Training Optimal Large Diffusion Language Models

Jinjie Ni[1]*, **Qian Liu, Chao Du**[2], **Longxu Dou**[2], **Hang Yan**[4], **Zili Wang**[3], **Tianyu Pang**[2], **Michael Qizhe Shieh**[1]
[1]National University of Singapore, [2]Sea AI Lab, [3]StepFun, [4]Shanghai Qiji Zhifeng Co., Ltd.

Abstract

We introduce Quokka, the first comprehensive scaling law for diffusion language models (DLMs), encompassing both compute- and data-constrained regimes alongside key modeling and optimization designs. Under compute constraints, we find that optimal parameter and dataset sizes scale equally with compute ($N_{\text{opt}} \propto C^{0.5}$, $D_{\text{opt}} \propto C^{0.5}$); however, DLMs are 2–5$\times$ more data-hungry than autoregressive (AR) models, requiring larger corpora and proportionally smaller models for a given FLOP budget. Under data constraints, validation loss follows a U-shaped curve across epochs, with the onset of overfitting scaling as $e_{\text{opt}} \propto U_D^{0.39}/N^{0.55}$. This indicates that optimally leveraging a fixed unique data budget (U_D) requires allocating both modestly larger models and more training epochs. Beyond data and parameter allocation, we provide actionable guidance on DLM design choices: absorbing-mask transitions outperform uniform diffusion, linear noise schedules are the most stable and performant, and principled diffusion ELBO objectives ultimately surpass MaskGIT losses despite slower initial convergence. Furthermore, we demonstrate that easy-to-hard noise curricula accelerate early learning, established AR scaling laws for batch size and learning rate transfer seamlessly to DLMs, and weight decay, while unhelpful for single-epoch runs, is essential for multi-epoch regularization.

1 Introduction

2025 marks the first year of diffusion language models (DLMs) scaling. Based on the great efforts that laid the theoretical foundation for DLMs [19, 26, 29, 32], [23] successfully trained the first large diffusion language model from scratch, competitive to state-of-the-art open-source autoregressive (AR) models [7]. Meanwhile, several commercial DLMs emerged, exhibiting superior coding and math performance with remarkably low generation latency at the same time [9, 16, 33]. Thereafter, [22] showed that DLMs exhibit much better data learning potential than AR models when data is the bottleneck, a.k.a. "intelligence crossovers", demonstrating a core advantage over AR models under token crisis [21, 38].

DLMs exhibit several modeling advantages over AR models. Their bidirectional attention and diffusion objective enable any-order modeling, allowing data to be modeled in arbitrary directions during both training and inference. This property is particularly beneficial for tasks requiring non-causal dependencies and back-and-forth reasoning, such as coding [35, 37], mathematics [9], report generation [10], etc. DLMs' bidirectional attention natively support on-the-fly context modification as new content is generated, a desirable feature in these tasks. Multi-token generation is also natively supported by DLMs, providing the foundation for their bleeding fast decoding. Moreover, DLMs spend more parallelable FLOPs at both the inference and training time, leading to their superior data learning capability and potentially stronger reasoning capabilities.

*Correspondence to: Jinjie Ni <jinjieni@nus.edu.sg>

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

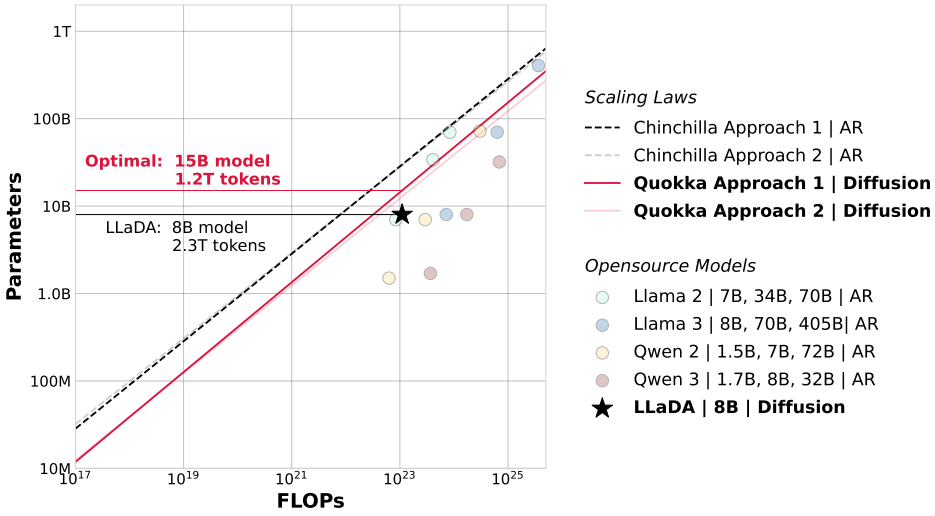

Figure 1: **Overlaid predictions from Chinchilla and Quokka (compute-constrained).** We overlay the predictions from our approach 1 and 2, along with those from [11]. Though scaling at the same pace, DLMs are *2–5*$\times$ more data-hungry than AR models at the same FLOPs—favor smaller models and larger corpora. We mark the position of LLaDA [23] in the same space, finding that it's severely over-trained with $2\times$ smaller models and $2\times$ more corpora against the Quokka efficient frontier. Meanwhile, wo show the positions of opensource models, finding that most models are over-trained compared with the Chinchilla efficient frontier, except some models from the Llama family. Note that the token statistics are based on the numbers in their reports, which might not be strictly unique tokens. More discussions are detailed in §D.

However, the knowledge on how to train large DLMs from scratch is still near blank. Existing studies are largely heuristic or simply extrapolate conclusions from AR models [23, 39]. In practice, two scaling laws are of primary interest: (1) the compute-constrained (or compute-optimal) scaling law [11], where compute is fixed while model and dataset size are unconstrained; and (2) the data-constrained scaling law [21], where dataset size is fixed while model size and compute are unbounded. Both regimes raise key questions about scaling behavior under these restrictions and, more critically, how to optimally allocate the remaining degrees of freedom. Moreover, beyond the classic trade-offs among data, parameters, and compute, additional modeling and optimization choices can also substantially affect the end-of-training performance of language models.

In this work, we will empirically investigate the dependence of language modeling loss and downstream evaluations on all of these factors. We introduce **Quokka**, the first large-scale scaling law for DLMs, covering both the compute-constrained and data-constrained regimes, and studying the key modeling and optimization designs. Specifically, the key contributions we made include:

Compute-constrained scaling laws for DLMs. Under compute-constraint, we revisit the question: Given a fixed FLOPs budget, how should one trade-off model size and the number of training tokens? To answer this question, we model the final pre-training loss as a function of the number of model parameters N, and the number of training tokens, D. Since the computational budget C is a deterministic function $FLOPs(N, D)$ of the number of these two variables, our objective is to minimize L subject to the constraint $FLOPs(N, D) = C$:

$$N_{\text{opt}}(C), D_{\text{opt}}(C) = \underset{N,D \text{ s.t. } \text{FLOPs}(N,D)=C}{\arg\min} L(N, D). \tag{1}$$

The functions $N_{\text{opt}}(C)$ and $D_{\text{opt}}(C)$ characterize the optimal allocation of a compute budget C. We estimate these functions empirically using results from a large set of models, spanning parameter counts from under 7M to over 11B and trained on datasets from 1B to over 260B tokens. Across two independent approaches, we consistently find that N and D should scale proportionally with C: doubling N requires doubling D, mirroring the scaling behavior observed in AR models. Meanwhile,

both approaches indicate that DLMs require roughly 2–5$\times$ more data than AR models under the same FLOPs budget (Figure 1).

Data-constrained scaling laws for DLMs. In the data-constrained regime—which represents the long-term practical bottleneck—we study the interactions among training performance, unique dataset size, model parameters, and data repetition. We focus on two central questions: (1) Given a fixed model size, a limited amount of unique data, and effectively unlimited compute, how many epochs can the model be trained before performance degradation occurs? (2) Given a fixed unique data budget and unlimited compute, what is the optimal allocation of parameters and data repetitions?

To address these questions, we model the U-shaped validation loss $L(N, U_D, e)$ as a function of parameters N, unique tokens U_D, and training epochs (or repetitions) e, using results from 21,345 training runs. For question (1), with N and U_D fixed, we seek the maximum number of epochs that minimizes validation loss along the U-curve. FLOPs are excluded from this formulation, as compute is assumed unconstrained:

$$e_{\text{opt}}(\hat{N}, \hat{U}_D) = \underset{e \text{ s.t. } N=\hat{N}, U_D=\hat{U}_D}{\arg\min} L(e). \tag{2}$$

For question (2), with U_D as the only constraint, we aim to determine the optimal allocation of model size N and epochs e. Since performance under data constraints is non-monotonic w.r.t. both N and e, the loss surface admits at least one minimum. We therefore fit the joint allocation of N and e that minimizes L:

$$e_{\text{opt}}(\hat{U}_D), N_{\text{opt}}(\hat{U}_D) = \underset{e,N \text{ s.t. } U_D=\hat{U}_D}{\arg\min} L(e, N). \tag{3}$$

In §3.2, we plot the predicted loss contour $L(N, U_D, e)$, and gave practical suggestions based on the results of Equation (2) and (3). E.g., we can train a 10B model for maximally 1098 epochs on 1T data before seeing a rise in the loss.

Key modeling and optimization designs. Beyond the interplays between parameters, dataset size, data repetition, and compute, we also ablate several critical modeling and optimization choices for DLMs. These include transition kernels (§B.1), diffusion schedules (§B.2), curriculum strategies (§B.2), loss formulation (§B.3), and optimization hyperparameters such as learning rate (§B.4), batch size (§B.4), and weight decay (§B.5). Our results show that while DLMs exhibit markedly different scaling coefficients from AR models, the established AR scaling laws for learning rate and batch size transfer directly.

2 Compute-Constrained Scaling Law for Diffusion Language Models

Constrained compute in model training is inevitable—every player in the AGI race faces limited compute budgets while having effectively unlimited model variants to explore. We therefore ask: *Given a fixed FLOPs budget, how should one optimally trade off model size against the number of training tokens?* Following [11], we model the DLM training loss, model size, and dataset size using power-law relationships under the limited-compute, infinite-data regime, where each model is trained for a single epoch.

We present two approaches to address this question. First, we conduct extensive IsoFLOPs runs across a range of compute budgets, varying model sizes up to 11B parameters and dataset sizes up to 260B tokens. This allows us to trace the efficient frontier for compute-optimal allocation between model size and dataset size. Second, we fit the power-law loss function to the final training losses obtained from these IsoFLOPs runs. Both approaches converge on the same conclusion: model size and dataset size should scale proportionally with training compute, i.e., doubling N requires doubling D, consistent with findings for AR models. However, both approaches also suggest a substantially higher fixed data allocation—roughly 2–5$\times$ that of AR models—for a given FLOPs budget, implying that DLMs are more data-hungry when trained for only a single epoch. Note that [22] shows that DLMs achieve higher data potential under multi-epoch training.

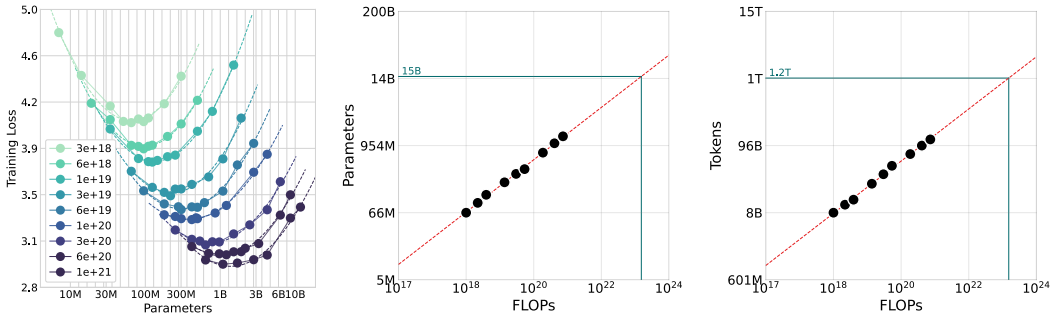

Figure 2: **IsoFLOP curves illustrating the final training loss for a fixed compute budget.** For each curve, we vary the model size and adjust the number of training tokens to maintain constant total training FLOPs. The left panel reveals a distinct performance valley, indicating an optimal trade-off between model size and data for a given compute budget. Leveraging the minima of these curves, we extrapolate the scaling law for the optimal number of parameters and training tokens to larger compute regimes (center and right). The green point highlights our projection for an optimally-scaled model trained with the LLaDA compute budget.

2.1 Approach 1: IsoFLOPs Profiles

In the first approach, we vary model size across nine fixed training FLOPs budgets, ranging from 3×10^{18} to 1×10^{21} FLOPs, and record the final training loss at each point. This directly answers the question: for a given FLOPs budget, what is the compute-optimal parameter count?

For each FLOPs budget, we plot the smoothed final loss against parameter count in Figure 2 (left). In all cases, we train a sufficiently diverse set of model sizes to ensure the loss curve exhibits a clear minimum. We fit a parabola to each IsoFLOPs curve to estimate the parameter count at which the minimum loss occurs (Figure 2, left). We then fit power laws relating compute to the loss-optimal model size and dataset size (Figure 2, center and right), both of which show near-perfect linearity in log-log space. The resulting scaling exponents are $N_{\text{opt}} \propto C^a$ and $D_{\text{opt}} \propto C^b$, with $a = 0.51$ and $b = 0.49$. The fitted formulas are $N \approx 0.0216C^{0.514}$ and $D \approx 7.7C^{0.486}$, as summarized in Table 1.

A head-to-head comparison is that the only dense diffusion language model trained from scratch, LLaDA [23], which consumed 1.1×10^{23} FLOPs, adopted a suboptimal parameter–data allocation. As shown in Figure 2 (center, right), the compute-optimal allocation at this budget would be a 15B-parameter model trained on 1.2T tokens, rather than the 8B model with 2.3T tokens they used. We provide a direct comparison between AR models and DLMs under compute constraints in §2.3.

2.2 Approach 2: Fitting a Parametric Loss Function

The second approach models final training loss as a parametric function of model size N (parameter count) and dataset size D. Following [11], we adopt a functional form based on classical risk decomposition, expressing the loss $L(N, D)$ as:

$$L(N, D) \triangleq E + \frac{A}{N^\alpha} + \frac{B}{D^\beta} \tag{4}$$

This formulation decomposes total loss into three components:

1. **Irreducible Error (E):** The entropy of the true data-generating process, representing the theoretical lower bound on loss, unattainable by any model.
2. **Model Error ($\frac{A}{N^\alpha}$):** Error due to limited model capacity. Even with infinite data, a finite transformer cannot perfectly capture the true distribution. This term decays as model size N increases.
3. **Training Error ($\frac{B}{D^\beta}$):** Error from finite dataset size D. It captures the gap between a finitely trained model and its fully converged counterpart, diminishing as D grows.

Table 1: **A comparison of scaling law coefficients between our model (Quokka) and Chinchilla.** Both DLMs and AR models exhibit similar scaling exponents, implying that the optimal model size and number of training tokens scale at a similar rate. However, for a compute-optimal configuration, our findings suggest allocating $2.2 - 6.7\times$ more training data with a correspondingly smaller model than prescribed by Chinchilla. We also observe that DLMs have a higher irreducible loss.

Approach	E	a	b	k_N	k_D
Chinchilla Approach 1	-	0.50	0.50	0.09	1.88
Chinchilla Approach 2	-	0.49	0.51	0.15	1.15
Chinchilla Approach 3	1.69	0.46	0.54	0.60	0.28
Quokka Approach 1	-	0.51	0.49	0.02	7.70
Quokka Approach 2	2.41	0.50	0.50	0.04	4.10

To estimate the five free parameters (A, B, E, α, β), we regress the functional form against our experimental results. Concretely, we minimize the Huber loss [13] between predicted and observed log-losses using the L-BFGS algorithm [25]:

$$\min_{A,B,E,\alpha,\beta} \sum_{\text{Runs } i} \text{Huber}_\delta \left(\log L(N_i, D_i) - \log L_i^{\text{obs}} \right), \tag{5}$$

where L_i^{obs} denotes the observed loss for run i. Log-loss is standard for fitting power-law relationships. We set $\delta = 10^{-3}$ to enhance robustness to outliers, improving predictive accuracy on held-out data. To avoid convergence to poor local minima, we perform a grid search over initial parameter values and retain the fit with the lowest objective value.

A key application of this parametric model is to derive the compute-optimal allocation of a fixed budget C between model size and dataset size. Assuming compute cost scales as FLOPs$(N, D) \approx 6ND = C$, the optimal N_{opt} and D_{opt} are obtained by minimizing Equation (4) under this constraint. The solution balances model error against training error, yielding a closed-form expression in which both N_{opt} and D_{opt} scale as power laws of C:

$$N_{\text{opt}}(C) = G \left(\frac{C}{6} \right)^a, \quad D_{\text{opt}}(C) = G^{-1} \left(\frac{C}{6} \right)^b, \tag{6}$$

where the scaling exponents a and b, and the constant G, are functions of the fitted parameters from our loss model:

$$G = \left(\frac{\alpha A}{\beta B} \right)^{\frac{1}{\alpha+\beta}}, \quad a = \frac{\beta}{\alpha+\beta}, \quad \text{and} \quad b = \frac{\alpha}{\alpha+\beta}.$$

By construction, $a + b = 1$. The contours of the fitted loss function $\hat{L}$ and the corresponding efficient frontier are shown in Figure 23 (left); Figure 23 (right) shows several isoFLOPs cross-sections of the loss surface, corresponding to the dashed lines in the left panel, with the real data points for a comparison. Our empirical fit, summarized in Table 1, yields exponents $a \approx 0.50$ and $b \approx 0.50$, suggesting that under a fixed compute budget, training data scales at the same pace of parameters. This outcome is fully consistent with approach 1, reinforcing the robustness of the conclusion. From approach 2, the fitted form of Equation (4) is:

$$L(N, D) \approx 2.413 + \frac{798.6}{N^{0.379}} + \frac{4604.9}{D^{0.378}} \tag{7}$$

2.3 Optimal Model Scaling

As detailed above, the optimal parameter count N_{opt} and token budget D_{opt} follow a power-law relationship with compute C: $N_{\text{opt}} \propto C^a$, $D_{\text{opt}} \propto C^b$. Introducing multipliers k_N and k_D, we write $N_{\text{opt}} = k_N C^a$ and $D_{\text{opt}} = k_D C^b$.

Table 1 summarizes the fitted coefficients and compares them directly with Chinchilla scaling. Despite methodological differences, both approaches of Quokka yield consistent exponents a and b, suggesting that model size and training data should scale nearly proportionally with compute.

Table 2: **Optimal FLOPs and training tokens allocation for compute-optimal models.** For a range of model sizes, we plot the estimated training FLOPs and number of tokens required to achieve compute optimal, as predicted by Approach 1, to provide a practical guidance for DLMs training. The estimates for both approach 1 and 2 are close, presented in Table 3. We also included the numbers predicted by Chinchilla scaling law to perform a head-to-head comparison.

	Quokka		Chinchilla	
Parameters	FLOPs	Tokens	FLOPs	Tokens
400 M	9.46e+19	39.3 B	1.92e+19	8.0 B
1 B	5.62e+20	93.5 B	1.21e+20	20.2 B
10 B	4.96e+22	825.2 B	1.23e+22	205.1 B
67 B	2.01e+24	5.0 T	5.76e+23	1.5 T
175 B	1.30e+25	12.4 T	3.85e+24	3.7 T
280 B	3.24e+25	19.3 T	9.90e+24	5.9 T
520 B	1.08e+26	34.6 T	3.43e+25	11.0 T
1 T	3.86e+26	64.2 T	1.27e+26	21.2 T
10 T	3.41e+28	566.4 T	1.30e+28	216.2 T

However, while the exponents align, the multipliers k_N and k_D differ, and these dominate the actual allocation under fixed C when $a \approx b$. Since $C = 6ND$, the constraint $k_N \times k_D = \frac{1}{6}$ holds.

Empirically, Quokka exhibits a 2.2–6.7$\times$ larger k_D than Chinchilla, implying substantially more data and correspondingly fewer parameters are optimal at fixed compute. In practice, for very large FLOPs budgets, even small exponent differences (e.g., 0.51 vs. 0.50) become increasingly important, eventually outweighing multiplier effects (Table 2).

DLMs also exhibit a higher irreducible loss than AR models (2.41 vs. 1.69). This is intuitive: beyond the intrinsic noise in real-world data, diffusion LMs optimize a variational upper bound (ELBO) on the negative log-likelihood. The forward noising process, discretization, and parameterization introduce a non-vanishing variational gap, so even at infinite scale the extrapolated irreducible loss under the diffusion objective remains higher than that of AR models trained directly on NLL.

Note that [11] employed three fitting methods. We merge their approaches 1 and 2 into Quokka approach 1, as they are effectively equivalent. Their approach 3, in contrast, reported negative curvature in the $N \to N_{\text{opt}}$ frontier, yielding lower N_{opt} estimates. Accordingly, for coefficients other than the irreducible loss E, we compare against Chinchilla approaches 1 and 2. The irreducible loss is reported only under their approach 3, i.e., the parametric fit.

Table 2 reports the estimated FLOPs and token counts required for models of different sizes to lie on the compute-optimal frontier, alongside Chinchilla's allocations. Across scales, DLMs consistently allocate 2–5$\times$ more tokens than AR models. This follows naturally: under compute constraints, data is not the bottleneck, so each example is used once. Unlike AR models, DLMs require corruption of inputs during training, effectively demanding more data to represent the same amount of information. As a result, DLMs favor comparatively smaller models trained on substantially larger corpora. These findings offer practical guidance for pre-training DLMs in compute-limited regimes.

3 Data-Constrained Scaling Law for Diffusion Language Models

In the long run, compute will not be the bottleneck in the pursuit of greater intelligence. According to Common Crawl's official statistics [6], web data grows roughly linearly, whereas compute for training AI models grows exponentially [30]. Since compute can be scaled both by increasing chip counts and by extending training time, it is effectively unbounded. By contrast, data constitutes the true limiting factor. In particular, certain domains face acute scarcity, including non-English language data, high-quality code, mathematical text, medical data, etc.

Under data constraints, a practical approach to improving model performance is repeated data usage, such as multi-epoch training. Our primary goal is to quantify the effect of multi-epoch training on performance and its relationship with unique dataset size and model parameter allocation. We address this by modeling the loss landscape with respect to training epochs e, model parameters N, and unique dataset size U. Beyond this, we focus on two key questions:

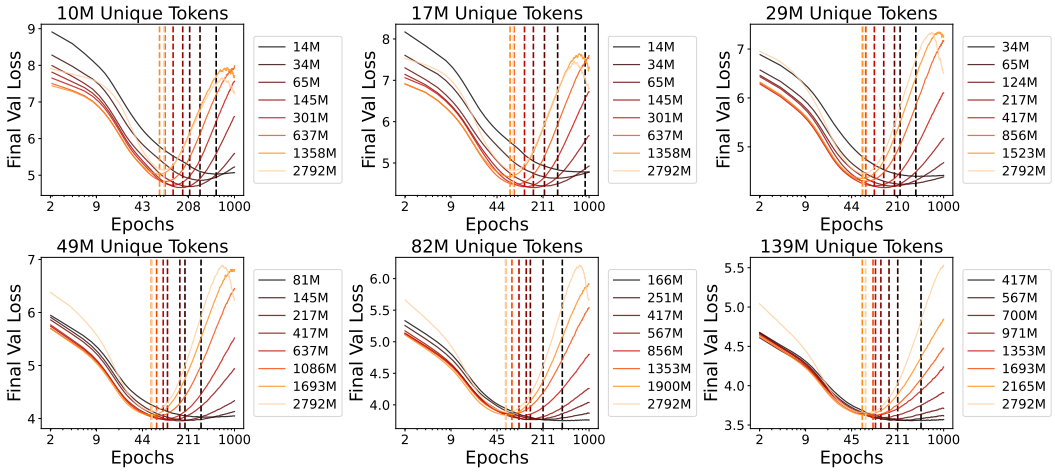

Figure 3: **Final-step validation losses for models of varying sizes trained with different unique data budgets and epochs.** We consistently observe a U-shaped relationship between model size and final validation loss for a fixed data budget, with a minority of runs exhibiting double descent. Larger model sizes tend to accelerate the onset of overfitting (the right side of the "U"), while increasing the number of unique tokens delays it. The minimum achievable loss improves as the amount of unique data increases. These empirical findings provide the motivation for our data-constrained scaling law.

- *Given a fixed model size, a fixed unique-data budget, and unbounded compute, how many epochs can we train before performance degrades?*
- *Given a fixed unique-data budget and unbounded compute, can we predict the optimal allocation between model size and number of training epochs?*

3.1 An Effort in Modeling the Validation Loss with Overfitting

Modeling validation loss is substantially more challenging than modeling pre-training loss. Muennighoff *et al.* [21] proposed Equation (13) to capture validation loss, introducing the notion of diminishing "effective model size" and "effective data size," which reflects the intuition that repeated exposure to the same data yields diminishing performance gains. However, this formulation has a critical flaw: it produces a monotonically non-increasing validation loss, which contradicts reality. In practice, repeated training on the same data inevitably leads to increased validation loss due to overfitting, a direct consequence of the bias–variance tradeoff.

To better characterize the validation loss landscape, we trained a suite of DLMs across varying parameter scales, unique data sizes, and epochs—amounting to 24,000 runs (Figure 3). The results clearly demonstrate the onset of overfitting when training on limited data for extended periods. At the same time, these experiments reveal several intriguing patterns that informed the design of our proposed formulation:

- For any model size and unique data budget, validation loss eventually increases once trained for sufficiently many epochs.
- With a fixed unique data budget, smaller models overfit more slowly.
- With fixed model size, larger unique data budgets delay overfitting.
- The minimum achievable loss decreases monotonically with unique data size.
- For a fixed unique data size, the minimum achievable loss is non-monotonic w.r.t. model size: it first decreases as capacity grows, then increases as overfitting dominates.

With that in mind, we proposed the below formula:

$$L(N, U_D, e) \triangleq E + \frac{A}{N^\alpha} + \frac{B}{D'^\beta}$$

$$\text{where} \quad D' = U_D \cdot e^{p_e} \cdot \exp\left(-\left(\frac{\max(0, e-1)}{e_p}\right)^\gamma\right) \quad \text{and} \quad e_p \quad = c_p \frac{U_D^{m_p}}{N^{k_p}} \tag{8}$$

Our formulation introduces ten coefficients to fit: the irreducible loss E, and the parameters α, β, A, B, c_p, m_p, k_p, p_e, and γ. This functional form extends the Chinchilla scaling law to capture the U-shaped validation loss curves characteristic of multi-epoch training under data constraints. The key modification is replacing the dataset size D in Chinchilla with an "effective dataset size" D', which depends on the number of epochs e, model size N, and unique data size U_D. This formulation has the following desirable properties:

Full learning–overfitting cycle modeling. The effective dataset size D' is defined as the product of a learning term (e^{p_e}) and an overfitting penalty ($\exp(\dots)$). At small e, the learning term dominates, D' increases, and validation loss decreases. At large e, the penalty dominates, D' shrinks, and validation loss rises—capturing the complete learning–overfitting cycle and aligning with the first observation.

Capturing the dynamics of the optimal epoch. The peak overfitting epoch e_p explicitly models the trade-off between model size and data budget. The numerator term $U_D^{m_p}$ ensures that more unique data postpones overfitting (larger e_p), while the denominator term N^{k_p} reflects that larger models overfit more quickly (smaller e_p). This directly accounts for the second and third observations.

Predicting optimal performance limits. The formulation preserves the core structure of a scaling law. A larger unique data budget U_D increases the attainable peak of D', yielding a lower minimum validation loss, consistent with the fourth observation. For the fifth observation, the interaction between the capacity term (A/N^α) and the data–overfitting term (B/D'^β, with D' dependent on N via e_p) reproduces the U-shaped dependence of optimal loss on model size under a fixed data budget.

Natural reduction to the compute-constrained law when $e \leq 1$. A key property of this formulation is that it generalizes compute-constrained formula in a consistent way. At one epoch of training ($e \leq 1$), the $\max(0, e-1)$ term vanishes, the exponential penalty equals 1, and the effective dataset size reduces to $D' = U_D \cdot 1^{p_e} = U_D$. The loss then simplifies to $L = E + A/N^\alpha + B/U_D^\beta$, exactly recovering the compute-constrained law for a model of size N trained on U_D tokens.

By adding only five parameters ($c_p, m_p, k_p, p_e, \gamma$) to the original five compute-constrained coefficients, the formulation effectively models the three-dimensional optimization space of model size, unique data budget, and training epochs.

3.2 Optimal Model Scaling

We fit the proposed formula on 23,145 runs spanning different values of N, U_D, and e, using the same fitting procedure as in the compute-constrained setting. The resulting fitted form is:

$$L(N, U_D, e) = \frac{1535.23}{N^{0.42}} + \frac{54.21}{\left(U_D \cdot e^{1.49} \cdot \exp\left(-\left(\frac{\max(0,e-1)}{254.35 \frac{U_D^{0.39}}{N^{0.55}}}\right)^{0.40}\right)\right)^{0.13}} \tag{9}$$

From the fitted formula, we can interpret that the onset of overfitting scales roughly as $e_{\text{opt}} \propto U_D^{0.39}/N^{0.55}$. The irreducible loss diminishes to a negligible value and is omitted, likely because the interaction among N, U_D, and e implicitly induces an effective lower bound. Using the fitted validation loss form, we plot loss contours in the (N, e) plane under varying U_D, which predict the validation landscape given (N, U_D, e) and provide guidance for the two central questions. The results reveal the existence of local optima when the unique-token budget is fixed. Moreover, larger unique-token budgets generally require both larger models and more epochs to be fully exploited.

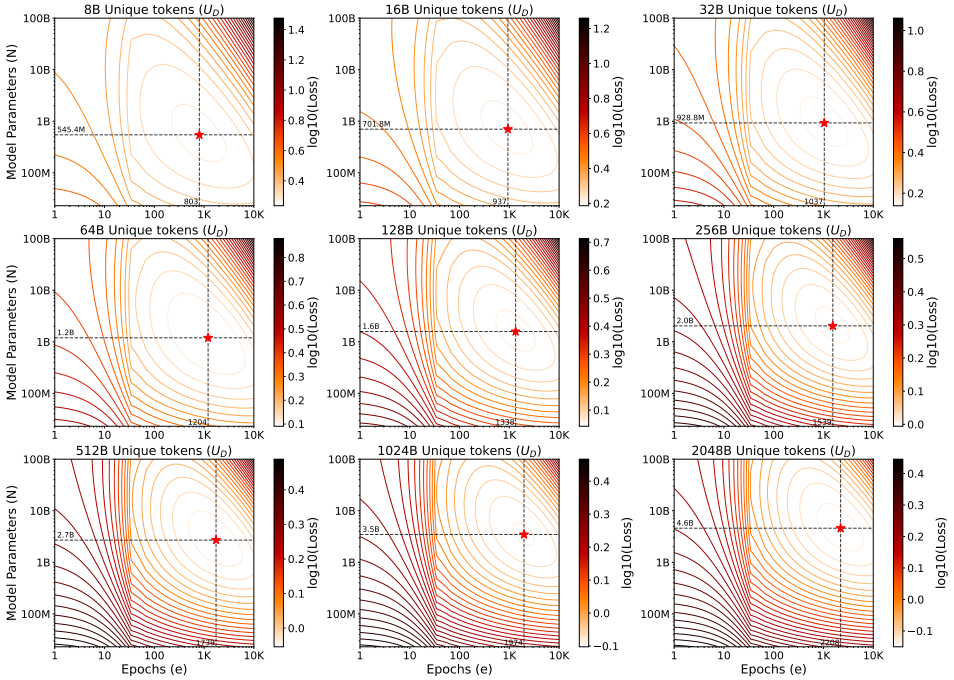

Figure 4: **The loss contours predicted by the fitted data-constrained loss $\hat{L}(N, U_D, e)$.** We exhibit the N - U_D contours with different unique data budgets U_D. We observe a local optima within each observation scope and the optimal N and e consistently grow with e.

However, the extremely low validation losses predicted in the contours may not be fully attainable in practice due to fitting error. More caveats of data-constrained scaling law are detailed in §D

The fitted formula also enables practical guidance for training DLMs under data constraints. Table 5 addresses the first question: *given model size, unique data budget, and unbounded compute, how many epochs can be run before performance degradation occurs?* For reference, we include representative model parameter counts aligned with the compute-constrained scaling law. Table 6 addresses the second question: *given a fixed unique data budget and unbounded compute, what is the optimal allocation of model size and training epochs?*

4 Key Modeling and Optimization Choices

Beyond the interplays between parameters, dataset size, data repetition, and compute, we also ablate several critical modeling and optimization choices for DLMs. These include transition kernels (§B.1), diffusion schedules (§B.2), curriculum strategies (§B.2), loss formulation (§B.3), and optimization hyperparameters such as learning rate (§B.4), batch size (§B.4), and weight decay (§B.5). Due to the space limitation, we defer the detailed discussions to §B.

5 Conclusion

We present Quokka, the first comprehensive scaling law for diffusion language models, establishing a rigorous foundation for both compute- and data-constrained training regimes. Our findings reveal a fundamental shift in optimal resource allocation: to maximize compute efficiency, DLMs require roughly 2–5$\times$ more training data and proportionally smaller models than their autoregressive (AR) counterparts. In scenarios where unique data is the bottleneck, we successfully map the complex U-shaped overfitting dynamics, enabling precise predictions for the optimal balance of model capacity, dataset size, and multi-epoch training. Paired with our evaluation of critical design choices, which highlights the superiority of masked transitions and linear noise schedules, alongside the seamless transferability of AR hyperparameter laws and the necessity of weight decay for numerical stability, this work offers an actionable blueprint for training state-of-the-art DLMs.

Acknowledgment

This project was partially supported by the Singapore Ministry of Education Academic Research Fund Tier 1 (Award Number: T1 251RES2514).

References

[1] A. N. Amin, N. Gruver, and A. G. Wilson, "Why masking diffusion works: Condition on the jump schedule for improved discrete diffusion," *arXiv preprint arXiv:2506.08316*, 2025.

[2] M. Arriola *et al.*, "Block diffusion: Interpolating between autoregressive and diffusion language models," *arXiv preprint arXiv:2503.09573*, 2025.

[3] Y. Bahri, E. Dyer, J. Kaplan, J. Lee, and U. Sharma, "Explaining neural scaling laws," *Proceedings of the National Academy of Sciences*, vol. 121, no. 27, e2311878121, 2024.

[4] T. Brown *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[5] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman, "Maskgit: Masked generative image transformer," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 11 315–11 325.

[6] Common Crawl. "Statistics of common crawl monthly archives: Crawl size." (2025), [Online]. Available: `https://commoncrawl.github.io/cc-crawl-statistics/plots/crawlsize`.

[7] A. Dubey *et al.*, "The llama 3 herd of models," *arXiv e-prints*, arXiv–2407, 2024.

[8] S. Gong *et al.*, "Diffucoder: Understanding and improving masked diffusion models for code generation," *arXiv preprint arXiv:2506.20639*, 2025.

[9] Google DeepMind, *Gemini diffusion: Our state-of-the-art, experimental text diffusion model*, `https://deepmind.google/models/gemini-diffusion/`, Accessed: 2025-09-23, 2025.

[10] R. Han *et al.*, "Deep researcher with test-time diffusion," *arXiv preprint arXiv:2507.16075*, 2025.

[11] J. Hoffmann *et al.*, "Training compute-optimal large language models," *arXiv preprint arXiv:2203.15556*, 2022.

[12] S. Hu *et al.*, "Minicpm: Unveiling the potential of small language models with scalable training strategies," *arXiv preprint arXiv:2404.06395*, 2024.

[13] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics: Methodology and distribution*, Springer, 1992, pp. 492–518.

[14] B. Isik, N. Ponomareva, H. Hazimeh, D. Paparas, S. Vassilvitskii, and S. Koyejo, "Scaling laws for downstream task performance of large language models," *arXiv preprint arXiv:2402.04177*, 2024.

[15] J. Kaplan *et al.*, "Scaling laws for neural language models," *arXiv preprint arXiv:2001.08361*, 2020.

[16] S. Khanna *et al.*, "Mercury: Ultra-fast language models based on diffusion," *arXiv preprint arXiv:2506.17298*, 2025.

[17] H. Li *et al.*, "Predictable scale: Part i–optimal hyperparameter scaling law in large language model pretraining," *arXiv e-prints*, arXiv–2503, 2025.

[18] Z. Liu *et al.*, "Llm360: Towards fully transparent open-source llms," *arXiv preprint arXiv:2312.06550*, 2023.

[19] A. Lou, C. Meng, and S. Ermon, "Discrete diffusion modeling by estimating the ratios of the data distribution," *arXiv preprint arXiv:2310.16834*, 2023.

[20] N. Muennighoff *et al.*, "Olmoe: Open mixture-of-experts language models," *arXiv preprint arXiv:2409.02060*, 2024.

[21] N. Muennighoff *et al.*, "Scaling data-constrained language models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 50 358–50 376, 2023.

[22] J. Ni and team, *Diffusion language models are super data learners*, `https://jinjieni.notion.site/Diffusion-Language-Models-are-Super-Data-Learners-239d8f03a866800ab196e49928c019ac`, Notion Blog, 2025.

[23] S. Nie *et al.*, "Large language diffusion models," *arXiv preprint arXiv:2502.09992*, 2025.

[24] S. Nie *et al.*, "Scaling up masked diffusion models on text," *arXiv preprint arXiv:2410.18514*, 2024.

[25] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of computation*, vol. 35, no. 151, pp. 773–782, 1980.

[26] J. Ou *et al.*, "Your absorbing discrete diffusion secretly models the conditional distributions of clean data," *arXiv preprint arXiv:2406.03736*, 2024.

[27] J. W. Rae *et al.*, "Scaling language models: Methods, analysis & insights from training gopher," *arXiv preprint arXiv:2112.11446*, 2021.

[28] C. Raffel *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[29] S. Sahoo *et al.*, "Simple and effective masked diffusion language models," *Advances in Neural Information Processing Systems*, vol. 37, pp. 130 136–130 184, 2024.

[30] J. Sevilla and E. Roldán. "Training compute of frontier ai models grows by 4–5× per year." (May 2024), [Online]. Available: `https://epoch.ai/blog/training-compute-of-frontier-ai-models-grows-by-4-5x-per-year`.

[31] U. Sharma and J. Kaplan, "Scaling laws from the data manifold dimension," *Journal of Machine Learning Research*, vol. 23, no. 9, pp. 1–34, 2022.

[32] J. Shi, K. Han, Z. Wang, A. Doucet, and M. Titsias, "Simplified and generalized masked diffusion for discrete data," *Advances in neural information processing systems*, vol. 37, pp. 103 131–103 167, 2024.

[33] Y. Song *et al.*, "Seed diffusion: A large-scale diffusion language model with high-speed inference," *arXiv preprint arXiv:2508.02193*, 2025.

[34] D. Su *et al.*, "Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset," *arXiv preprint arXiv:2412.02595*, 2024.

[35] C. Wu *et al.*, "Fast-dllm: Training-free acceleration of diffusion llm by enabling kv cache and parallel decoding," *arXiv preprint arXiv:2505.22618*, 2025.

[36] Y. Wu, Z. Sun, S. Li, S. Welleck, and Y. Yang, "Inference scaling laws: An empirical analysis of compute-optimal inference for llm problem-solving," *The Thirteenth International Conference on Learning Representations*, 2025.

[37] Z. Xie *et al.*, "Dream-coder 7b: An open diffusion language model for code," *arXiv preprint arXiv:2509.01142*, 2025.

[38] F. Xue, Y. Fu, W. Zhou, Z. Zheng, and Y. You, "To repeat or not to repeat: Insights from scaling llm under token-crisis," *Advances in Neural Information Processing Systems*, vol. 36, pp. 59 304–59 322, 2023.

[39] J. Ye *et al.*, "Dream 7b: Diffusion large language models," *arXiv preprint arXiv:2508.15487*, 2025.

[40] F. Zhu *et al.*, "Llada 1.5: Variance-reduced preference optimization for large language diffusion models," *arXiv preprint arXiv:2505.19223*, 2025.

[41] T. Zhu *et al.*, "Skyladder: Better and faster pretraining via context window scheduling," *arXiv preprint arXiv:2503.15450*, 2025.

A Preliminaries

A.1 Chinchilla Scaling Law and Its Data-Constrained Version for AR Models

Chinchilla Scaling Law. [11] studies *compute-constrained* (or compute-optimal) AR pre-training by triangulating evidence from three complementary approaches: (i) **Fixed-Parameters**: vary training tokens D while holding model size N fixed; (ii) **Fixed-FLOPs (IsoFLOP)**: keep total training compute C fixed while co-varying N and D; (iii) **Parametric Fit**: fit a two-factor loss surface $L(N, D)$ and derive the compute-optimal allocation. Its core parametric law is

$$L(N, D) \triangleq E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} \tag{10}$$

with compute $C \approx 6ND$. Minimizing (10) at fixed C yields the allocation

$$N_{\mathrm{opt}}(C) = G\left(\frac{C}{6}\right)^{a}, \qquad D_{\mathrm{opt}}(C) = G^{-1}\left(\frac{C}{6}\right)^{b}, \tag{11}$$

$$\text{where} \quad G = \left(\frac{\alpha A}{\beta B}\right)^{\frac{1}{\alpha+\beta}}, \quad a = \frac{\beta}{\alpha+\beta}, \quad \text{and} \quad b = \frac{\alpha}{\alpha+\beta}. \tag{12}$$

In practice, $a \approx b$, so compute-optimal training scales N and D in near lockstep.

A data-constrained generalization. When unique data is limited, repeated tokens and excess parameters have *diminishing* marginal value. [21] capture this by replacing the raw (N, D) in Equation (10) by their *effective* counterparts (N', D'):

$$L(N, D) \triangleq E + \frac{A}{N'^{\alpha}} + \frac{B}{D'^{\beta}} \tag{13}$$

where D' discounts repetitions and N' discounts parameters beyond those needed for the available unique data. Let $U_D = \min\{D, D_C\}$ be the unique tokens used under a data budget D_C, and let $R_D = \frac{D}{U_D} - 1$ be the number of repeats (epochs beyond the first). Symmetrically, define U_N as the parameters compute-optimal for U_D and $R_N = \frac{N}{U_N} - 1$. Then use simple exponential "half-life" forms:

$$D' \;=\; U_D \;+\; U_D\, R_D^{*}\Big(1 - e^{-R_D/R_D^{*}}\Big), \qquad N' \;=\; U_N \;+\; U_N\, R_N^{*}\Big(1 - e^{-R_N/R_N^{*}}\Big). \tag{14}$$

Here R_D^{*} and R_N^{*} are scale parameters: at $R_D = R_D^{*}$ (resp. $R_N = R_N^{*}$), each repeated token (resp. excess parameter) is worth roughly $(1 - 1/e)$ of a fresh one. A flaw of this formulation is that it assumes validation loss is non-increasing, which is not true in practice.

A.2 Masked Diffusion Language Models

Why masked diffusion? DLMs adopt a noising–denoising framework over sequences. Among their variants, *masked diffusion*—also known as absorbing discrete diffusion, which relies on an absorbing transition kernel—has emerged as the most effective formulation [1]. It preserves discreteness, supports any-order modeling, enables exact position-wise factorization during corruption, and allows flexible likelihood estimation and natively support multi-token prediction. These properties make masked diffusion a strong competitor to AR modeling while retaining many of its practical advantages. Moreover, Ni and team [22] demonstrate that masked DLMs consistently outperform AR models under data-constrained regimes through more repetitions on data. This advantage is likely rooted in DLMs' any-order modeling, high compute-parameter ratio, and inherent data augmentation.

Forward (corruption) process. Let K be the vocabulary size, L the sequence length, and m the mask token. Given a clean sequence $x_0 \in \{0, \ldots, K-1\}^L$, define a monotone diffusion schedule $\alpha_t \in [0, 1]$ with $\alpha_0 = 1$ and $\alpha_1 = 0$, where α_t is the probability that a token is *clean* (unmasked) at noise level $t \in [0, 1]$. The forward process independently masks tokens:

$$q_{t|0}(x_t \mid x_0) = \prod_{i=1}^{L} q_{t|0}\Big(x_t^{(i)} \mid x_0^{(i)}\Big), \qquad q_{t|0}\Big(x_t^{(i)} \mid x_0^{(i)}\Big) = \begin{cases} \alpha_t, & x_t^{(i)} = x_0^{(i)}, \\ 1 - \alpha_t, & x_t^{(i)} = m\,, \end{cases}$$

so that the expected unmasked fraction at level t equals α_t.

Reverse (denoising) process. Starting from the fully masked sequence x_1 and a decreasing schedule $1 = t_0 > t_1 > \cdots > t_N = 0$, the reverse dynamics from t to $s < t$ acts independently across positions:

$$q_{s|t}\Big(x_s^{(i)} \mid x_t\Big) = \begin{cases} 1, & x_t^{(i)} \neq m,\ x_s^{(i)} = x_t^{(i)}, \\ \dfrac{1-\alpha_s}{1-\alpha_t}, & x_t^{(i)} = m,\ x_s^{(i)} = m, \\ \dfrac{\alpha_s - \alpha_t}{1-\alpha_t}\ q_{0|t}\Big(x_s^{(i)} \mid x_t\Big), & x_t^{(i)} = m,\ x_s^{(i)} \in \mathcal{V} \setminus \{m\}, \\ 0, & \text{otherwise.} \end{cases}$$

i.e., already-revealed tokens stay fixed; masked tokens either remain masked with probability $\frac{1-\alpha_s}{1-\alpha_t}$ or are revealed by sampling from a *data-prediction* distribution $q_{0|t}(\cdot \mid x_t)$ with probability $\frac{\alpha_s-\alpha_t}{1-\alpha_t}$. A key *time-agnostic* property [26] of masked diffusion is that

$$q_{0|t}\Big(x_0^{(i)} \mid x_t\Big) = p_{\text{data}}\Big(x_0^{(i)} \,\Big|\, x_t^{\text{UM}}\Big),$$

the conditional distribution of the clean token depends only on the *unmasked* context x_t^{UM}; it does not depend on t beyond which tokens are visible. This allows the denoiser to be parameterized without an explicit time embedding.

Learning objective. Let $p_\theta\Big(x_0^{(i)} \mid x_t\Big)$ approximate $p_{\text{data}}\Big(x_0^{(i)} \mid x_t^{\text{UM}}\Big)$. Masked diffusion maximizes a variational bound on $\log p_\theta(x_0)$, which can be written as minimizing

$$\mathcal{L} \;=\; \int_0^1 w(t;\alpha)\ \mathbb{E}_{q_{t|0}(x_t|x_0)}\left[\sum_{i:\, x_t^{(i)}=m} -\log p_\theta\Big(x_0^{(i)} \mid x_t\Big)\right] \mathrm{d}t, \tag{15}$$

where the importance weight $w(t;\alpha)$ depends only on the schedule and, up to a constant factor, takes the natural form

$$w(t;\alpha) \;=\; \frac{\alpha_t'}{\alpha_t - 1}\,.$$

Intuitively, $w(t;\alpha)$ compensates for the varying expected number of masked positions across noise levels. For the widely used linear schedule $\alpha_t = 1 - t$, this reduces to the familiar integrand weight $w(t) = 1/t$.

B Key Modeling and Optimization Choices

Training optimal diffusion language models depends on more than parameter allocation, dataset size, and training epochs. Here, we ablate additional factors. Given resource constraints, a full ablation is infeasible; instead, we focus on the factors we consider most critical.

We report benchmark results on HellaSwag (commonsense reasoning) and MMLU (knowledge), chosen for their popularity and stability across model configurations [18, 20]. Their broad adoption allows direct comparison with prior work, making them reliable indicators for assessing the impact of our ablations.

B.1 Masked vs. Uniform Transition Kernel

In this ablation, we examine two key variants of discrete diffusion models for language: uniform diffusion and masked (absorbing) diffusion. Their primary difference lies in the state transition rules, which govern how text is corrupted in the forward process and reconstructed in the reverse process.

The **uniform diffusion** model corrupts a sentence by progressively replacing tokens with randomly sampled ones from the vocabulary, eventually reducing the text to uniform noise. Its reverse process learns to denoise this sequence, gradually refining random tokens into a coherent sentence. In contrast, the **masked diffusion** model corrupts text by replacing tokens with a special `[MASK]` token. Its

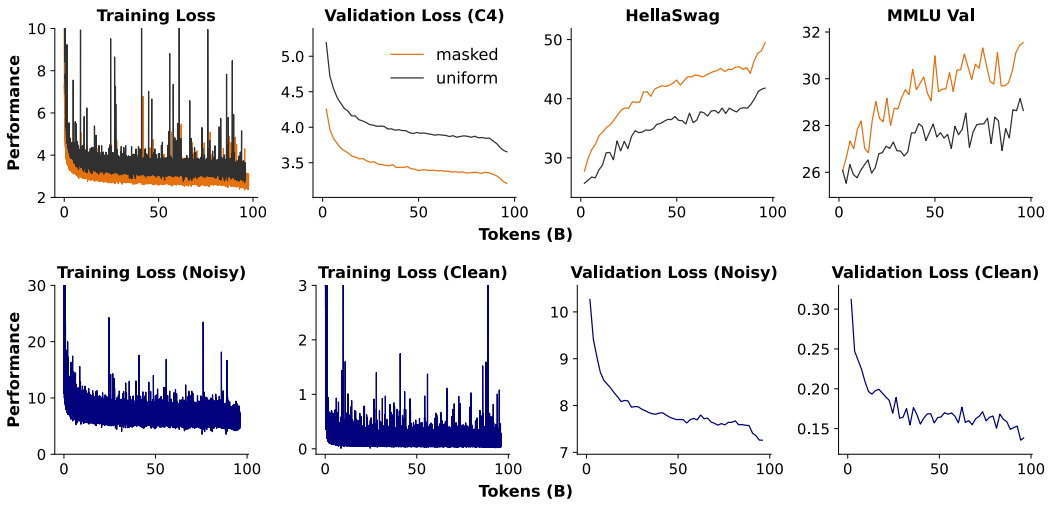

Figure 5: Upper: **Masked and uniform transition kernels.** 1B models are trained on 96B unique tokens. Masked DLMs significantly outperforms the uniform ones. Lower: **The training and validation loss for the clean and corrupted positions of the uniform DLM.**

reverse process resembles a fill-in-the-blank task, predicting the original words within a fully or partially masked sequence.

Formally, both dynamics are defined by a continuous-time Markov process with transition rate matrix Q. In uniform diffusion, transitions from any token to any other occur at a constant rate, yielding the following $N \times N$ matrix for vocabulary size N:

$$Q^{\text{uniform}} = \begin{pmatrix} 1-N & 1 & \cdots & 1 \\ 1 & 1-N & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1-N \end{pmatrix} \tag{16}$$

Here, the off-diagonal entries denote the uniform transition rate to any other token, while the diagonal entries capture the rate of leaving the current token state.

In contrast, masked diffusion restricts transitions to a single absorbing `[MASK]` state. The rate matrix is structured to enforce this one-way corruption in the forward process while simultaneously defining the generative dynamics for the reverse process. Assuming the final index corresponds to the `[MASK]` token, the matrix takes the form:

$$Q^{\text{absorb}} = \begin{pmatrix} -1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -1 & 0 \\ 1 & 1 & \cdots & 1 & 0 \end{pmatrix} \tag{17}$$

The diagonal -1s specify the transition rate from any token to the `[MASK]` state. In the forward process, this drives sequences to become fully masked within finite time. The final row of ones encodes the reverse transitions, allowing the `[MASK]` state to generate any vocabulary token, which the model learns to parameterize.

Masked diffusion is often easier to model, as the task reduces to filling in masked positions rather than distinguishing noise from clean tokens. Both theoretical and empirical studies suggest that masked diffusion models generally outperform uniform ones [1, 19]. However, direct large-scale comparisons under LLM evaluation settings remain absent.

We compare masked and uniform diffusion pre-training using a broad set of metrics. As shown in Figure 5, masked diffusion consistently outperforms uniform diffusion across all metrics by a wide

margin. To further probe the uniform variants, we plot their losses on both clean and noisy positions. Although the model is not explicitly given indicators for these positions, it learns to distinguish most noisy from clean tokens with low loss (around 0.15). This suggests that the main challenge lies not in identifying noisy versus clean tokens, but in transforming arbitrary embeddings into the correct ones.

It is worth noting that the uniform diffusion loss used here does not compute an exact ELBO, as multiple variants exist and complicate head-to-head comparisons. Instead, we adopt the same reweighting scheme as masked diffusion for noisy positions and average the loss over clean ones, enabling a direct comparison of the learning difficulty between masking and uniform transitions. Additionally, uniform diffusion includes a continuous-time embedding layer, which introduces minimal parameter overhead.

B.2 Diffusion Schedules

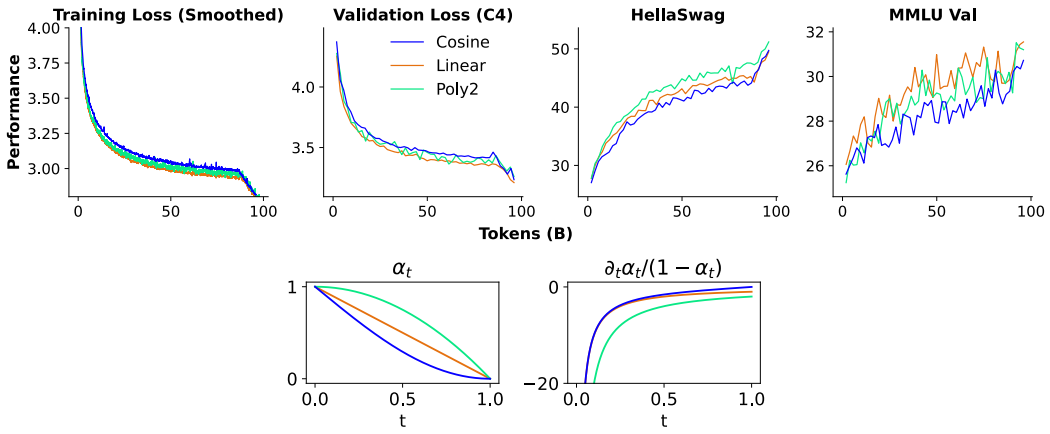

Figure 6: Upper: **Three commonly used diffusion schedules and their performances.** 1B models are trained on 96B unique tokens. Lower: **The shapes of α_t and cross entropy reweighting $\partial_t\alpha_t/(1-\alpha_t)$.**

The diffusion schedule is a central design choice in training DLMs. We ablate two types of schedules. The first is the standard diffusion schedule, a predefined sequence controlling the rate and manner of noise injection and removal at each step. The second is the noise-level sampling schedule across the training lifecycle, where different noise levels are sampled at different stages.

Diffusion schedule. We examine three common schedules: linear, poly2, and cosine, defined as $\alpha = 1 - t$, $\alpha = 1 - t^2$, and $\alpha = 1 - \cos\left(\frac{\pi}{2}(1-t)\right)$, respectively. Figure 6 (bottom) illustrates their shapes. Cosine assigns higher-than-average probability to masking, poly2 lower, and linear lies in between. From pre-training and evaluation results (Figure 6), cosine performs worst across metrics, while linear consistently outperforms the others in both train/val loss and MMLU. Linear also exhibits lower variance than nonlinear schedules, consistent with [32]. The poly2 schedule achieved better performance on HellaSwag.

Training-time noise schedule. A natural intuition in training DLMs is to begin with cleaner data and gradually increase noise, aiming for stronger end-of-training performance [41]. This is straightforward to implement by adjusting the sampling of t. In our setup, we use a moving Gaussian window to bias t toward lower values early in training, so the model first learns easier prediction tasks before progressively transitioning to harder ones as the Gaussian window shifts from 0 to 1. Results show that this schedule yields faster loss reduction in the early stages, followed by rising loss as noisier samples dominate. It achieves slightly better end-of-training performance across both benchmarks, suggesting this direction merits further study.

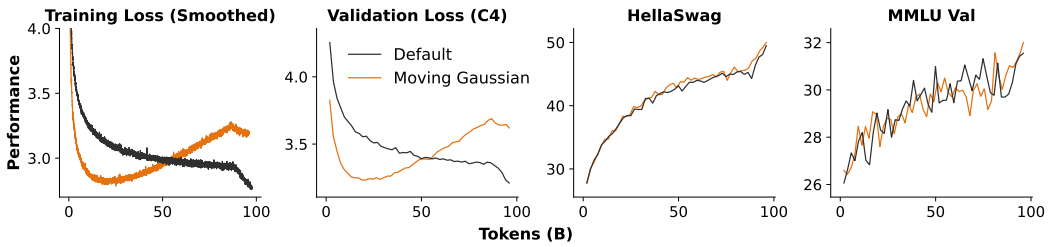

Figure 7: **Uniform t vs. clean-to-noisy t sampling**, where a moving Gaussian window gradually shifts from low-noise sampling early in training to high-noise sampling later, implementing an easy-to-hard curriculum. 1B models are trained on 96B unique tokens.

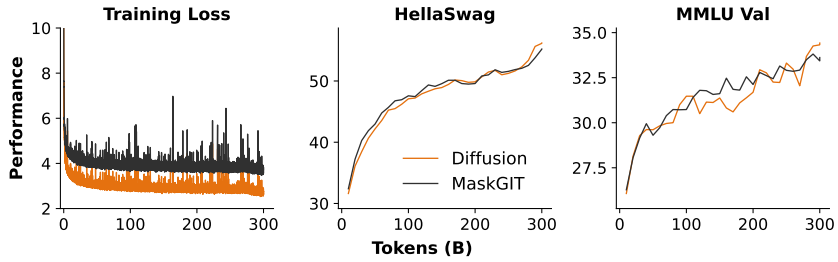

Figure 8: **Principled diffusion loss (Equation** (15)**) and MaskGIT loss (Equation** (15) **without reweighting).** 1B models are trained on 300B unique tokens.

B.3 Diffusion Loss Formula

Generative masked language models can be trained using either the principled diffusion loss [32] or the masked loss [5]. The diffusion loss is generally regarded as more faithful, since it optimizes a likelihood lower bound and is expected to yield better results. We compare masked generative models trained with diffusion loss (Equation (15)) and MaskGIT loss (Equation (15) without reweighting) over 300B tokens. Surprisingly, despite not optimizing a principled ELBO, MaskGIT achieves consistently comparable performance throughout training and even converges faster on both evaluations. While diffusion loss ultimately delivers stronger end-of-training performance on both benchmarks, this finding highlights the need for further study of how theoretical bounds influence training dynamics.

B.4 Batch Size and Learning Rate Transferability

Batch size. Training hyperparameters such as batch size are critical for stability and performance. AR models have well-established scaling laws for these settings [17]. Batch size is closely tied to dataset size, and diffusion language models (DLMs) effectively augment data through noise injection [22], often exhibiting higher variance in pre-training loss. Larger batches can mitigate both issues, raising the question of whether DLMs favor larger batch sizes than AR counterparts. Surprisingly, as shown in Figure 9, training dynamics for AR and DLMs are similar: batch size 4096 lags behind 256 and 1024, with the latter two performing comparably. This suggests that changing the training objective does not alter the optimal batch size when data and model architecture are fixed, implying that established AR scaling laws might be able to transfer directly to DLMs.

Learning rate. We also examine the transferability of learning rate, another key hyperparameter. We grid search three peak values ranging from small to large. As shown in Figure 10, both AR and DLM models show minimal differences in end-of-training performance across learning rates after annealing, with 1e-4 yielding a slight advantage. Convergence speed differences across learning rates are also consistent between AR and DLMs. These results further support that the training objective does not alter the optimal hyperparameter space, and that DLMs can directly reuse established learning rate practices from AR models.

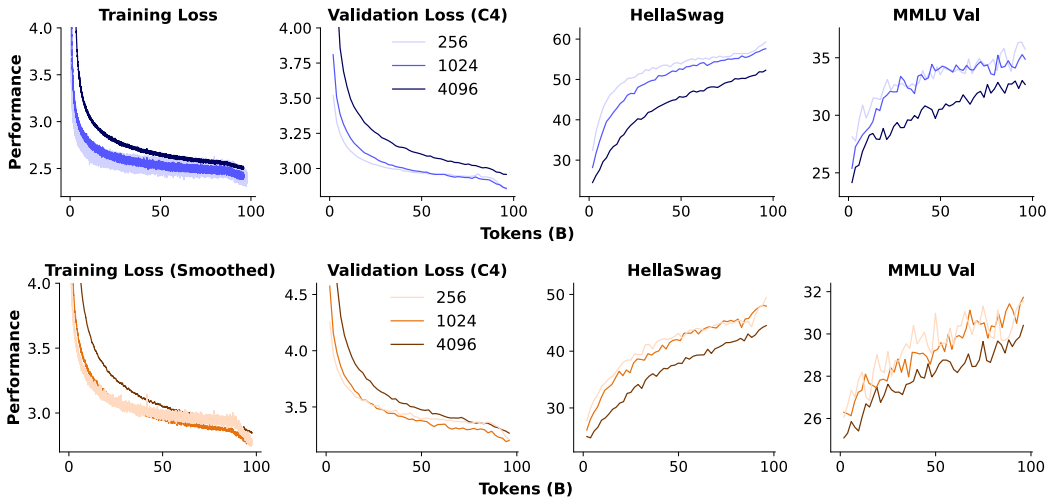

Figure 9: **Batch size transferability from AR models (upper) to DLMs (lower).** Both show consistent trends across batch sizes, suggesting that DLM training can leverage batch size laws from AR studies. 1B models are trained on 96B unique tokens.

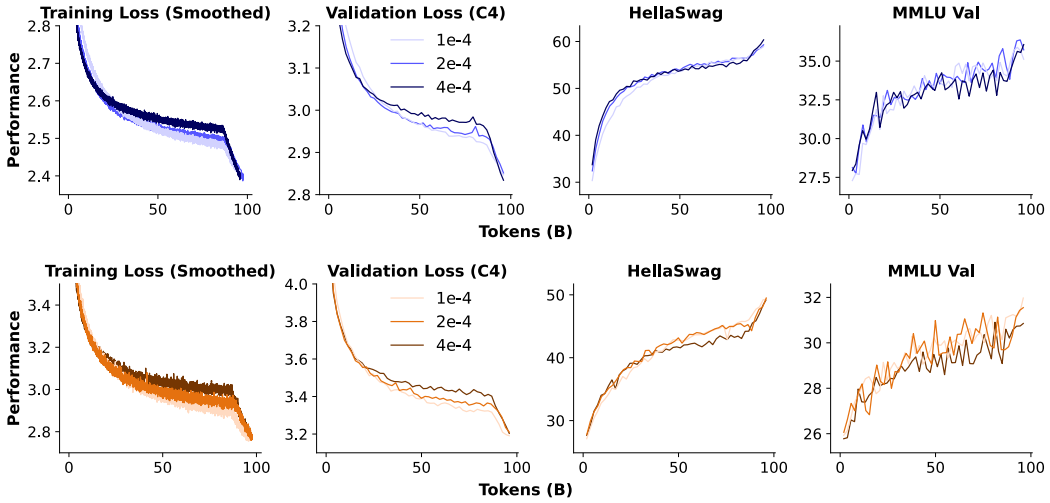

Figure 10: **Learning rate transferability from AR models (upper) to DLMs (lower).** Both show consistent trends across learning rates, suggesting that DLM training can leverage learning rate laws from AR studies. 1B models are trained on 96B unique tokens.

B.5 Weight decay

Weight decay is a standard technique in LLM pre-training to keep parameter norms stable and mitigate issues such as overfitting and numerical instability. We investigate its effect in DLM pre-training by comparing AR and DLM models with and without weight decay under two settings: single-epoch and multi-epoch training.

Single-epoch training. We train models for 96B tokens over 1 epoch. As shown in Figure 11 and Figure 12, neither AR nor DLM benefits from weight decay in this regime; in fact, removing weight decay leads to faster convergence.

Multi-epoch training. We train models on 1B unique tokens for 96 epochs, a setting prone to overfitting where weight decay is expected to play a larger role. As shown in Figure 13, removing weight decay severely degrades AR models' validation loss and benchmark performance. In contrast,

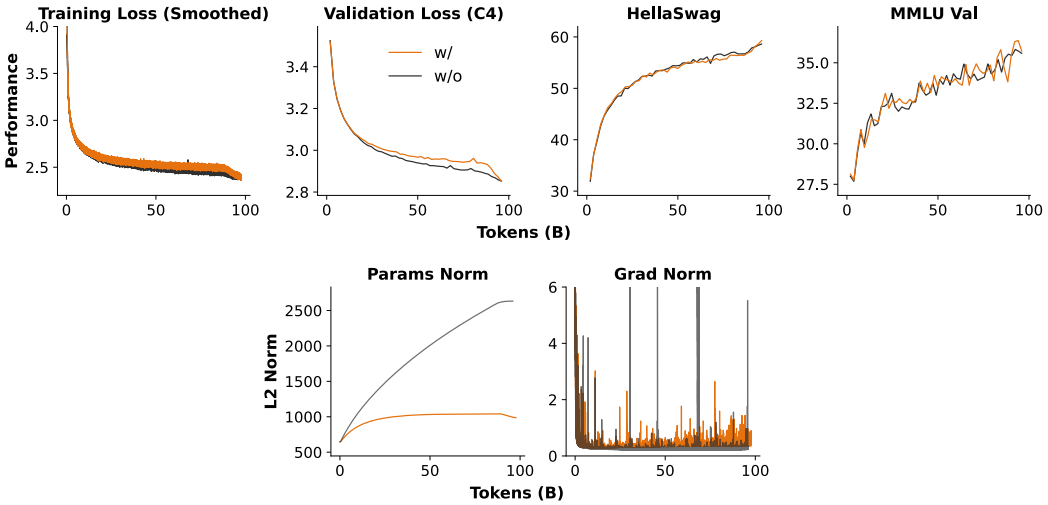

Figure 11: **The impact of weight decay on AR models in single epoch scenarios.** 1B AR models are trained with and without weight decay, on 96B unique tokens.

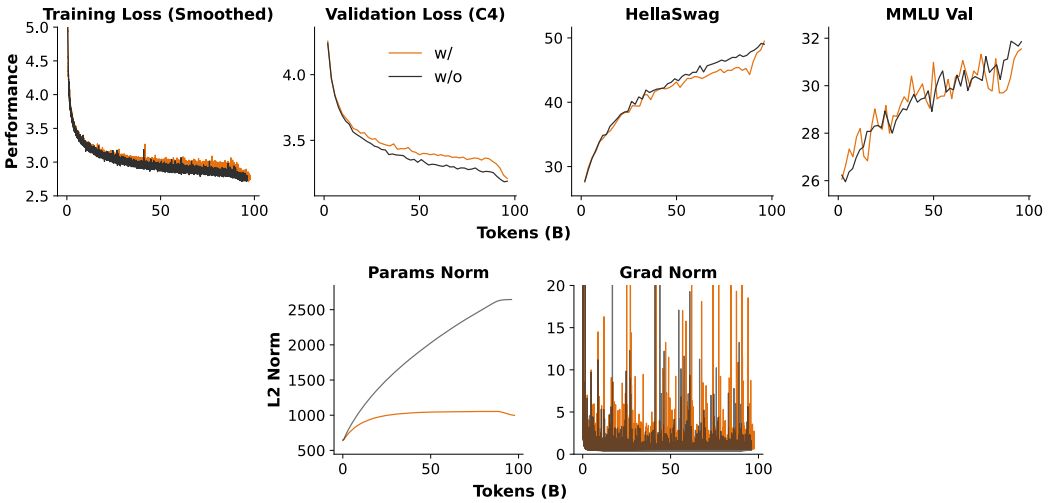

Figure 12: **The impact of weight decay on DLMs in single epoch scenarios.** 1B DLMs are trained with and without weight decay, on 96B unique tokens.

DLMs remain largely unaffected and appear robust to data repetition even without weight decay. That said, applying weight decay still yields better end-of-training results across all metrics.

Although weight decay shows limited benefit in 3 of 4 ablations, maintaining healthy parameter norms remains important. As shown in Figure 11–14, removing weight decay consistently increases the L2 norm of parameters, risking numerical instability. In practice, this can cause logits before softmax to collapse, since `bf16` provides only 7 mantissas bits and quantization becomes coarse for values above 128, substantially harming both training and inference performance.

C Related Work

C.1 Scaling Laws

Understanding how scaling affects large language model (LLM) performance has been a central research focus. The seminal work of Kaplan *et al.* [15] showed that model performance follows

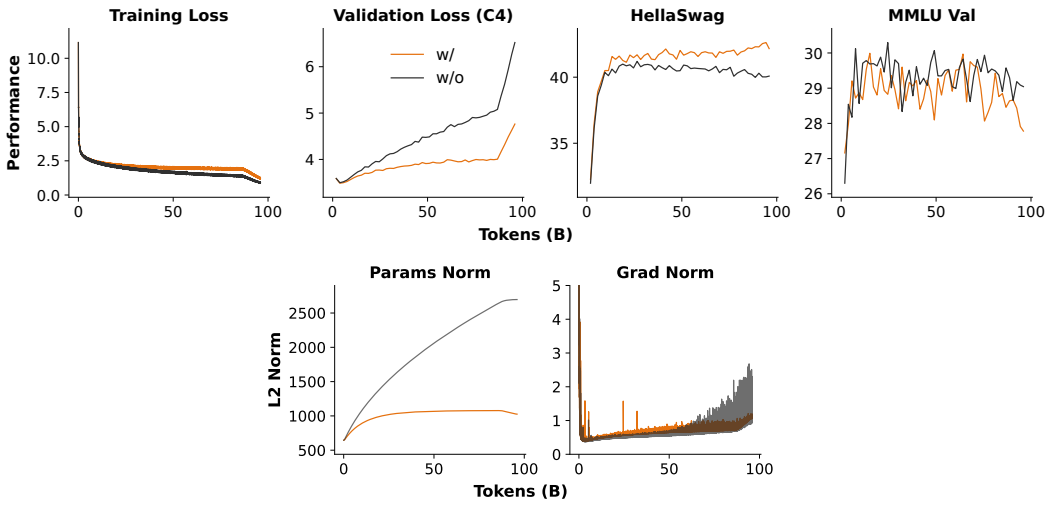

Figure 13: **The impact of weight decay on AR models in multi-epoch scenarios.** 1B AR models are trained with and without weight decay, on 1B unique tokens for 96 epochs.

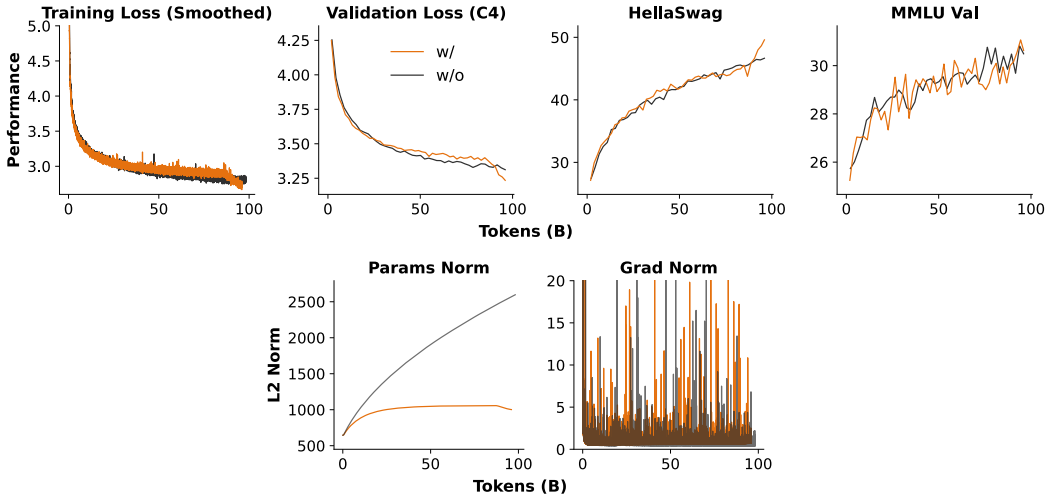

Figure 14: **The impact of weight decay on DLMs in multi-epoch scenarios.** 1B DLMs trained are with and without weight decay, on 1B unique tokens for 96 epochs.

predictable power-law trends with respect to model size, compute, and training data, implying that ever-larger models should yield better results. This paradigm shaped the development of models such as GPT-3 [4]. However, Hoffmann *et al.* [11] challenged this view with Chinchilla, demonstrating that, under a fixed compute budget, optimal performance arises from scaling model size and training data in tandem. This revealed that many prior models, including Gopher [27], were undertrained, shifting the field's understanding toward balanced scaling and more efficient compute utilization.

Subsequent research has refined our understanding of scaling laws, particularly in data-constrained regimes. The Chinchilla laws, though influential, assume effectively unlimited training data. As models scale further, the scarcity of unique, high-quality data has emerged as a critical bottleneck. Muennighoff *et al.* [21] introduced a data-constrained scaling law (Equation (13)) to model validation loss under limited data, where repeated exposure reduces the "effective model size" and "effective data size." While this captures diminishing returns, the formulation has a key limitation: it enforces a non-increasing validation loss, whereas in practice repeated epochs inevitably induce overfitting, increasing validation loss due to the bias–variance tradeoff. In this work, we propose a new formulation that addresses this flaw.

Beyond pre-training loss, scaling law research is expanding to downstream task performance [14], inference dynamics [36], and theoretical grounding, linking empirical trends to concepts such as data manifold dimensionality [3, 31]. This broadening scope underscores the need for more refined laws that integrate model architecture, data quality, and task-specific requirements. Li *et al.* [17] further explored hyperparameter scaling, offering practical guidance for pre-training choices.

For DLMs, systematic scaling laws were lacking prior to Quokka. Nie *et al.* [24] trained models at low FLOPs budgets to compare scaling trends of AR models and DLMs, marking an important step toward DLM scaling, though their study provided only limited scaling law coefficients & insights.

C.2 Diffusion Language Models

Building on the theoretical foundations of DLMs [19, 26, 29, 32], Nie *et al.* [23] trained the first large-scale DLM from scratch, achieving performance competitive with leading open-source AR models [7]. In parallel, several commercial DLMs have emerged, demonstrating strong coding and math capabilities while offering significantly lower generation latency [9, 16, 33]. Ni and team [22] further showed that DLMs possess substantially higher data potential than AR models under limited data, enabling so-called "intelligence crossovers" that highlight their advantage in the face of the token crisis [21, 38].

Efforts have also explored hybrid approaches bridging AR and diffusion. Block diffusion [2] performs block-wise diffusion, with block size 1 similar to AR modeling without shift. Dream [39] initialized DLMs with AR priors and employed a "shift-by-one" strategy to better retain AR knowledge, offering another effective training paradigm. Recent work has also advanced DLM coders [8, 37], DLM RL scaling [40], accelerated inference techniques [35], pushing DLMs toward greater practicality and competitiveness.

D Discussions

In practice, model training is often constrained by resources beyond compute–leading to deviations from the allocations prescribed by scaling laws. For instance, Llama 3 [7] trained an 8B model with 15T tokens, whereas the Chinchilla law would suggest a 70B model for 2T tokens. Several factors contribute to such deviations: (1) Scaling-optimal allocation is not the only consideration for commercial models; factors such as deployability, customer adoption, and hardware compatibility (e.g., GPU/TPU memory limits) play a decisive role. (2) Compute budgets are not always strict. In many cases, one can effectively "expand" compute by extending training time, making smaller models with more data or epochs more practical than adhering rigidly to scaling predictions. (3) Current compute- and data-constrained scaling laws are limited in scope, and their coefficients can shift across architectures and datasets. Thus, scaling laws should be viewed as high-level guidance on balancing model size, data, and training duration, while precise choices require empirical tuning under specific constraints.

Data-Constrained Scaling Law Caveats (1) The validation loss landscape remains poorly understood, and its mathematical form is far from established. We do not have a strict theoretical justification for our formulation, and thus cannot claim it holds universally. For instance, we observed double-descent behavior in a small subset of long-epoch runs. In our fitting, we assume a single descent and truncate the second peak for these cases. (2) In Figure 15, we compared actual N–e optima across data budgets against the fitted ones. The fitted contours tend to overshoot epochs and underestimate model size. Similarly, in Figure 16, we show actual vs. predicted validation losses for randomly sampled (N, U_D, e). While Equation (9) captures the overall loss shape, noticeable gaps remain in some cases. §F provides alternative formulations and predictions that may also be plausible but resulted in higher loss in the fitting. (3) Validation loss values depend heavily on the choice of validation set and tokenizer, making absolute values less meaningful. The emphasis should instead be on trends and the dynamic interplay among variables.

E Implementation Details

All experiments were conducted with a heavily modified Megatron-LM codebase. Compute-constrained runs and ablations were trained on a subset of the Nemotron-CC corpus [34], while

data-constrained runs used a subset of the c4-en corpus [28]. Validation losses were consistently evaluated on the c4-val split, following Muennighoff *et al.* [21]. Token budgets were randomly sampled from the respective corpora without additional filtering. Model parameters were initialized from a normal distribution with standard deviation 0.02. Architecturally, we adopted a performant configuration combining the GPT-2 tokenizer, RoPE, SwiGLU, pre-layer RMSNorm, bias-free layers, and qk normalization.

For compute-constrained runs, we applied Gaussian smoothing with a window size of 301 (vs. 10 in Chinchilla), reducing variance by $\sim 13\times$. This substantially improved fitting stability at the cost of a mild bias, corresponding to a $\sim$40-step lag. Learning rates were set to $2\mathrm{e}{-4}$ for models $<$8B and $1.25\mathrm{e}{-4}$ for models $>$8B, with a cosine decay schedule. To reuse prior epoch runs and collect stable data points, we employed the Warmup-Stable schedule [12] with peak learning rate $2\mathrm{e}{-4}$.

All models were trained with sequence length 2048. Batch size was scaled with model size: 256 for $\leq$1.5B, 512 for 1.5B–5B, and 1024 for $>$5B, the latter chosen for stability.

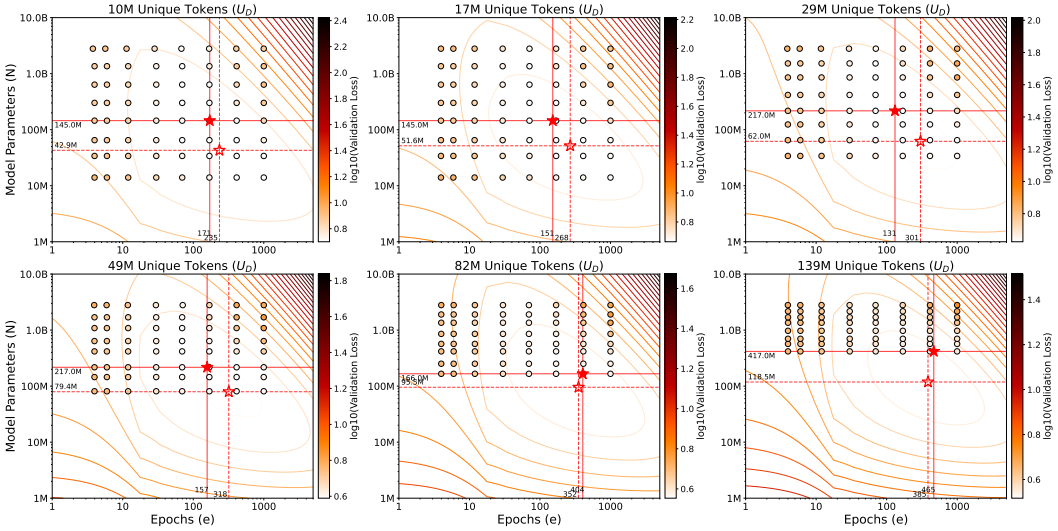

Figure 15: **The contours predicted by Equation** (9) **v.s. the real data points.** Equation (9) tend to overshoot the epochs and under-estimate the model sizes.

F Alternative Data-Constrained Formulas and Fitting Results

In this section we present the alternative data-constrained validation loss formulas, which are among the most effective ones we tried, but the losses (31.52 and 23.8 over 23145 data points) are still far from Equation (9) (9.78 over 23145 data points).

F.1 Additive Overfitting Term v1

Equation (18) presents an additive formula breaking down the data-constrained scaling law into learning loss and overfitting penalty, with the fitted form in Equation (21). The predicted contours are presented in Figure 17 and the contours v.s the actual data points are in Figure 19.

$$L(e, N, U_D) \triangleq \underbrace{E + \frac{A}{N^\alpha} + \frac{B}{(D'(e, U_D))^\beta}}_{\text{Learning Loss}} + \underbrace{\mu \left(\frac{N}{U_D}\right)^\delta (\log(\max(1, e)))^\gamma}_{\text{Overfitting Penalty}}, \tag{18}$$

$$\text{where} \quad D'(e, U_D) = U_D \left(1 + R_D^* \left(1 - exp(\frac{-\max(0, e-1)}{R_D^*})\right)\right), \tag{19}$$

$$D'(e, U_D) \approx U_D \cdot \max(1, e), \text{as} \quad R_D^* \quad \text{is very large.} \tag{20}$$

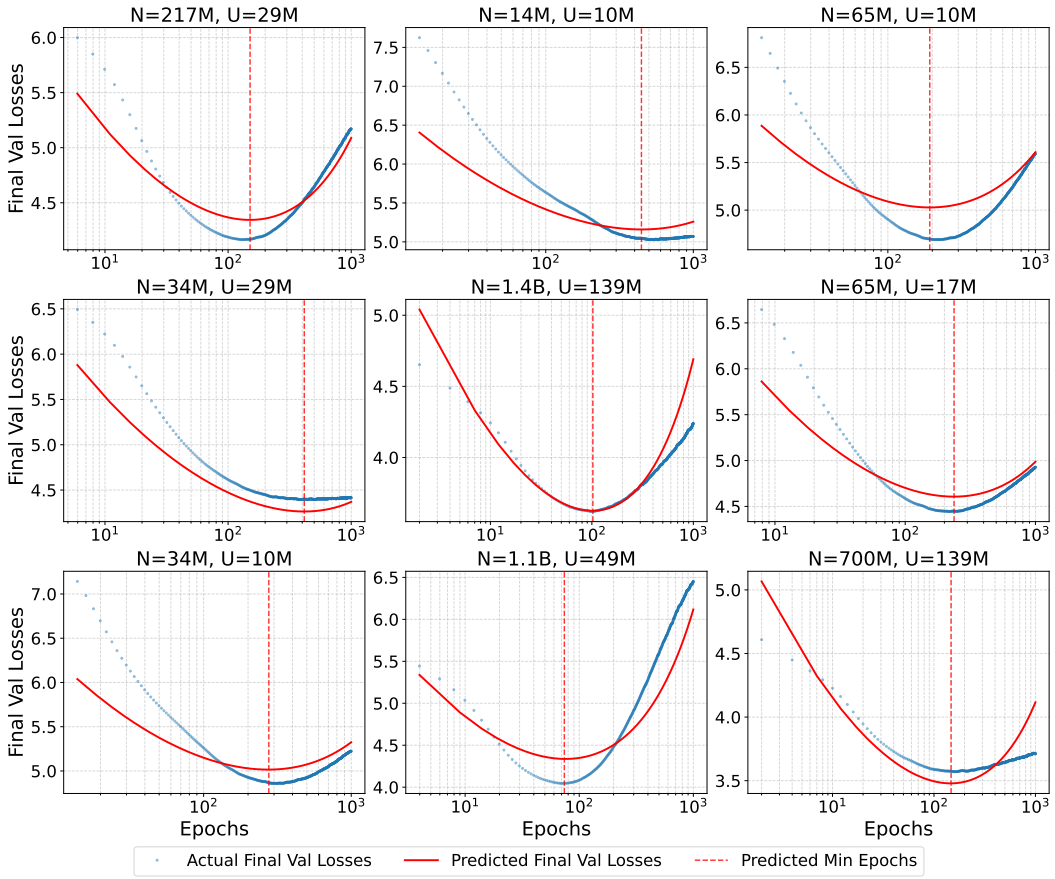

Figure 16: The validation losses predicted by Equation (9) v.s. the real validation losses.

$$L(e, N, U_D) \approx \frac{145962.2}{N^{0.73}} + \frac{61.1}{[U_D \cdot e]^{0.13}} + 58 \times 10^{-4} \left(\frac{N}{U_D}\right)^{0.43} (\log(\max(1, e)))^{4.49} \tag{21}$$

F.2 Additive Overfitting Term v2

Similarly, Equation (22) presents an additive formula breaking down the data-constrained scaling law into learning loss and a more complicated overfitting penalty (after trials), with the fitted form in Equation (23). The predicted contours are presented in Figure 21 and the contours v.s the actual data points are in Figure 22.

$$L(e, N, U_D) = \underbrace{E + \frac{A}{N^\alpha} + \frac{B}{D'^\beta}}_{\text{Learning Loss}} + \underbrace{\mu \left(\frac{N}{D'}\right)^\delta \left[\text{softplus}\left(\frac{e - \kappa \left(\frac{U_D}{N}\right)^\eta}{\tau}\right)\right]^\gamma}_{\text{Overfitting Penalty}} \tag{22}$$

$$\text{where} \quad D'(e, U_D) = U_D \left(1 + R_D^* \left(1 - \exp\left(-\frac{e-1}{R_D^*}\right)\right)\right)$$

$$\text{and} \quad \text{softplus}(x) = \log(1 + e^x)$$

$$L(e, N, U_D) \approx 9.505 \times 10^{-66} + \frac{2.738}{N^{1.240}} + \frac{53.58}{D'^{0.1207}} + 0.1610 \left(\frac{N}{D'}\right)^{0.3073} \left[\text{softplus}\left(\frac{e - 12642\left(\frac{U_D}{N}\right)^{1.486}}{26.56}\right)\right]^{0.8106} \tag{23}$$

$$\text{where} \quad D'(e, U_D) = U_D \left(1 + 33.62\left(1 - \exp\left(-\frac{e-1}{33.62}\right)\right)\right)$$

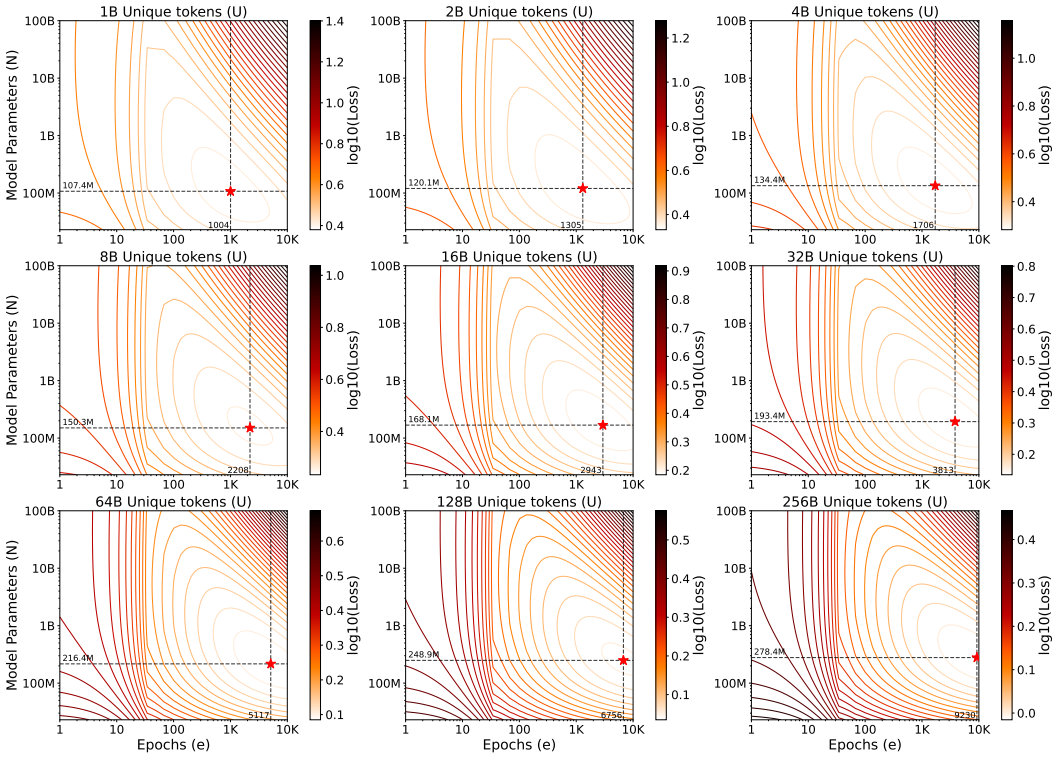

Figure 17: The contours predicted by Equation (18) and the optimal allocations.

Table 3: **The FLOPs and Tokens allocation predicted by approach 2 and 3.** Similar to [11], the loss fitting approach under-estimates $N \to N_{opt}$ for very large models.

	Approach 1		Approach 2	
Parameters	FLOPs	Tokens	FLOPs	Tokens
400 M	9.46e+19	39.3 B	1.06e+20	44.3 B
1 B	5.62e+20	93.5 B	6.68e+20	111.2 B
10 B	4.96e+22	825.2 B	6.74e+22	1.1 T
67 B	2.01e+24	5.0 T	3.05e+24	7.6 T
175 B	1.30e+25	12.4 T	2.09e+25	19.9 T
280 B	3.24e+25	19.3 T	5.35e+25	31.8 T
520 B	1.08e+26	34.6 T	1.85e+26	59.3 T
1 T	3.86e+26	64.2 T	6.86e+26	114.3 T
10 T	3.41e+28	566.4 T	6.93e+28	1153.9 T

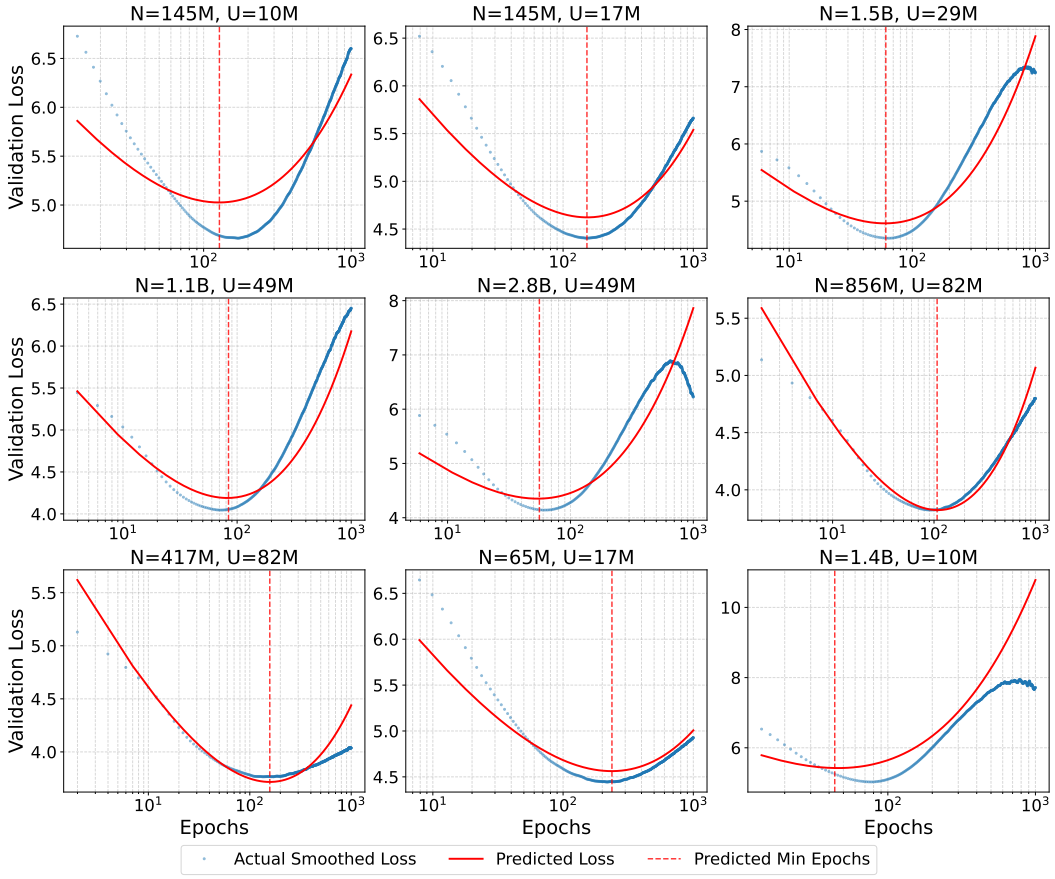

Figure 18: The contours predicted by Equation (18) v.s. the real data points.

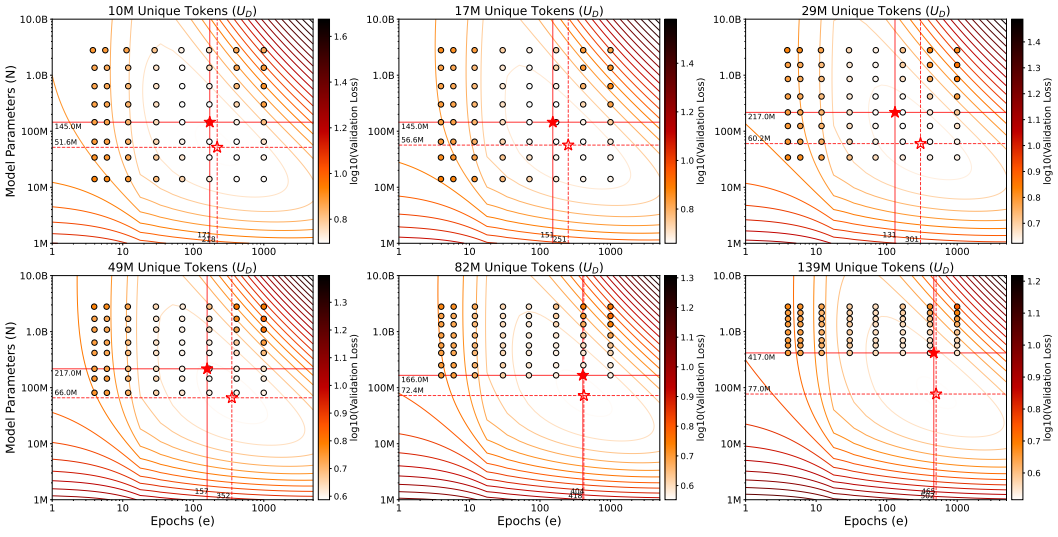

Figure 19: The validation losses predicted by Equation (18) v.s. the real validation losses.

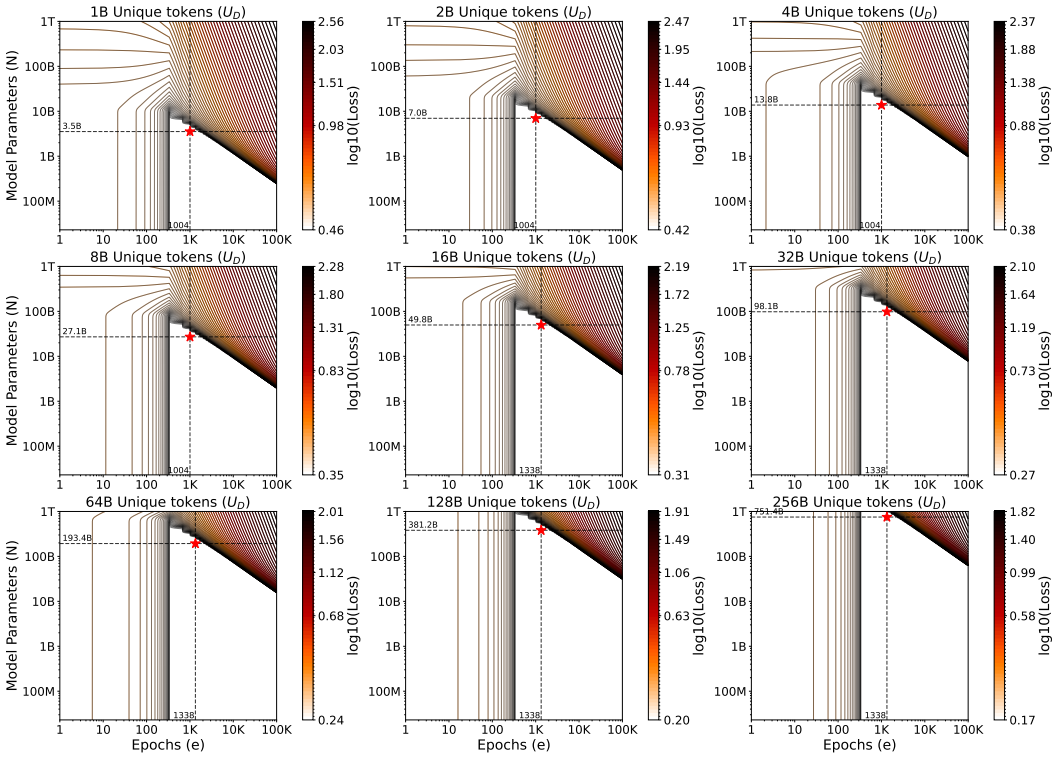

Figure 20: The contours predicted by Equation (22) and the optimal allocations.

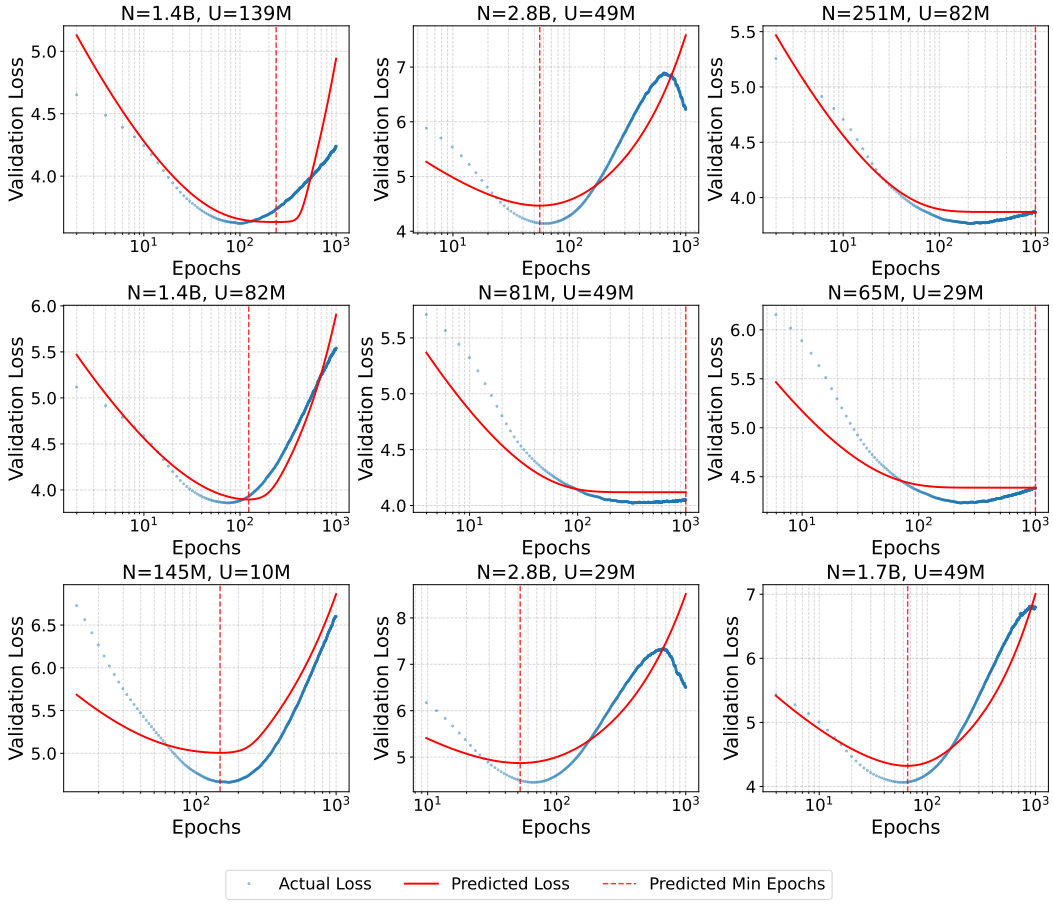

Figure 21: The contours predicted by Equation (22) v.s. the real data points.

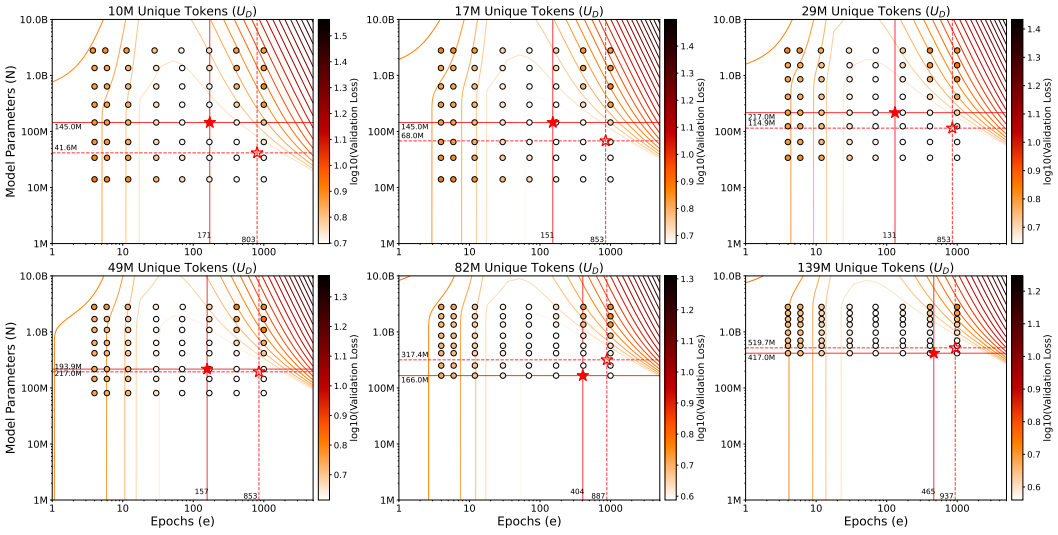

Figure 22: The validation losses predicted by Equation (22) v.s. the real validation losses.

Table 4: The model arch details for all models trained in this work. All models used archs described in §E.

Parameters (million)	d_model	ffw_size	kv_size	n_heads	n_layers
1	128	512	32	4	3
2	224	896	32	7	4
5	288	1,152	32	7	5
7	320	1,280	32	10	6
14	448	1,792	32	7	6
25	512	2,048	64	8	8
36	576	2,304	64	9	9
49	640	2,560	64	10	10
64	640	2,560	64	10	13
79	640	2,560	64	10	16
85	768	3,072	64	12	12
106	768	3,072	64	12	15
127	768	3,072	64	12	18
135	896	3,584	64	14	14
154	896	3,584	64	14	16
173	896	3,584	64	14	18
201	1,024	4,096	64	16	16
226	1,024	4,096	64	16	18
252	1,024	4,096	64	16	20
354	1,280	5,120	128	10	18
413	1,280	5,120	128	10	21
428	1,408	5,632	128	11	18
472	1,280	5,120	128	10	24
500	1,408	5,632	128	11	21
538	1,536	6,144	128	12	19
571	1,408	5,632	128	11	24
623	1,536	6,144	128	12	22
708	1,536	6,144	128	12	25
771	1,792	7,168	128	14	20
886	1,792	7,168	128	14	23
1,002	1,792	7,168	128	14	26
1,107	2,048	8,192	128	16	22
1,250	2,176	8,704	128	17	22
1,258	2,048	8,192	128	16	25
1,409	2,048	8,192	128	16	28
1,420	2,176	8,704	128	17	25
1,529	2,304	9,216	128	18	24
1,591	2,176	8,704	128	17	28
1,784	2,304	9,216	128	18	28
2,038	2,304	9,216	128	18	32
2,045	2,560	10,240	128	20	26
2,359	2,560	10,240	128	20	30
2,674	2,560	10,240	128	20	34
3,121	2,688	10,752	128	21	36
3,426	2,816	11,264	128	22	36
3,744	2,944	11,776	128	23	36
4,077	3,072	12,288	128	24	36
6,166	3,584	14,336	128	28	40
8,456	4,096	16,384	128	32	42
10,682	4,352	17,408	128	32	47
11,211	4,608	18,432	128	36	44
11,976	4,608	18,432	128	32	47
13,343	4,864	19,456	128	32	47
14,653	4,992	19,968	128	32	49
14,785	5,120	20,480	128	40	47

Table 5: **The maximum epochs one can train given the model parameters N and unique tokens U_D**, predicted by the fitted data-constrained loss function 9, answering question 1.

N / U_D	10 M	100 M	1 B	10 B	100 B	1 T	10 T	100 T	1000 T
400 M	70	175	430	1057	2593	6357	15585	38205	93651
1 B	42	105	260	641	1572	3857	9456	23180	56821
10 B	11	29	73	181	447	1098	2693	6603	16187
67 B	1	9	25	63	157	388	953	2339	5736
175 B	1	4	14	37	93	229	564	1385	3398
280 B	1	1	10	28	71	177	436	1072	2629
520 B	1	1	7	20	50	126	311	764	1875
1 T	1	1	4	13	35	88	217	535	1312
10 T	1	1	1	1	9	24	61	151	373

Table 6: **The optimal model parameters N and epochs e allocation under different unique tokens U_D**, predicted by the fitted data-constrained loss function 9, answering question 2.

Unique Tokens	Parameters	Epochs	FLOPs
10 M	41 M	247	6.07e+17
100 M	95 M	356	2.04e+19
1 B	222 M	569	7.59e+20
10 B	518 M	910	2.83e+22
100 B	1.6 B	1151	1.11e+24
1 T	3.7 B	1842	4.12e+25
10 T	8.7 B	2947	1.54e+27
100 T	20.2 B	4715	5.72e+28
1000 T	47.1 B	7543	2.13e+30

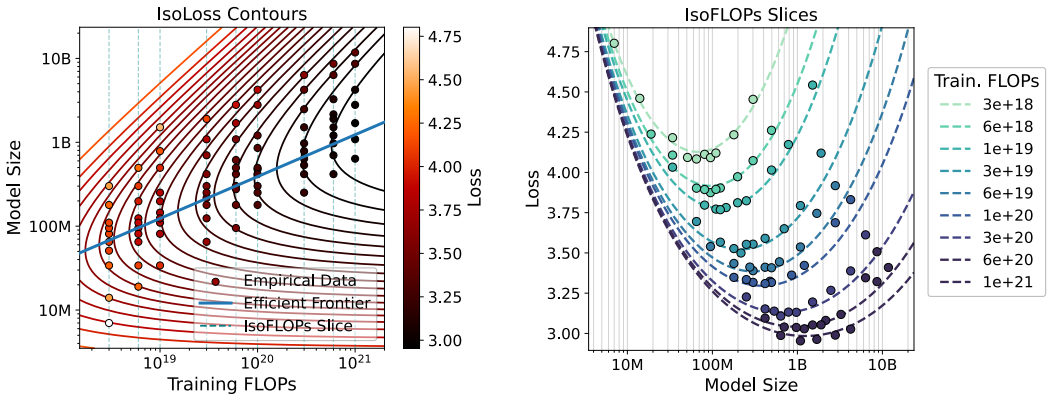

Figure 23: **Parametric fit of the loss function $L(N, D)$.** Left: Iso-loss contours of our fitted model. The blue line indicates the efficient frontier—the trajectory of minimal compute (FLOPs) required to achieve a given loss value, which is linear in log-log space. Right: Several isoFLOPs cross-sections of the loss surface, corresponding to the dashed lines in the left panel. The real data points are also plotted for a comparison.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly summarize the compute-constrained and data-constrained scaling laws, as well as the design choices, which are directly supported by the empirical results detailed in Sections 2, 3, and 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations and caveats are explicitly discussed in Section D (Discussions), specifically acknowledging the lack of strict theoretical justification for the validation loss formula and empirical gaps in the contour fitting.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper is fundamentally empirical, focusing on deriving and fitting parametric scaling laws to large-scale experimental data rather than providing formal theoretical proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section E (Implementation Details) explicitly details the codebase, datasets, validation splits, model architectures, and training hyperparameters used to generate the experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The proprietary code and model checkpoints are not fully open-sourced with this submission; however, we heavily detail the implementation (using standard datasets like c4-en and Nemotron-CC) to ensure reproducibility.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Yes, Section E (Implementation Details) explicitly specifies the datasets, splits (c4-val), batch sizes scaling, initialization metrics, and learning rate schedules.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We evaluate models at scale across 23,145 independent pre-training runs, making explicit error bars or confidence intervals for each run computationally prohibitive. However, the smooth parametric fits and aggregated data points themselves demonstrate the overall statistical stability of the scaling laws.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper provides exact model scales and token budgets, explicitly outlining the precise FLOP requirements for the models (e.g., Tables 2, 4, 5), which universally quantify the compute resources required.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research fully conforms to the NeurIPS Code of Ethics, utilizing widely accepted text datasets and focusing solely on understanding empirical scaling behaviors and training methodologies.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: As a foundational study focused on mathematical scaling laws and the optimization behaviors of diffusion language models, the direct societal applications and consequences are limited, thus a dedicated broader impacts section was omitted.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release high-risk, fully trained consumer-facing models or novel datasets that require strict safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

Justification: We properly cite the datasets (Nemotron-CC, c4-en) and architectural codebases (Megatron-LM) utilized throughout our empirical evaluations.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [N/A]

Justification: The paper focuses on empirical laws and framework configurations rather than releasing new distinct datasets or standalone assets.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The research does not involve any crowdsourcing or human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The research does not involve human subjects, making IRB approval non-applicable.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: While the paper is *about* scaling language models, it does not use external LLMs as a functional or non-standard component within its experimental methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.