A Notation

We use the following notation:

- $\mathcal{L}$: Cross-entropy loss.
- D: Dataset size in tokens.
- N: Number of non-embedding parameters in the model.
- $\hat{N}$: Total number of parameters in the model, which excludes embedding layer but includes the model's head layer.
- C: Compute budget in FLOPs.
- N_{layer}: Number of layers in the Transformer model.
- d_{ff}: Dimension of the feed-forward network hidden layer in the Transformer.
- d_{model}: Hidden dimension of the Transformer model.
- N_{head}: Number of attention heads in the Transformer model.
- LR: Learning rate.
- BS: Batch size (in tokens).
- $\eta_{opt}(N, D)$: Optimal peak learning rate for a given parameter count N and dataset size D.
- $B(N, D)$: Optimal batch size (in tokens) for given parameter N and dataset size D.
- $\mathbb{A}$: Model architecture space defined by N_{layer}, d_{ff}, d_{model}, and N_{head}.
- $\mathbb{D}$: Training data distribution governed by the data-generating probability distribution.

B Model Scale Dominates Optimal Hyperparameter Selection Over Computational Complexity

To investigate how model architecture variations affect optimal hyperparameter settings, we conducted two sets of control experiments. In the first set, we maintained a constant parameter count (N), while in the second set, we kept the computational complexity (M) constant. Both sets used identical training configurations with 8B training tokens, varying only in their architectural proportions.

Tab. 4 presents the detailed configurations and results for both experimental groups. For each model, we systematically varied the hidden dimension (d_{model}), feed-forward dimension (d_{ff}), number of attention heads (N_{head}), and number of layers (N_{layer}) while maintaining either constant N or M. The embedding dimension (D) was fixed at 8.00×10^9 across all experiments.

To visualize the impact of hyperparameters across different architectural configurations, we generated heatmaps of the loss landscape with respect to LR and BS in Fig. 5 and 8. The heatmaps reveal consistent patterns in the optimal hyperparameter regions across different architectural configurations within each experimental group.

The experimental results reveal several key findings: (1) Models with constant N demonstrate remarkably consistent optimal hyperparameter regions, with minimal variation in minimum loss values (ranging from 2.4294 to 2.4776) despite architectural differences. (2) The constant M experiments show slightly more variation in optimal hyperparameter regions and minimum loss values (ranging from 2.4346 to 2.5089), suggesting that parameter count N may be a more robust indicator for hyperparameter selection than computational complexity M. (3) Across both experimental groups, the optimal learning rates typically fall within a narrow range (6.91×10^{-4} to 1.95×-3), and batch sizes cluster around either 131,072 or 262,144, regardless of the specific architectural configuration.

These findings strongly suggest that the fundamental scale metrics, particularly the parameter count N, are more influential in determining optimal hyperparameter settings than specific architectural choices. This observation motivates our discussion of hyperparameter scaling laws in relation to N in Sec. 3.4.

C Model Structural Parameters

Table 5 and Table 6 summarizes the precise architectural settings for all models evaluated in our study. In the **dense model** group (Models 1-18), we cover models ranging from 2.15×10^8 to 1.07×10^9 total parameters by varying the hidden dimension (d_{model}), feed-forward width (d_{ff}), number of attention heads (N_{head}) and number of layers (N_{layer}). Each entry also lists the corresponding dataset size D used during training. In the **MoE model** group (Models 1-16), we hold the overall parameter count fixed at approximately 2.15×10^9 while sweeping dataset size D from 2×10^9 up to 2×10^{10}. We further vary the number of experts (N_{expert}), per-expert hidden size (d_{moe}), top-k routing, and the resulting active parameter count (N_a). This systematic variation allows direct comparison of dense versus sparse Mixture-of-Experts architectures under matched compute budgets and data scales.

D Composition of Training Datasets

Tab. 7 details the dataset weight percentages for four training recipes: Baseline, Code+Math, More Code+Math, and En-CN (English-Chinese bilingual).

Bilingual Corpus: We augmented the original English-only dataset with Chinese data, creating a bilingual distribution to test the law's validity in multilingual settings.
Code Integration: We reduced English content and incorporated 32.36% of the code-the-stack dataset, examining the law's adaptability to code-heavy distributions.
Code-Dominant: We further decreased English content and increased code-the-stack to 57.05%, representing an extreme shift towards code-based data.

E Statistical Validation of Batch Size Scaling Relationships

Empirical validation. This part provides a statistical analysis to examine the claim in the main text that batch size B is independent of model parameter count N but dependent on training dataset size D. We conducted a multivariate regression analysis on all experimental configurations. All

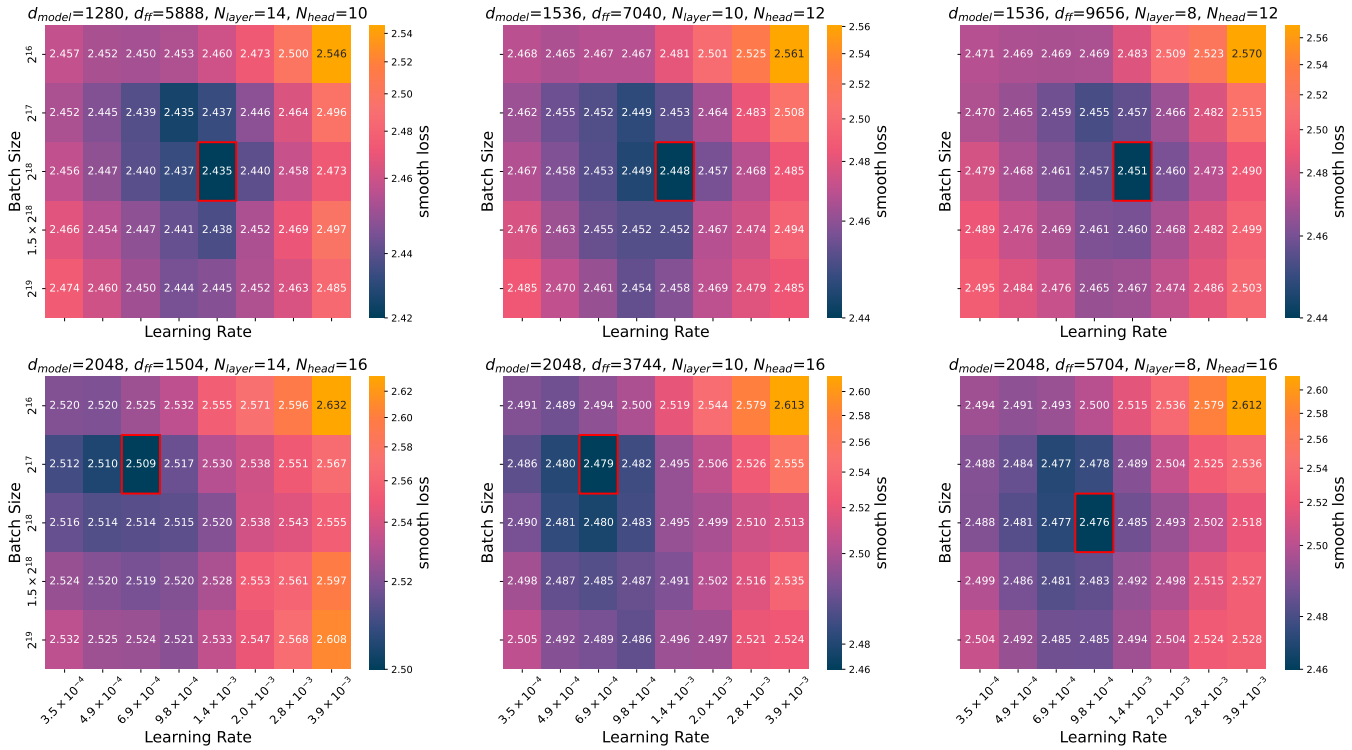


Figure 8: Loss landscapes visualized as heatmaps across learning rate (x-axis) and batch size (y-axis) configurations. Darker colors indicate lower loss values. Shows results for models with constant computational complexity M, exhibiting slightly more variance in optimal hyperparameter regions.

variables are log-transformed to linearize power-law relationships (logB, logN, logD), consistent with the main text. Three regression formulations were compared:

$$\begin{aligned} \text{N-only Formulation:} \quad & \log B = \beta_0 + \beta_1 \log N \\ \text{D-only Formulation:} \quad & \log B = \beta_0 + \beta_2 \log D \\ \text{Full Formulation:} \quad & \log B = \beta_0 + \beta_1 \log N + \beta_2 \log D \end{aligned}$$

We fit each of these models using ordinary least squares (OLS) regression and perform hierarchical F-tests to assess the contribution of $\log N$ and $\log D$ to predicting $\log B$. The results are summarized in Table 8 and Table 9 .

As shown in Appendix Table 8 and Table 9, the D-only formulation achieves nearly identical explanatory power as the full model ($R^2 = 0.821$ vs. 0.823), while the N-only model performs poorly ($R^2 < 0$). Moreover, in the full model, the coefficient of $\log D$ is highly significant ($p < 0.001$), whereas that of $\log N$ is not ($p = 0.257$). These results confirm that the optimal batch size scales with D, but not with N.

F Theoretical Foundations of the Step Law

While our primary contribution is an empirical one, the discovered Step Law is not merely a curve fit to data points. It is grounded in the fundamental principles of stochastic optimization and scaling analysis, akin to the reasoning used in works like u P (**?**). Here, we provide a theoretical justification for the functional form of our scaling laws, arguing that they emerge from the necessity to maintain stable training dynamics as model size (N) and data scale (D) change.

The Optimal Learning Rate $\eta_{opt}(N, D)$ The learning rate is arguably the most critical hyperparameter, governing the magnitude of parameter updates. An optimal learning rate must balance the speed of convergence with the stability of training, adapting to both the geometry of the loss landscape (influenced by N) and the statistical properties of the training data (influenced by D).

Scaling with Model Size N: Stabilizing Parameter Updates Let $\boldsymbol{\theta}$ be the model parameters, and let $\mathcal{L}(\boldsymbol{\theta})$ be the loss. The basic parameter update rule for a parameter group is $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla \mathcal{L}(\boldsymbol{\theta}_t)$. To ensure training stability, particularly in large models, the magnitude of the change in the function output, $\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}(\boldsymbol{\theta}_t)$, should remain well-behaved and not diverge as $N \to \infty$. A first-order Taylor expansion gives:

$$\mathcal{L}(\boldsymbol{\theta}_{t+1}) \approx \mathcal{L}(\boldsymbol{\theta}_t) - \eta \left\| \nabla \mathcal{L}(\boldsymbol{\theta}_t) \right\|_2^2 \tag{5}$$

The stability of this update step depends on the product $\eta \left\| \nabla \mathcal{L} \right\|_2^2$. In many neural network parameterizations, the squared norm of the gradient scales with the number of parameters. For a simplified model, let's assume the contribu-

tions from individual parameter gradients are roughly independent, leading to $\|\nabla\mathcal{L}\|_2^2 \propto N$. This is a common observation in wide networks where gradients accumulate across neurons.

To keep the functional change $\mathcal{L}(\boldsymbol{\theta}_{t+1}) - \mathcal{L}(\boldsymbol{\theta}_t)$ of a constant order, i.e., $\mathcal{O}(1)$, as N grows, we must counteract the growth of the gradient norm. This implies the scaling relationship:

$$\eta \cdot N \approx \text{const} \quad \Longrightarrow \quad \eta \propto N^{-1} \tag{6}$$

This derivation, while simplified, provides a theoretical basis for why the optimal learning rate should decrease as model size increases. The precise exponent, which we empirically find to be $\alpha \approx -0.713$, depends on the specific architecture and interactions between layers, but the inverse relationship is a direct consequence of maintaining stable updates in the face of growing gradient magnitudes. This mirrors the core logic of μP, which adjusts learning rates for different parameter groups to stabilize training dynamics at scale.

Scaling with Dataset Size *D*: Adapting to Training Horizon The relationship between η and D is more subtle. Our empirical result, $\eta \propto D^{\beta}$ with $\beta > 0$, suggests that one can use a *larger* learning rate for a *larger* dataset. This can be understood from two perspectives:

1. **Smoother Loss Landscape**: Training on more data (D) exposes the model to a richer, more diverse set of examples. This has a regularizing effect, often resulting in a smoother loss landscape with fewer sharp, spurious local minima. A smoother landscape can tolerate a larger step size without causing divergence, allowing for more aggressive, faster convergence.
2. **Longer Training Horizon**: With a fixed batch size, a larger dataset D implies a greater number of training steps $S \propto D$. Modern learning rate schedules, such as our cosine decay, are defined over this entire horizon. A higher peak learning rate η is permissible when the number of decay steps is large, as the model has ample time to anneal the learning rate to a small value for fine-tuning in the final stages of training. A short training horizon would necessitate a smaller peak η to avoid ending the training with a still-large learning rate.

Combining these scaling arguments, we arrive at the functional form for the optimal learning rate, which jointly depends on model size and data scale:

$$\eta_{opt}(N, D) \propto N^{\alpha} D^{\beta} \tag{7}$$

This form captures the trade-off between architectural complexity and data-driven regularization.

The Optimal Batch Size $B_{opt}(D)$: Managing Gradient Noise Our second key finding is that the optimal batch size B depends strongly on the dataset size D but is largely independent of the model size N. This can be explained by viewing the roles of batch size and learning rate as fundamentally decoupled.

Decoupling Batch Size from Model Size We posit that the primary role of the **batch size** is to manage the **statistical variance of the stochastic gradient**, which is a property of the data sampling process. In contrast, the primary role of the **learning rate** is to adapt to the **geometric curvature of the loss landscape**, which is determined by the model architecture (N).

The variance of the stochastic gradient estimator $\mathbf{g}_B$ for a batch of size B is inversely proportional to B:

$$\text{Var}(\mathbf{g}_B) = \frac{\text{Var}(\mathbf{g}_1)}{B} \tag{8}$$

where $\mathbf{g}_1$ is the gradient for a single example. The "signal-to-noise ratio" (SNR) of the gradient can be defined as $\text{SNR} \propto \frac{\|\mathbb{E}[\mathbf{g}_B]\|^2}{\text{Var}(\mathbf{g}_B)} \propto B$.

While model size N dramatically alters the loss landscape's curvature (requiring η to adapt), it does not fundamentally change the statistical properties of the data distribution from which we sample. Therefore, the task of controlling gradient noise falls to B, and the task of navigating the resulting landscape falls to η. This explains the observed independence of B from N.

Scaling with Dataset Size *D* The scaling relationship $B \propto D^{\gamma}$ with $\gamma > 0$ arises from the need to maintain a sufficient gradient SNR over the course of training. As the total amount of data D increases, we are effectively trying to optimize over a much larger and more complex empirical distribution. To obtain a proportionally reliable estimate of the true gradient at each step, it is natural to increase the sample size (B) used for the estimate. If we were to keep B constant while increasing D, the gradient noise would remain the same, but we would be taking many more noisy steps. Scaling B with D ensures that as the optimization problem grows in scope, the quality of our gradient estimate scales with it, leading to a more stable and effective training process. This aligns with findings in other large-scale training studies, which distinguish between the hardware-centric "critical batch size" and the loss-centric "optimal batch size" that we investigate here (**??**).

In summary, this theoretical framework provides a principled explanation for the Step Law's structure. It frames hyperparameter scaling not as an arbitrary fitting exercise, but as a necessary adaptation to maintain stable and efficient optimization dynamics across vast scales of model and data.

d_{model}	d_{ff}	N_{head}	N_{layer}	$\eta_{opt}(N, D)$	$B(N, D)$	D	N	M
Constant N Experiments								
1280	12264	10	8	1.95×10^{-3}	262,144	8.00×10^9	4.29×10^8	2.83×10^9
1280	6280	10	14	1.38×10^{-3}	262,144	8.00×10^9	4.29×10^8	3.02×10^9
1536	9600	12	8	9.77×10^{-4}	131,072	8.00×10^9	4.29×10^8	2.88×10^9
1536	7264	12	10	1.38×10^{-3}	262,144	8.00×10^9	4.29×10^8	2.95×10^9
1536	4608	12	14	9.77×10^{-4}	131,072	8.00×10^9	4.29×10^8	3.10×10^9
2048	6000	16	8	9.77×10^{-4}	262,144	8.00×10^9	4.29×10^8	2.98×10^9
2048	4256	16	10	9.77×10^{-4}	262,144	8.00×10^9	4.29×10^8	3.08×10^9
2048	2256	16	14	9.77×10^{-4}	262,144	8.00×10^9	4.29×10^8	3.28×10^9
Constant M Experiments								
1280	12608	10	8	1.38×10^{-3}	262,144	8.00×10^9	4.40×10^8	2.89×10^9
1280	5888	10	14	1.38×10^{-3}	262,144	8.00×10^9	4.08×10^8	2.89×10^9
1536	9656	12	8	1.38×10^{-3}	262,144	8.00×10^9	4.31×10^8	2.89×10^9
1536	7040	12	10	1.38×10^{-3}	262,144	8.00×10^9	4.19×10^8	2.89×10^9
1536	4056	12	14	9.77×10^{-4}	262,144	8.00×10^9	3.94×10^8	2.89×10^9
2048	5704	16	8	9.77×10^{-4}	262,144	8.00×10^9	4.15×10^8	2.89×10^9
2048	3744	16	10	6.91×10^{-4}	131,072	8.00×10^9	3.98×10^8	2.89×10^9
2048	1504	16	14	6.91×10^{-4}	131,072	8.00×10^9	3.64×10^8	2.89×10^9

Table 4: **Model configurations and results for constant N and constant M experiments.** The first group (top) maintains constant parameter count $N \approx 4.29 \times 10^8$, while the second group (bottom) maintains constant computational complexity $M \approx 2.89 \times 10^9$. M: non-embedding FLOPs/token.

Model	N	D	d_{model}	d_{ff}	N_{head}	N_{layer}
1	2.15×10^8	1.14×10^{10}	960	9368	15	7
2	4.29×10^8	5.00×10^{10}	1280	9472	10	10
3	2.68×10^8	8.00×10^{10}	1024	9552	16	8
4	4.29×10^8	8.00×10^9	1280	9472	10	10
5	1.07×10^9	2.00×10^{10}	2048	8192	16	16
6	5.37×10^8	1.00×10^{10}	1280	9048	10	13
7	2.15×10^8	4.00×10^9	960	9368	15	7
8	2.68×10^8	5.00×10^9	1024	9552	16	8
9	2.68×10^8	1.42×10^{10}	1024	9552	16	8
10	1.07×10^9	5.69×10^{10}	2048	8192	16	16
11	2.15×10^8	1.00×10^{11}	960	9368	15	7
12	4.29×10^8	2.27×10^{10}	1280	9472	10	10
13	5.37×10^8	2.84×10^{10}	1280	9048	10	13
14	2.15×10^8	2.00×10^{10}	960	9368	15	7
15	4.29×10^8	4.00×10^{10}	1280	9472	10	10
16	2.68×10^8	2.50×10^{10}	1024	9552	16	8
17	5.37×10^8	5.00×10^{10}	1280	9048	10	13
18	1.07×10^9	1.00×10^{11}	2048	8192	16	16

Table 5: **Dense Model Configuration.**

Model	N	D	d_{model}	N_{head}	N_{layer}	N_{expert}	d_{moe}	Top-k	N_a
1	2.151×10^9	2.00×10^9	1408	11	16	89	352	1	1.88×10^8
2	2.151×10^9	2.00×10^9	1408	11	16	88	352	2	2.33×10^8
3	2.155×10^9	2.00×10^9	1408	11	16	8	3528	1	5.90×10^8
4	2.156×10^9	2.00×10^9	1408	11	16	8	2888	3	1.24×10^9
5	2.151×10^9	4.00×10^9	1408	11	16	89	352	1	1.88×10^8
6	2.151×10^9	4.00×10^9	1408	11	16	88	352	2	2.33×10^8
7	2.155×10^9	4.00×10^9	1408	11	16	8	3528	1	5.90×10^8
8	2.156×10^9	4.00×10^9	1408	11	16	8	2888	3	1.24×10^9
9	2.151×10^9	8.00×10^9	1408	11	16	89	352	1	1.88×10^8
10	2.151×10^9	8.00×10^9	1408	11	16	88	352	2	2.33×10^8
11	2.155×10^9	8.00×10^9	1408	11	16	8	3528	1	5.90×10^8
12	2.156×10^9	8.00×10^9	1408	11	16	8	2888	3	1.24×10^9
13	2.151×10^9	2.00×10^{10}	1408	11	16	89	352	1	1.88×10^8
14	2.151×10^9	2.00×10^{10}	1408	11	16	88	352	2	2.33×10^8
15	2.155×10^9	2.00×10^{10}	1408	11	16	8	3528	1	5.90×10^8
16	2.156×10^9	2.00×10^{10}	1408	11	16	8	2888	3	1.24×10^9
17	6.510×10^9	1.00×10^{10}	2048	16	24	82	512	2	7.26×10^8
18	6.510×10^9	1.30×10^{11}	2048	16	24	82	512	2	7.26×10^8

Table 6: **MoE Model Configuration.** N_{expert} denotes the number of experts. d_{moe} denotes the hidden size of experts. Top-k denotes the number in the routing algorithm. N_a denotes the activate parameters.

Dataset	**Baseline**	**Code+Math**	**More Code+Math**	**En-CN**
Web-data-en	79.53	44.75	20.00	44.99
Web-data-cn	–	–	–	34.52
Code-the-stack	4.62	32.36	57.05	4.63
Web-data-math	–	7.07	7.07	–
Book-non-novel-en	4.35	4.34	4.34	4.35
Paper	3.38	3.37	3.37	3.38
Wikipedia-mtlg	3.24	3.24	3.24	3.25
Stackexchange	2.21	2.21	2.21	2.22
Wikipedia-en	1.69	1.69	1.69	1.69
Book-novel-en	0.83	0.83	0.83	0.83
Wikipedia-cn	0.13	0.13	0.13	0.13

Table 7: **Comparison of dataset weights (%) across different training recipes.** Each recipe represents a different focus: baseline, enhanced code and mathematics capability, and English-Chinese bilingual ability.

Table 8: Regression formulation comparisons

Comparison	**Adj.** R^2	ΔR^2 **VS Full**	**F-statistic**
N-only	-0.032	-85.4%	0.08
D-only	0.821	-0.2%	138.2
Full	0.823	0.0%	70.59

Table 9: Coefficient analysis for Full formulation

Predictor	**Coeff.**	**Std. Error**	**t-value**	**p-value**	**95% CI**
$\log N$	-0.087	0.075	-1.158	0.257	[-0.241, 0.067]
$\log D$	0.580	0.049	11.863	$<$0.001	[0.480, 0.680]

G Loss Landscape Convexity

Experiments are carried out on 18 unique model architectures, with their precise parameter settings summarized in Appendix 5. For each model, the Smooth Loss metric is derived through a grid search over the hyperparameters of learning rate and batch size. Visualization of the smooth loss landscapes substantiates the claims in the main text, demonstrating a consistent convex relationship between the hyperparameters (learning rate and batch size) and the training loss across all models. These findings confirm the robustness and stability of the optimization process as asserted in the primary discussion.

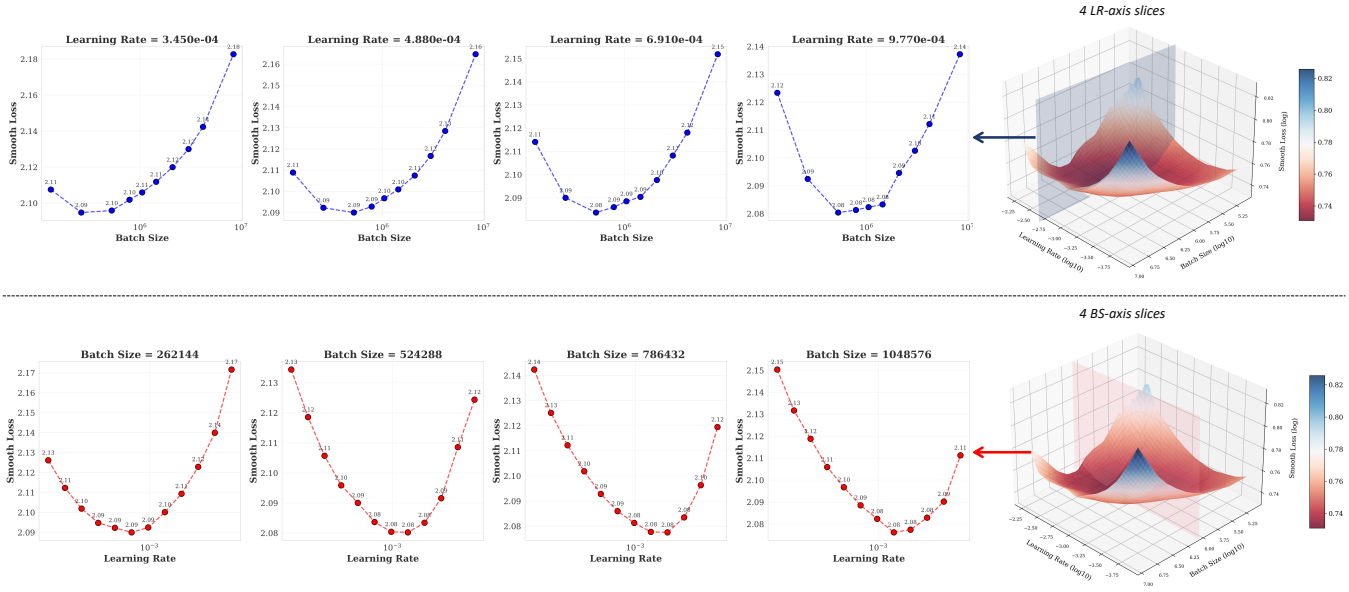
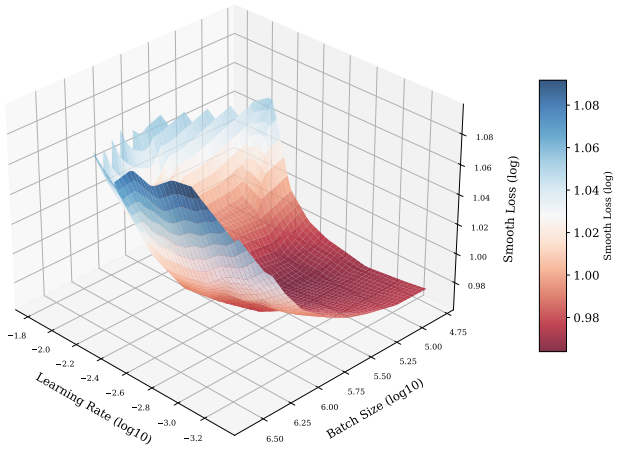
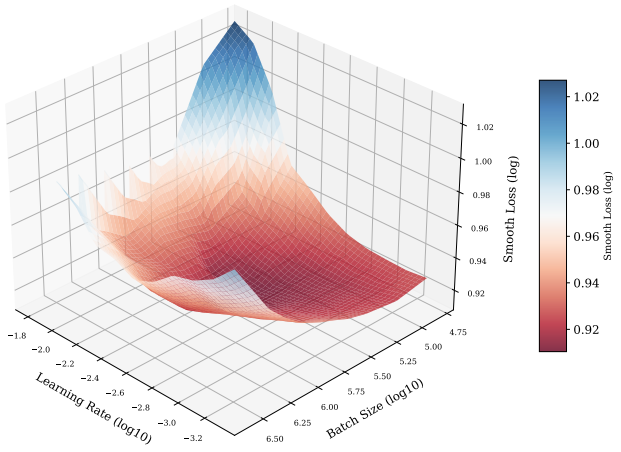


Figure G: Learning Rate vs. Batch Size Loss Landscape Analysis for 1B Model (Trained on 100B Tokens): Scatter Plots and 3D Surface Visualizations of Hyperparameter Sensitivity.

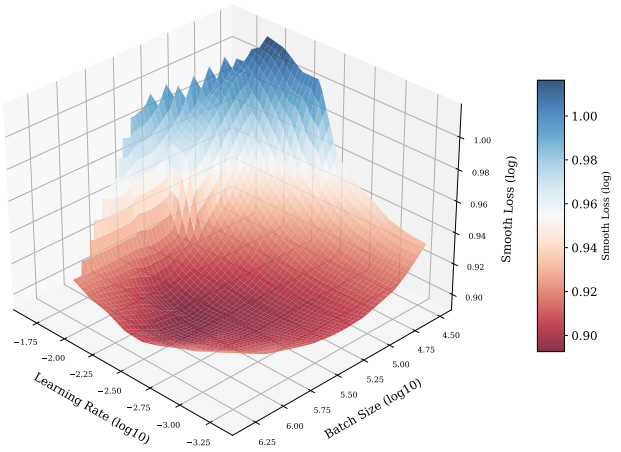


(a) Model 1

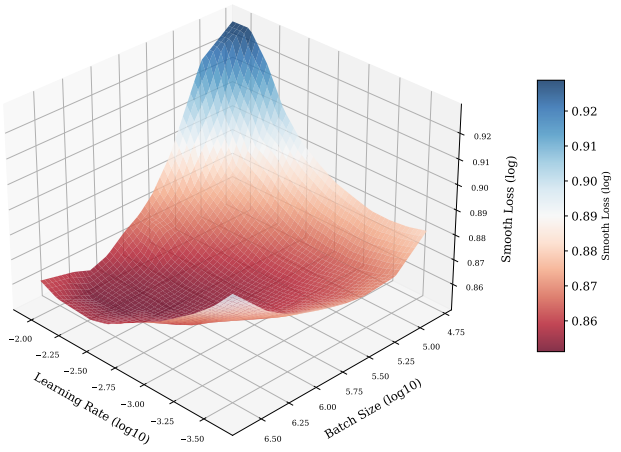


(b) Model 2

Figure 9: Illustration of Hyperparameter Configuration Spaces for Models 1 and 2.

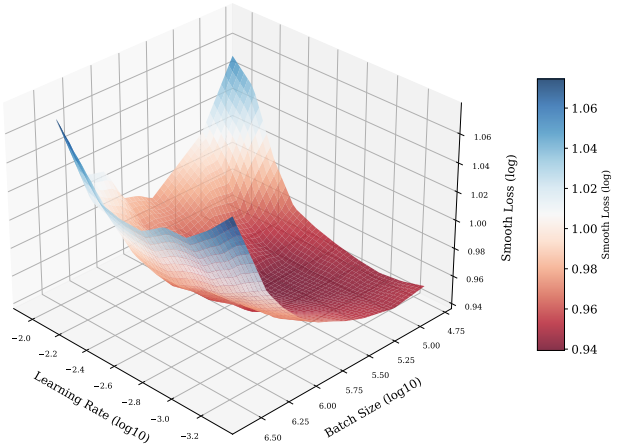


(a) Model 3

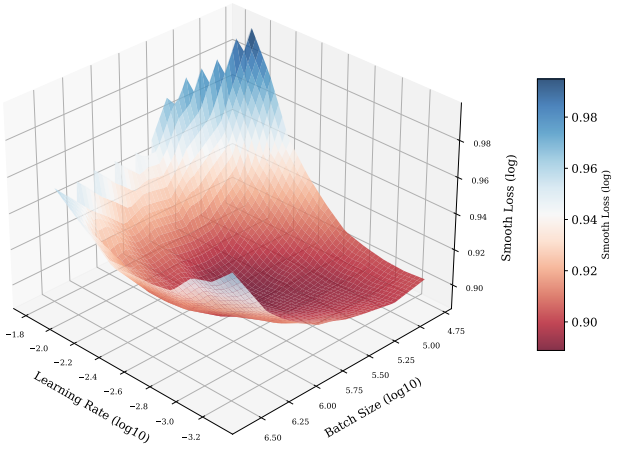


(b) Model 4

Figure 10: Illustration of Hyperparameter Configuration Spaces for Models 3 and 4.

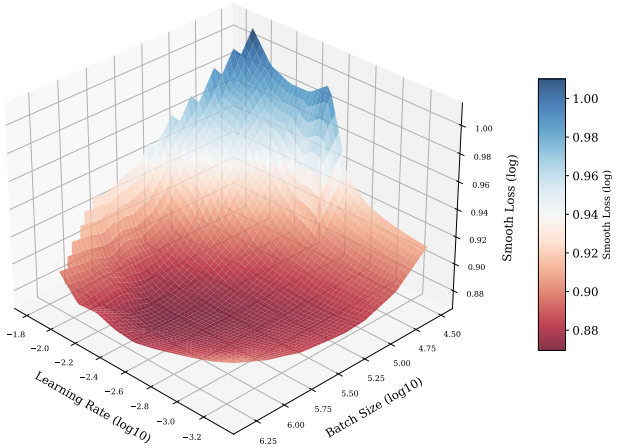


(a) Model 5

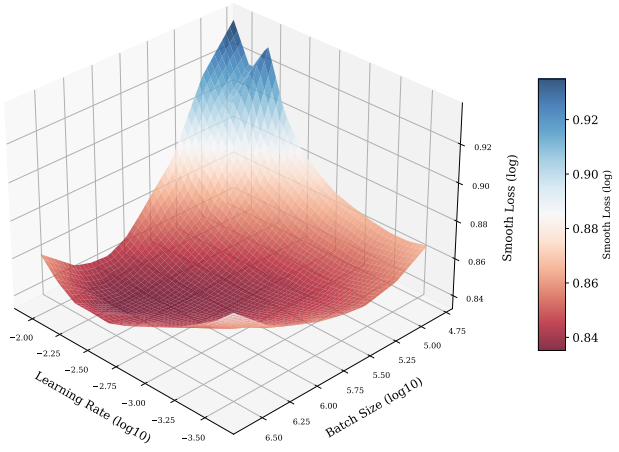


(b) Model 6

Figure 11: Illustration of Hyperparameter Configuration Spaces for Models 5 and 6.

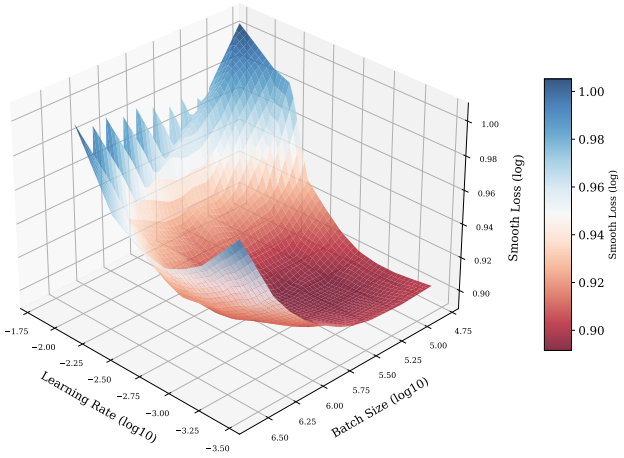


(a) Model 7

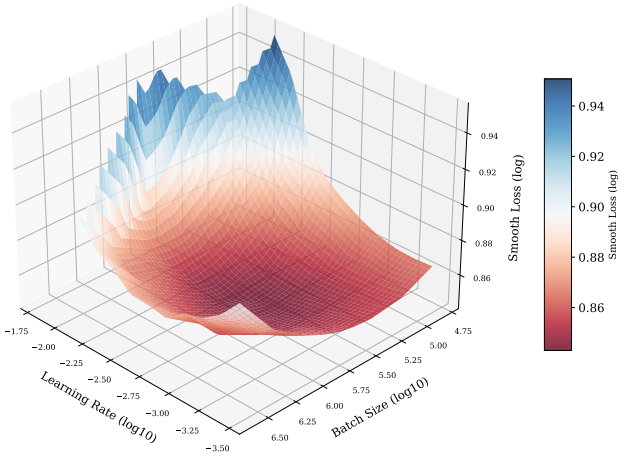


(b) Model 8

Figure 12: Illustration of Hyperparameter Configuration Spaces for Models 7 and 8.

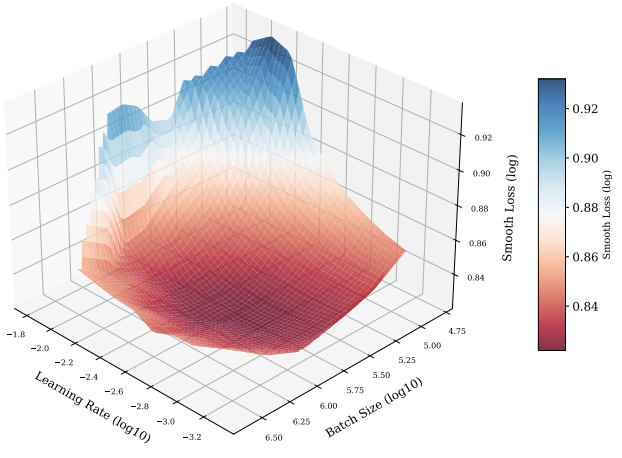


(a) Model 9

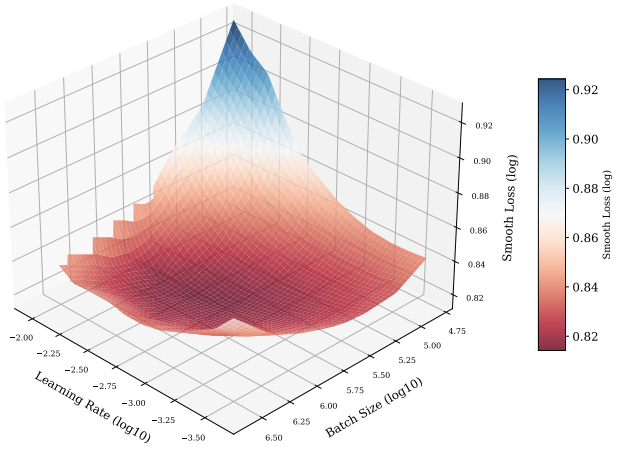


(b) Model 10

Figure 13: Illustration of Hyperparameter Configuration Spaces for Models 9 and 10.

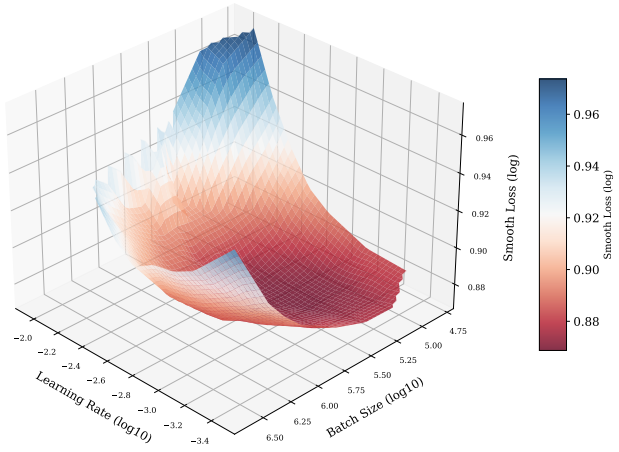


(a) Model 11

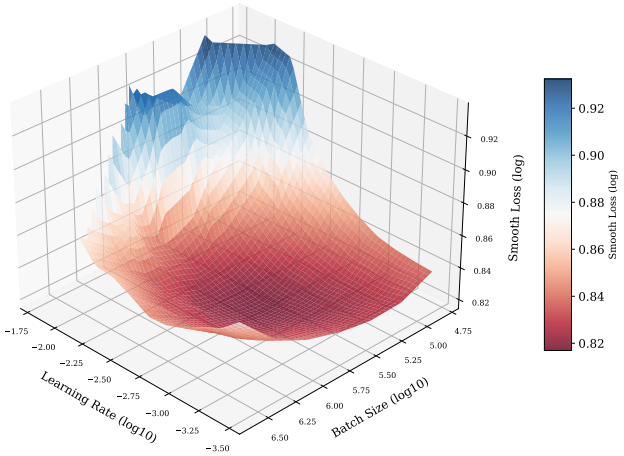


(b) Model 12

Figure 14: Illustration of Hyperparameter Configuration Spaces for Models 11 and 12.

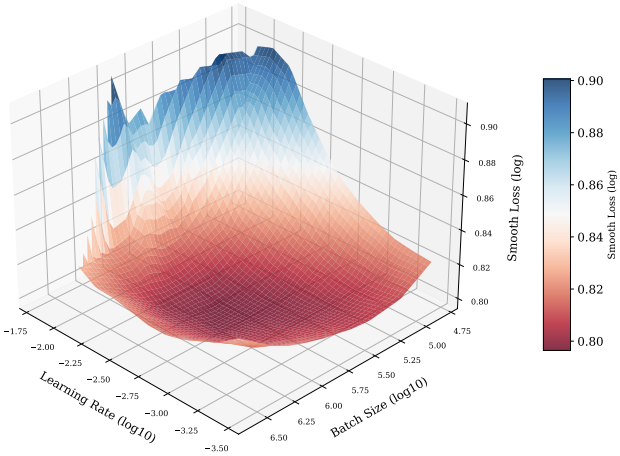


(a) Model 13

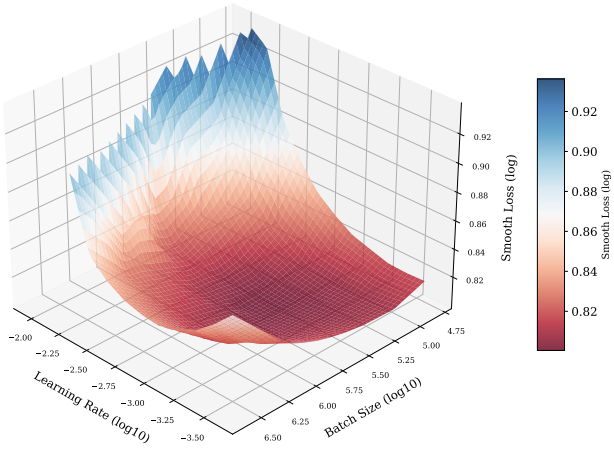


(b) Model 14

Figure 15: Illustration of Hyperparameter Configuration Spaces for Models 13 and 14.

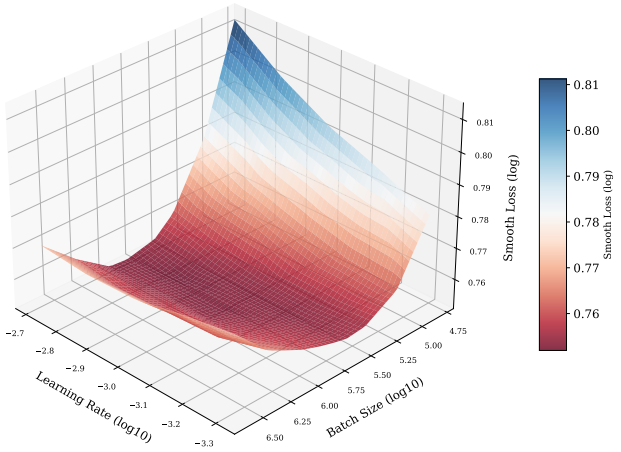


(a) Model 15

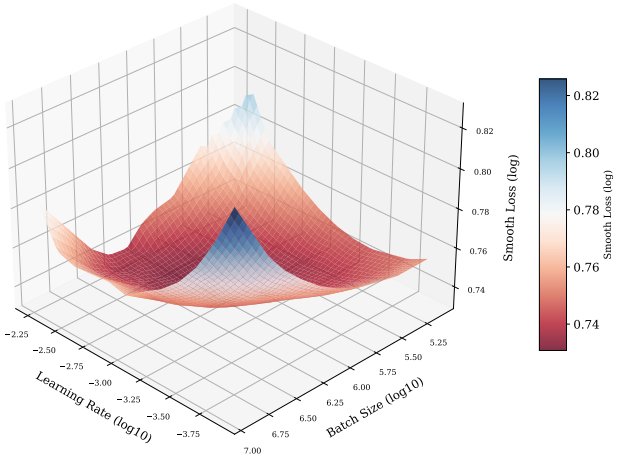


(b) Model 16

Figure 16: Illustration of Hyperparameter Configuration Spaces for Models 15 and 16.



(a) Model 17

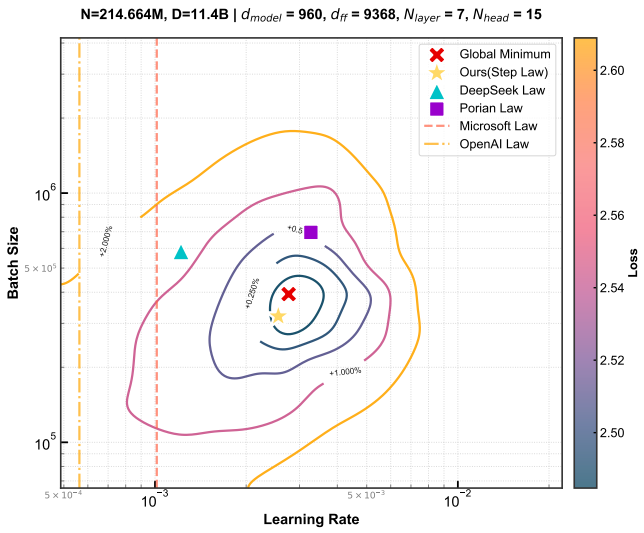


(b) Model 18

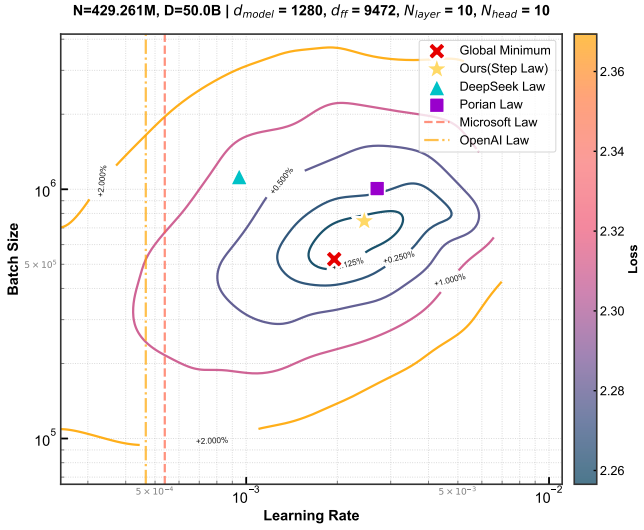
Figure 17: Illustration of Hyperparameter Configuration Spaces for Models 17 and 18.

H Dense Models

This section presents the full set of hyperparameter configuration space visualizations for the 18 dense Transformer models described in Table 5. Each plot illustrates the validation loss surface as a function of learning rate (LR) and batch size (BS), using a log–log scale for both axes. These visualizations reveal consistent trends across model scales, including the emergence of convex, bowl-shaped minima and smooth shifts in optimal hyperparameter regions. They serve as empirical evidence for the predictive structure captured by the scaling law, and demonstrate its robustness across a wide range of dense model sizes.

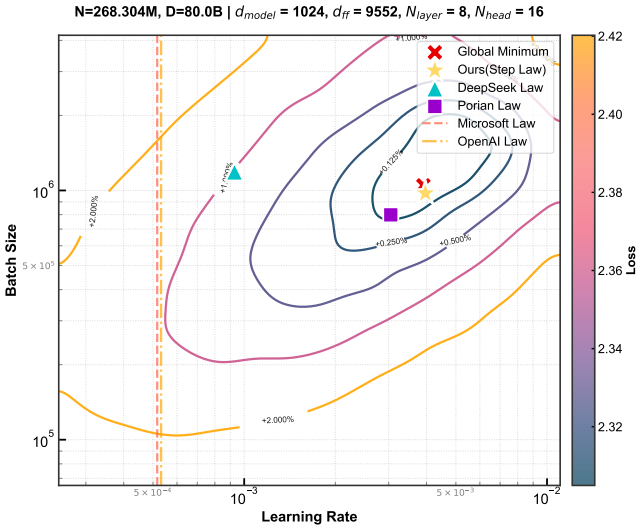


(a) Model 1

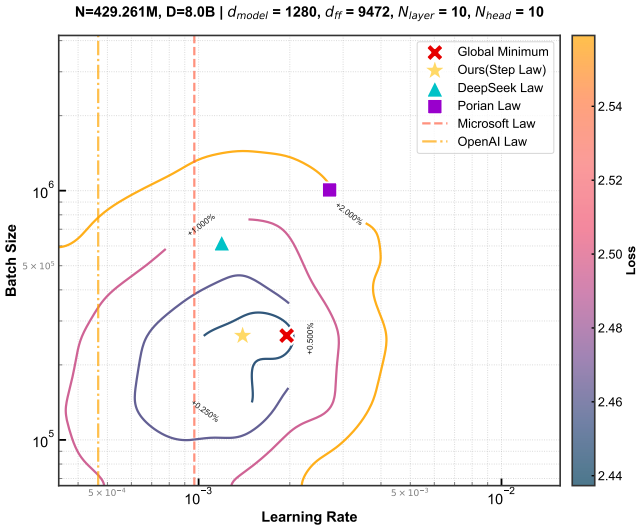


(b) Model 2

Figure 18: Illustration of Hyperparameter Configuration Spaces for Models 1 and 2.

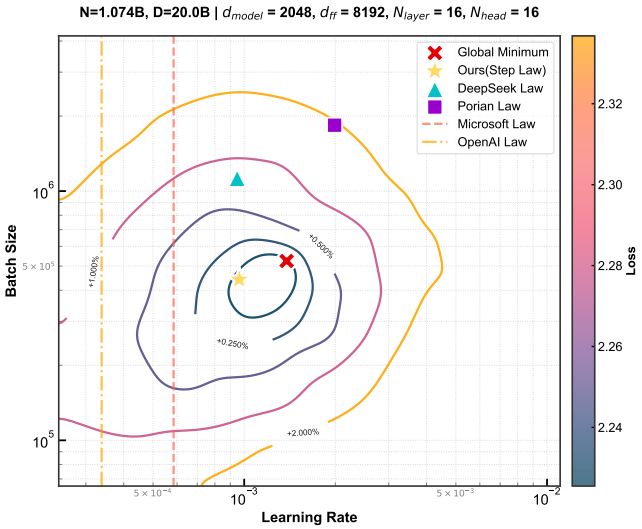


(a) Model 3

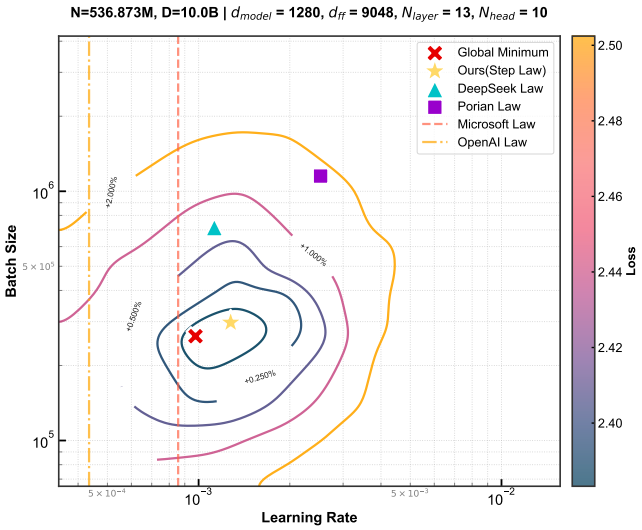


(b) Model 4

Figure 19: Illustration of Hyperparameter Configuration Spaces for Models 3 and 4.

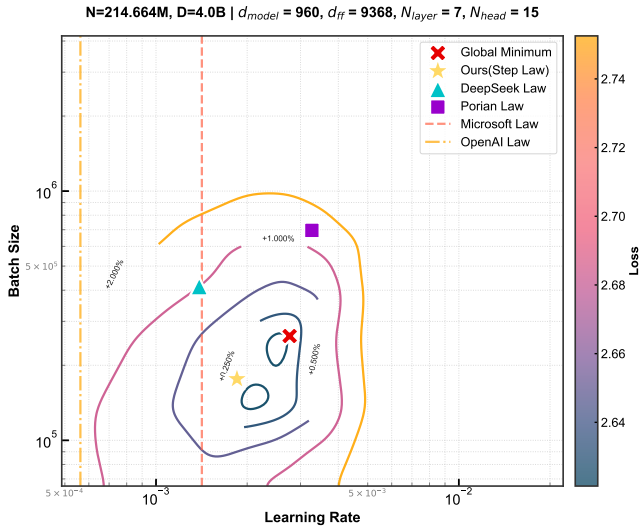


(a) Model 5

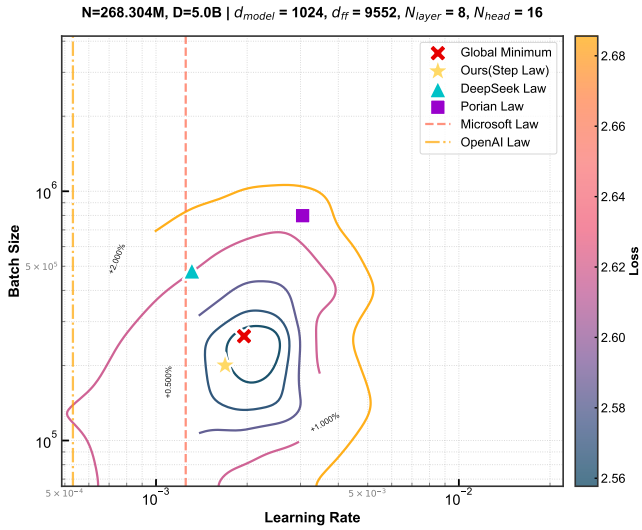


(b) Model 6

Figure 20: Illustration of Hyperparameter Configuration Spaces for Models 5 and 6.

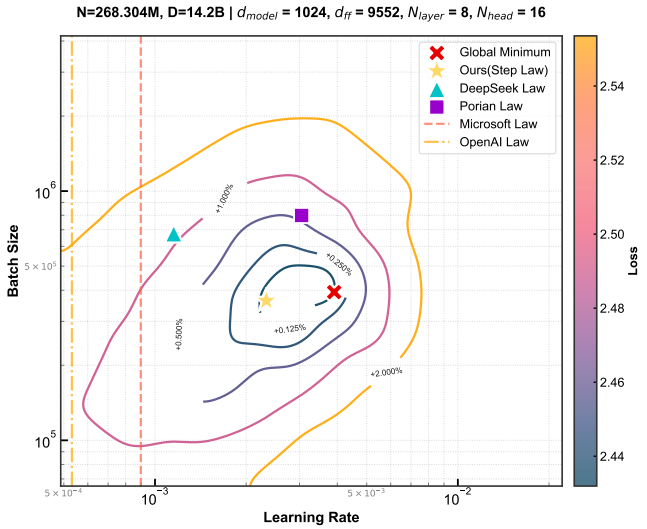


(a) Model 7

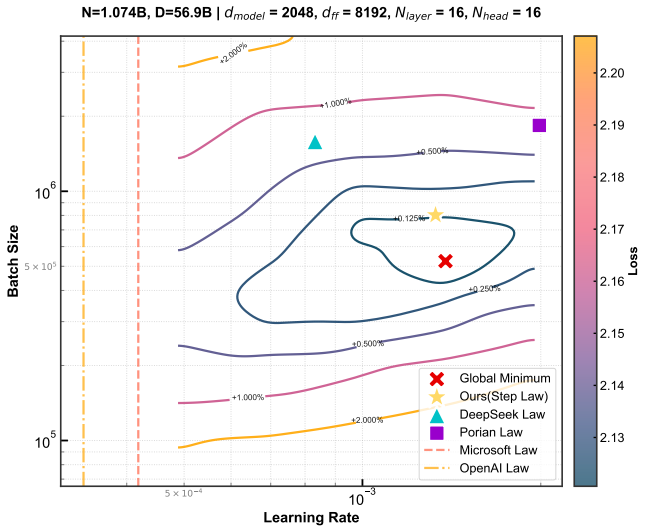


(b) Model 8

Figure 21: Illustration of Hyperparameter Configuration Spaces for Models 7 and 8.

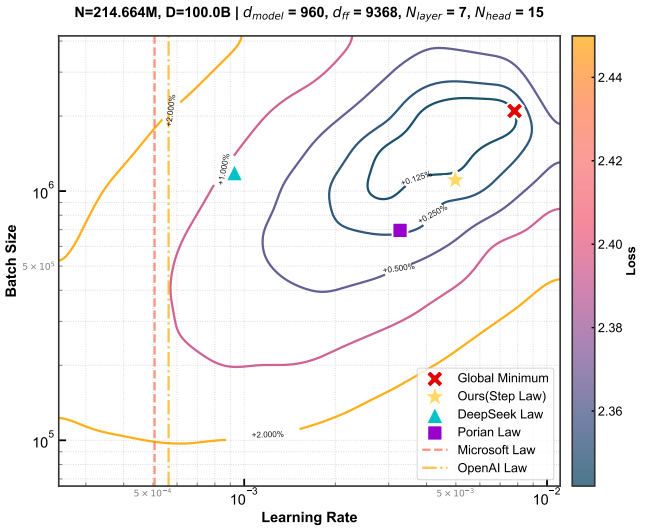


(a) Model 9

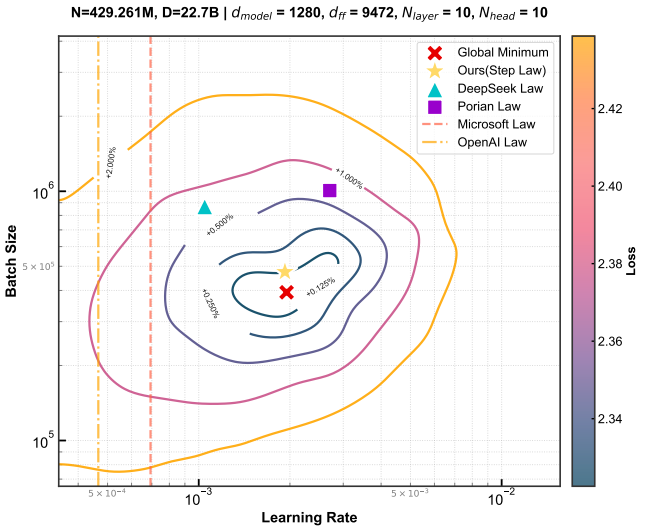


(b) Model 10

Figure 22: Illustration of Hyperparameter Configuration Spaces for Models 9 and 10.

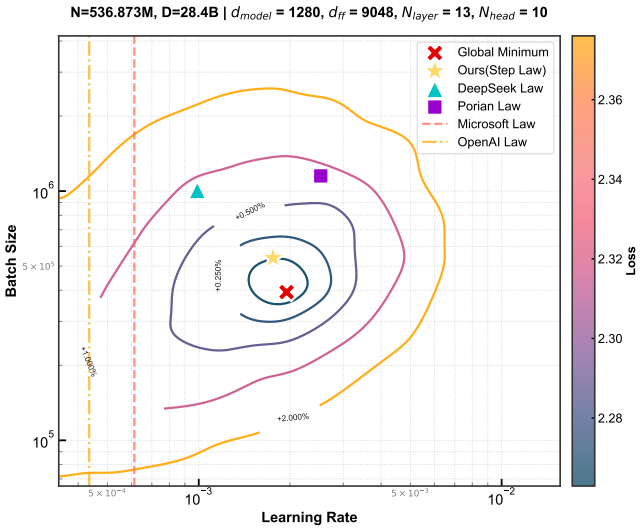


(a) Model 11

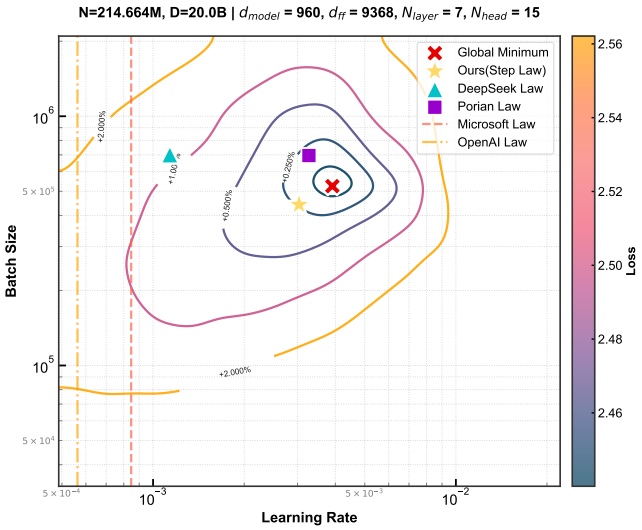


(b) Model 12

Figure 23: Illustration of Hyperparameter Configuration Spaces for Models 11 and 12.

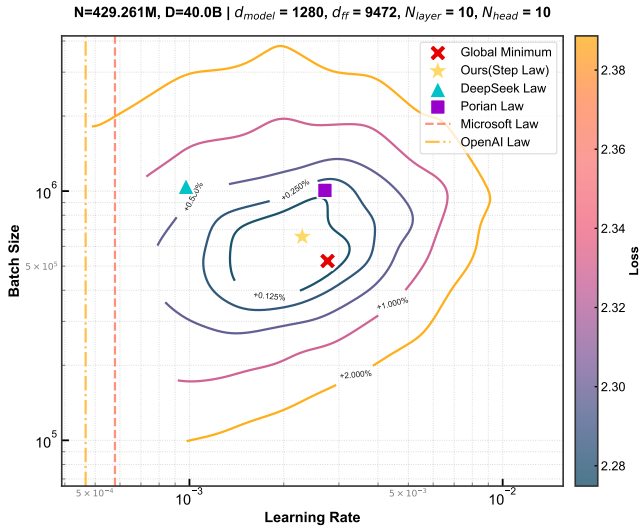


(a) Model 13

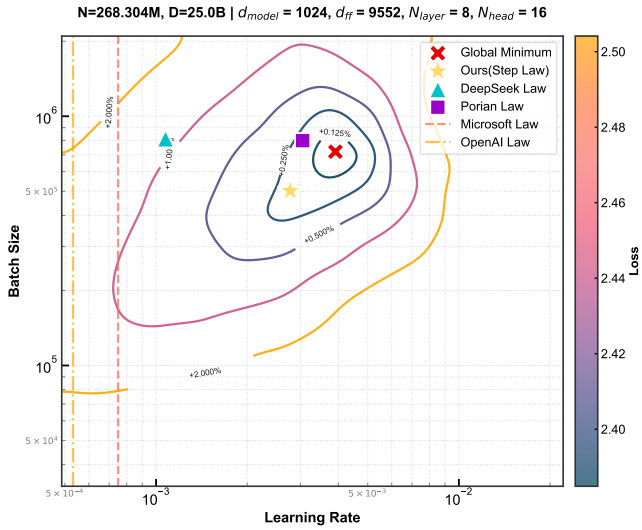


(b) Model 14

Figure 24: Illustration of Hyperparameter Configuration Spaces for Models 13 and 14.

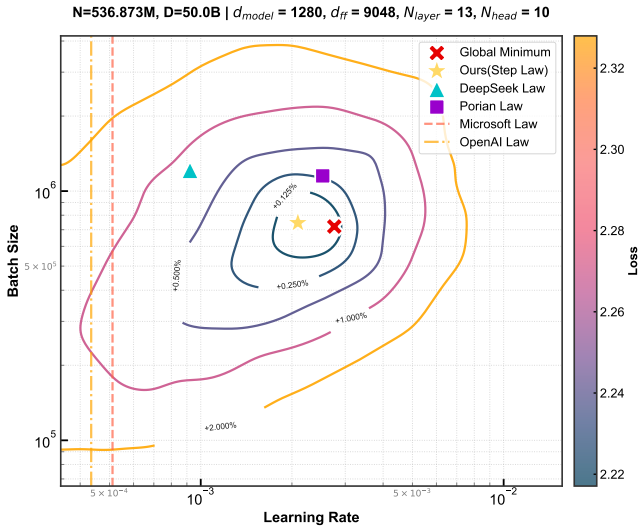


(a) Model 15

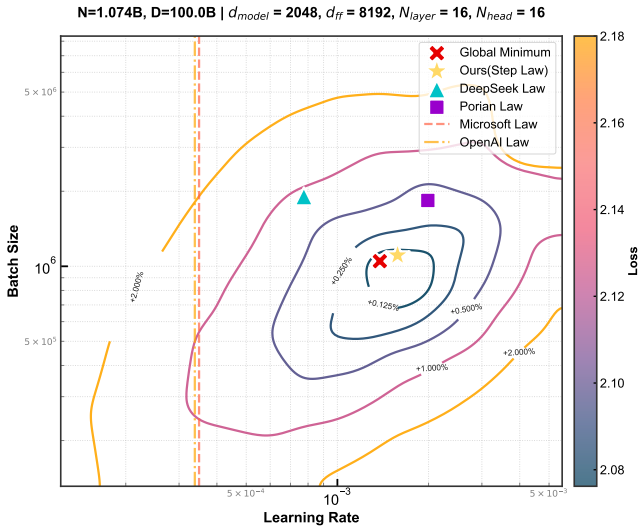


(b) Model 16

Figure 25: Illustration of Hyperparameter Configuration Spaces for Models 15 and 16.



(a) Model 17

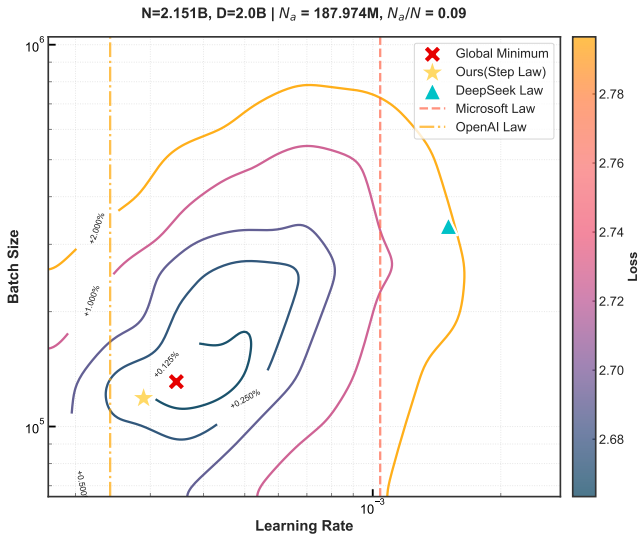


(b) Model 18

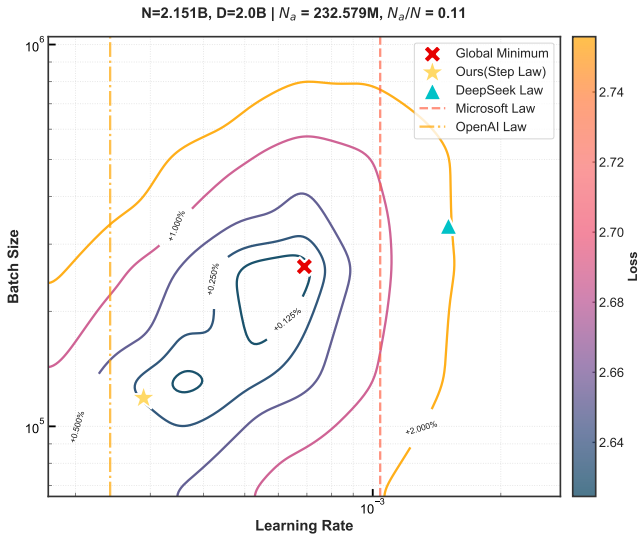
Figure 26: Illustration of Hyperparameter Configuration Spaces for Models 17 and 18.

I MoE Models

To assess the generality of HP scaling laws beyond dense Transformers, we conduct a comprehensive study on MoE models, which activate only a subset of experts per token. We evaluate 16 distinct configurations (see Table 6), varying both total parameter count and sparsity. For each model, we sweep learning rate (LR) and batch size (BS) over the same logarithmic grid used in our dense experiments.

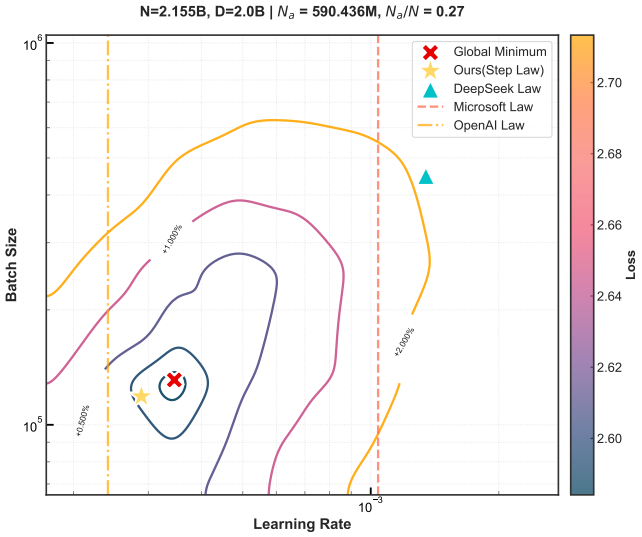


(a) MoE Model 1

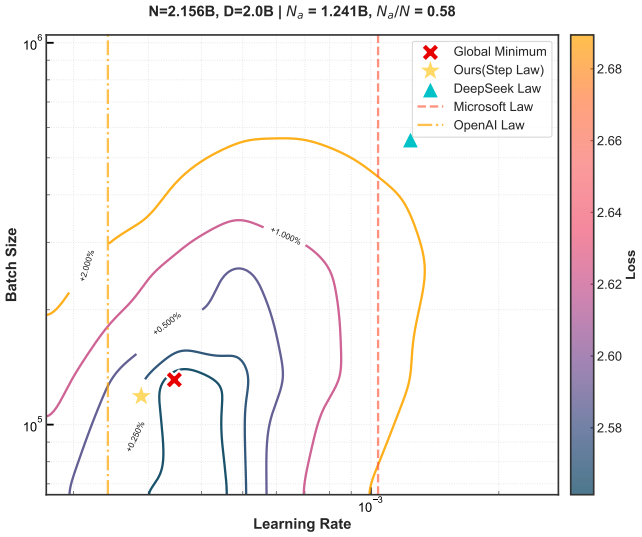


(b) MoE Model 2

Figure 27: Illustration of Hyperparameter Configuration Spaces for MoE Models 1 and 2.

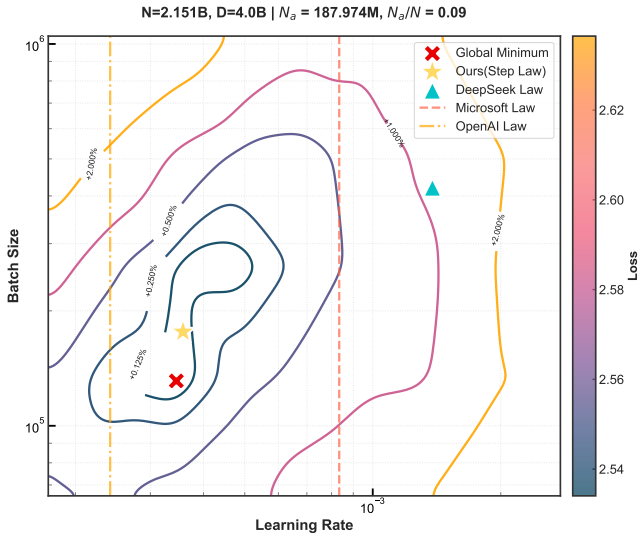


(a) MoE Model 3

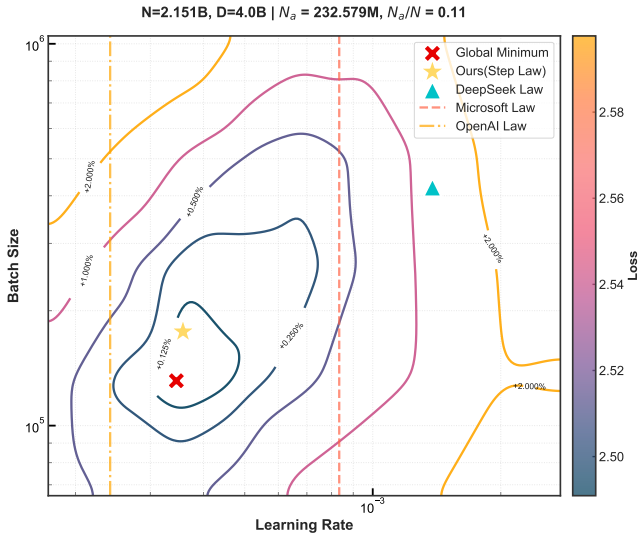


(b) MoE Model 4

Figure 28: Illustration of Hyperparameter Configuration Spaces for MoE Models 3 and 4.

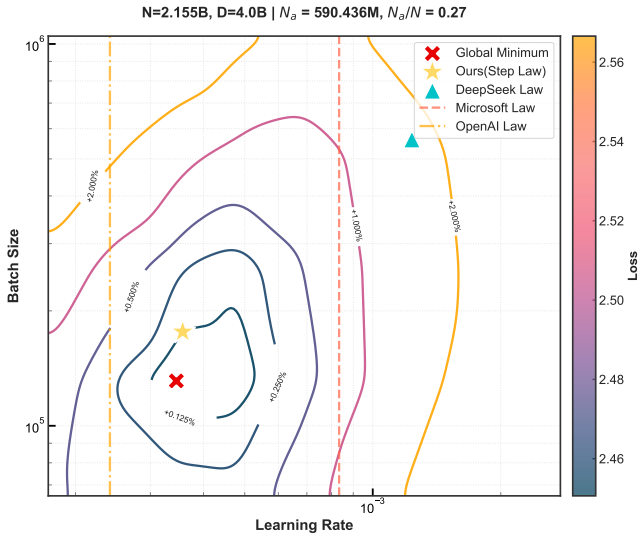


(a) MoE Model 5

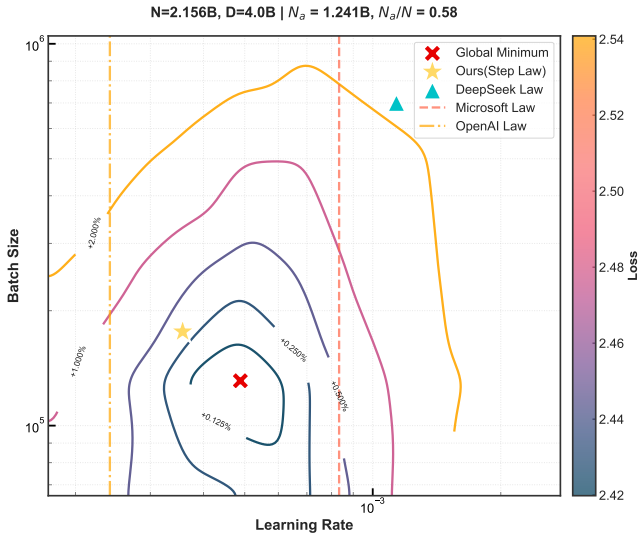


(b) MoE Model 6

Figure 29: Illustration of Hyperparameter Configuration Spaces for MoE Models 5 and 6.

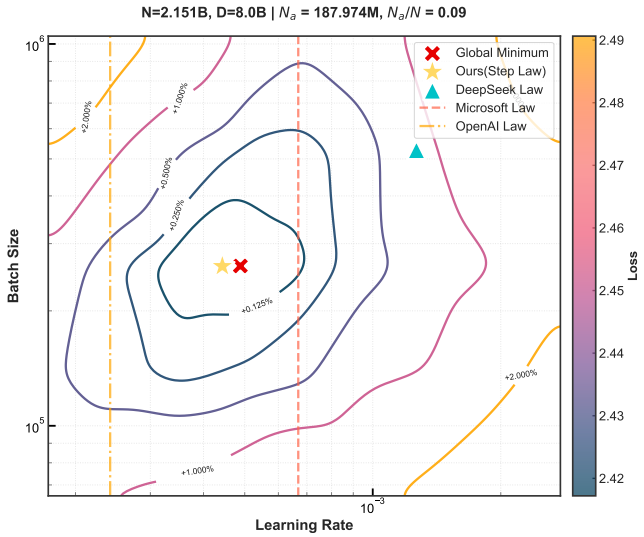


(a) MoE Model 7

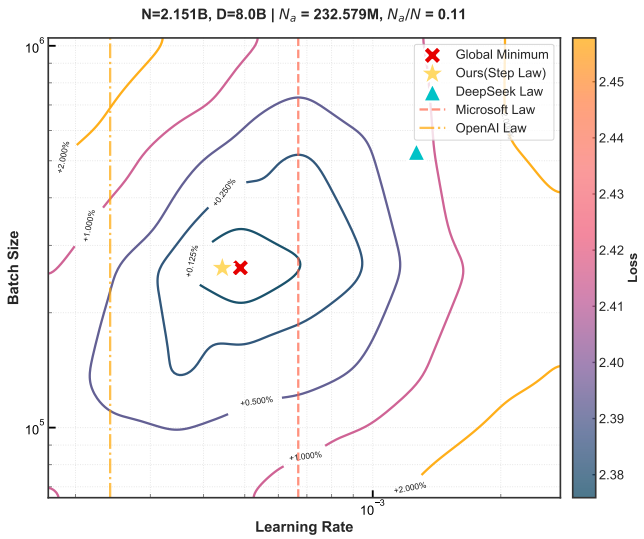


(b) MoE Model 8

Figure 30: Illustration of Hyperparameter Configuration Spaces for MoE Models 7 and 8.

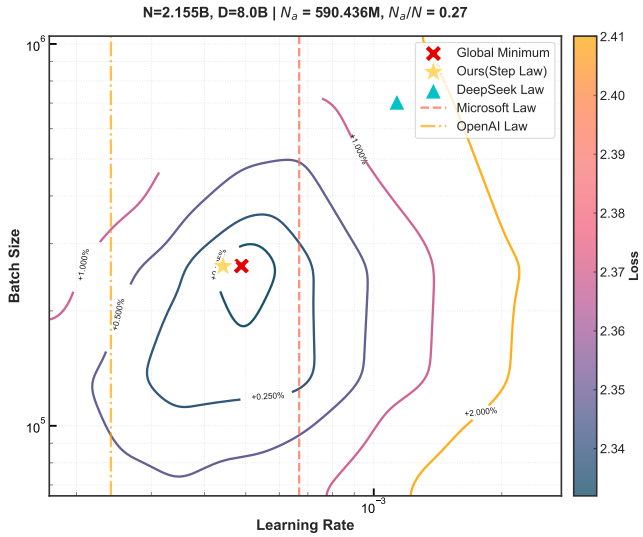


(a) MoE Model 9

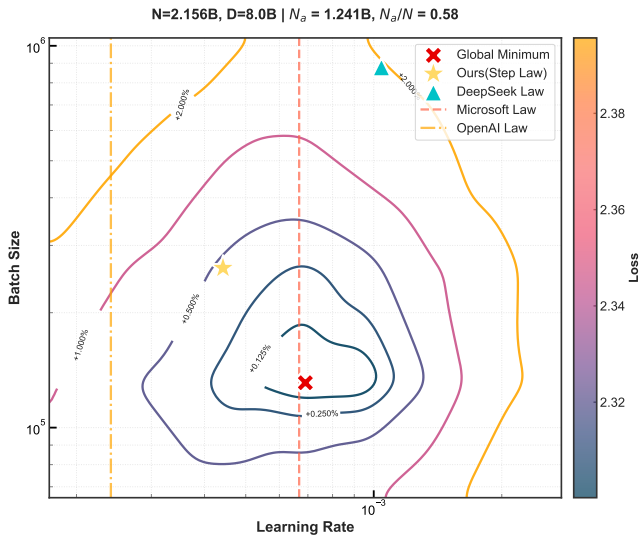


(b) MoE Model 10

Figure 31: Illustration of Hyperparameter Configuration Spaces for MoE Models 9 and 10.

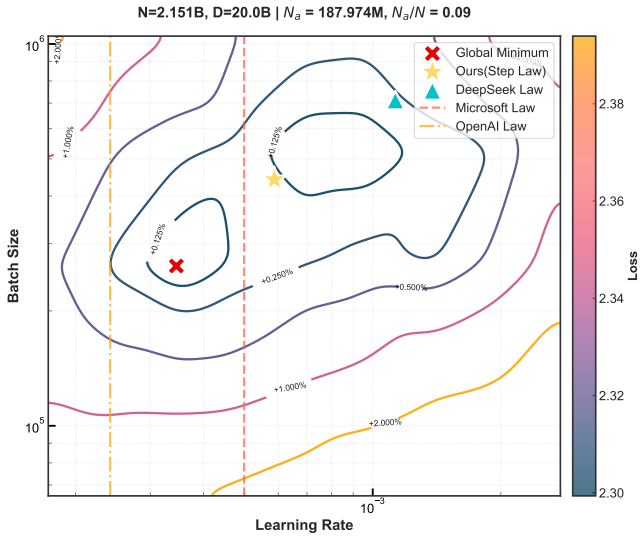


(a) MoE Model 11

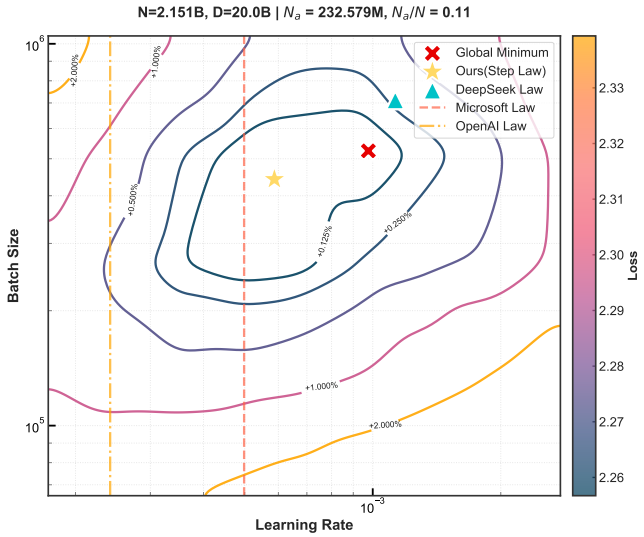


(b) MoE Model 12

Figure 32: Illustration of Hyperparameter Configuration Spaces for MoE Models 11 and 12.

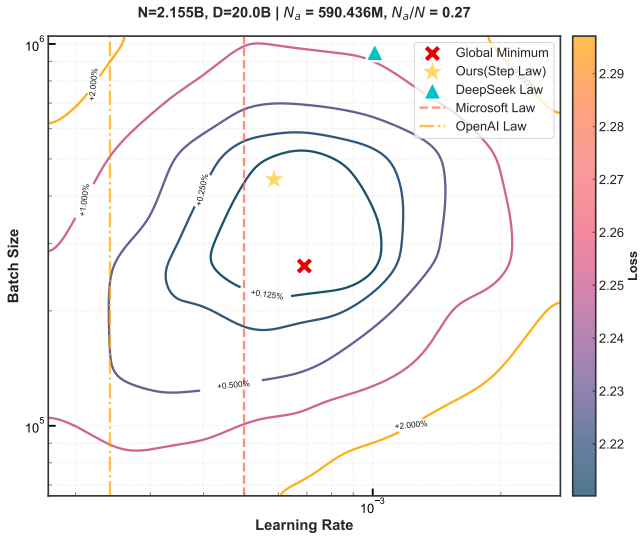


(a) MoE Model 13

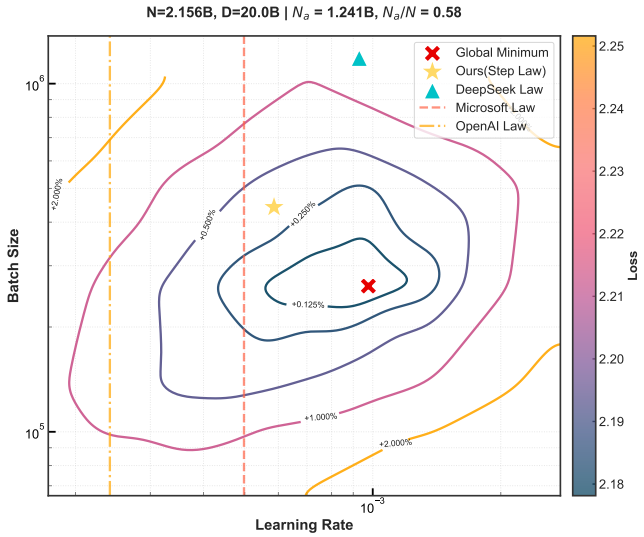


(b) MoE Model 14

Figure 33: Illustration of Hyperparameter Configuration Spaces for MoE Models 13 and 14.



(a) MoE Model 15



(b) MoE Model 16

Figure 34: Illustration of Hyperparameter Configuration Spaces for MoE Models 15 and 16.

J Limitations

While our empirical study provides valuable generalizable HP scaling laws and demonstrates their practical efficacy, it is essential to acknowledge the limitations inherent in an empirical approach. Our findings are primarily data-driven. Future work should focus on developing a more theoretical understanding of the observed power-law relationships, potentially deriving them from first principles to enhance their predictive power and generalizability beyond the empirically validated domain.

Loss Landscape Convexity Analysis Through extensive empirical analysis, we identify a key property of the loss landscape with respect to hyperparameters: both the learning rate and batch size exhibit convex relationships with the training loss under fixed model parameters and dataset size conditions. Fig. G illustrates a representative experimental result, with comprehensive results further elaborated in Appendix G.

Furthermore, we observe that the loss surface demonstrates a stable region around the optimal configuration, evidenced by the plateau-like behavior shown in Fig. 3. This stability provides practical tolerance for small deviations in hyperparameter selection while maintaining near-optimal performance. These properties form the empirical foundation for our subsequent development of scaling laws and validate their applicability across different architectural configurations.