights from DoGE with various scale of proxy models.** The results are consistent among three different scale of proxy models.

Domain	DoGE (60M)	DoGE(82M)	DoGE(124M)	DoReMi (60M)	DoReMi (82M)	DoReMi (124M)
Arxiv	0.0997	0.0880	0.0890	0.0781	0.0424	0.0434
Book	0.0467	0.0450	0.0456	0.0830	0.0819	0.0546
C4	0.2455	0.2693	0.2789	0.1343	0.1141	0.1127
CommonCrawl	0.2004	0.2135	0.1968	0.2683	0.3811	0.3781
Github	0.0767	0.0703	0.0714	0.1055	0.0654	0.0753
Stackexchange	0.1968	0.1658	0.1703	0.1157	0.0847	0.0919
Wikipedia	0.1342	0.1482	0.1480	0.2150	0.2307	0.2440
MAE from 82M proxy	1.45%	/	0.48%	3.66%	/	0.91%
Computation Time (hours)[1]	4.5	6.0	10.5	20.5	39.0	51.5

C.5. Performance of Proxy Model.

We also compare the performance of the proxy model, which rescales the gradient from each domain at each single step, and the base model trained with resampled training corpus. According to Fig. (8), the performance of the proxy model falls behind the base model with resampled dataset with DoGE domain sampling weights. It is even worse than the baseline with uniform domain weights.

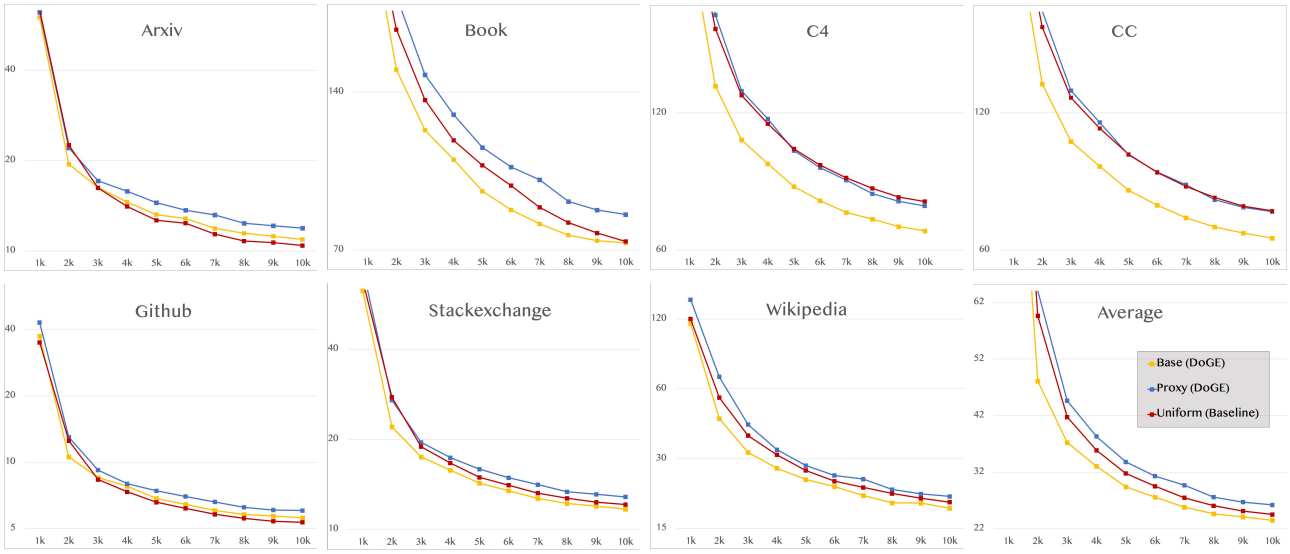


Figure 8. **Comparison of the proxy model and the base model with the same scale (82M).** The proxy model shows worse performance on average perplexity than both the base model with resampled training data and uniform sampling weights.

D. Out-of-Domain Generalization Evaluation

D.1. Evaluation Curves on OoD tasks.

Fig. 9 shows detailed curve of validation perplexity during the training process. On all 7 domains, DoGE outperforms uniform baseline without target domain. On Book, Github, C4 and CC, DoGE gets comparable or better perplexity than the baseline with access to the target domain. However, on Arxiv, Stackexchange and Wikipedia, both DoGE and uniform baseline without target have a large performance gap from the oracle. It indicates learning these domains requires more domain-specific knowledge, which can hardly be obtained from the other source domains. In that case, the gain from source domain reweighting could be limited.

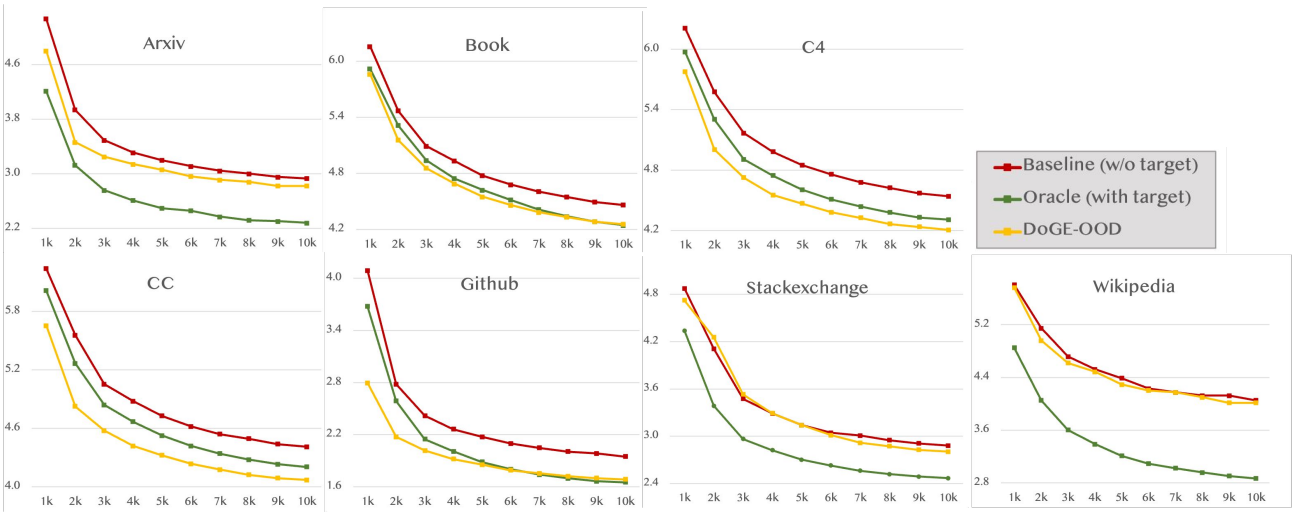


Figure 9. **Target domain perplexity on validation set.** DoGE outperforms baseline with uniform domain weight by a large margin across all target tasks. Notably, DoGE is comparable or even outperforms ORACLE (which the target domain is accessible) on several target tasks (Book, C4, CC, Github).

D.2. Domain Weights on OoD tasks.

We present all detailed domain weights on out-of-domain generalization on SlimPajama in Table (8), where each of 7 domains is set as the target iteratively. The domain weights on Wiki40b is shown in Table (9), where the low-resource languages (Catalan and Dutch) are set as the target while five mainstream languages are used as the training datasets.

Table 8. **Domain weights from DOGE (82M) on SlimPajama.** The results are averaged by two random seeds with standard error.

	Target Domain						
	Arxiv	Book	C4	CommonCrawl	Github	Stackexchange	Wikipedia
Arxiv	0	0.063±0.0006	0.035±0.0002	0.045±0.0017	0.084±0.0003	0.095±0.0030	0.091±0.0201
Book	0.010±0.0018	0	0.117±0.0008	0.124±0.0071	0.051±0.00002	0.048±0.0015	0.149±0.0102
C4	0.125±0.0010	0.341±0.0036	0	0.674±0.0145	0.058±0.0008	0.099±0.0045	0.201±0.0056
CommonCrawl	0.142±0.0009	0.345±0.0047	0.721±0.0025	0	0.059±0.0004	0.076±0.0020	0.251±0.0258
Github	0.161±0.0005	0.055±0.0008	0.029±0.0002	0.035±0.0015	0	0.637±0.0124	0.103±0.0222
Stackexchange	0.366±0.0031	0.081±0.0019	0.063±0.0008	0.059±0.0018	0.691±0.0019	0	0.204±0.0221
Wikipedia	0.106±0.0019	0.113±0.0006	0.035±0.0004	0.063±0.0023	0.057±0.0003	0.045±0.0013	0

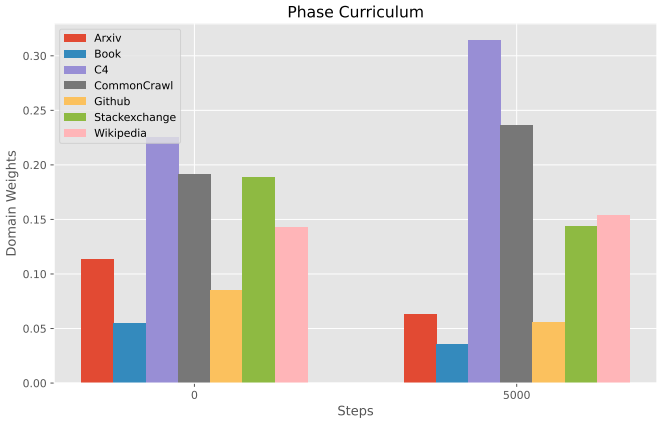
Table 9. **Domain weights from DOGE (82M) on Wiki40B.** The results are averaged by two random seeds with standard error.

	English (en)	German (de)	French (fr)	Spanish (es)	Russian (ru)
Catalan (ca)	0.073±0.008	0.043±0.008	0.344±0.103	0.516±0.102	0.024±0.0001
Dutch (nl)	0.259±0.003	0.267±0.037	0.176±0.043	0.203±0.025	0.095±0.022

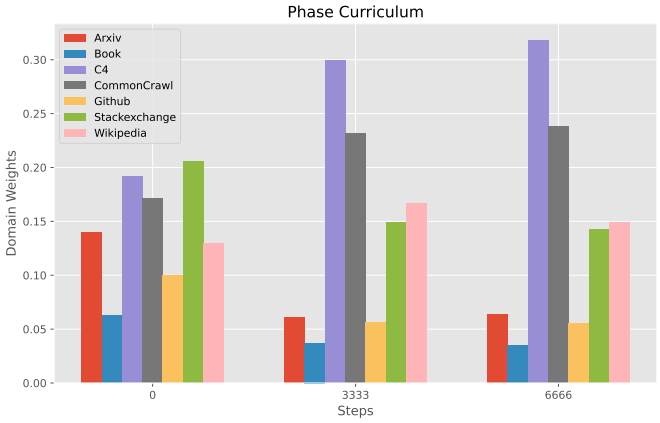
E. Stage-wise Curriculum

We provide the implementation and evaluation details of stage-wise curriculum learning in this section. Specifically, we firstly train a 82M proxy model applying DOGE for 10k steps. We then divide the whole training process of the proxy model into $K = 2, 3, 10$ stages, with $5000, 3333, 1000$ training steps in each stage. By average the domain weights by number of steps within each stage, we get the stage-wise sampling weights distribution as Fig. (10). We then train another 124M model from scratch for 10k steps, where we map the stage-wise sampling weights within the corresponding training steps. We compare the validation perplexity between the model trained with stage-wise curriculum and applying a globally-averaged domain weights. The models trained by each curriculum should have seen the same amount of tokens from each domains in expectation.

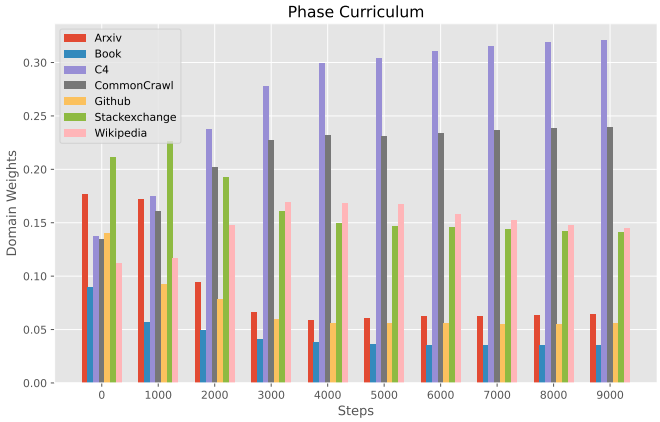
With $K = 2, 3$, the stage-wise curriculum keeps comparable performance as original DOGE, which applies the global average as the sampling weights throughout the whole training process. It is worth noting that the models learns *hard* domains (C4, CC, Book) slightly better than the global curriculum, while sacrificing the performance on easier domains (Arxiv, Github). However, with an extremely find-grained curriculum ($K = 10$), the curriculum severely hurt the performance on all the domains by a large margin. It suggests that given the same set of data covering diverse knowledge fields, the order of training data does impact the language modelling effectiveness, so that we have to carefully determine the granularity of the curriculum.



(a) $K = 2$



(b) $K = 3$



(c) $K = 10$

Figure 10. **Stage-wise curriculum with $K = 2, 3, 10$ learning stages.**

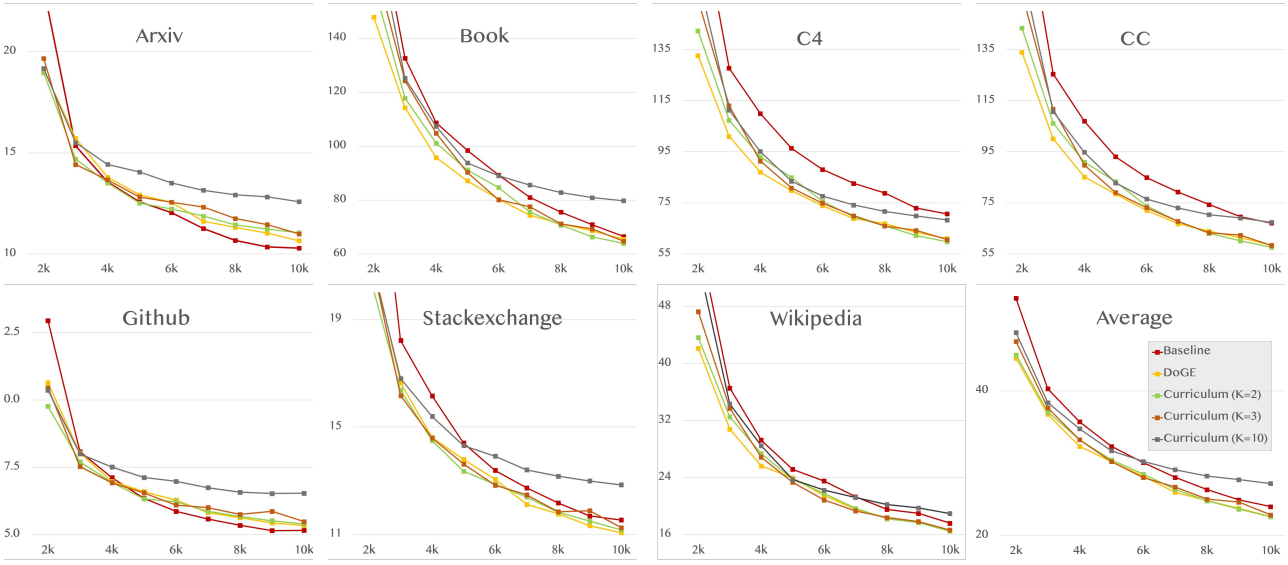


Figure 11. **Per-domain perplexity on validation set with stage-wise curriculum.**

F. Cancellation Effect

Following Yeh et al. (2022), at each time step t, we measure the ratio of the actual weight change and the summation of gradient among the mini-batch $B^{(t)}$ for each module of model weights $\boldsymbol{w}$. We sum up the ratio across the first $T_c = 1000$ steps to obtain the score of cancellation effect $C(\boldsymbol{w})$ as:

$$C(\boldsymbol{w}) = \sum_{t \in [T_c]} \frac{\|\boldsymbol{w}^{(t+1)} - \boldsymbol{w}^{(t)}\|}{\sum_{(x_i \in B^t)} \frac{\partial l(x_i)}{\partial \boldsymbol{w}}} \tag{16}$$

During the measurement of cancellation effect, the mini-batch is sampled uniformly from all domains. After the first 1000 steps, we re-initialize the proxy model and compute gradient estimation $\mathcal{W}$ only using the gradient of the selected parameter modules.

We then rank all 76 modules from the parameters of the 82M proxy model, and apply five parameter selection strategies: (1) We select $K = 10, 30, 50$ modules with the *lowest cancellation effect* scores, denoting DoGE-($low10$,$low30$,$low50$); (2) We select $K = 10, 30$ modules with the *highest cancellation effect* scores, denoting DoGE-($high10$,$high30$).

According to Table. (11), none of the parameter selection strategies could outperform the original DoGE, where the gradient estimation $\mathcal{W}$ is computed using the full gradient of the proxy model. However, the domain weights from different parameter selection strategies shows an intriguing pattern (Fig. 12): modules with low cancellation effect incline to upweigh *unique* domains, which contain more domain specific knowledge (e.g. Wikipedia, Arxiv, Stackexchange), while modules with high cancellation effect tend to upweigh *diverse* domains, which have broader knowledge coverage (e.g. CC, C4). It aligns with the out-of-domain generalization experiment (§ 3.2), where Wikipedia and Stackexchange get least improvement from domain reweighting, which indicates the *uniqueness* of the domain-specific knowledge.

Table 10. **Domain weights with parameter selection based on cancellation effect.** DoGE-low[k] (DoGE-high[k]) denotes the k modules with lowest (highest) cancellation effect are selected to compute $\mathcal{W}$.

Domain	DoGE-*full*	DoGE-*low10*	DoGE-*low30*	DoGE-*low50*	DoGE-*high30*	DoGE-*high10*
Arxiv	0.08800	0.2071	0.1571	0.1094	0.07855	0.05635
Book	0.04500	0.04734	0.04601	0.04708	0.05304	0.05783
C4	0.2693	0.1139	0.1425	0.2209	0.2871	0.3406
CommonCrawl	0.2135	0.09111	0.1142	0.1658	0.2537	0.3316
Github	0.07027	0.1123	0.1005	0.07782	0.06943	0.04462
Stackexchange	0.1658	0.2061	0.1994	0.1726	0.1494	0.1050
Wikipedia	0.1482	0.2221	0.2402	0.2063	0.1088	0.06402

Table 11. **Validation perplexity by domains with parameter selection based on cancellation effect.** DoGE-low[k] (DoGE-high[k]) denotes the k modules with lowest (highest) cancellation effect are selected to compute $\mathcal{W}$.

Domain	Baseline (Uniform)	DoGE-*low10*	DoGE-*low30*	DoGE-*low50*	DoGE-*full*	DoGE-*high30*	DoGE-*high10*
Arxiv	8.672	**8.106**	8.447	8.735	8.954	9.092	9.413
Book	51.535	59.359	56.146	52.700	51.564	50.030	**48.526**
C4	56.424	61.225	57.866	53.652	49.937	48.422	**45.789**
CommonCrawl	52.661	57.891	54.500	49.487	47.297	45.696	**42.990**
Github	**4.266**	4.268	4.298	4.460	4.533	4.543	4.796
Stackexchange	9.555	**9.075**	9.102	9.336	9.365	9.494	9.982
Wikipedia	14.043	12.793	**12.399**	12.755	13.567	14.471	16.334
Average	18.566	18.848	18.442	18.151	**18.066**	18.067	18.359
Computation saved for $\mathcal{W}$	/	66.9%	42.8%	24.4%	/	97.5%	99.9%

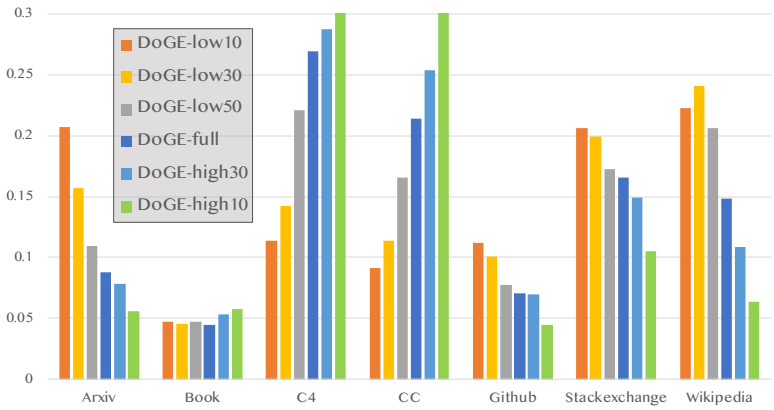


Figure 12. **Domain weights with parameter selection by cancellation effect.**

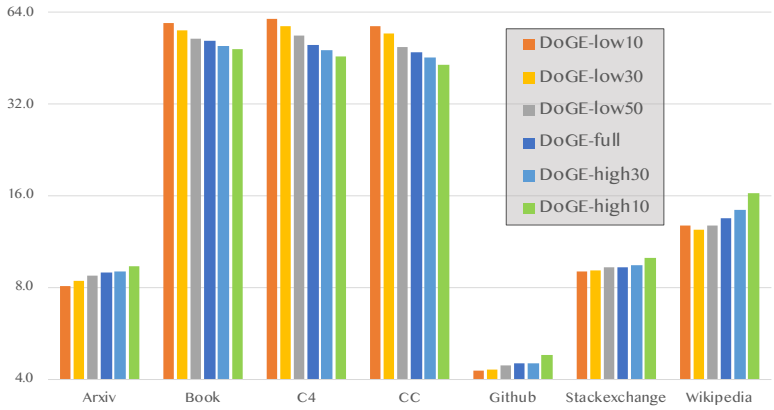


Figure 13. **Validation perplexity by domain with parameter selection by cancellation effect.**