Gemstones: A Model Suite for Multi-Faceted Scaling Laws

Sean McLeish [1] **John Kirchenbauer** [1] **David Yu Miller** [1] **Siddharth Singh** [1] **Abhinav Bhatele** [1]
Micah Goldblum [2] **Ashwinee Panda** [1] **Tom Goldstein** [1]

Abstract

Scaling laws are typically fit using a family of models with a narrow range of frozen hyperparameter choices. In this work we study scaling laws using a wide range of architecture and hyperparameter choices, and highlight their impact on resulting prescriptions. As a primary artifact of our research, we release the ***Gemstones***: the most comprehensive open-source scaling law dataset to date, consisting of over 4000 checkpoints from transformers with up to 2 billion parameters; these models have been trained with different learning rates, cooldown schedules, and architectural shapes. Our checkpoints enable more complex studies of scaling, such as a law that predicts language modeling performance as a function of model width and depth. By examining the various facets of our model suite, we find that the prescriptions of scaling laws can be highly sensitive to the experimental design process and the specific model checkpoints used during fitting.
Code: github.com/mcleish7/gemstone-scaling-laws

1. Introduction

Existing works on scaling laws often restrict Transformer architectures to a small range of width-depth ratios (Porian et al., 2024), train on a small number of tokens, and fix training hyperparameters such as cooldown schedule across training runs (Hoffmann et al., 2022). These design choices, in turn, can dramatically influence the resulting scaling laws. If a scaling law is sensitive to such design choices, then it may only be useful for practitioners implementing similar setups to those that produced the scaling law. In practice, practitioners often take guidance from scaling laws that assume completely different design choices than their own implementation, often without understanding to degree to which these choices may impact optimal scaling.

In this work, we produce a vast array of model checkpoints for studying how model design and model selection impact scaling laws. Our models, called the *Gemstones* because they are loosely based on scaled-down variants of the Gemma architecture, vary in their parameter count, width/depth ratio, training tokens, learning rates, and cooldown schedules. By fitting scaling laws to these checkpoints, we confirm that scaling law parameters and interpretations indeed depend strongly on the selection of models and fitting procedure used, and we quantify the degree to which these decisions impact predictions. By exploiting the variation among our model checkpoints, we also fit a number of unique scaling laws and analyze their predictions to discern whether they are consistent with design choices we see in industry models. Our contributions are summarized as follows:

- **We open-source more than 4000 checkpoints cumulatively trained on over 10 trillion tokens.** The models we provide are diverse across architectural and training hyperparameter axes, enabling more granular studies than previous work (see Figure 2).
- **We highlight the fragility and common pitfalls of prior scaling laws.** There are many decisions to make when choosing points to fit scaling laws that significantly change the slope of the law (see Figure 6).
- **We modify standard scaling laws to account for the width and depth of models.** Our scaling laws estimate not only parameter count, but also the optimal aspect ratio for any given training budget. We compare our estimates to industrial models (see Figure 5).
- **We modify scaling laws to account for practical considerations like GPU efficiency and overtraining.** When distorted to account for training and inference costs, scaling laws predict considerably wider and shallower models (see Figure 7).
- **We consider what happens when scaling laws are inaccurate.** Given that scaling laws are quite sensitive to design decisions, it is natural to ask how best to cope with their imperfections. We find that it is much better to err on the side of wider models than narrow ones. We also find that it is better to err on the side of smaller models, as overtraining is quite efficient (see Figure 8).

[1]University of Maryland [2]Columbia University. Correspondence to: Sean McLeish <smcleish@umd.edu>.

2. Related Work

2.1. Scaling Laws

Scaling laws address the trade-off between parameter count and number of training tokens, attempting to find the minimum loss possible for a language model with a constrained FLOP budget. Unfortunately, scaling laws treat model design and training as if it has a single dimension (parameter count). In reality, training is sensitive to many choices. Notably, Hoffmann et al. (2022) find significantly different fitted laws (Equation (1)) compared to Kaplan et al. (2020). Pearce & Song (2024) and Porian et al. (2024) attribute most of this discrepancy to the choice to exclude embedding parameters from the parameter count, both showing one law can be transformed into the other via controlled changes. Kaplan et al. (2020) justify including embedding parameters by showing that non-embedding parameters have a cleaner relationship with test loss. Scaling laws are also commonly included in many large model releases (Hu et al., 2024; Bi et al., 2024; Dubey et al., 2024).

Choshen et al. (2024) collect both loss and benchmark performance metrics for a multitude of models and offer a practitioner's guide to fitting scaling laws. Most notably, they suggest 5 models are ample to fit a scaling law, and the early period of training should be excluded from the analysis. In contrast, Li et al. (2024b) show that selecting data according to different tokens-per-parameter ratios and using small grid searches when fitting scaling laws can cause big swings in outcomes. Hägele et al. (2024) suggest that a constant learning rate plus cooldown is preferable to a cosine learning rate schedule. The authors also find that stochastic weight averaging should be encouraged in scaling law analysis as it tends to lead to better models. Furthermore, Inbar & Sernau (2024) observe that FLOPs cannot be used to predict wall-clock time nor memory movement, and suggest that fast-training architectures may be preferred over those prescribed by scaling laws.

There are multiple works analyzing whether scaling laws can be used to predict downstream performance. Ruan et al. (2024) show that scaling laws can be predictive of benchmark performance. Caballero et al. (2023) suggest broken scaling laws that predict performance of both downstream and upstream tasks. Works in this vein are myriad; see our extended literature review in Appendix B.

2.2. The Role of Model Shape

Levine et al. (2020) find that, for large models, optimal depth grows logarithmically with width. Henighan et al. (2020) find there is an optimal aspect ratio for each modality they study which gives maximum performance, for example they find 5 to be optimal for math models. Petty et al. (2024) claim small (<400M) transformers have diminishing benefits from depth. Brown et al. (2022) show that in some cases shallower models can beat their parameter equivalent deep models on tasks for encoder-decoder transformer architectures. This differs from Kaplan et al. (2020) who suggest aspect ratio is not a determining factor for final loss.

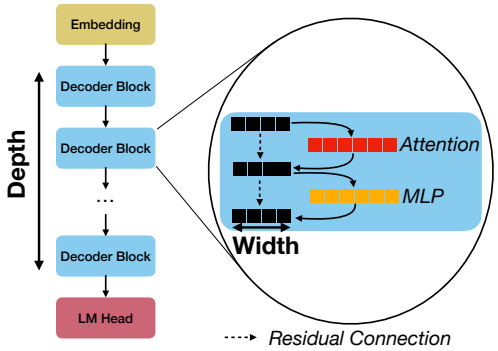

Figure 1. **The meaning of width and depth.** We visualize a standard transformer architecture, highlighting the "width" as the size of the hidden dimension and the "depth" as the number of transformer blocks. These are the quantities we're interested in providing a prescription for with our new scaling laws.

Most similarly to our work, Alabdulmohsin et al. (2024) study the impact of width and depth for encoder-decoder vision transformers, using their laws to create a smaller transformer model which has competitive downstream performance when compared with much larger models. The architecture found in this study has since been used by Beyer et al. (2024) in PaliGemma.

Model shape has also been analyzed for sparse mixture of expert models and in the context of finetuning. Krajewski et al. (2024) use "granularity" to allow their law for mixture of expert models to predict the width of the experts. Tay et al. (2022) show that downstream performance strongly depends on shape when finetuning but pretraining perplexity does not.

2.3. Zero-shot Hyperparameter Transfer

The ability to train a series of models with extremely different parameter counts is an implicit requirement of any scaling law analysis. Work on zero-shot hyperparameter transfer across transformer model *widths* is mature (Yang et al., 2021; Everett et al., 2024; Hayou & Yang, 2023; Dey et al., 2024), but achieving transfer across diverse model *depths* is less well studied, especially in transformer language models (Bordelon et al., 2024).

3. Designing Our Scaling Laws

We detail the design of our scaling laws, including model selection, the choice of learning rate, and curve fitting schemes

in the subsequent sections.

Architecture. To reduce the search space of all possible models we add some constraints, each of which are either based on precedent from a popular model series like Gemma (Team et al., 2024a;b), Llama (Touvron et al., 2023), Pythia (Biderman et al., 2023), or practical considerations such as hardware details.

All models have a head size of 128 because 256 is the maximum head dimension supported by the AMD implementation of Flash Attention 2 we utilize and we constrain our search to models with > 1 attention heads. We assume the simple convention of the Llama series where the head dimension is always the embedding dimension divided by the number of heads, implying that the embedding dimension (width) must be divisible by 128. Following conventions from the Gemma suite, we constrain the head count to be even to enable Grouped Query Atten
2023) with a query to key ratio of 2 : 1
mediate size to be $4\times$ the width of the m
vocabulary size to match the $50,304$ tok
kenizer. While many of the architecture
from Gemma, for simplicity we do not u
nor do we tie the embedding and langu
weight matrices.

Within these constraints, we searc
sible models within target parame
$50M, 100M, 500M, 1B$ and $2B$ with
At smaller scales we train up to 5 mod
and depths. At large parameter counts
models, aiming for one "standard" asp
existing models), one "wide" model, an
We visualize the models we choose to tr
laid with a selection of existing models
the Appendix we plot the entire discret
models under our constraints (Figure 2
models range from $50M$ to $2B$ paran
widths from 256 to 3072 and 18 depths from 3 to 80.

Polishing the Gemstones. For the main set of training runs, we train each model for $350B$ tokens of Dolma (Soldaini et al., 2024) data with a context length of 2048 and a world batch size of 2048 sequences. We use a linear learning rate warm up over 80 million tokens, and then train at a constant learning rate, which we adjust for model size as described in Section 3.1.

In service of future research based on our model suite, we open source checkpoints for all models at 2 billion token intervals, amounting to over $4,000$ checkpoints in total. We also open source the fitting code and logged metrics for all runs.

We also perform ablations over both cooldown and learning rate. For the cooldown ablation, we take the checkpoints saved every 10 billion tokens for the the first 100 billion tokens of training and cool these down creating a second set of models which have had their learning rate annealed to 0 linearly. Specifically, we cool each model down by training for a further 10% of the total tokens which it has seen during training, i.e. our cooled down set of models have training budgets ranging from 11 to 110 billion tokens.

Training Details We train with AdamW (Loshchilov & Hutter, 2017) with β parameters 0.9 and 0.95 and a weight decay of 0.1. We do not apply weight decay to the bias or normalization parameters. All models are trained with tensor parallelism (Singh & Bhatele, 2022; Singh et al., 2024) over multiple nodes of AMD MI250X GPUs. To the best of our knowledge, this makes the Gemstone suite of models the largest collection trained on AMD GPUs.

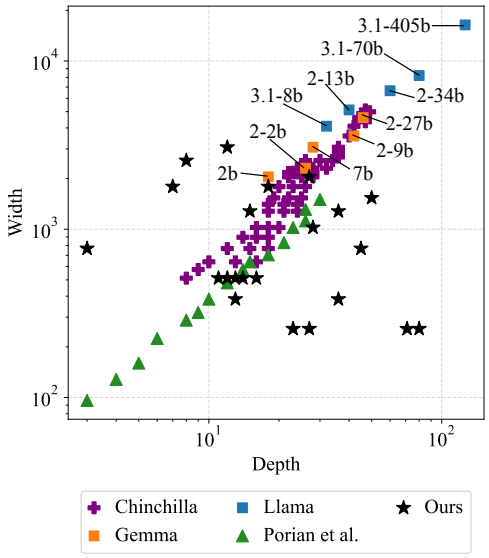

Figure 2. **Distribution of prior scaling law models, industry models, and our models in terms of width and depth.** Prior work (purple and green) and industry models (blue and orange) mostly lie on a fixed width-depth line. If we want to prescribe the optimal width-depth ratio, we need to select models with different widths and depths (our models, black).

3.1. Optimal Learning Rates for Gemstones

Training models across diverse architectures and scales requires learning rates that ensure both stability and near-optimal performance. Suboptimal learning rates risk misrepresenting scaling laws, as they could conflate architectural preferences with hyperparameter sensitivity. For the Gemstone models—varying in width, depth, and size—we address this challenge through a unified learning rate scal-

ing rule and a parameter initialization scheme tailored for stability.

Unified Learning Rate Scaling Rule Existing scaling rules prescribe learning rates (lr) as $lr_{\text{base}}/\text{width}$ for width scaling or $lr_{\text{base}}/\sqrt{\text{depth}}$ for depth scaling. Since Gemstone models vary both dimensions, we propose a hybrid rule: $lr_{\text{eff}} = lr_{\text{base}}/(\text{width} \times \sqrt{\text{depth}})$ This accounts for the compounding effect of gradient dynamics across width and depth, balancing update magnitudes during optimization.

Empirical Validation To validate lr_{base}, we stress-test four extreme model shapes: wide (64 layers, 768 width) and deep (128 layers, 512 width) at 100M and 2B parameter scales. Each is trained for 2B tokens with lr_{eff} swept from 10^{-4} to 5×10^{-2}. As shown in Figure 3 (left), optimal lr_{eff} varies widely across architectures. However, rescaling the x-axis by width $\times \sqrt{\text{depth}}$ collapses all curves onto a shared trend, revealing $lr_{\text{base}} = 5$ as the consistent optimum (right panel). This confirms our rule's efficacy for width-depth transfer.

Flaws in the Gemstones. While $lr_{\text{base}} = 5$ achieves stable training for most models under the scheme described above, wider architectures (e.g., 256 width-depth ratio) occasionally exhibit loss spikes nonetheless. Despite these instabilities, via rollbacks and minor modifications to the learning rates for the most extreme models, all models in the suite are trained to 350B tokens without divergence. We discuss these issues and our solutions further in Appendix F.2.

Ablation Study To assess sensitivity to lr_{base}, we replicate training for a subset of models with $lr_{\text{base}} = 2.5$ (e.g. dividing lr_{eff} by 2). While losses are marginally higher, scaling law fits remain robust, suggesting our conclusions are not artifacts of aggressive learning rates.

Scalable Parameter Initialization Rules. Finally, stable training across model shapes and scales also requires model specific tweaks to parameter initialization (Yang et al., 2021). Following OLMo(1) (Groeneveld et al., 2024), we apply a parameter initialization strategy intended to enable stable training and learning rate transfer across scales. We initialize all parameters as truncated normal ($\mu = 0, a = -3 \cdot \sigma, b = 3 \cdot \sigma$) with modified variances dependent on the parameter type. We use $\sigma = 1/\sqrt{\text{width}}$ except for the attention projections which are initialized as $\sigma = 1/\sqrt{2 \cdot \text{width} \cdot (l+1)}$ and the MLP projections as $\sigma = 1/\sqrt{2 \cdot (4 \times \text{width}) \cdot (l+1)}$ where in each case l is the layer index (not the total model depth) and the $4\times$ factor comes from the relation of width to MLP intermediate dimension.

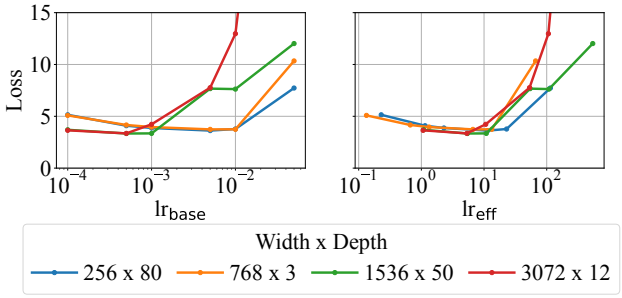

Figure 3. **Learning rate scaling is necessary for width-depth transfer.** Left: Preliminary training runs with initialization rules active, but no learning rate scaling. Right: Same data, but with x-axis rescaled to simulate the application of learning rate scaling with $lr_{\text{base}} = lr_{\text{eff}} \times (\text{width} \times \sqrt{\text{depth}})$.

3.2. Fitting Scaling Laws

We fit scaling laws using methods similar to approach 1 and 3 from Chinchilla (Hoffmann et al., 2022). We fit all laws using the log perplexity of all trained models on a sample of 100 million tokens from a fixed, held-out validation set from the training distribution. We also collect log perplexity values for a range of open source models (Team et al., 2024a;b; Touvron et al., 2023; Dubey et al., 2024; Yang et al., 2024a;b) on the same validation data to allow for a comparison between our predictions and a selection of widely used models. We design a specialized FLOP counting function as we find that simple rules of thumb (e.g., FLOPs= $6 \times parameters$ (Hoffmann et al., 2022)) do not accurately account for differences in FLOPs between extremely wide and narrow architectures. We discuss this further and present our function in Appendix G.

Following prior work, we plot the Epoch AI Replication (Besiroglu et al., 2024) of Chinchilla (Hoffmann et al., 2022) on all plots and use the coefficients for Kaplan plotted by Porian et al. (2024) which were extracted from the original paper (Kaplan et al., 2020).

A More Robust Approach to Fitting Compute-Optimal Laws. The first approach in Hoffmann et al. (2022) fits a scaling law by plotting the loss against FLOPs for a range of architectures, and then fitting a line to the pareto-optimal architecture for each FLOP count (see Figure 4). Following Hoffmann et al. (2022), we refer to this as "Approach 1". As we use a constant learning rate, we can use all recorded validation losses to fit our law. Hoffmann et al. (2022) and Kaplan et al. (2020) select model shapes so densely that they have a near-optimal architecture at each FLOP count. This works when all architectures lie in a 1D space (parameterized by parameter count), as each model is optimal in some FLOP regime, and the lower envelope is densely populated. In our two dimensional exploration (varying width and depth), some models are never optimal, and the ones that

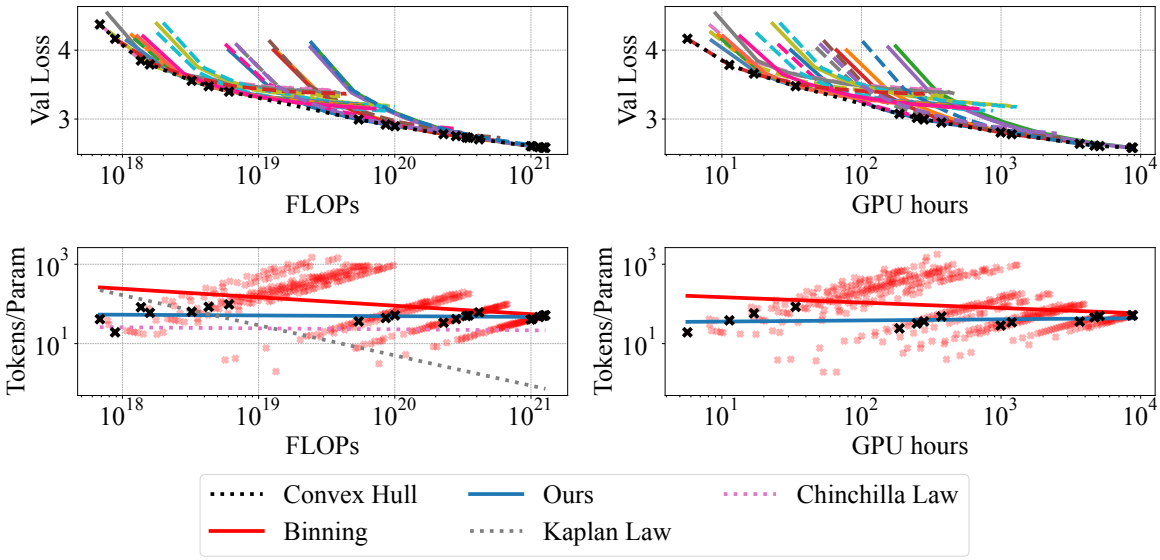

Figure 4. **Approach 1 prescriptions.** Row one: Validation loss over FLOPs (left) and GPU hours (right). We use Approach 1 to find the optimal points on the convex hull in each setting, marked with **black crosses**. Row two: We fit a line to the tokens per parameter of empirically optimal models and find a slightly higher, but still constant, tokens per parameter prescription than Hoffmann et al. (2022). Hoffmann et al. (2022)'s Approach 1 creates 250 logarithmically-spaced FLOPs bins per order of magnitude, and in **red** we plot the minimizers over these bins, and the scaling law fitted to these minimizers (binning). Clearly, their Approach 1 is not well-suited for our data, and our convex hull approach is better when we select fewer models to fit our law on. Extended plot in Figure 9.

are do not densely populate the envelope. We therefore need to propose a new fitting method for less data-dense regimes.

Our New Method: The Convex Hull. We fit a lower *convex hull* to our loss curves. This hull is only supported by a sparse set of optimal models. This naturally excludes sub-optimal models that lie above the convex hull of optimality, and as we will show, this makes the resulting scaling law far more robust to model selection choices.

Why We Skip Approach 2. Another method to fit scaling laws is to put model runs into isoFLOP bins and choose the best parameter count in each bin. Hoffmann et al. (2022) call this "Approach 2". Our 2-dimensional set of models do not finely cluster into isoFLOP bins, meaning our data is not easily amenable to Approach 2, hence we exclude this approach from our analysis. Hu et al. (2024) also eschew this approach.

Prescribing Optimal Widths and Depths by Fitting Power Laws. The final approach described by Hoffmann et al. (2022) is to fit a parametric formula to the loss values with the ansatz

$$L(p, T) = \frac{A}{p^{\alpha}} + \frac{B}{T^{\beta}} + \varepsilon \tag{1}$$

where p is parameter count and T is tokens. We fit our models using L-BGFS (Liu & Nocedal, 1989) with a Huber loss ($\delta = 10^{-4}$) between the empirical log loss and the model prediction, and use multiple initializations following Besiroglu et al. (2024). We ablate to check that our fitting procedure is robust to the size of the grid of initializations and the choice of delta in Appendix D.5.

Our broad selection of model architectures enables us to study scaling laws that predict loss as a function of not only parameter count, but also model aspect ratio. For this purpose, we consider a perturbation of the standard scaling law with additional terms to account for the impact of model width and depth.

$$L(w, d, p, T) = \frac{A}{w^{\alpha}} + \frac{B}{d^{\beta}} + \frac{C}{p^{\gamma}} + \frac{D}{T^{\zeta}} + \varepsilon. \tag{2}$$

Here, w is the hidden dimension of the model, d is the number of layers. All of $A, \alpha, B, \beta, C, \gamma, D, \zeta$ and ε are optimized to fit our scaling runs. We choose this form as it predicts the same behaviors as the standard law if width and depth are not implicated in any systematic trends. We ablate another possible form of this law in Appendix D.6.

4. Experiments

In Section 4.1 we use our new convex hull fitting method to make a scaling law for the compute-optimal tokens-to-parameters ratio, and our new power law approach to provide a prescription for the compute-optimal width-to-depth ratio. We show how many seemingly innocuous design choices such as the learning rate schedule can significantly

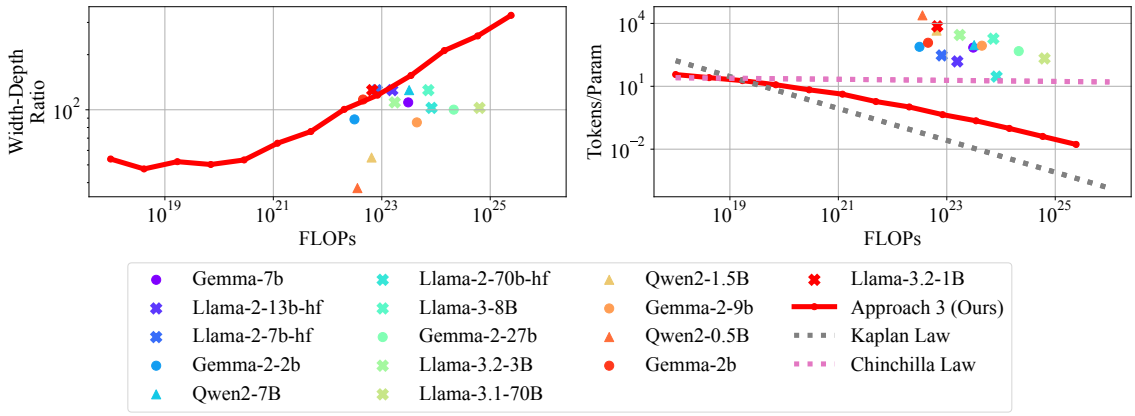

Figure 5. **Approach 3 laws with the parametrization shown in Equation (2).** We see the prescribed optimal width-depth ratio increases with the FLOPs (left) budget and the optimal tokens per parameter decreases as the FLOPs budget increases (right). We see slight bumpiness in the lines due to the integer constraints we enforce on the attention heads, we also plot with this constraint removed in Figure 16.

impact these prescribed scaling laws in Section 4.2. Finally, we reflect on the benefits of training compute-optimal and time-optimal models in Section 4.3.

4.1. Sizing Up Our Scaling Laws Against Prior Laws and Industry Models

Approach 1. In Figure 4 (row one), we see our validation losses plotted as both a function of FLOPs (left) and GPU hours (right) for the first 100 billion tokens of training. We calculate GPU hours from the average recorded optimizer step time for each model.

Our convex hull fits the data better than prior approaches. Hoffmann et al. (2022)'s Approach 1 creates 250 logarithmically-spaced FLOPs bins per order of magnitude and then uses the models that achieve the best loss in each FLOPs bin to fit the scaling law (a line). However, for our data, their approach does not work very well because it includes many points that are strictly suboptimal with respect to the minimal loss envelope. Our convex hull method omits these points, and fits the line with far fewer points.

In Figure 4 (row two), we see the prescription of the fitted laws for tokens per parameter. We see that the tokens per parameter prescription of our Approach 1 fitting is also close to constant, like Hoffmann et al. (2022), but slightly higher, suggesting more tokens should be used per parameter in the model. We extend this plot showing predicted total parameters, tokens, and over multiple ablations in Appendix C. We give a more detailed plot of each model's individual validation loss in Appendix F.

Approach 3. This uses a parametric function to predict the loss when given parameter count and number of tokens. As our data is intentionally designed to cover a wide variety of widths and depths, we can also predict the optimal aspect ratio of a model for a given FLOP budget. We do this by optimizing over the law shown in Equation (2) with four terms so we can explicitly optimize the width and depth.

We find that convex optimization struggles to find optimal points due to the integer constraint we impose on the number of attention heads in the model. Instead, we use a brute force approach by selecting approximately 10^8 model shapes and tokens counts, then use the law to predict the loss for each model, and choose the minimal loss among those in each FLOP range. In Figure 5 we plot the output of this process as a solid line labeled "Approach 3 (Ours)". In the left of the figure, we see that the prescribed width-depth ratio increases with the FLOP budget, but that width-depth ratio increases relatively slowly even as FLOPs is increased by many orders of magnitude (corroborating certain observations in Levine et al. (2020)). We also plot a selection of widely used models trained by industry estimating their cumulative FLOP expenditures (x-axis values) using FLOPs/token = $6 \times parameters$ to calculate FLOPs with their published training token counts. In Figure 5 (right), we see that our optimal tokens per parameter ratio tracks the prescription of Kaplan et al. (2020) more closely than Hoffmann et al. (2022); the prescribed tokens per parameter decreases as the number of available FLOPs increases.

4.2. A Rainbow of Scaling Laws

To demonstrate the sensitivity of scaling laws to design choices, we fit laws with various assumptions and model selection rules. We begin by fitting laws of the classical form (without width-depth), and use equation 4 from Hoffmann et al. (2022) to provide compute-optimal parameter count prescriptions. In Figure 6 we show the optimal predictions

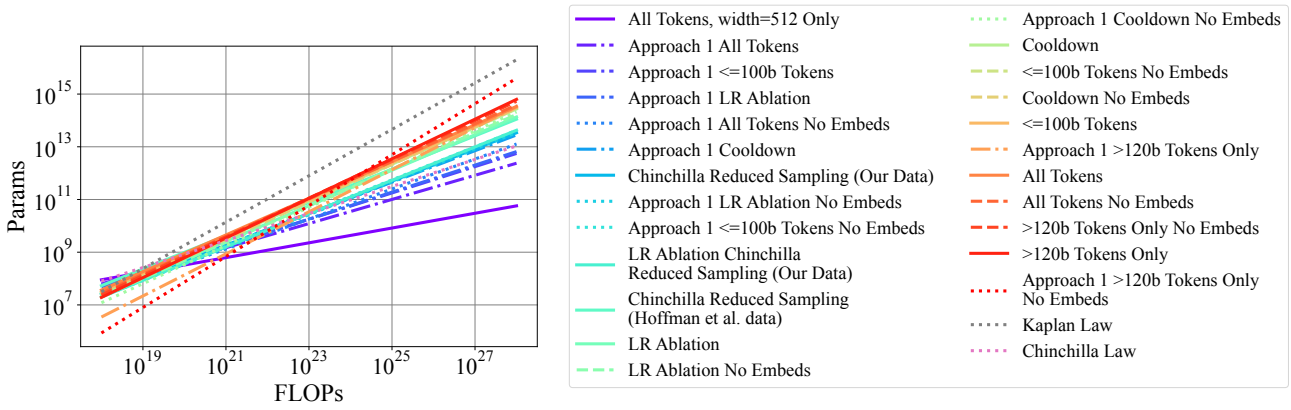

Figure 6. **We demonstrate the variability in fitting scaling laws by resampling our data many different ways.** We label prescriptions found using Approach 1 with "Approach 1" in the legend, otherwise approach 3 is used. All tokens counts available are used to fit the laws unless stated otherwise in the legend, for example $\leq 100B$ means that only token counts less than or equal to $100B$ are used in fitting.
No Embeds: Embedding parameters are not counted when fitting these laws.
Cooldown: Only data from the cooldown ablation is used to fit this law.
LR Ablation: Only data from the learning rate ablation training runs, where the learning rate is halved, is used to fit these laws.
width=512 *Only*: Only models with width 512 are used to fit these laws.
Chinchilla Reduced Sampling: We subsample our data to be as close as possible to the token counts and model sizes that Hoffmann et al. (2022) use to fit their scaling laws and also fit new scaling laws on this subset of Hoffmann et al. (2022) data. Details in Section 4.2.

of multiple possible laws fitted on different data subsets, visualizing the amount of variation possible under an array of configurations.

One particular dimension of variability we wish to highlight briefly here is the interplay between model selection and the derived law. To do this, we select 5 models from Gemstones that have an analogous model in Hoffmann et al. (2022) (using data extracted by Besiroglu et al. (2024)) with similar parameter count and aspect ratio. We select Gemstones checkpoints with token counts nearly matching the Hoffman points. We call this "Chinchilla Reduced Sampling." We then fit scaling laws to both of these sub-sampled datasets. We find that fitting Hoffmann's data using this reduced sampling results in an increased slope relative to fitting on all data. Meanwhile this subsampling reduces the slope of the line fit on Gemstones. This highlights that scaling law fitting can be quite sensitive to seemingly innocuous changes in model selection for both the Gemstones and the simpler model family selected by Hoffman. Notably, there are 5 models in this subset for both Hoffmann et al. (2022) and our data, this meets the rule of thumb given by Choshen et al. (2024) for the minimum number of models should be used to fit a scaling law.

We also perform many other ablations in Figure 6, and we summarize these here. First, we fit laws both including and excluding embedding parameter count, which both Pearce & Song (2024) and Porian et al. (2024) find to be a primary explanation of the discrepancies between the prescriptions found by Kaplan et al. (2020) and Hoffmann et al. (2022). We also show the impact of fitting on our cooldown and learning rate ablation datasets in turn, seeing that both choices have a noticeable impact on the prescription for optimal parameter count. Next, we remove checkpoints from our data to simulate having only trained for 100 billion tokens or only having data for token counts greater than 120 billion. Finally, we also fit to only models with a width of 512 to isolate the role of depth. Fitting on only these models creates a drastic difference in the law, significantly reducing the predicted optimal rate of growth of parameters as a function of FLOPs, highlighting the need for including diverse model shapes in the fitting data.

4.3. The Price of Stepping Off the Scaling Law

In the previous sections we provide prescriptions for the optimal width, depth, parameters, and tokens as a function of total training FLOPs. As we have seen, industry models don't always comply with our prescriptions. At the same time, the variability in our analyses suggests that scaling laws are inherently fuzzy, providing only rough estimates of optimal settings. With this in mind, we ask a natural question: if one happens to choose a *sub-optimal* point in design space, how bad can it really be?

By analyzing the cost of stepping off of the scaling law, we find that some kinds of design errors are more damaging than others. In particular, one pays a steep price for training models that are too narrow, and it is better to err on the side of too wide. We also find that training on more tokens than is strictly recommended (aka "overtraining") is typically

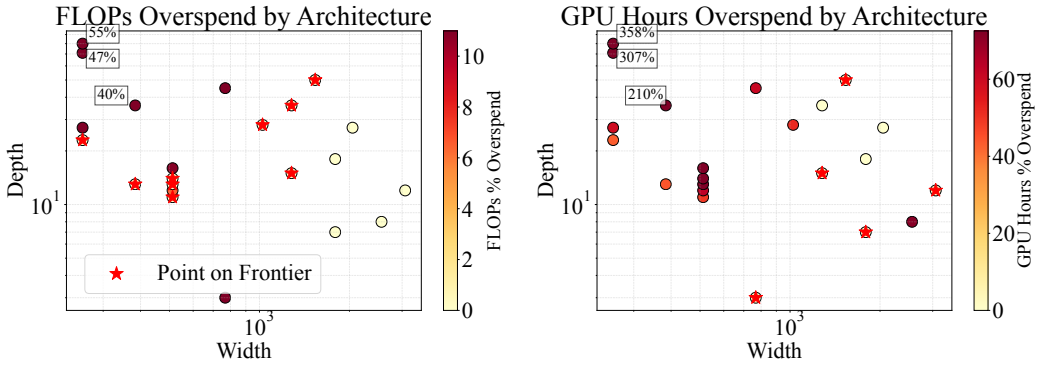

Figure 7. **The inefficiency of training models with suboptimal widths and depths.** We plot the FLOPs (left) and GPU Hours (right) *overspend* after training our Gemstones for 300 billion tokens. We define the overspend as how many resources (FLOPs or GPU hours) are required for a model with a given width-depth configuration to reach some target loss, relative to the models that achieve that target loss the fastest (the "points on (pareto)-frontier"). We can see that the skinny models (top-left, dark points) use many more FLOPs or GPU hours to reach a target loss than the wide models. We note that these inefficiencies exist in our training setup because we only use tensor parallelism and not pipeline parallelism.

quite efficient in terms of pushing down loss.

If You Value Your Time, Train Wide Models. We first show that in our training setup, training wider models is far more efficient than training deep models. In Figure 7, we reflect on the consequences of suboptimal architectural choices, by considering how much of a given resource—FLOPs or GPU hours—would be "overspent" to reach any target loss value with the plotted architecture rather than the prescribed width and depth. We find that choosing to train "skinny" models (top left) wastes many FLOPs and GPU hours. The scale of overspend is quite different however, with the least efficient models only overspending about 50% on FLOPs but wasting more than 200% of the GPU hours spent by the best configuration. In other words, in the time taken to train a single (very) suboptimal model to the desired loss value, one could train three optimal-width-depth models. We note that while the time-optimal models tend to be the wider ones, this is probably due to our training scheme. Similar to other open-source efforts such as OLMo et al. (2024), we do not make any use of pipeline parallelism, and only employ tensor parallelism (using a hybrid data and tensor parallel algorithm similar to the ubiquitous Fully Sharded Data Parallel strategy). In summary, for standard parallelism implementations, wider models are simply easier to scale, but as a result our observations regarding resource overspending may not generalize to other parallelism strategies.

Scaling Laws Predict That Overtraining Is Efficient.

In Figure 5, we see the optimal predictions from our law of the form shown in Equation (2). We can shift those optimal points to simulate overtraining. To do this, we fix a FLOP budget and trace out a path of model sizes and corresponding token counts to remain within that budget. For each model size and token count, we record the "overtraining factor," which is the selected number of training tokens divided

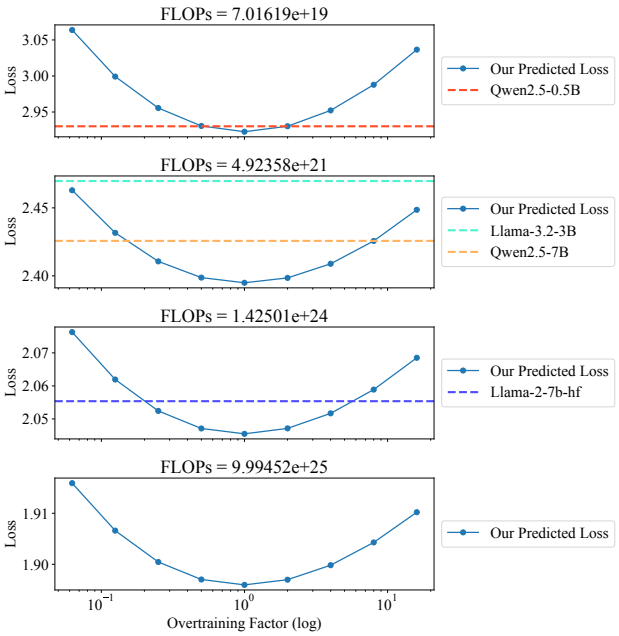

Figure 8. **Quantifying the cost of overtraining.** We simulate deviations from our prescriptions to assess their impact on model performance by increasing the optimal token count prescribed by Equation (2) by an overtraining factor. We then optimize the model shape to achieve the lowest loss possible at each FLOP budget and overtraining factor. Note that 10^0, or $1\times$, is the prescribed optimal point. We take four FLOP budgets (title of each plot) and plot the loss as a function of overtraining factor and see that under or overtraining increases predicted loss but by only a small amount. We plot the losses of selected open source models on our validation set to help ground the y-axis ranges.

by the optimal number of tokens for that model shape. An overtraining factor of less than one corresponds to undertraining the model, and a factor greater than one represents overtraining. We show the results of this process in Figure 8. We see that overtraining does increase predicted loss at a given FLOP count but that these curves are actually quite flat. We include the loss values of open source models on our own validation set to allow readers to contextualize the y-axis values. Especially at high FLOP counts, overtraining becomes quite efficient in that it results in fairly small elevations in loss for a relatively large reduction in model size.

Industry models often use fewer parameters and train on more tokens than prescribed in prior work (see Figure 5). We find the impact of overtraining a smaller model on predicted loss to be small. Combining this with Figure 7, where wider models are predicted to be optimal in terms of GPU hours, reinforces the message that FLOPs optimality is not the end of the story for training models. Trading some FLOPs optimality for time optimality necessarily means over-/under-training, but Figure 8 suggests the difference is marginal. We believe this combined evidence makes significant progress towards explaining the differences between the prescriptions found in prior work and training choices observed in the wild out in industry.

5. Limitations and Conclusions

We hope this work encourages a rich range of future work on the impact of width and depth within modern transformer architectures using the large amount of open source artifacts we produce. Future work should also extend to other hyperparameters involved in training transformer architectures, such as the expansion factor. Although we endeavor to make our laws as generalizable as possible, we still expect that their applicability declines in training set-ups very different from our own.

Acknowledgements

We thank (in alphabetical order): Brian Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Sachin Shah and Abhimanyu Hans for helpful feedback.

An award for computer time was provided by the U.S. Department of Energy's (DOE) Innovative and Novel Computational Impact on Theory and Experiment (INCITE) Program. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

This work was made possible by the ONR MURI program, DAPRA TIAMAT, the National Science Foundation (IIS-2212182), and the NSF TRAILS Institute (2229885). Commercial support was provided by Capital One Bank, the Amazon Research Award program, and Open Philanthropy.

Impact Statement

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

References

Aghajanyan, A., Yu, L., Conneau, A., Hsu, W.-N., Hambardzumyan, K., Zhang, S., Roller, S., Goyal, N., Levy, O., and Zettlemoyer, L. Scaling laws for generative mixed-modal language models. In *International Conference on Machine Learning*, pp. 265–279. PMLR, 2023.

AI, L. Litgpt. https://github.com/Lightning-AI/litgpt, 2023.

Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

Alabdulmohsin, I. M., Zhai, X., Kolesnikov, A., and Beyer, L. Getting vit in shape: Scaling laws for compute-optimal model design. *Advances in Neural Information Processing Systems*, 36, 2024.

Bansal, Y., Ghorbani, B., Garg, A., Zhang, B., Cherry, C., Neyshabur, B., and Firat, O. Data scaling laws in nmt: The effect of noise and architecture. In *International Conference on Machine Learning*, pp. 1466–1482. PMLR, 2022.

Besiroglu, T., Erdil, E., Barnett, M., and You, J. Chinchilla scaling: A replication attempt. *arXiv preprint arXiv:2404.10102*, 2024.

Beyer, L., Steiner, A., Pinto, A. S., Kolesnikov, A., Wang, X., Salz, D., Neumann, M., Alabdulmohsin, I., Tschannen, M., Bugliarello, E., et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

Bi, X., Chen, D., Chen, G., Chen, S., Dai, D., Deng, C., Ding, H., Dong, K., Du, Q., Fu, Z., et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Biderman, S., Schoelkopf, H., Anthony, Q. G., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., et al. Pythia: A suite for analyzing large language models across training and scaling.

In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Bordelon, B., Noci, L., Li, M. B., Hanin, B., and Pehlevan, C. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=KZJehvRKGD.

Brown, J. R., Zhao, Y., Shumailov, I., and Mullins, R. D. Wide attention is the way forward for transformers? *arXiv preprint arXiv:2210.00640*, 2022.

Caballero, E., Gupta, K., Rish, I., and Krueger, D. Broken neural scaling laws. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=sckjveqlCZ.

Choshen, L., Zhang, Y., and Andreas, J. A hitchhiker's guide to scaling law estimation, 2024. URL https://arxiv.org/abs/2410.11840.

Clark, A., de Las Casas, D., Guy, A., Mensch, A., Paganini, M., Hoffmann, J., Damoc, B., Hechtman, B., Cai, T., Borgeaud, S., et al. Unified scaling laws for routed language models. In *International conference on machine learning*, pp. 4057–4086. PMLR, 2022.

Dey, N., Anthony, Q., and Hestness, J. The practitioner's guide to the maximal update parameterization, September 2024. URL https://www.cerebras.ai/blog/the-practitioners-guide-to-the-maximal-update-parameterization.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Everett, K. E., Xiao, L., Wortsman, M., Alemi, A. A., Novak, R., Liu, P. J., Gur, I., Sohl-Dickstein, J., Kaelbling, L. P., Lee, J., and Pennington, J. Scaling exponents across parameterizations and optimizers. In Salakhutdinov, R., Kolter, Z., Heller, K., Weller, A., Oliver, N., Scarlett, J., and Berkenkamp, F. (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 12666–12700. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/everett24a.html.

Frantar, E., Riquelme, C., Houlsby, N., Alistarh, D., and Evci, U. Scaling laws for sparsely-connected foundation models. *arXiv preprint arXiv:2309.08520*, 2023.

Ghorbani, B., Firat, O., Freitag, M., Bapna, A., Krikun, M., Garcia, X., Chelba, C., and Cherry, C. Scaling laws for neural machine translation. *arXiv preprint arXiv:2109.07740*, 2021.

Gordon, M. A., Duh, K., and Kaplan, J. Data and parameter scaling laws for neural machine translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 5915–5922, 2021.

Groeneveld, D., Beltagy, I., Walsh, P., Bhagia, A., Kinney, R., Tafjord, O., Jha, A. H., Ivison, H., Magnusson, I., Wang, Y., Arora, S., Atkinson, D., Authur, R., Chandu, K., Cohan, A., Dumas, J., Elazar, Y., Gu, Y., Hessel, J., Khot, T., Merrill, W., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Pyatkin, V., Ravichander, A., Schwenk, D., Shah, S., Smith, W., Subramani, N., Wortsman, M., Dasigi, P., Lambert, N., Richardson, K., Dodge, J., Lo, K., Soldaini, L., Smith, N. A., and Hajishirzi, H. Olmo: Accelerating the science of language models. *Preprint*, 2024.

Hägele, A., Bakouch, E., Kosson, A., Allal, L. B., Von Werra, L., and Jaggi, M. Scaling laws and compute-optimal training beyond fixed training durations. *arXiv preprint arXiv:2405.18392*, 2024.

Hayou, S. and Yang, G. Width and depth limits commute in residual networks. In *International Conference on Machine Learning*, pp. 12700–12723. PMLR, 2023.

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Hernandez, D., Kaplan, J., Henighan, T., and McCandlish, S. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., et al. An empirical analysis of compute-optimal large language model training. *Advances in Neural Information Processing Systems*, 35: 30016–30030, 2022.

Hu, S., Tu, Y., Han, X., He, C., Cui, G., Long, X., Zheng, Z., Fang, Y., Huang, Y., Zhao, W., et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Inbar, I. and Sernau, L. Time matters: Scaling laws for any budget, June 2024. URL http://arxiv.org/abs/2406.18922. arXiv:2406.18922 [cs].

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Krajewski, J., Ludziejewski, J., Adamczewski, K., Pióro, M., Krutul, M., Antoniak, S., Ciebiera, K., Król, K., Odrzygóźdź, T., Sankowski, P., et al. Scaling laws for fine-grained mixture of experts. *arXiv preprint arXiv:2402.07871*, 2024.

Levine, Y., Wies, N., Sharir, O., Bata, H., and Shashua, A. The depth-width interplay in self-attention. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 22640–22651. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/ff4dfdf5904e920ce52b48c1cef97829-Paper.pdf.

Li, B., Liang, H., Meng, Z., and Zhang, W. Are bigger encoders always better in vision large models? *arXiv preprint arXiv:2408.00620*, 2024a.

Li, M., Kudugunta, S., and Zettlemoyer, L. (mis)fitting scaling laws: A survey of scaling law fitting techniques in deep learning. In *The Thirteenth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=xI71dsS3o4. https://iclr.cc/virtual/2025/poster/27795.

Liang, Z., He, H., Yang, C., and Dai, B. Scaling laws for diffusion transformers. *arXiv preprint arXiv:2410.08184*, 2024.

Liu, D. C. and Nocedal, J. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, 1989. doi: 10.1007/BF01589116.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Muennighoff, N., Rush, A., Barak, B., Le Scao, T., Tazi, N., Piktus, A., Pyysalo, S., Wolf, T., and Raffel, C. A. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36:50358–50376, 2023.

OLMo, T., Walsh, P., Soldaini, L., Groeneveld, D., Lo, K., Arora, S., Bhagia, A., Gu, Y., Huang, S., Jordan, M., et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Pearce, T. and Song, J. Reconciling kaplan and chinchilla scaling laws, 2024. URL https://arxiv.org/abs/2406.12907.

Petty, J., van Steenkiste, S., Dasgupta, I., Sha, F., Garrette, D., and Linzen, T. The impact of depth on compositional generalization in transformer language models, 2024. URL https://arxiv.org/abs/2310.19956.

Porian, T., Wortsman, M., Jitsev, J., Schmidt, L., and Carmon, Y. Resolving discrepancies in compute-optimal scaling of language models. *arXiv preprint arXiv:2406.19146*, 2024.

Ruan, Y., Maddison, C. J., and Hashimoto, T. Observational scaling laws and the predictability of language model performance, 2024. URL https://arxiv.org/abs/2405.10938.

Singh, S. and Bhatele, A. Axonn: An asynchronous, message-driven parallel framework for extreme-scale deep learning. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 606–616, 2022. doi: 10.1109/IPDPS53621.2022.00065.

Singh, S., Singhania, P., Ranjan, A. K., Sating, Z., and Bhatele, A. A 4d hybrid algorithm to scale parallel training to thousands of gpus, 2024. URL https://arxiv.org/abs/2305.13525.

Soldaini, L., Kinney, R., Bhagia, A., Schwenk, D., Atkinson, D., Authur, R., Bogin, B., Chandu, K., Dumas, J., Elazar, Y., Hofmann, V., Jha, A. H., Kumar, S., Lucy, L., Lyu, X., Lambert, N., Magnusson, I., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Ravichander, A., Richardson, K., Shen, Z., Strubell, E., Subramani, N., Tafjord, O., Walsh, P., Zettlemoyer, L., Smith, N. A., Hajishirzi, H., Beltagy, I., Groeneveld, D., Dodge, J., and Lo, K. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *arXiv preprint*, 2024.

Tay, Y., Dehghani, M., Rao, J., Fedus, W., Abnar, S., Chung, H. W., Narang, S., Yogatama, D., Vaswani, A., and Metzler, D. Scale efficiently: Insights from pretraining and finetuning transformers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=f2OYVDyfIB.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024a.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024b.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. Huggingface's transformers: State-of-the-art natural language processing, 2020. URL https://arxiv.org/abs/1910.03771.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., Dong, G., Wei, H., Lin, H., Tang, J., Wang, J., Yang, J., Tu, J., Zhang, J., Ma, J., Yang, J., Xu, J., Zhou, J., Bai, J., He, J., Lin, J., Dang, K., Lu, K., Chen, K., Yang, K., Li, M., Xue, M., Ni, N., Zhang, P., Wang, P., Peng, R., Men, R., Gao, R., Lin, R., Wang, S., Bai, S., Tan, S., Zhu, T., Li, T., Liu, T., Ge, W., Deng, X., Zhou, X., Ren, X., Zhang, X., Wei, X., Ren, X., Liu, X., Fan, Y., Yao, Y., Zhang, Y., Wan, Y., Chu, Y., Liu, Y., Cui, Z., Zhang, Z., Guo, Z., and Fan, Z. Qwen2 technical report, 2024a. URL https://arxiv.org/abs/2407.10671.

Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024b.

Yang, G., Hu, E., Babuschkin, I., Sidor, S., Liu, X., Farhi, D., Ryder, N., Pachocki, J., Chen, W., and Gao, J. Tuning large neural networks via zero-shot hyperparameter transfer. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17084–17097. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8df7c2e3c3c3be098ef7b382bd2c37ba-Paper.pdf.

Yun, L., Zhuang, Y., Fu, Y., Xing, E. P., and Zhang, H. Toward inference-optimal mixture-of-expert large language models. *arXiv preprint arXiv:2404.02852*, 2024.

Zhang, B., Ghorbani, B., Bapna, A., Cheng, Y., Garcia, X., Shen, J., and Firat, O. Examining scaling and transfer of language model architectures for machine translation. In *International Conference on Machine Learning*, pp. 26176–26192. PMLR, 2022.

Zhang, B., Liu, Z., Cherry, C., and Firat, O. When scaling meets llm finetuning: The effect of data, model and finetuning method. *arXiv preprint arXiv:2402.17193*, 2024.

A. Software and Data

We train all models using a fork of `litgpt` (AI, 2023) enhanced with AxoNN (Singh & Bhatele, 2022; Singh et al., 2024) tensor parallelism. We open source all models used in our analysis to Hugging Face (Wolf et al., 2020) and the logging from training on Weights and Biases in json format. All scaling-law fitting code is released on GitHub, with training code to be released shortly after publication.

B. Extended Related Works

Scaling laws are broadly applied to many areas of machine learning, such as machine translation. Ghorbani et al. (2021) split the parameters term into two, one for each of encoder and decoder, and similarly to Gordon et al. (2021) analyze the relationship between BLEU scores and scaling laws. Zhang et al. (2022) and Bansal et al. (2022) study the impact of architecture choice on the scaling law, finding increasing data or parameters can compensate for worse architectural decisions.

Scaling laws have also been applied to sparse architectures. Clark et al. (2022) analyze how the number of experts can be used in the law, studying both linear and quadratic interactions for many types of routing models. Yun et al. (2024) extend this, analyzing the trade offs between optimal training and optimal inference. Krajewski et al. (2024) find that with optimal settings a Mixture of Experts model always outperforms a transformer model at any computational budget. Frantar et al. (2023) focus on weight sparsity within foundation models, adding a multiplicative parameter on the parameters term in the law.

These techniques are not limited to generative text modeling only; they have also been applied to multi-model models. Henighan et al. (2020) find optimal model size can be described as a power law for model modeling including images and video. The authors also find that model size does not help 'strong generalization' for problem solving. Aghajanyan et al. (2023) analyze text, images, code and speech, presenting a scaling law to describe the competition between these modalities and describe a regime for optimal hyperparameter transfer from the unimodal to multimodal regimes. Liang et al. (2024) look at scaling laws for diffusion transformer models. Li et al. (2024a) analyze scaling laws for vision encoder commonly used to encode image inputs for transformer model backbones, finding increasing the size of the encoder alone can lead to performance degradation in some cases.

Further analyses using scaling laws have extended to analyzing finetuning and data limited scaling. Hernandez et al. (2021) find that finetuning is much more compute efficient when the pretraining ignored. Zhang et al. (2024) study parameter efficient finetuning regimes find a multiplicative law is better for the finetuning setting than the classical additive law used by others. Muennighoff et al. (2023) analyze the data constrained training regimes, finding epoching data up to four times is as good as training on deduplicated data in terms of reducing loss.

C. Ablations for Approach 1

C.1. Extended Paper Figures

In Figure 9, we plot an extended version of the Approach 1 plot we present in Figure 4.

C.2. Alternative Learning Rates

In Figure 10, we present the Approach 1 prescription when fitting on the learning rate ablation data.

C.3. Cooldown

In Figure 11, we present the Approach 1 prescription when fitting on the cooldown ablation data.

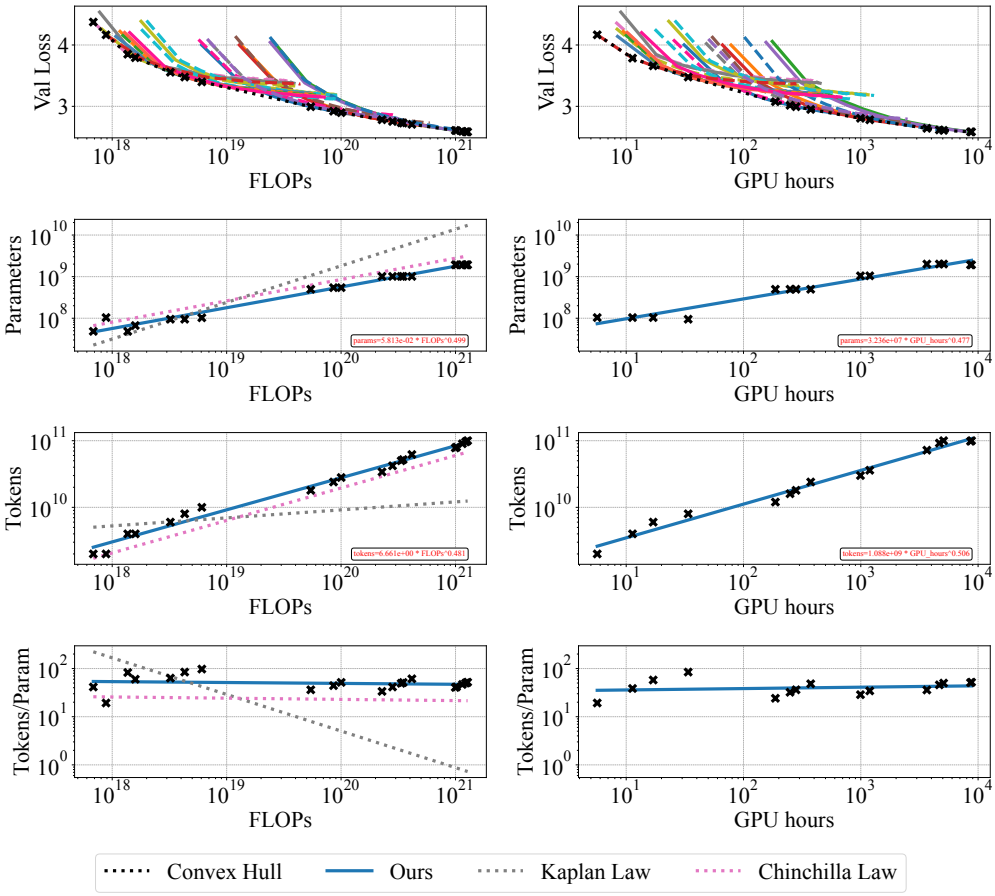

Figure 9. Extended Approach 1 plot from Figure 4, including tokens and parameters axes. As in Figure 4, we present an analysis over FLOPs on the left and over GPU hours take to train on the right.

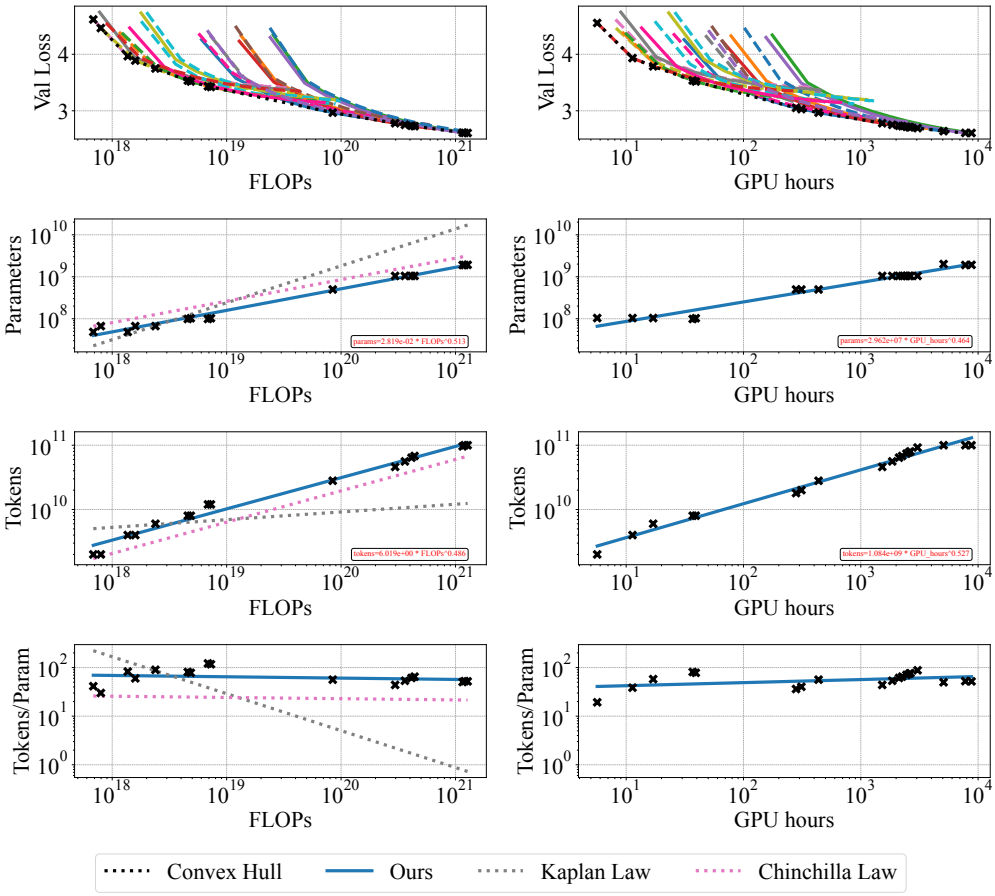

Figure 10. Approach 1 fitted on the learning rate ablation dataset. As in Figure 4, we present an analysis over FLOPs on the left and over GPU hours take to train on the right.

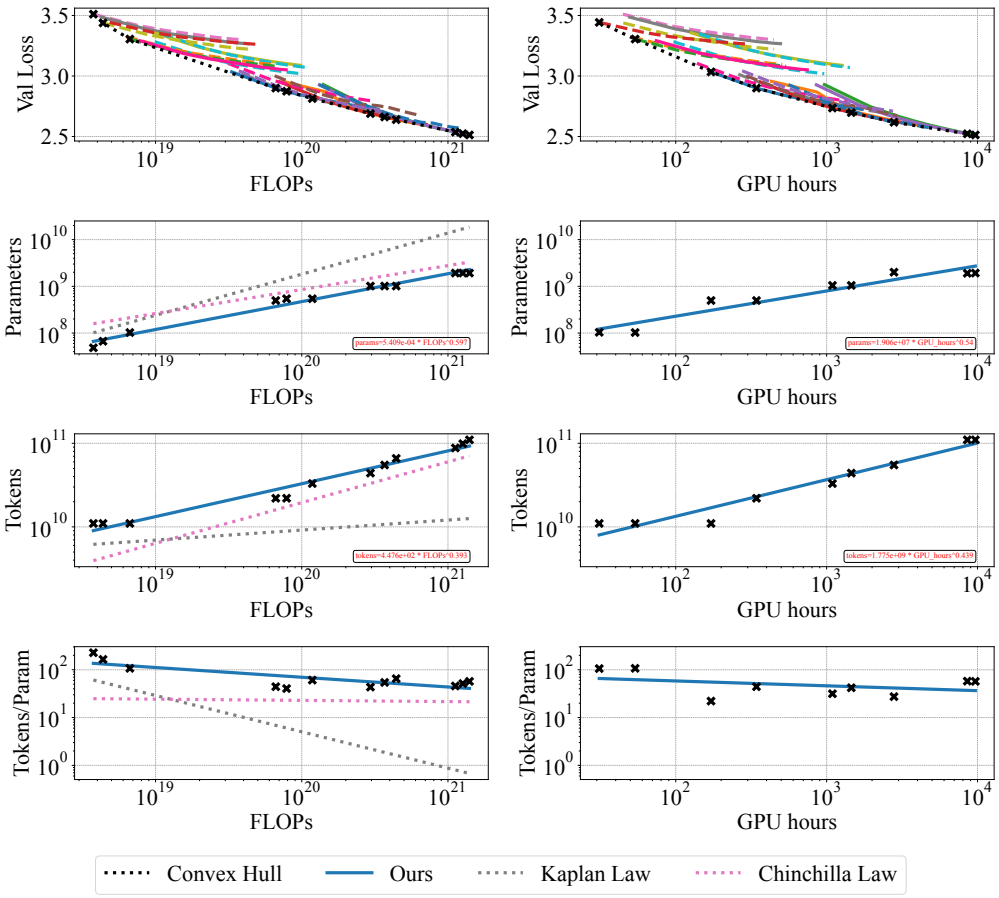

Figure 11. Approach 1 fitted on the cooldown ablation dataset. As in Figure 4, we present an analysis over FLOPs on the left and over GPU hours take to train on the right.

D. Ablations for Approach 3

D.1. Extended Paper Figures

In Figure 12, we plot an extended version of the Approach 3 plot we present in Figure 5.

D.2. Alternative Learning Rates

In Figure 13, fit Approach 3 laws to the dataset we trained with half of the optimal learning rate as described in Section 3.1. We again use a brute force approach as above to plot the results but to allow for precise comparison with later ablations we ignore the integer constraints on the number of heads, still enforcing that at least one head number be in every model. We remove this constraint for all models shown in Appendix D.2, Appendix D.3, Appendix D.4 and Appendix D.5.

D.3. Cooldown

In Figure 14, we fit Approach 3 laws to the subset of data for models for which we linearly decreased the learning rate to zero for.

D.4. Removing Smaller Token Counts

In Figure 15, we present the Approach 3 prescription when fitting on a dataset where all token counts less than 120 billion are removed.

D.5. Varying Delta in the Huber loss

So far we have fit all approach three laws with a Huber loss delta of 10^{-4}. We now ablate this decision by refitting all laws with a delta of 10^{-3}. We use an extremely large grid search of over 4 million initializations for the width-depth based law when fitting.

To begin we show the prescriptions of the Approach 3 laws if the integer constraints are removed, as we did for the learning rate and cooldown ablations in Figures 13 and 14 respectively.

We now compare all Approach 3 laws found with the increased delta. Specifically, we plot the full dataset laws with delta of 10^{-4} in Figure 16 and with 10^{-3} in Figure 17. We plot the learning rate ablation laws with delta of 10^{-4} in Figure 13 and with 10^{-3} in Figure 18. We plot the cooldown ablation laws with delta of 10^{-4} in Figure 14 and with 10^{-3} in Figure 19. In these figures, we see the difference for the full dataset, cooldown and learning rate ablations laws is minimal when changing the delta. We conclude with a cautionary figure about size of the grid search and the delta used in the Huber loss. In Figure 20, where we plot the exponents found by optimizing the Huber loss versus the size of the grid search used for optimization. We see that a delta of 10^{-5} is unstable for smaller grid sizes and including more tokens in the fitting data generally increases stability of the exponents found during optimization.

D.6. Alternative Law Forms

We also experiment with laws of the form shown in Equation (3). In Figure 21, we see that the prescriptions of this law are approximately in line with those of Kaplan et al. (2020). Unlike the laws for shown in Equation (2), these laws tend to prescribe that the width-depth ratio should go to zero as the FLOPs budget increases, i.e. prescribing an infinite-depth model in the infinite-budget limit.

$$L(w, d, p, T) = \frac{A}{w^{\alpha}} + \frac{B}{d^{\beta}} + \frac{C}{T^{\gamma}} + \varepsilon \tag{3}$$

E. Data Sampling

We plot the entire space of all possible models subject to our design constraints discussed in Figure 22. While exploring the impact of finer grained depth differences during our experiments, we decided to add two additional models slightly outside the $\pm 5\%$ tolerance band at the 100M scale; for $width = 512$, in addition to the originally chosen depths of 12 and 13, we added 11 and 14; these appear as a dense collection of 4 points at the same $width$.

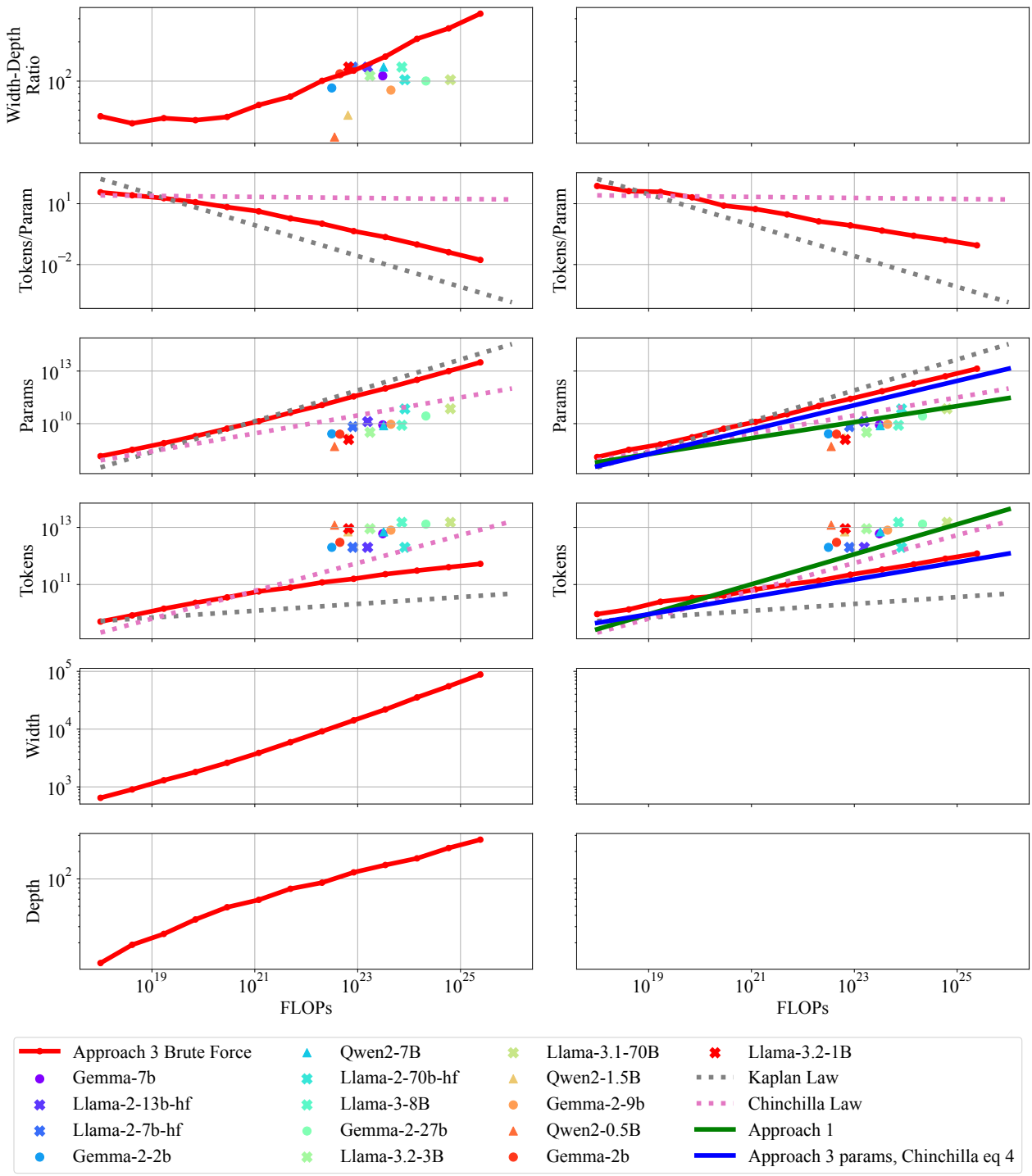

Figure 12. Extended Approach 3 plot from Figure 5, including tokens, parameters, width and depth axes. This also includes a comparison of equation 4 that Hoffmann et al. (2022) propose and a brute force plotting method which includes our FLOPs counting for laws of the form seen in Equation (1). We show these are blue and red lines respectively in the right column seeing only a minor difference in the outcome. We remark that industry models fall systematically below our parameter per FLOPs prescriptions, or equivalently above our token per FLOPs prescriptions.

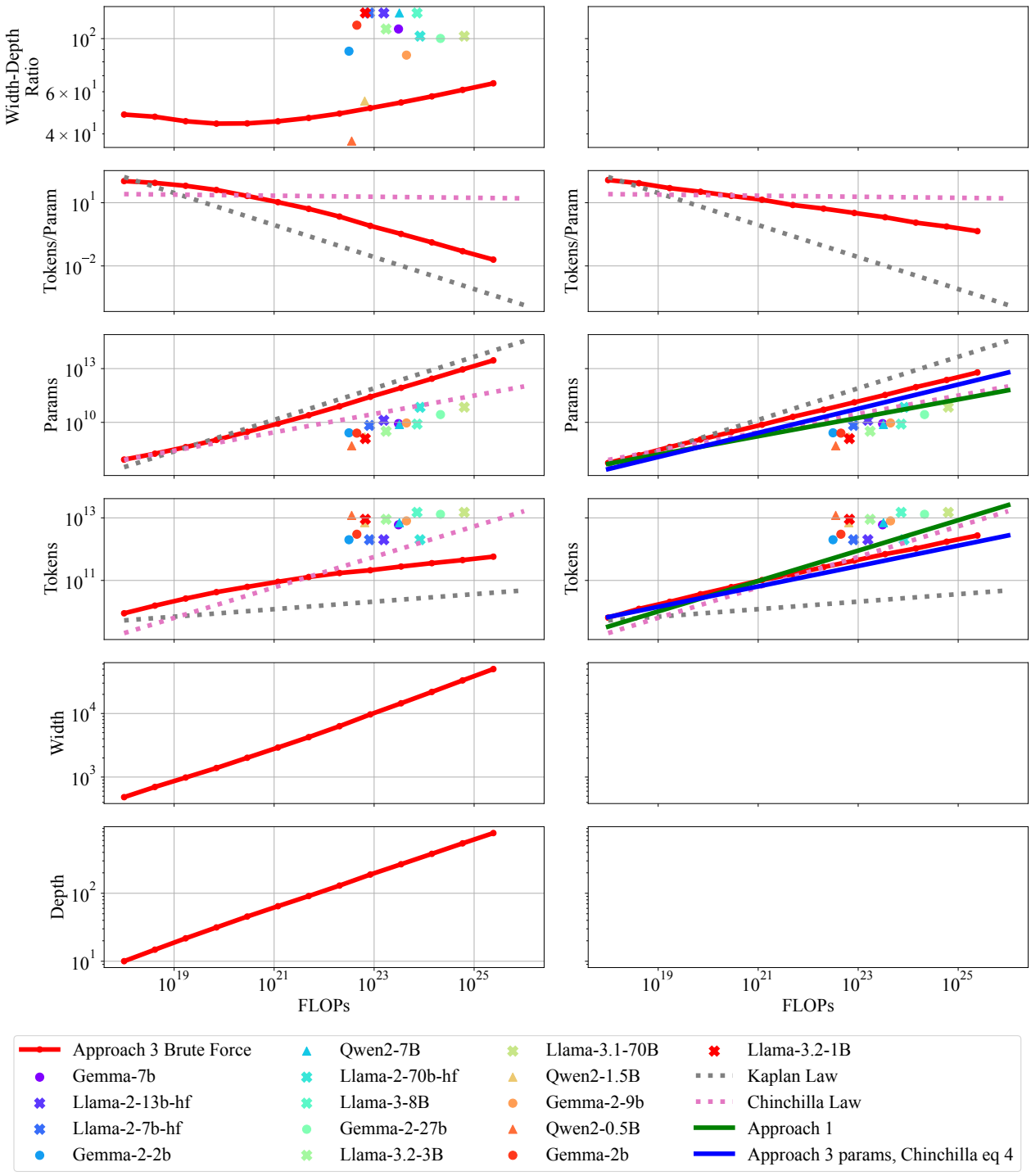

Figure 13. Approach 3 fitted on the learning rate ablation dataset. As in Figure 12, we present an analysis over laws of the from shown in Equation (2) on the left and laws of the form shown in Equation (1) on the right.

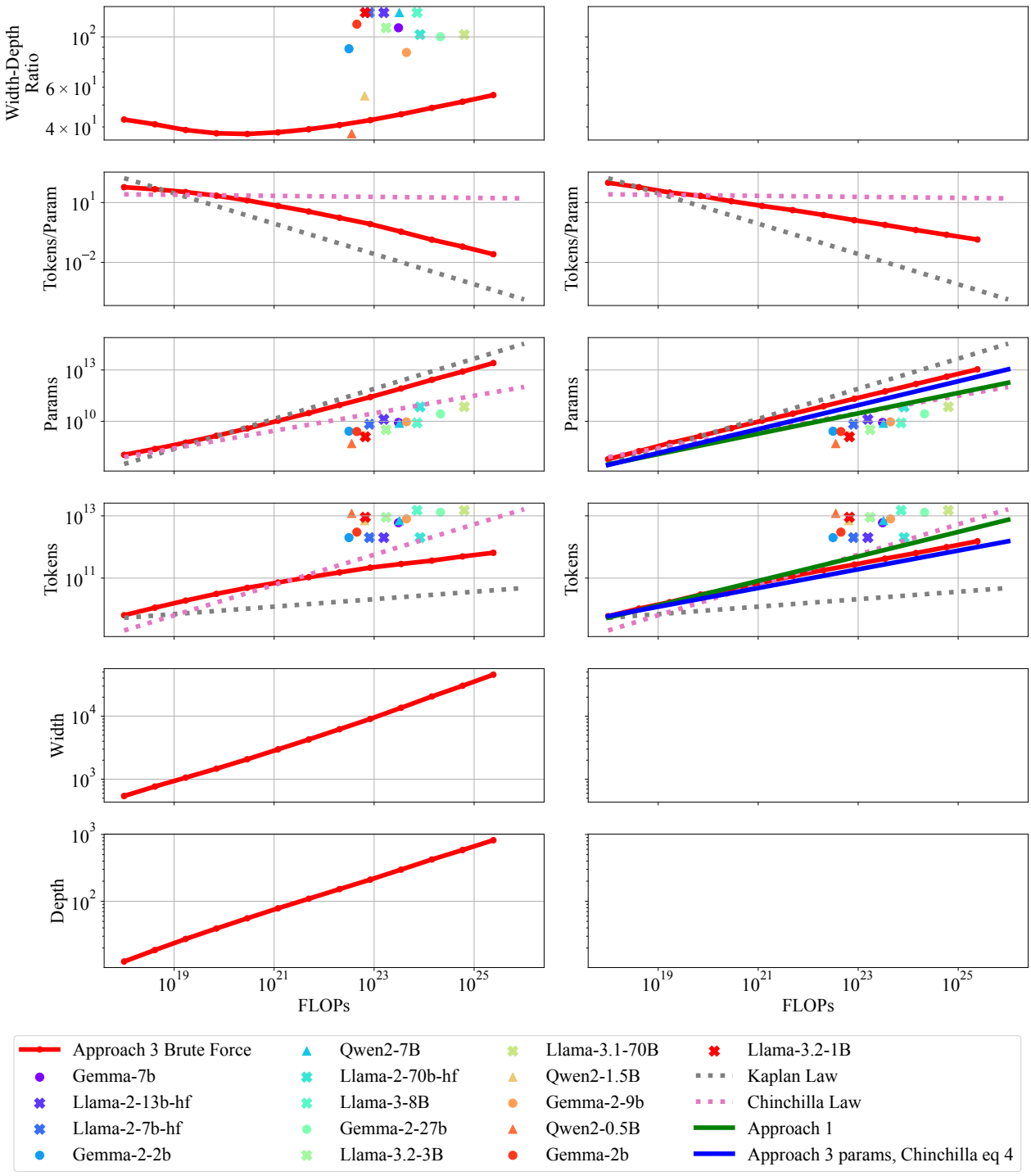

Figure 14. Approach 3 fitted on the cooldown ablation dataset. As in Figure 12, we present an analysis over laws of the from shown in Equation (2) on the left and laws of the form shown in Equation (1) on the right.

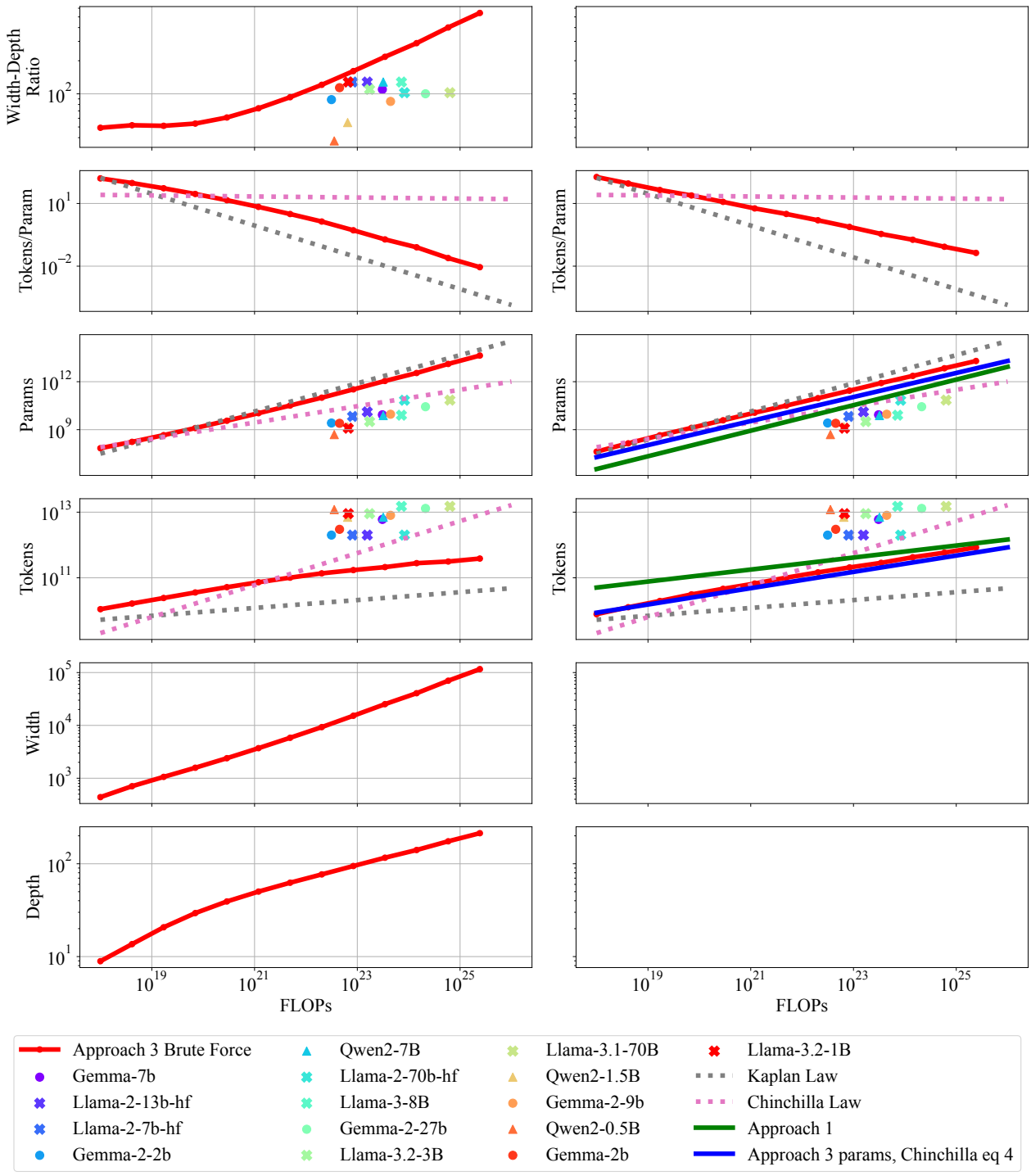

Figure 15. Approach 3 fitted on a dataset where all token counts less than 120 billion are removed. As in Figure 12, we present an analysis over laws of the from shown in Equation (2) on the left and laws of the form shown in Equation (1) on the right.

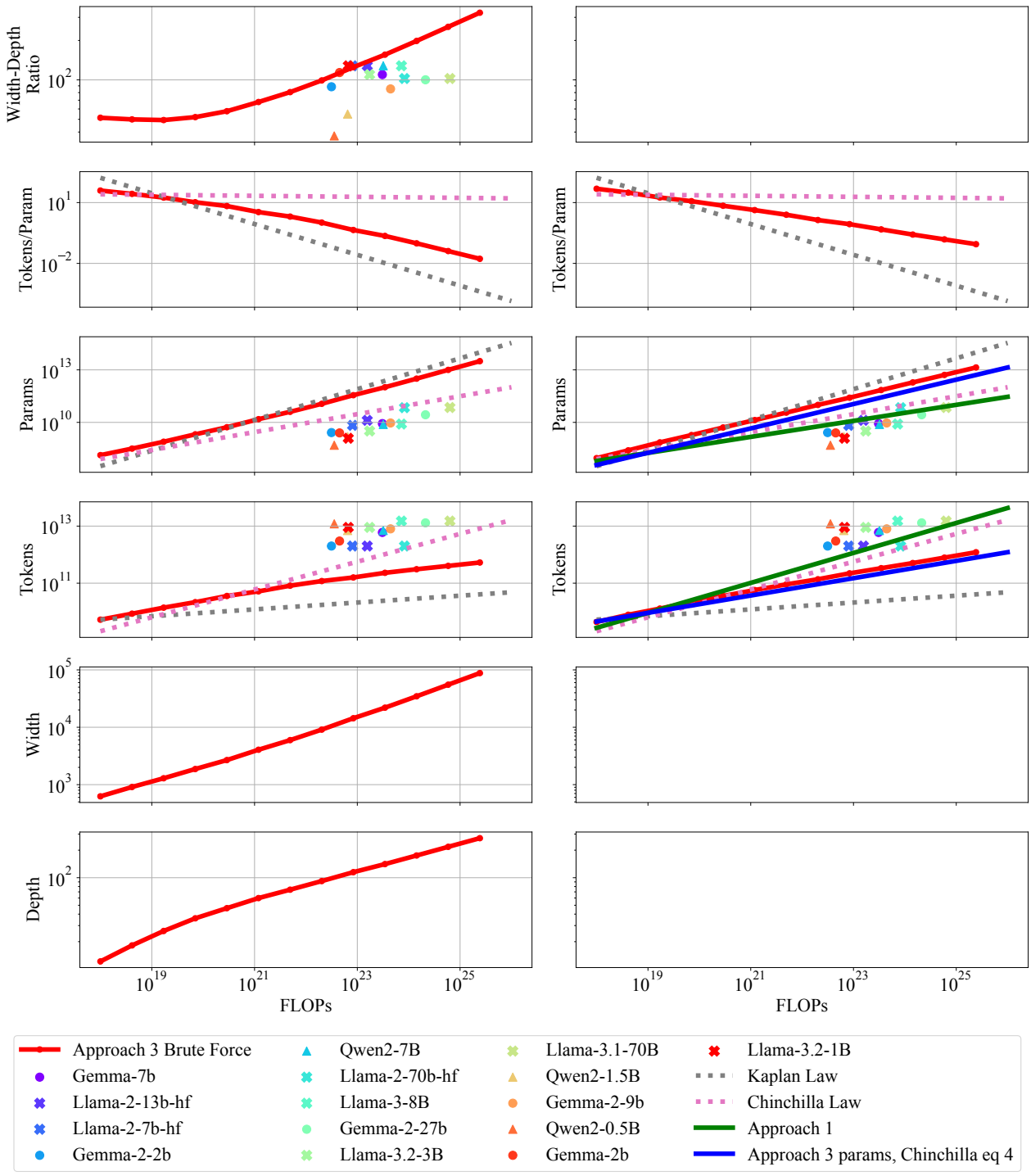

Figure 16. Following from Figure 5, this plot removes the integer constraints when optimizing. As in Figure 12, we present an analysis over laws of the from shown in Equation (2) on the left and laws of the form shown in Equation (1) on the right.

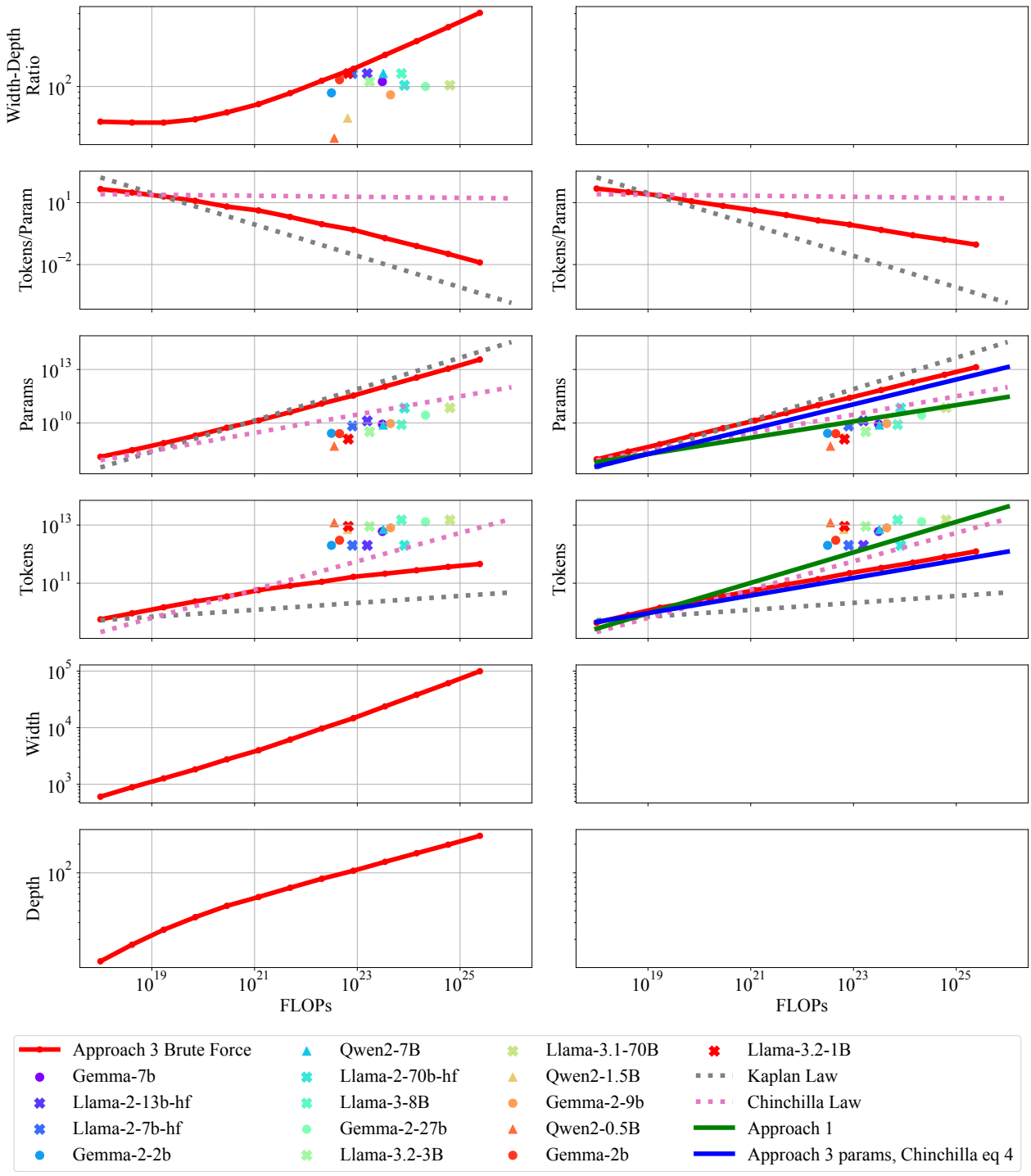

Figure 17. Approach 3 fitted on our main dataset using $\delta = 10^{-3}$ in the Huber loss. Corresponds to Figure 16. As in Figure 12, we present an analysis over laws of the from shown in Equation (2) on the left and laws of the form shown in Equation (1) on the right.

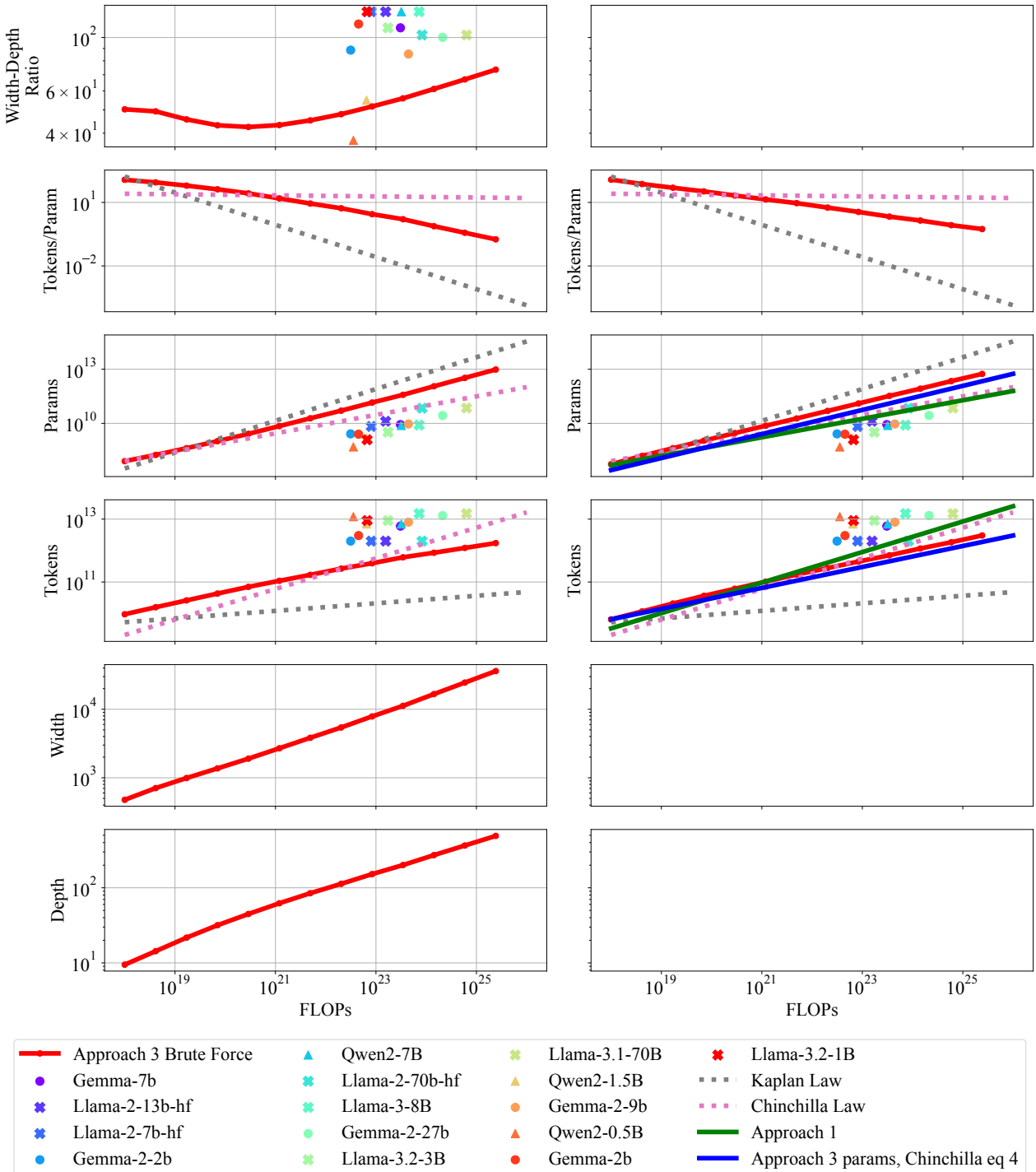

Figure 18. Approach 3 fitted on the learning rate ablation dataset using $\delta = 10^{-3}$ in the Huber loss. Corresponds to Figure 13. As in Figure 12, we present an analysis over laws of the from shown in Equation (2) on the left and laws of the form shown in Equation (1) on the right.

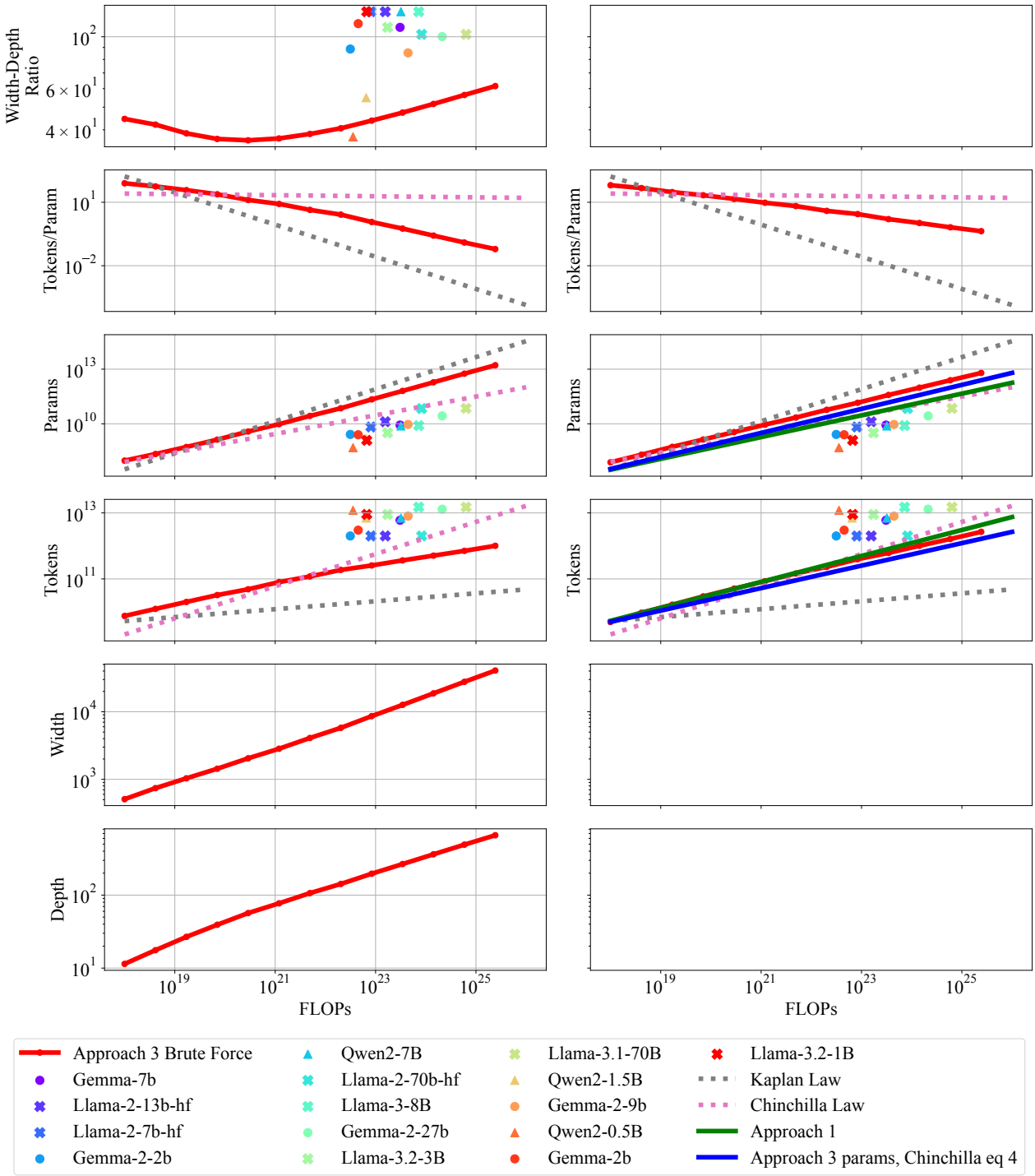

Figure 19. Approach 3 fitted on the cooldown ablation dataset using $\delta = 10^{-3}$ in the Huber loss. Corresponds to Figure 19. As in Figure 12, we present an analysis over laws of the from shown in Equation (2) on the left and laws of the form shown in Equation (1) on the right.

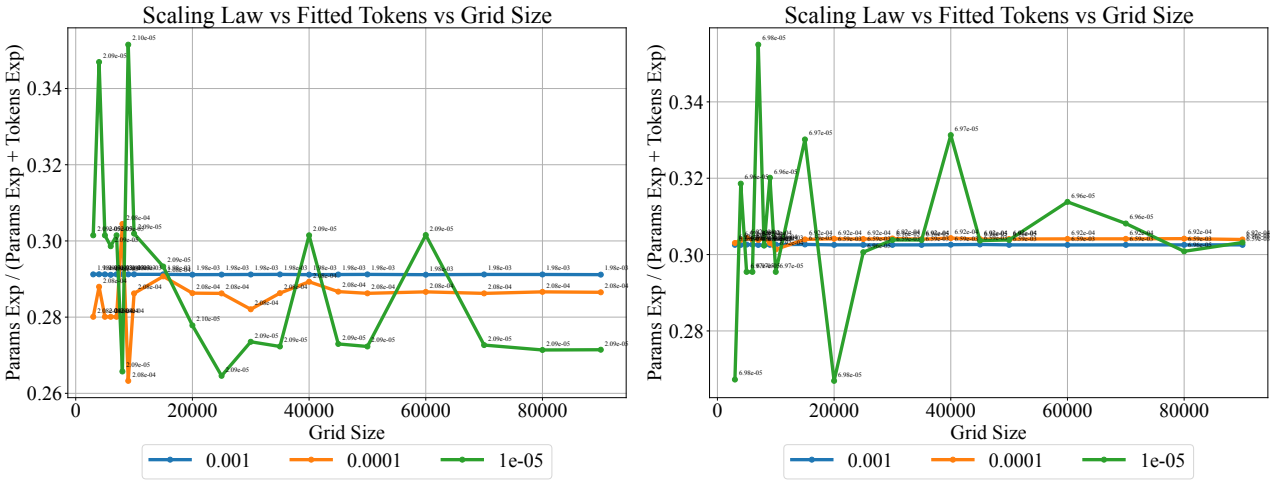

Figure 20. We plot the size of the grid search as the x axis and the gradient of the prescribed tokens as the y axis. We vary delta and see that a delta of 10^{-5} is highly unstable when fitting on smaller grid sizes. On the left, we plot only fitting on data less than 100 billion tokens. On the right, we plot fitting on all data up to 350 billion tokens. We see that including more data increases the stability of the exponents found for smaller grid sizes for deltas $10^{-4}, 10^{-5}$.

F. Training

Despite our best efforts to sufficiently mix the training data, we still see slight jumps in the global training loss when the training switches between chunks of data, hence we use validation loss to fit all laws as this is smooth.

F.1. Loss Curves

F.2. Additional Training Complications

Any gemstone naturally contains a small number of inclusions or fractures. We discuss a few of the minor imperfections in our model collection below.

Dealing with Training Instabilities After some of the widest models were trained beyond $50B$ tokens we began to observe unrecoverable loss spikes that were proceeded by small wobbles in the loss trajectory. Under the general intuition that the culprit was most likely that the $width/depth$ ratios considered were simply *too extreme* for existing initialization and learning rate scaling approaches to handle, we reran some of the models with a "patch" in place.

We modified the initialization rules and learning rate scaling factors to rescale the depth and layer indices of the model such that if $width/depth > 256$ scale variances and learning rates as if the depth of the model was actually $depth' = \lceil (width/100) \rceil$. The overall effect of the patch is to initialize and scale learning rates more conservatively, as if the aspect ratio were only 100 while keeping the original $width$ of the model. We found this allowed us to complete training for a full set of 22 models out to 350B tokens for even our most extreme models.

However, after $350B$ tokens, despite these efforts we observed that most extreme models which were patched still diverged anyway. While a partial cause of this could be the constant learning rate scheduler employed during training, concurrent work, from the authors of the original OLMo paper and codebase (Groeneveld et al., 2024) from which we derived some of our choices, reported that the initialization scheme dubbed the "Mitchell-init" is indeed systematically prone to instabilities later on in training (OLMo et al., 2024). While an unfortunate finding, we were unable to rerun all of our experiments due to the consumption of significant non-fungible compute resources in the original experiments.

Models Lacking Ablations Our cooldown ablation is from initial experiments below $100B$ tokens of training which do not use the patched learning rates scaling rules. This means there are minor discrepancies between the cooldown ablation and main set of training runs for the widest models from the three largest parameter count groups (1792×7, 2560×8, 3072×12). We also do not cool down the $100B$ token checkpoint for the 3072×12 model as it was experiencing a loss spike at that final point. Finally, we do not include ablations for the two width 512 models which do not fall into the $\pm 5\%$

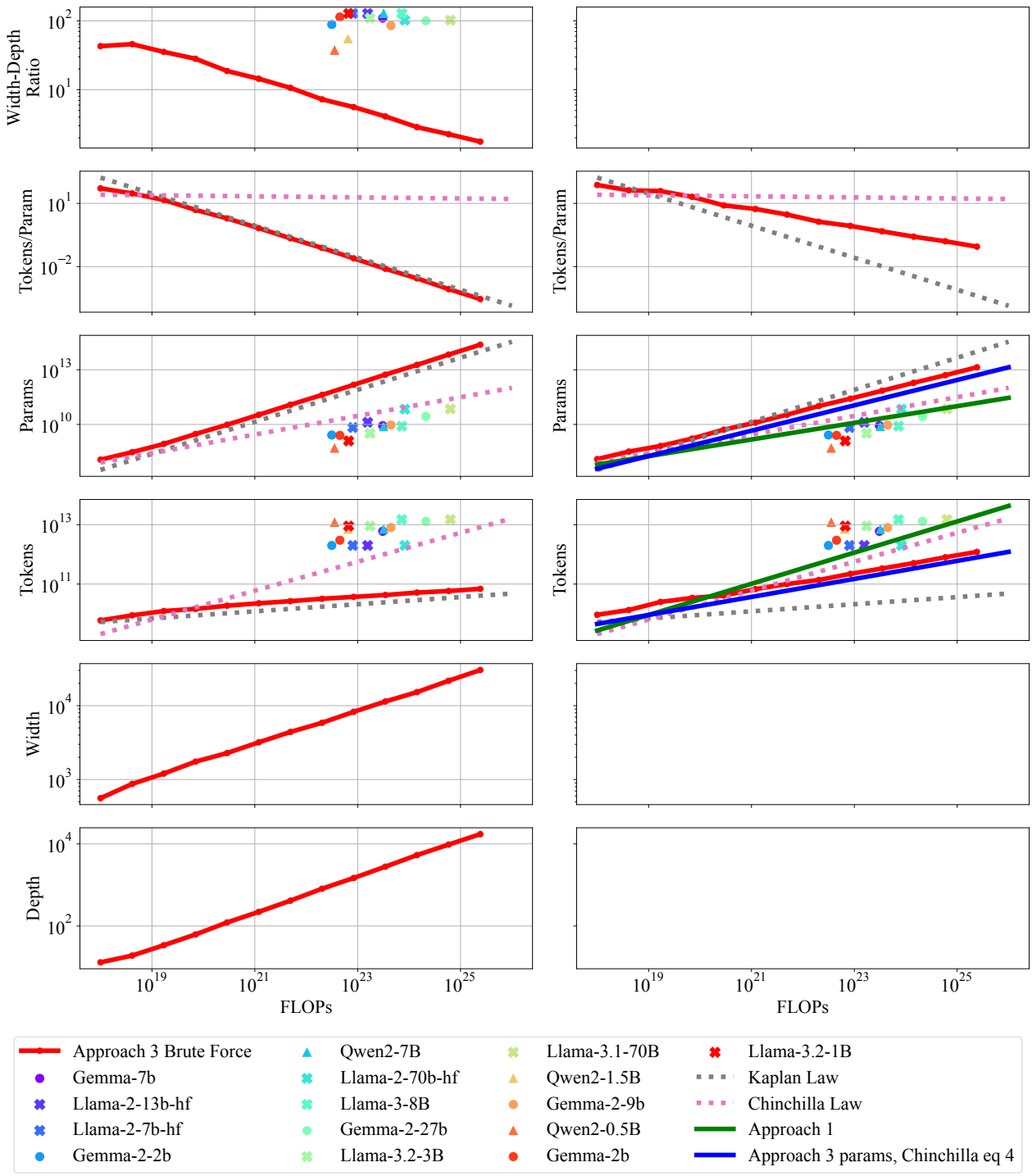

Figure 21. Approach 3 fitted on our main dataset using an ansatz of the form shown in Equation (3).

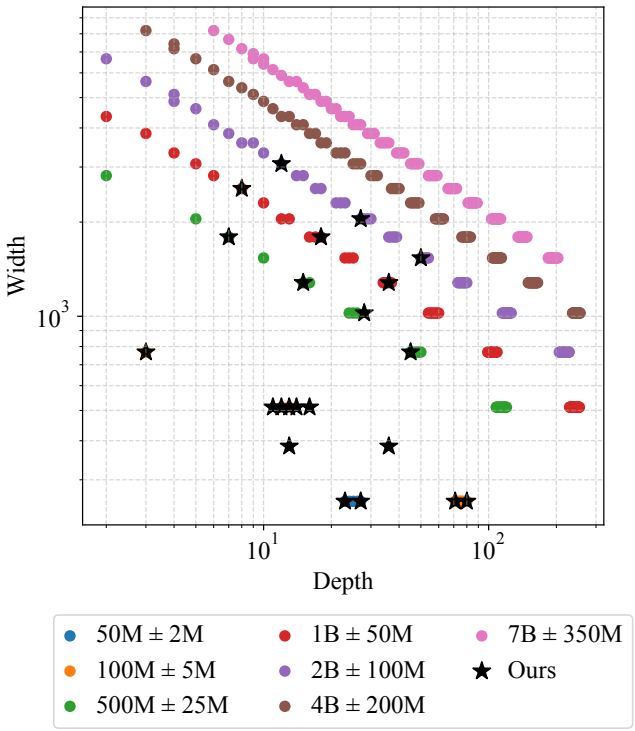

Figure 22. All possible model shapes we could have chosen based on our architecture within $\pm 5\%$ are shown as circles. The points we selected are highlighted as stars, including the two extra points we select to have four models of width 512.

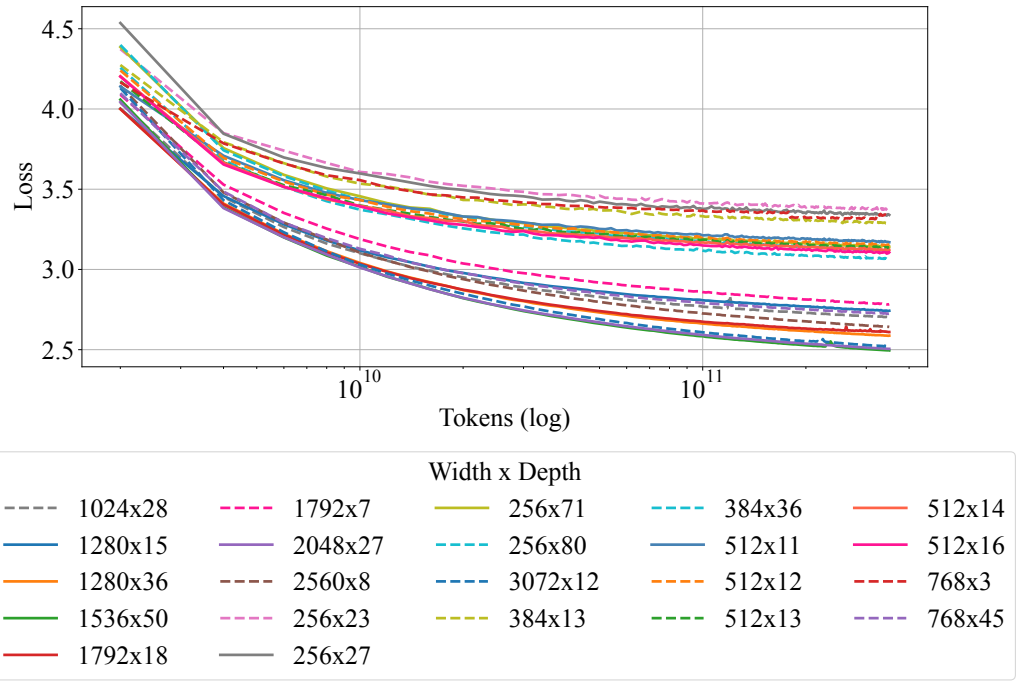

Figure 23. Loss curves for the main 22 training runs.

boundary of the $100M$ parameter count (512×11, 512×14) as they were only added to the collection in later experiments.

G. FLOP counting matters

In Figure 24 we show that the common approximation of FLOPs per token$= 6 \times parameters$, miscounts the true FLOPS by a significant amount.

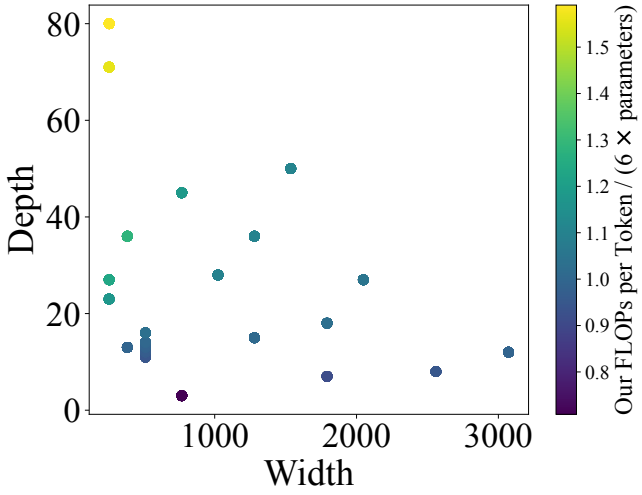

Figure 24. We color the points based on the ratio of our calculated FLOPs per token which is shown in the code below and using $6 \times parameters$. We see counting the FLOPs properly becomes more important for aspect ratios off outside of the standard regime.

```
VOCAB_OURS = 50304
SEQ_LEN = 2048
WORLD_BATCH_SIZE = 2048.0
HEAD_SIZE = 128
EXPAND_FACTOR = 4.0

def flops_per_token_gqa(
    width: NDArray[number] | number,
    depth: NDArray[number] | number,
    vocab_size=VOCAB_OURS,
    queries_per_group=2,
    seq_len=SEQ_LEN,
):
    """
    Some details (negligible even for extremely wide models) omitted, including:
    * numerically stable softmax
    * softmax addition only being over rows
    * dot products being only n-1 additions (fused multiply-add exists anyway)
    """
    num_qheads = width / HEAD_SIZE
    num_kvheads = (
        2 * num_qheads / queries_per_group
    )

    embeddings = 0  # 0 if sparse lookup, backward FLOPs negligible

    attention = 2.0 * seq_len * (num_qheads + num_kvheads) * width * HEAD_SIZE
    attention += (
        3.5 * seq_len * (num_qheads + num_kvheads / 2) * HEAD_SIZE
```

```
)  # RoPE, as implemented here/GPT-NeoX
# score FLOPs are halved because causal => triangular mask => usable sparsity
kq_logits = 1.0 * seq_len * seq_len * HEAD_SIZE * num_qheads
softmax = 3.0 * seq_len * seq_len * num_qheads
softmax_q_red = 2.0 * seq_len * seq_len * HEAD_SIZE * num_qheads
final_linear = 2.0 * seq_len * width * HEAD_SIZE * num_qheads
attn_bwd = (
    2.0 * attention
    + 2.5 * (kq_logits + softmax + softmax_q_red)
    + 2.0 * final_linear
) * depth
attention += kq_logits + softmax + softmax_q_red + final_linear

ffw_size = EXPAND_FACTOR * width
dense_block = (
    6.0 * seq_len * width * ffw_size
)  # three matmuls instead of usual two because of GEGLU
dense_block += (
    10 * seq_len * ffw_size
)  # 7 for other ops: 3 for cubic, two additions, two scalar mults
dense_block += 2.0 * width * seq_len  # both/sandwich residual additions
rmsnorm = 2 * 7.0 * width * seq_len

final_rms_norm = 7.0 * width * seq_len  # one last RMSNorm
final_logits = 2.0 * seq_len * width * vocab_size
nonattn_bwd = 2.0 * (
    embeddings + depth * (dense_block + rmsnorm) + final_rms_norm + final_logits
)
forward_pass = (
    embeddings
    + depth * (attention + dense_block + rmsnorm)
    + final_rms_norm
    + final_logits
)
backward_pass = attn_bwd + nonattn_bwd  # flash attention

return (forward_pass + backward_pass) / seq_len
```