

Params	275M	468M	932M	1.63B	2.28B	3.35B	8.13B
width	800	1088	1632	2208	2624	3232	5120
depth				24			
Learning rate	1.5e-3	1.5e-3	5e-4	4.2e-4	4e-4	3.5e-4	2.4e-4
Training tokens				2.5B-600B			
Optimizer				Fully decoupled AdamW [39]			
Optimizer Momentum				$\beta_1 = 0.9, \beta_2 = 0.95$			
Minimum Learning rate				0			
Weight decay				1e-4			
Batch size				2k			
Patch size				(14, 14)			
Gradient clipping				1.0			
Warmup iterations				1k			
Augmentations:							
RandomResizedCrop							
size				224px			
scale				[0.4, 1.0]			
RandomHorizontalFlip				$p = 0.5$			

Table 4: **Pre-training hyperparameters** used for pre-training of NMM to conduct the scaling laws study.

A Implementation details

A.1 Multimodal pretraining

Implementation details. We closely follow the implementation of [52] and present in Table 4 the pre-training hyperparameters for the model configurations used in our scaling laws study. Our models range from 100M to 8B parameters, with width scaling accordingly while maintaining a constant depth of 24 layers. We use causal attention for text tokens and bidirectional attention for image tokens. Learning rates are adjusted based on model size, generally decreasing for larger models. These values were determined through empirical testing. Optimization is handled using a fully decoupled AdamW optimizer with momentum values set to $\beta_1 = 0.9$, $\beta_2 = 0.95$, and a weight decay of 1×10^{-4} . Each training batch consists of 2,000 samples, totaling 2 million tokens with a 1,024-token context length. Gradients are clipped at 1.0, and training begins with a warmup phase of 1,000 iterations, followed by a constant learning rate schedule to reduce the number of experiments.

For vision inputs, we process images as (14, 14) patches with augmentations including Random Resized Crop (224px, scale range of [0.4, 1.0]) and Random Horizontal Flip with a 50% chance. Model training benefits from efficiency techniques such as `bf16` precision, Fully Sharded Data Parallel (FSDP) [65], activation checkpointing, gradient accumulation, and sequence packing to minimize padding in the image-captioning dataset.

We assess model performance on three held-out data subsets: interleaved data (Obelics), image-caption data (HQITP), and text-only data (DCLM), following prior works [2, 3, 29]. This setup provides a robust evaluation of model generalization across diverse data types.

A.2 LLM pretraining

We borrow architectures from [8]. We use a fixed depth of 24 and change the latent dimension of the network to obtain different model scales. All hyperparameters are described in Table 5.

B Detailed extrapolation results

We report the detailed per-domain and per-model size MRE corresponding to each experiment and scaling law in the paper.

B.1 Comparison to the laws of [62]

[62] propose four laws to model the behavior of the loss on a domain *as a function of h only*, that is, for a fixed N, D budget. They propose the following formulas, rewritten with notations consistent

Params	90M	200M	350M	700m	1.3B	3B
Width	512	768	1024	1536	2048	3072
Depth				24		
Learning rate			constant after warmup: 1e-4			
Training tokens	8.4B	8.4B	8.4B	16.8B	33.6B	67.2B
Optimizer			AdamW [39]			
Optimizer Momentum			$\beta_1 = 0.9, \beta_2 = 0.95$			
Batch size				128		
Sequence length				1024		
Gradient clipping				1.0		
Warmup iterations				1k		

Table 5: **Pre-training hyperparameters** used for the pre-training of LLM to conduct the scaling laws study.

Scaling Law	Domain	Train MRE(%)	2B MRE(%)	8B MRE(%)
Simple	Text	0.15	0.44	0.50
Additive	Text	0.12	0.40	0.51
Joint	Text	0.10	0.39	0.32
Full	Text	0.09	0.38	0.33
Simple	Image-Captions	0.52	0.89	1.36
Additive	Image-Captions	0.47	0.83	1.23
Joint	Image-Captions	0.43	0.85	1.17
Full	Image-Captions	0.43	0.90	1.33
Simple	Interleaved	0.22	0.65	0.80
Additive	Interleaved	0.14	0.44	0.58
Joint	Interleaved	0.10	0.41	0.45
Full	Interleaved	0.10	0.40	0.45

Table 6: Full results of experiments in Section 4 for the multimodal experiment.

with our notation:

$$\mathcal{L}(h) = E + \sum_{i=1}^k C_i \exp(\gamma_i h_i) \quad (\text{Ye M1})$$

$$\mathcal{L}(h) = E + C \sum_{i=1}^k \exp(\gamma_i h_i) \quad (\text{Ye M2})$$

$$\mathcal{L}(h) = E + C \exp\left(\prod_{i=1}^k \gamma_i h_i\right) \quad (\text{Ye M3})$$

$$\mathcal{L}(h) = E + C \exp\left(\sum_{i=1}^k \gamma_i h_i\right) \quad (\text{Ye M4})$$

where the parameters of the law are the E, C, C_i, γ_i . We compare this with the form of our scaling law Equation (4) when N and D are fixed:

$$\mathcal{L}(h) = E + \frac{1}{\sum_{i=1}^k C_i h_i^{\gamma_i}} \quad (\text{Additive, fixed (N, D)})$$

We fit all those scaling laws on the LLM training data, keeping only one value for N, D (we take $N = 200m, D = 8B$ tokens). We only keep 25 training mixtures to fit the laws, and report the MRE on the $84 - 25 = 59$ remaining in Tab. 9. Note that we had trouble fitting the M3 law, which is also reported to underperform in [62]. Overall, our formula gives systematically better training errors, and it most of the time translates to better testing errors on unseen mixtures. We stress once again that one of the core contributions of our work is to explain how scale interacts with data mixtures, by proposing scaling laws that take N, D , and h as inputs. [62] only consider scaling laws with respect to h .

Scaling Law	Domain	Train MRE(%)	700m MRE(%)	1B MRE(%)
Simple	Wikipedia	0.28	0.75	1.13
Additive	Wikipedia	0.24	0.77	1.11
Joint	Wikipedia	0.13	0.24	0.39
Full	Wikipedia	0.12	0.23	0.39
Simple	GitHub	0.60	1.38	3.28
Additive	GitHub	0.42	1.10	1.69
Joint	GitHub	0.23	0.38	1.46
Full	GitHub	0.22	0.49	1.91
Simple	StackExchange	0.40	0.90	1.50
Additive	StackExchange	0.33	0.88	1.31
Joint	StackExchange	0.17	0.26	1.05
Full	StackExchange	0.16	0.30	1.17
Simple	PG-19	0.21	0.53	0.91
Additive	PG-19	0.16	0.55	0.94
Joint	PG-19	0.15	0.40	0.71
Full	PG-19	0.14	0.35	0.54

Table 7: Full results of experiments in Section 4 for the LLM experiment.

Scaling Law	Domain	Train MRE(%)	700m MRE(%)	1B MRE(%)
Additive	Wikipedia	0.45	0.53	0.53
Joint	Wikipedia	0.22	0.18	0.19
Additive	GitHub	0.70	0.88	1.49
Joint	GitHub	0.39	0.31	1.09
Additive	StackExchange	0.50	0.55	0.50
Joint	StackExchange	0.29	0.23	0.44
Additive	PG-19	0.24	0.37	0.53
Joint	PG-19	0.18	0.25	0.44

Table 8: Full results of experiments in Section 4 for the cosine schedule experiment.

B.2 Comparison to [21]

[21] propose a scaling law to evaluate the loss the i -th training domains, as a function of both h and number of tokens D :

$$\mathcal{L}_i(h, D) = \left(\frac{B}{D^\beta} + E \right) \frac{C}{h_i^\gamma} \quad (\text{Ge 24})$$

where the coefficients B, β, E, C and γ have to be fitted. Here, the loss must be on domain i , and it only involves the proportion of that domain h_i , not that of the other domains. In our view, this is a caveat since it implies that all other domains would have the same impact on the loss for domain i , while one other training domain might be very useful for that task, and another might be useless.

Since this law does not take into account model scale, we compare it to our additive law for a fixed model scale:

$$\mathcal{L}(h, D) = E + \frac{1}{\sum_{i=1}^k C_i h_i^{\gamma_i}} + \frac{B}{D^\beta} \quad (\text{Additive, fixed N})$$

We consider a fixed size of model (N=200M) on the LLM training experiment, take only 25 mixtures to fit the scaling laws, and report the MRE on the remaining testing set in Tab. 10. We see that our proposed scaling law achieves a significantly lower MRE on both train and testing data, highlighting the importance of taking all other domains into account.

Scaling law	Domain	Train (MRE%)	Test (MRE%)
Additive, fixed (N, D)	Wikipedia	0.07	0.18
Ye M1	Wikipedia	0.08	0.14
Ye M2	Wikipedia	0.09	0.17
Ye M3	Wikipedia	2.54	4.20
Ye M4	Wikipedia	0.17	0.31
Additive, fixed (N, D)	GitHub	0.10	0.19
Ye M1	GitHub	0.20	0.61
Ye M2	GitHub	0.22	0.40
Ye M3	GitHub	5.45	5.26
Ye M4	GitHub	0.36	0.44
Additive, fixed (N, D)	StackExchange	0.07	0.18
Ye M1	StackExchange	0.14	0.32
Ye M2	StackExchange	0.14	0.21
Ye M3	StackExchange	4.11	3.20
Ye M4	StackExchange	0.22	0.34
Additive, fixed (N, D)	PG-19	0.08	0.12
Ye M1	PG-19	0.09	0.17
Ye M2	PG-19	0.13	0.18
Ye M3	PG-19	2.21	3.14
Ye M4	PG-19	0.16	0.21

Table 9: Comparison of our scaling laws for a fixed (N, D) budget with those of [62] on the LLM data.

Scaling law	Domain	Train (MRE%)	Test (MRE%)
Additive, fixed N	Wikipedia	0.13	0.17
Ge 24	Wikipedia	0.51	0.62
Additive, fixed N	GitHub	0.20	0.26
Ge 24	GitHub	1.99	2.26
Additive, fixed N	StackExchange	0.14	0.19
Ge 24	StackExchange	0.94	1.21
Additive, fixed N	PG-19	0.13	0.15
Ge 24	PG-19	0.49	0.54

Table 10: Comparison of our scaling laws for a fixed model size N with that of [21] on the LLM data.

C Additional analysis

C.1 Optimal domain weights

Recall that section 5 defines the optimal domain weights $h^*(\cdot)$ as function of compute budget (N, D) . In the main text, we assumed uniform weights of the target domains \mathcal{D}_i . However, we can also consider a weighted average scenario, with an arbitrary weight vector w_i .

$$h^*(w, N, D) \in \arg \min_{h \in \Delta_k} \sum_{i=1}^m w_i \mathcal{L}^i(N, D, h). \quad (13)$$

Once again, this objective is seamlessly optimizable with mirror descent. When training domains and target domains are the same, w and h^* are a probability distribution over the same simplex. Therefore, we can study the mapping $w \mapsto h^*(w, N, D)$.

Behavior at the corners We predict the optimal domain weights for both laws, choosing each training domain j as the target, that is, putting $w_i = 1$ if $i = j$ and $w_i = 0$ otherwise. The results are given in Figure 6.



Figure 6: **Optimal domain weights for a single (pure) domain, typically lies at the corner of the probability simplex.** Scaling law predictions for a 1.3B model trained on 10B tokens.

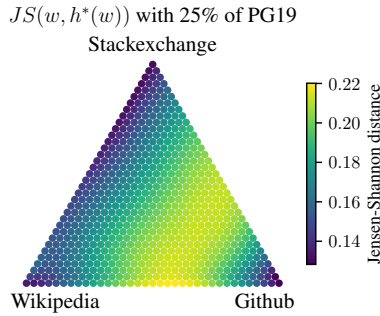


Figure 7: **The optimal domain weights are always different from the target mixture, except at the corners of the simplex.** 1.3B model with 10B tokens, on 4 domains of The Pile.

716 We see that the data mixture law predicts that the optimal domain weights are typically located in the
717 corresponding corner of the simplex - which is not too surprising when there is little domain overlap.
718 This justifies the rule of thumb $h_i^* \approx 1$ when the target domain is \mathcal{D}_i .

719 **Fixed-points** For a given (N, D) pair, the optimization problem of Equation 13 defines a function
720 $w \mapsto h^*(w)$ that maps the simplex onto itself. The fact that $h^*(w) \neq w$ indicates the surprising
721 phenomenon that, in order to minimize the loss $\mathcal{L}^{\text{ERM}} = \sum_{i=1}^m w_i \mathcal{L}^i(\theta)$, it is faster to instead
722 minimize $\mathcal{L}^* = \sum_{i=1}^m h^*(w)_i \mathcal{L}^i(\theta)$, rather than minimizing directly \mathcal{L}^{ERM} .

723 Fixed points of the map $w \mapsto h^*(w)$ correspond to target mixtures w that are minimized by training
724 on w itself: this is the empirical risk minimizer.

725 To find these points, we compute the Jensen-Shannon distance, defined as $JS(w, h^*) =$
726 $\sqrt{1/2(KL(w||m) + KL(h^*||m))}$ with $m = (w + h^*)/2$, and we look for near-zero values, in Fig-
727 ure 7 and Figure 8.

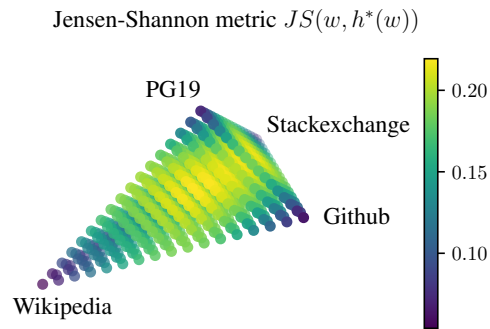


Figure 8: **Jensen-Shannon distance between target mixture h and its optimum training mixture $h^*(w)$** on the 4D simplex. 1.3B model with 10B tokens, on 4 domains of the Pile. No fixed-point are visible, except at the corners. This suggests that in general, it is better not to train on the mixture you want to be good on.