Configuration-to-Performance Scaling Law with Neural Ansatz

Anonymous authors
Paper under double-blind review

Abstract

Researchers build scaling laws to forecast the training performance of expensive large-scale runs with larger model size N and data size D. These laws assume that other training hyperparameters are optimally chosen, which can require significant effort and, in some cases, be impossible due to external hardware constraints. To improve predictability across a broader set of hyperparameters and enable simpler tuning at scale, we propose learning a *Configuration-to-Performance Scaling Law* (CPL): a mapping from the *full training configuration* to training performance. Because no simple functional form can express this mapping, we parameterize it with a large language model (LLM), and fit it with diverse open-source pretraining logs across multiple sources, yielding a *Neural* Configuration-to-Performance Scaling Law (NCPL). NCPL accurately predicts how training configurations influence the final pretraining loss, achieving 20-40% lower prediction error than the configuration-agnostic Chinchilla law and generalizing to runs using up to $10\times$ more compute than any run in the training set. It further supports joint tuning of multiple hyperparameters with performance comparable to hyperparameter scaling law baselines. Finally, NCPL naturally and effectively extends to richer prediction targets such as loss-curve prediction.

1 Introduction

As pretraining large language models is extremely costly (Liu et al., 2024a; Kimi Team et al., 2025a), it is critical to have predictability of performance before executing the training run with the biggest models. People use training runs of smaller models to build a scaling law—oftentimes a power law—that maps the number of model parameters N and the amount of data (or tokens) D to the predicted pretraining loss (Kaplan et al., 2020; Hoffmann et al., 2022). With an accurate scaling law, researchers predict the pretraining performance for N and D that are larger than those that have been experimented with, and can decide the optimal choice of N and D for a given target compute (which depends on ND). Recent work also extends it to include a few other training hyperparameters, such as the learning rate, as inputs or outputs to facilitate choosing these hyperparameters at scale Luo et al. (2025); Tissue et al. (2024); Xie et al. (2024); DeepSeek-AI et al. (2024); Porian et al. (2024); Li et al. (2025d).

This paper proposes to build a more comprehensive scaling law that accurately maps the **full** training configuration C to performance metrics P , such as the final loss, which we call the *Configuration-to-Performance Scaling Law* (CPL). It addresses limitations of the standard scaling law and provides better predictability. Standard scaling implicitly assumes that all the hyperparameters related to the training algorithms are optimally tuned for the existing runs as well as the hypothetical large-scale runs. However, researchers don't always have resources to tune all hyperparameters, at least not optimally. Thus the scaling law implicitly depends on and varies over the hyperparameter scaling strategy (Everett et al., 2024; Yang et al., 2021; Blake et al., 2025; Li et al., 2025d). Moreover, some hyperparameters cannot be arbitrarily tuned due to hardware considerations. For instance, batch size needs to be large enough to fully utilize a large compute cluster (Shuai et al., 2024; Zhang et al., 2025; McCandlish et al., 2018). In contrast, researchers can use a CPL, which maps out the dependency of the performance metric on all hyperparameters, to predict optimal hyperparameters at scale under any external constraints. This simply involves maximizing the predicted performance metric over the choice of the hyperparameters, and thus is simpler than building scaling laws for each individual hyperparameter.

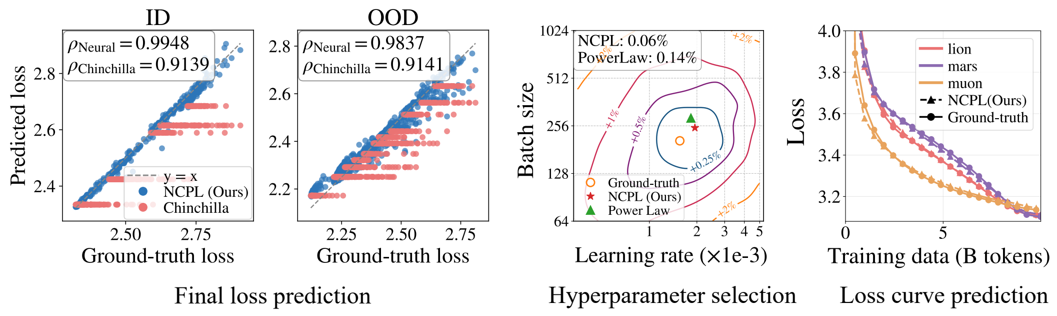


Figure 1: **An Overview of NCPL's Performance Across Tasks.** We split the collected pretraining logs by the model size. In-distribution (ID) means the model size is within the range of the model size in the training set used for NCPL and out-of-distribution (OOD) means the model size is larger. **Left: NCPL predicts final loss more accurately than the Chinchilla Law.** Predicted vs. ground-truth loss on validation sets from the Steplaw dataset. The Chinchilla law yields configuration-agnostic prediction whereas NCPL takes in the full configuration as inputs and therefore achieves better prediction. **Middle: NCPL enables hyperparameter tuning.** Optimal learning rate and batch size prediction in an OOD setup (Steplaw dataset; $N = 536$M, $D = 28.4$B). Configuration-dependent predictions naturally enable joint tuning over these two hyperparameters, achieving comparable performance to hand-designed functional form (Li et al., 2025d). **Right: NCPL can predict the entire loss curve beyond a single loss value.** Predicted vs. ground-truth pretraining loss curves under different optimizers on the validation sets (Marin dataset; $N = 520$M, $D = 10$B). NCPL predicts optimizer-specific curve shapes accurately.

A priori, building a CPL seems to be an overly ambitious goal. The relationship between the configuration and performance is difficult, if not impossible, to have a pre-specified functional form like power law. Thus, we propose to use a *neural ansatz*: a neural network that maps C to P , with parameters learned from data collected across many existing experiments.

It may appear that we won't have sufficient data from expensive training runs to build such a CPL. Fortunately, open-source pretraining studies, such as Marin (Marin, 2025), Step Law (Li et al., 2025d) and OLMo (OLMo Team et al., 2024), have recently released much more diverse public pretraining data. Unlike standard scaling law, CPL can benefit from training on the suboptimal runs (in fact, suboptimal runs are required). Moreover, modern foundation models, as the base models for the training of CPL, may encode prior or theoretical understanding of training dynamics, enhancing transferability and reducing the need for massive run logs.

Encouragingly, we find that training CPL is already feasible with current open pretraining logs and foundation models. We finetune Qwen3-1.7B (Yang et al., 2025) to predict performance metrics, including the final pretraining loss and the full loss curve, from a training configuration using a regression objective. We train on over 3,000 pretraining logs from two open-source projects, Marin (Marin, 2025) and StepLaw (Li et al., 2025d), and obtain a predictor that we call *Neural Configuration-to-Performance Law (NCPL)*. To test the generalization of this method, we split the data into in-distribution (ID) and out-of-distribution (OOD) sets based on model size, and train only on runs with model size below 430M parameters. NCPL generalizes to OOD runs that use up to 10× more compute than any run in the training set (section 3). It improves over classical methods along multiple axes:

1. NCPL learns how configurations affect the final loss, achieving higher accuracy than the Chinchilla scaling law, which only takes N and D as inputs (Figure 1, Left). On the StepLaw dataset, NCPL achieves over 40% lower MAE than Chinchilla, and on the Marin dataset it achieves over 20% lower MAE.

2. NCPL supports joint tuning of multiple hyperparameters. When restricted to learning rate and batch size, NCPL matches the predictive performance of StepLaw (Li et al., 2025d), a hyperparameter scaling law specifically designed for tuning these two hyperparameters (Figure 1, Middle).

3. NCPL can also be extended to predict the full loss curve, not just the final loss. Previously, achieving this typically required hand-designing complex functional forms (Tissue et al., 2024; Luo et al., 2025; Li et al., 2025a) (Figure 1, Right).
4. NCPL qualitatively learns nuanced interactions between hyperparameters, including a rarely noticed interaction between the optimizer choice and weight-decay strength.

As more public training runs are released, the community can build a shared NCPL from pooled data, and users can further fine-tune their own NCPL starting from this shared base. Looking ahead, an NCPL may ingest orders of magnitude more training runs than any individual human researcher, while possessing principled understanding comparable to that of a human researcher through the knowledge in the base model. While an NCPL likely cannot extrapolate to completely unknown scenarios such as runs testing a novel model architecture or a new dataset, one can continue training it with training logs data from the new scenarios, and expect some level of transfer based on the prior knowledge encoded in the network. This may be more data-efficient than the existing approach of building a brand-new scaling law with all new experiments.

2 Preliminaries

Classical scaling laws. Kaplan et al. (2020) observed that the pretraining loss of large language models decreases monotonically and in a predictable manner as the number of model parameters N and the number of training tokens D increase. More specifically, they proposed that the pretraining loss approximately follows a power-law scaling with respect to N and D. Subsequently, Hoffmann et al. (2022) introduced a revised formulation, known as the *Chinchilla law*:

$$\ell_{\text{chinchilla}}(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}}. \tag{1}$$

A key ingredient for accurate scaling-law fitting is proper hyperparameter tuning for each pair (N, D) used to fit the law.

However, the Chinchilla law itself does not provide guidance on how to tune hyperparameters during pretraining, nor does it predict the pretraining loss for suboptimal training configurations. In this work, we address this limitation by modeling the pretraining loss as a function of the full training configuration using a neural network.

Hyperparameter scaling law. Hyperparameter selection plays a crucial role in LLM pretraining. Recent work approaches this problem by fitting parametric functions that map training scale (model and data size) to the optimal choice of hyperparameters (Kaplan et al., 2020; DeepSeek-AI et al., 2024; Bjorck et al., 2025; Porian et al., 2024; Hu et al., 2024; Wang et al., 2024; Li et al., 2025d; Zhou et al., 2026). For instance, Li et al. (2025d) models the optimal learning rate and batch size as power-law functions of model size and data size (with batch size modeled as a function of data size):

$$\eta(N, D) = c\, N^{\alpha} D^{\beta}, \quad B(D) = d\, D^{\gamma}. \tag{2}$$

Such approaches rely on strong inductive assumptions about the functional form of the parametric scaling laws. In contrast, NCPL enables hyperparameter selection without specifying an explicit functional form a priori by modeling the pretraining loss from heterogeneous training logs in a data-driven manner (section 3.3).

3 Methodology

3.1 Formulation

Large language model training outcomes are influenced by a wide range of factors, including model size N, data size D, model architecture, data recipe, optimization algorithms, training hyperparameters, etc. We collectively refer to these factors as the training configuration C. We study how to learn a predictive model that maps configurations C to performance metrics P , such as the final pretraining loss or downstream task scores. We refer to this question as learning a *Configuration–to-Performance Scaling Law (CPL)*.

Classical scaling laws(Kaplan et al., 2020; Hoffmann et al., 2022) can be viewed as a restricted special case of CPL, where the input is limited to only two factors in C, the model size N and data size D, and the relationship is assumed to be a power law (e.g., in eq. (1)), while many other influential factors are left unmodeled. However, pre-specifying a closed-form functional relationship for the entire high-dimensional and heterogeneous configuration space is extremely challenging, if not impossible, due to complex and nonlinear interactions among hyperparameters. Motivated by this perspective, we parameterize the CPL using a generic neural network (specifically, a language model) and train it on a large collection of open-source pretraining runs (Hall et al., 2025; Li et al., 2025d). This yields what we refer to as the *Neural Configuration–Performance Scaling Law (NCPL)*.

3.2 Neural configuration-to-performance scaling law

We now proceed to the concrete design of NCPL, in which we fine-tune a pretrained language model as regressor f_θ to map full training configurations C to training outcomes P .

Input features and output metrics. We use the training configuration of each pretraining run as the input features, including:

1. A source identifier indicating which open-source training project the run comes from, to account for source-specific factors not explicitly represented by other features (e.g., data recipe).
2. Model architecture: model size N (in our case, the number of non-embedding parameters), the number of layers, the number of heads, and the hidden dimension.
3. Data scale: the number of training tokens (D).
4. Optimizer and training hyperparameters: optimizer, peak learning rate, learning-rate schedule, final learning rate after decay, weight decay, batch size, warmup ratio, gradient clipping threshold, and optimizer-specific hyperparameters (e.g., β_1, β_2, and ϵ for AdamW).

An example input instance is provided in fig. 2. In this work, we show that language models used as regressors can leverage these configuration features to make accurate, configuration-aware performance predictions.

```
source:  steplaw
data size:  25.0
model size:  268.0
num layers:  8
num heads:  16
hidden dim:  9552
optimizer:  adamw
learning rate:  0.000977
lr schedule:  cosine
weight decay:  0.1
batch size:  960
max_grad_norm:  1.0
min_lr:  1e-05
max step:  127155
final loss:  0.0235
```

Figure 2: An illustrative training configuration used as input to the model. Numbers are embedded with a two-layer MLP, while other text uses standard token embeddings. The 0.0235 value denotes the target label that the model needs to predict. Note that this number is the residual loss with respect to a Chinchilla baseline (Equation (3)). A full example and additional details are provided in section B.1.

For the main part of the paper, NCPL's target output is the final pretraining loss of each run. However, NCPL naturally extends to other objectives, and we present results in modeling the entire training loss curve in the last paragraph of section 4.2.

Architecture. We parameterize the regressor f_θ with a language model. Compared to training from scratch, our ablation study shows that fine-tuning a pretrained model yields better performance on datasets with diverse configurations. We therefore adopt fine-tuning as our default approach (Qwen3-1.7B as the base model in our experiments (Yang et al., 2025); see section 4.3 and table 1). Given the serialized input sequence x, the model embeds textual and numerical fields differently (fig. 2). Textual fields (including field names and categorical values such as optimizer type and learning-rate schedule) use the backbone language model's standard tokenizer and token embeddings. Numerical values (e.g., N, D, learning rate, and weight decay) are mapped to the model embedding space via a two-layer MLP. We obtain the scalar prediction by applying a linear layer to the last-layer hidden state at the last input position. Our approach is different from text-to-text regression (Akhauri et al., 2025), which parameterizes any numerical values as a sequence of digit tokens.

Training. Let $\mathcal{D}_{\text{train}} = \{(\mathrm{C}^{(i)}, \mathrm{P}^{(i)})\}_{i=1}^{n}$ denote a collection of pretraining runs, where $\mathrm{C}^{(i)}$ is the training configuration of run i, and $\mathrm{P}^{(i)} \in \mathbb{R}$ is the observed final pretraining loss (or an intermediate pretraining loss for the loss-curve prediction task, see section B for details).

Rather than predicting the observed loss $\mathrm{P}^{(i)}$ directly, we train the model to predict the residual relative to a Chinchilla-law baseline (Hoffmann et al., 2022). This design biases learning toward configuration-specific effects beyond the coarse dependence on model size and data size, and empirically improves extrapolation across scales (see ablations in section C.2). Concretely, we first fit a Chinchilla-law baseline $\hat{\ell}_{\text{chinchilla}}(N, D)$ on $\mathcal{D}_{\text{train}}$, which depends only on the number of model parameters N and the number of training tokens D. As described in section 2, for each (N, D), we select the run with the lowest final pretraining loss across different configurations, and fit the Chinchilla-law form (eq. (1)) to this collection of selected runs.[1] We emphasize that the baseline is fit using only the training set, so no information from validation runs is leaked. For each run i, we define the residual regression target as

$$y^{(i)} = \mathrm{P}^{(i)} - \hat{\ell}_{\text{chinchilla}}(N^{(i)}, D^{(i)}). \tag{3}$$

NCPL is trained by minimizing the mean squared error (MSE):

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \left(f_\theta(\mathrm{C}^{(i)}) - y^{(i)} \right)^2 / n.$$

At inference time, we recover the loss prediction by adding the baseline back.

To stabilize training, we adopt a two-stage fine-tuning scheme similar to the LP-FT method (Kumar et al., 2022): in Stage 1 updates only the two-layer MLP encoder for numerical fields and the linear prediction head, and Stage 2 fine-tunes all model parameters.

Evaluation. We report the performance of our NCPL and baselines on in-distribution (ID) and out-of-distribution (OOD) validation sets, where the OOD split consists of runs whose model sizes N exceed those observed during NCPL training. The pretraining runs we collected roughly follow the Chinchilla scaling law, and the data-to-model ratio N/D lies in the range $[1, 8]$. As a consequence, we are testing the extrapolation of NCPL to both larger model size and data size at the same time.

Remark 1 (Rationale for using language model as the regressors) *Previous work has shown that transformer-based foundation models achieve strong performance as generic regressors (see section 5). Two key advantages are their flexibility in handling heterogeneous inputs (e.g., numerical and categorical features) and their ability to leverage large-scale pretraining on diverse data sources. Moreover, the same Transformer backbone readily supports richer prediction targets (e.g., downstream-task performance prediction) and continual fine-tuning. We provide ablation studies against classical tabular baselines such as XGBoost in section 4.3.*

[1] When training on multiple sources, we fit the baseline separately per source.

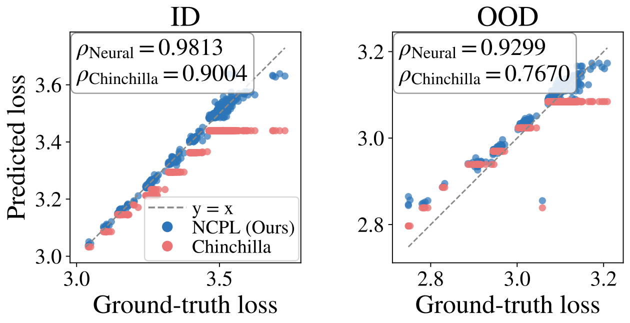


Figure 3: **Predicted loss vs. ground-truth loss.** Each point visualizes the predicted vs. ground-truth final pretraining loss of an individual run from the Marin dataset (for StepLaw, see fig. 1 left). NCPL uses the full training configuration as input, whereas the Chinchilla law only depends on (N, D) and therefore gives the same prediction for all runs sharing the same (N, D). As a result, NCPL achieves substantially higher Spearman correlation ρ.

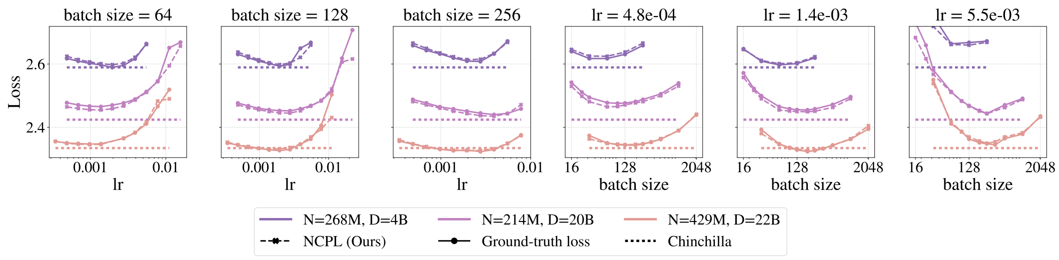


Figure 4: **Predicted vs. ground-truth loss across hyperparameters.** Predicted and ground-truth losses are shown across different learning rates and batch sizes for three held-out (N, D) pairs from the StepLaw dataset. Across different N and D, NCPL accurately predicts how training hyperparameters modulate the final loss, whereas the Chinchilla law yields a single configuration-agnostic prediction for each (N, D) pair.

3.3 NCPL FOR HYPERPARAMETER SELECTION

NCPL predicts a loss value from the full training configuration. This enables selecting multiple hyperparameters jointly without running expensive training sweeps. Specifically, for a target model size N and data size D, one can enumerate a discrete grid of candidate training configurations and select the configuration that minimizes NCPL's estimated final loss, which serves as a proxy for the true final pretraining loss. Experimental results are provided in section 4.2.

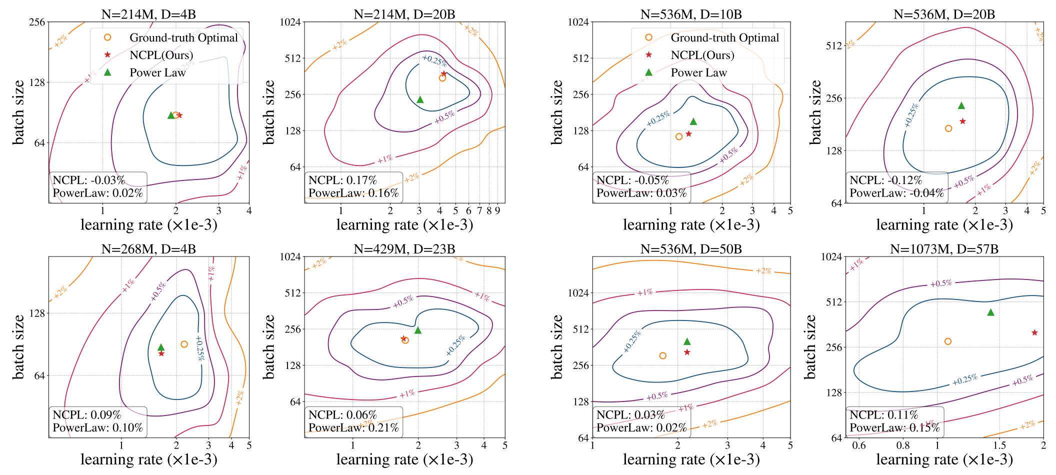
4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Dataset. We train and evaluate NCPL on pretraining logs collected from two open-source pretraining projects, from which we construct the *Marin Dataset* and the *Steplaw Dataset*, respectively.

1. **Marin Dataset.** Collected from the Marin Fantastic Optimizers Project (Hall et al., 2025; Wen et al., 2025), which systematically studies the performance and scalability of different optimizers for language model pretraining. The project conducts extensive hyperparameter sweeps across model sizes ranging from 130M to 1.2B parameters and data scales up to 193B tokens.[2]
2. **Steplaw Dataset.** Collected from the Steplaw Project (Li et al., 2025d). The project performs fine-grained sweeps over learning rate and batch size for models ranging from 215M to 1B pa-

[2]Pretraining logs are available at `https://wandb.ai/marin-community/optimizer-scaling`.



(a) In-distribution (b) Out-of-distribution

Figure 5: **Hyperparameter selection with NCPL.** Predicted optimal learning rates and batch sizes from NCPL and the power-law fitting baseline (eq. (2)) on held-out ID and OOD (N, D) pairs. The contour lines show the true loss landscape, with labels indicating relative losses to the minimum. The bottom legend reports the relative losses of the hyperparameters selected by NCPL and by the power law baseline. NCPL's prediction aligns with the true optima, and achieves comparable losses to the power-law baseline.

rameters and data scales up to 100B tokens, while fixing all other hyperparameters and using the AdamW optimizer (Loshchilov & Hutter, 2019).[3] The extracted training logs contain model size (with architectural specifications such as number of layers, attention heads, and hidden dimension), data size, learning rate, batch size, pretraining loss.

After excluding runs that are unstable, non-converged, or accidentally terminated (details in section B.1), we obtain the Marin dataset and Steplaw dataset of 2549 and 2581 training logs respectively. We designate all runs with model sizes larger than 430M parameters as an out-of-distribution (OOD) validation set. The remaining runs are split into training and in-distribution (ID) validation sets with an 8:2 ratio. To make the splits between training set and ID validation set more meaningful and challenging, we randomly split on the level of *(optimizer, model size, data size)* tuples. Concretely, we first group runs that share the same (optimizer, N, D) tuple, and then randomly assign each group to either the training set or the ID validation set. This strategy makes sure that every run in the ID validation set (or the OOD validation set) does not have another run with the same optimizer, model size, data size in the training dataset. In total, the dataset comprises 3,225 training runs, 796 ID validation runs, and 1,109 OOD validation runs.

Model and fine-tuning details. We use Qwen3-1.7B as the base model (Yang et al., 2025) and adopt the 2-stage LP-FT training described in Section 3.2. We find that training with multiple passes of data improves generalization, and therefore train for 20 epochs in Stage 1 and 1000 epochs in Stage 2 on the training set. More details are provided in section B.2.

4.2 Main Results

Configuration-dependent final loss prediction We first examine NCPL's ability to predict the final pretraining loss based on the full training configuration. In fig. 3, each point shows the predicted versus ground-truth final pretraining loss for a single training run of a particular training configuration. NCPL takes the full training configuration of each run as the input, whereas the Chinchilla Law baseline yields a single configuration-agnostic prediction for each model-data size pair. Consequently, NCPL aligns much more closely with the ground truth, achieving lower prediction errors and higher Spearman correlations on both ID and OOD validation sets, as summarized in table 1.

[3]Pretraining logs are available at `https://wandb.ai/billzid/predictable-scale`.

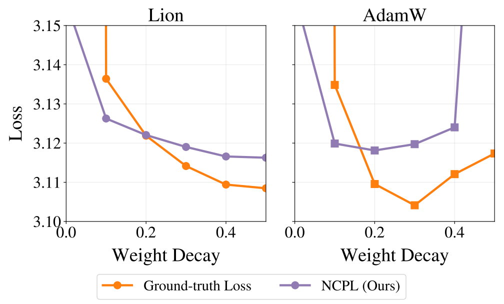


Figure 6: **NCPL learns interactions between weight decay and optimizer choice.** It's known that Lion requires substantially larger weight decay than AdamW (Chen et al., 2023; Loshchilov & Hutter, 2019; Wen et al., 2025). NCPL predicts this phenomenon on the OOD validation sets (Marin Dataset, N=520M, D=10B).

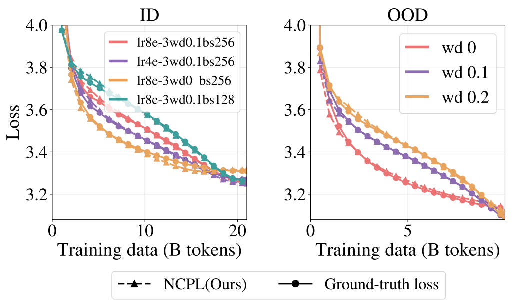


Figure 7: **Loss curve prediction.** Ground-truth and predicted pretraining loss curves under different hyperparameter settings on the ID and OOD validation sets. NCPL closely tracks the overall trajectories and learns hyperparameter-specific curve shapes. Setting: Marin Dataset. **Left:** ID validation, $N = 130\text{M}$, $D = 21\text{B}$, AdamW with varying learning rate, weight decay, and batch size. **Right:** OOD validation, $N = 520\text{M}$, $D = 10\text{B}$, Muon with learning rate 8×10^{-3} and varying weight decay. Results under different optimizers are shown in figs. 1 and 15.

Figure 4 further visualizes fine-grained predictions over learning rate and batch size sweeps for held-out in-distribution model-data size pairs from the StepLaw dataset. NCPL closely tracks the ground-truth U-shaped loss curves across hyperparameters, illustrating its ability to learn how training hyperparameters affect the final loss.

Hyperparameter selection. Hyperparameter selection plays a crucial role in large-scale language model pretraining and is a main application of modeling Configuration-to-Performance scaling laws. With our learned NCPL, one can sweep over candidate training configurations and use NCPL's estimated loss as a proxy for the true pretraining loss, thereby identifying optimal hyperparameters without performing expensive training sweeps.

In this section, we evaluate this capability on held-out model-data size pairs from the StepLaw dataset. We sweep over NCPL's predictions to select the optimal learning rate and batch size, and compare its predictions with the power-law scaling rule proposed in Li et al. (2025d) (eq. (2)). As shown in Figure 5, the hyperparameters predicted by NCPL closely match the true optima obtained from exhaustive sweeps, in both in-distribution and out-of-distribution settings. The prediction error is comparable to that of the fitted power-law baseline. Additional results are provided in section C.1.

We observe that, in the out-of-distribution setting, the neural predictor tends to suggest larger learning rates, especially for model-data pairs that are farther from the training distribution (e.g., Figure 5b, N=1073M, D=57B). We hypothesize that this behavior is because NCPL biases towards its training set of small models which use larger learning rates, and discuss it further in Section 6.

Characterizing interactions between training configurations Different hyperparameters in the training configuration do not affect final pretraining loss independently; instead, their effects can interact in complicated ways. For example, different optimizers often prefer different hyperparameter choices (e.g., Liu et al. (2025b); Wen et al. (2025); Marek et al. (2025)). By learning a mapping from full training configurations to pretraining performance, we expect NCPL to learn such interactions from large-scale pretraining logs. Here we illustrate this potential with a concrete example. The Lion optimizer (Chen et al., 2023) is known to require substantially larger weight decay than AdamW (Loshchilov & Hutter, 2019; Wen et al., 2025). As shown in fig. 6, when using larger weight decay (e.g., ≥ 0.4), AdamW's performance deteriorates markedly, whereas Lion improves. NCPL learns this interaction from the training set and generalizes it to the OOD setting.

Table 1: Comparison of various versions of NCPL, XGBoost, and Chinchilla-law baselines for final-loss prediction and loss-curve prediction on both ID and OOD splits. For NCPL, we ablate fine-tuning (*ft*) versus training from scratch (*scratch*) with two backbone sizes (1.7B and 135M). We report mean absolute error (MAE), root mean squared error (RMSE), and Spearman correlation (ρ). Overall, NCPL achieves substantially lower error and higher rank correlation than the Chinchilla-law by leveraging full training configurations. On Steplaw dataset, where only a small set of hyperparameters vary (learning rate and batch size), NCPL (training from scratch) are competitive with NCPL with fine-tuning; in contrast, on Marin dataset, which has more diverse configurations, NCPL with fine-tuning provides a clear advantage over other variants and baselines.

Data	**Method**	**Final-loss prediction**						**Loss Curve Prediction**			
		In-distribution			Out-of-distribution			In-distribution		Out-of-distribution	
		MAE↓	RMSE↓	$\rho\uparrow$	MAE↓	RMSE↓	$\rho\uparrow$	MAE↓	RMSE↓	MAE↓	RMSE↓
Marin	NCPL (ft, 1.7B)	**0.0109**	**0.0169**	**0.9813**	**0.0168**	**0.0239**	0.9299	**0.0252**	**0.0475**	**0.0363**	**0.0644**
	NCPL (scratch, 1.7B)	0.0119	0.0171	0.9774	0.0207	0.0301	0.9324	0.0289	0.0554	0.0386	0.0707
	NCPL (scratch, 135M)	0.0140	0.0211	0.9777	0.0266	0.0347	**0.9421**	0.0306	0.0527	0.0376	0.0713
	XGBoost	0.0188	0.0277	0.9754	0.0325	0.0375	0.9227	0.0305	0.0541	0.0385	0.0667
	Chinchilla Law	0.0566	0.0676	0.9004	0.0240	0.0326	0.7670	–	–	–	–
	Per-optimizer Chinchilla	0.0416	0.0575	0.8268	0.0211	0.0278	0.8062	–	–	–	–
Steplaw	NCPL (ft, 1.7B)	0.0097	0.0158	0.9948	0.0223	0.0345	0.9837	0.0278	0.0981	0.0415	0.1696
	NCPL (scratch, 1.7B)	0.0090	0.0150	**0.9956**	0.0225	0.0313	0.9876	0.0258	**0.0941**	0.0384	0.1709
	NCPL (scratch, 135M)	**0.0082**	**0.0147**	0.9953	**0.0199**	**0.0284**	**0.9910**	**0.0265**	0.0943	**0.0376**	**0.1666**
	XGBoost	0.0095	0.0162	0.9947	0.0246	0.0332	0.9863	0.0275	0.1068	0.0471	0.1834
	Chinchilla Law	0.0704	0.0949	0.9139	0.0440	0.0670	0.9141	–	–	–	–

NCPL for loss curve prediction Beyond accepting the full training as inputs, the generality of NCPL also allows us to target other metrics beyond final pretraining loss. We train a variant of NCPL to perform loss curve prediction, and fig. 7 demonstrates the true loss curves and NCPL's predictions from the Marin Dataset. NCPL accurately predicts the loss curve under different hyperparameter choices on both ID and OOD generalization settings. This task previously required hand-designing complex multi-component power laws (Luo et al., 2025; Tissue et al., 2024), and we show that the shapes of loss curves can be learned directly from data. This also resonates with the recent finding on scaling collapse (Qiu et al., 2025), which suggests that the shapes of loss curves are the same across scales up to an affine transformation. Moreover, NCPL can also predict loss curves for different optimizers; see fig. 1 and fig. 15.

4.3 Ablation on the Backbone Architecture of NCPL

In this section, we ablate the effect of the chosen backbone for NCPL, with the results shown in Table 1. We compare NCPL with XGBoost (Chen & Guestrin, 2016) and Chinchilla Law (Hoffmann et al., 2022) baselines on final-loss prediction and loss curve prediction tasks. For NCPL, we further ablate fine-tuning versus training from scratch with different model sizes, using the Qwen3-1.7B (Yang et al., 2025) and SmolLM2-135M (Allal et al., 2025) architectures.

Comparing with XGBoost and Chinchilla Law, NCPL achieves substantially lower prediction error and higher Spearman correlation than the configuration-agnostic Chinchilla-law baseline by leveraging the full training configuration, rather than only (N, D). On the Steplaw dataset, where the configuration variation is restricted to a small set of continuous factors (N, D, learning rate, and batch size), training from scratch can outperform fine-tuning. In contrast, on the Marin dataset, where many heterogeneous fields vary jointly (e.g., optimizer choices and a wide range of hyperparameters), fine-tuning yields a clear advantage over both training from scratch and XGBoost. Therefore, we adopt the fine-tuned NCPL as our main method.

5 Related work

Scaling law. Kaplan et al. (2020); Hoffmann et al. (2022) showed the pretraining loss of the Transformer-based LLMs follow a power-law relation with the model size and data size. Subsequent work explores refined fitting protocols and functional forms (Porian et al., 2024; Li et al., 2025c; Caballero et al., 2023), different model architectures (Ludziejewski et al., 2024; Wang et al., 2024), incorporating other training hyperparameters such as learning rate (Tissue et al., 2024; Xie et al., 2024; Luo et al., 2025; Li et al., 2025a) and loss curve prediction (Luo et al., 2025; Tissue et al., 2024; Qiu et al., 2025). Scaling law is also applied to broader domains and settings, including multi-modal models (Henighan et al., 2020) , hyperparameter optimization (Kadra et al., 2023; Hashimoto, 2021), (Jain et al., 2024), adversarial attacks (Liu et al., 2024b), transfer learning(Hernandez et al., 2021), etc. Recent works extends scaling laws to model downstream tasks performance rather than the pretraining loss while some other works point out practical challenges (Gadre et al., 2025; Ruan et al., 2024; Bhagia et al., 2024; Khatri et al., 2025; Lourie et al., 2025) Scaling laws are also extended to predict the optimal hyperparameters with the scale of training resources (DeepSeek-AI et al., 2024; Li et al., 2025d; Bergsma et al., 2025; Bjorck et al., 2025; Marek et al., 2025; Li et al., 2025b; Zhou et al., 2026). However, these approaches fit only a small subset of hyperparameters, making it difficult to ensure that the remaining hyperparameters stay optimal during fitting or extrapolation; moreover, they do not model non-parametric configuration choices such as the optimizers.

Recently, Lin et al. (2025) proposes to use advanced LLMs in a scaffold to propose the functional form of scaling laws under different setups. However, identifying the correct functional form mapping from the *entire* configuration C to performance P can be extremely difficult due to the complicated interactions of different factors. We adopt a different methodology to directly use LLM as the regressor to predict the performance from training configurations.

Foundation models as regressors. Transformer-based foundation models have demonstrated strong capability in modeling complex input-output relationships beyond natural language processing. Recent work has shown that pretrained language models can be used as general-purpose regressors by direct prompting pretrained models (Vacareanu et al., 2024) or through supervised learning (Garg et al., 2022; Hollmann et al., 2022; Huang et al., 2020). Compared to classical regression approaches, foundation models offer several key advantages, including leveraging the semantic meaning of features to perform feature selection (Jeong et al., 2025), enabling large-scale pretraining on diverse data sources (Hollmann et al., 2025), and supporting online adaptation fine-tuning (Song et al., 2024). At the same time, recent studies have highlighted potential limitations, including sensitivity to data representation that are irrelevant to the underlying learning task (Liu et al., 2025a). Foundation models as regressors have been explored in a wide range of domains, for instance, time-series forecasting (Gruver et al., 2023; Jin et al., 2024; Das et al., 2024; Goswami et al., 2024) and performance prediction of complex engineered systems (Akhauri et al., 2025). Our work focuses on predicting the pretraining outcomes of LLMs from full training configurations. To the best of our knowledge, this setting has not been systematically studied in prior work.

6 Conclusion and Discussion

In this work, we propose to learn the mapping from full training configurations to training outcomes using generic neural networks, by training on large-scale and diverse open-source pretraining logs. We instantiate this idea by fine-tuning a pretrained language model (Qwen3-1.7B (Yang et al., 2025)), resulting in the *Neural Configuration–Performance Scaling Law* (NCPL). Empirically, NCPL accurately predicts how configuration choices affect final pretraining performance, supports joint hyperparameter tuning and loss-curve prediction, and qualitatively reveals interaction effects among hyperparameters.

Despite these promising results, our current study should be viewed as a proof of concept due to the limited accessibility of open-source pretraining logs. In particular, our training dataset only includes models with up to 430M parameters and OOD validation set only includes models with at most 1.2B parameters. Moreover, the diversity of the configurations in the pretraining logs are limited. For example, in the open-source pretraining logs, for AdamW, hyperparameter β_1, β_2, and ϵ are rarely tuned, making it difficult to reliably learn their effects and interactions from the logs. Another example is that the pretraining logs do not have any MoE models or models with linear attention (Dai et al., 2024; Kimi Team et al., 2025b; Yang et al., 2024), and only contain two choices of pre-training datasets (Li et al., 2024)

Looking ahead, we expect NCPL-style predictors to improve with the community's collective efforts to open-source pretraining spanning diverse setups. Continuously incorporating new data should ultimately enable more reliable prediction for large-scale pretraining.

References

Adriaensen, S., Rakotoarison, H., Müller, S., and Hutter, F. Efficient bayesian learning curve extrapolation using prior-data fitted networks. *Advances in Neural Information Processing Systems*, 36:19858–19886, 2023.

Akhauri, Y., Lewandowski, B., Lin, C.-H., Reyes, A. N., Forbes, G. C., Wongpanich, A., Yang, B., Abdelfattah, M. S., Perel, S., and Song, X. Performance prediction for large systems via text-to-text regression. *arXiv preprint 2506.21718*, 2025.

Allal, L. B., Lozhkov, A., Bakouch, E., Blázquez, G. M., Penedo, G., Tunstall, L., Marafioti, A., Kydlíček, H., Lajarín, A. P., Srivastav, V., et al. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Athanasiadis, T., Adriaensen, S., Müller, S., and Hutter, F. Tune my adam, please! *arXiv preprint arXiv:2508.19733*, 2025.

Bergsma, S., Dey, N., Gosal, G., Gray, G., Soboleva, D., and Hestness, J. Power lines: Scaling laws for weight decay and batch size in llm pre-training. In *Advances in Neural Information Processing Systems*, 2025.

Bhagia, A., Liu, J., Wettig, A., Heineman, D., Tafjord, O., Jha, A. H., Soldaini, L., Smith, N. A., Groeneveld, D., Koh, P. W., et al. Establishing task scaling laws via compute-efficient model ladders. *arXiv preprint arXiv:2412.04403*, 2024.

Bjorck, J., Benhaim, A., Chaudhary, V., Wei, F., and Song, X. Scaling optimal lr across token horizons. In *International Conference on Learning Representations*, 2025.

Blake, C., Eichenberg, C., Dean, J., Balles, L., Prince, L. Y., Deiseroth, B., Cruz-Salinas, A. F., Luschi, C., Weinbach, S., and Orr, D. u-μ p: The unit-scaled maximal update parametrization. In *International Conference on Learning Representations*, 2025.

Bordelon, B., Noci, L., Li, M., Hanin, B., and Pehlevan, C. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. In *International Conference on Learning Representations*, 2024.

Caballero, E., Gupta, K., Rish, I., and Krueger, D. Broken neural scaling laws. In *International Conference on Learning Representations*, 2023.

Chen, T. and Guestrin, C. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL `http://doi.acm.org/10.1145/2939672.2939785`.

Chen, X., Liang, C., Huang, D., Real, E., Wang, K., Pham, H., Dong, X., Luong, T., Hsieh, C.-J., Lu, Y., et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.

Dai, D., Deng, C., Zhao, C., Xu, R., Gao, H., Chen, D., Li, J., Zeng, W., Yu, X., Wu, Y., et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

Das, A., Kong, W., Sen, R., and Zhou, Y. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

DeepSeek-AI, Bi, X., Chen, D., Chen, G., Chen, S., Dai, D., Deng, C., Ding, H., Dong, K., Du, Q., Fu, Z., Gao, H., Gao, K., Gao, W., Ge, R., Guan, K., Guo, D., Guo, J., Hao, G., Hao, Z., He, Y., Hu, W., Huang, P., Li, E., Li, G., Li, J., Li, Y., Li, Y. K., Liang, W., Lin, F., Liu, A. X., Liu, B., Liu, W., Liu, X., Liu, X., Liu, Y., Lu, H., Lu, S., Luo, F., Ma, S., Nie, X., Pei, T., Piao, Y., Qiu, J., Qu, H., Ren, T., Ren, Z., Ruan, C., Sha, Z., Shao, Z., Song, J., Su, X., Sun, J., Sun, Y., Tang, M., Wang, B., Wang, P., Wang, S., Wang, Y., Wang, Y., Wu, T., Wu, Y., Xie, X., Xie, Z., Xie, Z., Xiong, Y., Xu, H., Xu, R. X., Xu, Y., Yang, D., You, Y., Yu, S., Yu, X., Zhang, B., Zhang, H., Zhang, L., Zhang, L., Zhang, M., Zhang, M., Zhang, W., Zhang, Y., Zhao, C., Zhao, Y., Zhou, S., Zhou, S., Zhu, Q., and Zou, Y. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint 2401.02954*, 2024.

Dey, N., Zhang, B. C., Noci, L., Li, M., Bordelon, B., Bergsma, S., Pehlevan, C., Hanin, B., and Hestness, J. Don't be lazy: Completep enables compute-efficient deep transformers. *arXiv preprint arXiv:2505.01618*, 2025.

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., et al. Scaling exponents across parameterizations and optimizers. In *International Conference on Machine Learning*, pp. 12666–12700. PMLR, 2024.

Fan, Z., Liu, Y., Zhao, Q., Yuan, A., and Gu, Q. Robust layerwise scaling rules by proper weight decay tuning. *arXiv preprint arXiv:2510.15262*, 2025.

Gadre, S. Y., Smyrnis, G., Shankar, V., Gururangan, S., Wortsman, M., Shao, R., Mercat, J., Fang, A., Li, J., Keh, S., Xin, R., Nezhurina, M., Vasiljevic, I., Soldaini, L., Jitsev, J., Dimakis, A., Ilharco, G., Koh, P. W., Song, S., Kollar, T., Carmon, Y., Dave, A., Heckel, R., Muennighoff, N., and Schmidt, L. Language models scale reliably with over-training and on downstream tasks. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=iZeQBqJamf`.

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. What can transformers learn in-context? a case study of simple function classes. *Advances in neural information processing systems*, 35: 30583–30598, 2022.

Gargiani, M., Klein, A., Falkner, S., and Hutter, F. Probabilistic rollouts for learning curve extrapolation across hyperparameter settings. *arXiv preprint arXiv:1910.04522*, 2019.

Goswami, M., Szafer, K., Choudhry, A., Cai, Y., Li, S., and Dubrawski, A. Moment: A family of open time-series foundation models. In *Forty-first International Conference on Machine Learning*, 2024.

Gruver, N., Finzi, M., Qiu, S., and Wilson, A. G. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36:19622–19635, 2023.

Hall, D., Ahmed, A., Chou, C., Garg, A., Kuditipudi, R., Held, W., Ravi, N., Shandilya, H., Wang, J., Bolton, J., Karamcheti, S., Kotha, S., Lee, T., Liu, N., Niklaus, J., Ramaswami, A., Salahi, K., Wen, K., Wong, C. H., Yang, S., Zhou, I., and Liang, P. Introducing marin: An open lab for building foundation models, 2025. URL `https://marin.community/blog/2025/05/19/announcement/`.

Hashimoto, T. Model performance scaling with multiple data sources. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4107–4116. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/hashimoto21a.html`.

Henighan, T., Kaplan, J., Katz, M., Chen, M., Hesse, C., Jackson, J., Jun, H., Brown, T. B., Dhariwal, P., Gray, S., Hallacy, C., Mann, B., Radford, A., Ramesh, A., Ryder, N., Ziegler, D. M., Schulman, J., Amodei, D., and McCandlish, S. Scaling laws for autoregressive generative modeling. *arXiv preprint 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., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O., and Sifre, L. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

Hollmann, N., Müller, S., Eggensperger, K., and Hutter, F. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Hollmann, N., Müller, S., Purucker, L., Krishnakumar, A., Körfer, M., Hoo, S. B., Schirrmeister, R. T., and Hutter, F. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Hu, M. Y., Pan, J., Jhaveri, A. R., Lourie, N., and Cho, K. Neural neural scaling laws. *arXiv preprint arXiv:2601.19831*, 2026.

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

Huang, X., Khetan, A., Cvitkovic, M., and Karnin, Z. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Jain, A., Montanari, A., and Sasoglu, E. Scaling laws for learning with real and surrogate data. *Advances in Neural Information Processing Systems*, 37:110246–110289, 2024.

Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Jeong, D. P., Lipton, Z. C., and Ravikumar, P. K. Llm-select: Feature selection with large language models. *Transactions on Machine Learning Research*, 2025.

Jin, M., Wang, S., Ma, L., Chu, Z., Zhang, J. Y., Shi, X., Chen, P.-Y., Liang, Y., Li, Y.-F., Pan, S., et al. Time-llm: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Kadra, A., Janowski, M., Wistuba, M., and Grabocka, J. Scaling laws for hyperparameter optimization. *Advances in Neural Information Processing Systems*, 36:47527–47553, 2023.

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.

Khatri, D., Madaan, L., Tiwari, R., Bansal, R., Duvvuri, S. S., Zaheer, M., Dhillon, I. S., Brandfonbrener, D., and Agarwal, R. The art of scaling reinforcement learning compute for llms. *arXiv preprint 2010.14701*, 2025.

Kimi Team, Bai, Y., Bao, Y., Chen, G., Chen, J., Chen, N., Chen, R., Chen, Y., Chen, Y., Chen, Y., et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025a.

Kimi Team, Zhang, Y., Lin, Z., Yao, X., Hu, J., Meng, F., Liu, C., Men, X., Yang, S., Li, Z., et al. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025b.

Klein, A., Falkner, S., Springenberg, J. T., and Hutter, F. Learning curve prediction with bayesian neural networks. In *International Conference on Learning Representations*, 2017.

Kosson, A., Welborn, J., Liu, Y., Jaggi, M., and Chen, X. Weight decay may matter more than mup for learning rate transfer in practice. *arXiv preprint arXiv:2510.19093*, 2025.

Kumar, A., Raghunathan, A., Jones, R., Ma, T., and Liang, P. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

Li, B., Chen, F., Huang, Z., Wang, L., and Wu, L. Functional scaling laws in kernel regression: Loss dynamics and learning rate schedules. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

Li, B., Wen, J., Zhou, Z., Zhu, J., and Chen, J. Efficient hyperparameter tuning via trajectory invariance principle. *arXiv preprint arXiv:2509.25049*, 2025b.

Li, H., Zheng, W., Wang, Q., Ding, Z., Wang, H., Wang, Z., Xuyang, S., Ding, N., Zhou, S., Zhang, X., and Jiang, D. Predictable scale: Part ii, farseer: A refined scaling law in large language models. *arXiv preprint 2506.10972*, 2025c.

Li, H., Zheng, W., Wang, Q., Zhang, H., Wang, Z., Xuyang, S., Fan, Y., Ding, Z., Wang, H., Ding, N., Zhou, S., Zhang, X., and Jiang, D. Predictable scale: Part i, step law – optimal hyperparameter scaling law in large language model pretraining. *arXiv preprint arXiv:2503.04715*, 2025d.

Li, J., Fang, A., Smyrnis, G., Ivgi, M., Jordan, M., Gadre, S. Y., Bansal, H., Guha, E., Keh, S. S., Arora, K., et al. Datacomp-lm: In search of the next generation of training sets for language models. *Advances in Neural Information Processing Systems*, 37:14200–14282, 2024.

Lin, H., Ye, H., Feng, W., Huang, Q., Li, Y., Lim, H., Li, Z., Wang, X., Ma, J., Zou, J., et al. Can language models discover scaling laws? *arXiv preprint arXiv:2507.21184*, 2025.

Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Liu, C., Chen, H., Zhang, Y., Dong, Y., and Zhu, J. Scaling laws for black box adversarial attacks. *arXiv preprint arXiv:2411.16782*, 2024b.

Liu, H., Yang, M., and Adomavicius, G. Robustness is important: Limitations of llms for data fitting. *arXiv preprint arXiv:2508.19563*, 2025a.

Liu, J., Su, J., Yao, X., Jiang, Z., Lai, G., Du, Y., Qin, Y., Xu, W., Lu, E., Yan, J., et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025b.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Lourie, N., Hu, M. Y., and Cho, K. Scaling laws are unreliable for downstream tasks: A reality check. In Christodoulopoulos, C., Chakraborty, T., Rose, C., and Peng, V. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 16167–16180, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.877. URL `https://aclanthology.org/2025.findings-emnlp.877/`.

Ludziejewski, J., Krajewski, J., Adamczewski, K., Pióro, M., Krutul, M., Antoniak, S., Ciebiera, K., Król, K., Odrzygóźdź, T., Sankowski, P., Cygan, M., and Jaszczur, S. Scaling laws for fine-grained mixture of experts. 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. 33270–33288. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/ludziejewski24a.html`.

Luo, K., Wen, H., Hu, S., Sun, Z., Liu, Z., Sun, M., Lyu, K., and Chen, W. A multi-power law for loss curve prediction across learning rate schedules. In *International Conference on Learning Representations*, 2025.

Malladi, S., Lyu, K., Panigrahi, A., and Arora, S. On the sdes and scaling rules for adaptive gradient algorithms. *Advances in Neural Information Processing Systems*, 35:7697–7711, 2022.

Marek, M., Lotfi, S., Somasundaram, A., Wilson, A. G., and Goldblum, M. Small batch size training for language models: When vanilla sgd works, and why gradient accumulation is wasteful. In *Advances in Neural Information Processing Systems*, 2025.

Marin. Introducing marin: An open lab for building foundation models, 2025. URL `https://marin.community/blog/2025/05/19/announcement/`. Accessed: 2025-07-20.

McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

OLMo Team, 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.

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

Qiu, S., Xiao, L., Wilson, A. G., Pennington, J., and Agarwala, A. Scaling collapse reveals universal dynamics in compute-optimally trained neural networks. In *International Conference on Machine Learning*, 2025.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Rakotoarison, H., Adriaensen, S., Mallik, N., Garibov, S., Bergman, E., and Hutter, F. In-context freeze-thaw bayesian optimization for hyperparameter optimization. In *International Conference on Machine Learning*, pp. 41982–42008. PMLR, 2024.

Ruan, Y., Maddison, C. J., and Hashimoto, T. B. Observational scaling laws and the predictability of langauge model performance. *Advances in Neural Information Processing Systems*, 37:15841–15892, 2024.

Shuai, X., Wang, Y., Wu, Y., Jiang, X., and Ren, X. Scaling law for language models training considering batch size. *arXiv preprint arXiv:2412.01505*, 2024.

Song, X., Li, O., Lee, C., Yang, B., Peng, D., Perel, S., and Chen, Y. Omnipred: Language models as universal regressors. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=t9c3pfrR1X.

Tissue, H., Wang, V., and Wang, L. Scaling law with learning rate annealing. *arXiv preprint arXiv:2408.11029*, 2024.

Vacareanu, R., Negru, V. A., Suciu, V., and Surdeanu, M. From words to numbers: Your large language model is secretly a capable regressor when given in-context examples. In *First Conference on Language Modeling*, 2024.

Wang, S., Chen, Z., Li, B., He, K., Zhang, M., and Wang, J. Scaling laws across model architectures: A comparative analysis of dense and MoE models in large language models. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 5583–5595. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.emnlp-main.319. URL https://aclanthology.org/2024.emnlp-main.319/.

Wang, X. and Aitchison, L. How to set adamw's weight decay as you scale model and dataset size. In *International Conference on Machine Learning*, 2025.

Wen, K., Hall, D., Ma, T., and Liang, P. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*, 2025.

Wistuba, M. and Pedapati, T. Learning to rank learning curves. In *International Conference on Machine Learning*, pp. 10303–10312. PMLR, 2020.

Xie, X., Ding, K., Yan, S., Toh, K.-C., and Wei, T. Optimization hyper-parameter laws for large language models. *arXiv preprint arXiv:2409.04777*, 2024.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

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.

Yang, S., Kautz, J., and Hatamizadeh, A. Gated delta networks: Improving mamba2 with delta rule. *arXiv preprint arXiv:2412.06464*, 2024.

Zhang, H., Morwani, D., Vyas, N., Wu, J., Zou, D., Ghai, U., Foster, D., and Kakade, S. M. How does critical batch size scale in pre-training? In *International Conference on Learning Representations*, 2025.

Zhou, Y., Xing, S., Huang, J., Qiu, X., and Guo, Q. How to set the learning rate for large-scale pre-training? *arXiv preprint arXiv:2601.05049*, 2026.

A Additional Related Work

Theory-motivated studies of hyperparameter effects and transfer. As complement to data-driven scaling-law fitting, an orthogonal approach studies the effect of hyperparameters and its transfer across training scales through theoretical lens. Using SDEs as a proxy for stochastic-gradient–based optimization, prior work has proposed scaling rules for how the learning rate should grow with batch size (Jastrzębski et al., 2017; Malladi et al., 2022). The critical batch size, beyond which large-batch training stops yielding proportional efficiency gains, can be characterized via the gradient-noise-scale proxy (McCandlish et al., 2018), and its scaling behavior has been further analyzed under simplified assumptions (Zhang et al., 2025). Wang & Aitchison (2025) interpret model weights as an exponential moving average of recent updates, and use this view to derive practical scaling rules for weight decay as model and dataset size grow. μP tackles how optimal learning rate can be transferred from smaller to larger models (Yang et al., 2021). Later works extended from the original width scaling setup to depth scaling and other architectural variants (Bordelon et al., 2024; Dey et al., 2025; Blake et al., 2025; Everett et al., 2024). However, μP primarily focuses on model scaling and does not directly address learning-rate transfer under data scaling, and offers limited guidance for hyperparameters beyond the learning rate. Recent work argues weight decay plays an important role in stabilizing the update dynamics and thus facilitating learning rate transfer (Kosson et al., 2025), and propose joint transfer principle of learning rate and weight decay (Fan et al., 2025). While these works offer valuable insights into how hyperparameters influence pretraining performance and provide practical tuning and transfer guidelines, they typically address only a limited subset of hyperparameters and do not predict the training loss for a given configuration, as NCPL does.

Learning-curve extrapolation and hyperparameter optimization. Beyond fitting parametric scaling laws, another line of work aims to *predict training trajectories* (learning curves) from partially-observed training trajectories, enabling early stopping and grey-box resource allocation in hyperparameter optimization. Early neural and probabilistic approaches include Bayesian neural networks with specialized learning-curve parameterizations (Klein et al., 2017) and probabilistic rollouts for learning-curve extrapolation across hyperparameter settings using models such as Bayesian RNNs (Gargiani et al., 2019). Ranking-based formulations further learn to rank partially observed curves to guide early termination (Wistuba & Pedapati, 2020). More recently, transformer-based Prior-Data Fitted Networks (PFNs) enable fast approximate Bayesian learning-curve extrapolation in a single forward pass (Adriaensen et al., 2023), and extend naturally to freeze-thaw Bayesian optimization via in-context surrogates (Rakotoarison et al., 2024). These PFN surrogates can also be specialized to optimizer hyperparameters, e.g., Adam tuning (Athanasiadis et al., 2025). In parallel, Deep Power Laws exploit power-law structure for grey-box HPO and budget allocation (Kadra et al., 2023). A very recent work Hu et al. (2026) tries to train a language model to predict downstream performance with the entire accuracy trajectories from a partial run, using token-level validation loss as input.

Table 2: Training configuration fields used as inputs to NCPL. For numerical fields, we report the scaling factor applied before feeding them into the numerical encoder.

Field group	Included fields	Field type	Scaling Factor
Source tag	Source identifier indicating which open-source training project the run comes from (e.g., Marin or StepLaw)	Categorical	–
Model architecture	Model size N (number of non-embedding parameters, in millions)	Numerical	$0.01\times$
	Number of layers	Numerical	$1\times$
	Number of attention heads	Numerical	$1\times$
	Hidden dimension	Numerical	$0.01\times$
Training scale	Number of training tokens D (in billions)	Numerical	$1\times$
	Total number of training steps	Numerical	$0.001\times$
	Current ratio of total training steps (for loss curve prediction)	Numerical	$1\times$
Optimizer and hyperparameters	Optimizer	Categorical	–
	Peak learning rate	Numerical	$10^4\times$
	Learning-rate schedule	Categorical	–
	Final learning rate after decay	Numerical	$10^4\times$
	The ratio between final learning rate and peak learning rate	Numerical	$200\times$
	Weight decay	Numerical	$10^2\times$
	Batch size	Numerical	$10^{-1}\times$
	Warmup ratio	Numerical	$10^{-2}\times$
	Gradient clipping threshold	Numerical	$1\times$
	Momentum coefficients (β_1, β_2) for AdamW	Categorical	–
	Numerical-stability constant ϵ for AdamW	Categorical	–
	Adam learning rate used inside Muon optimizer	Numerical	$10^4\times$
	Block size for the Soap optimizer	Numerical	$0.02\times$
	Preconditioner learning rate for Kron optimizer	Categorical	–

B EXPERIMENTAL DETAILS

B.1 DATA PROCESSING

Output Metrics Since the per-step pre-training loss exhibits high variance, we use less noisy metrics. For the Marin dataset, we use the language modeling loss on the English split of the C4 dataset (Raffel et al., 2020). For the StepLaw dataset, we use the exponentially smoothed training loss with a smoothing coefficient of 0.99.

Filtering. The open-source runs we collect include diverged or failed runs, so we apply a filtering procedure with the following criteria: (i) unfinished runs; (ii) diverged runs, defined as those with final pretraining loss > 4, or with final loss more than 0.3 above the best loss achieved at the same data size and model size (a large gap in language-model pretraining); (iii) unstable runs, defined as those with an average loss slope larger than 0.001 over any 5% window of training steps.

Training configuration used. In table 2, we list all training-configuration fields used as inputs to NCPL. Categorical fields are encoded using the backbone language model's standard tokenizer and token embeddings. Numerical values are mapped into the embedding space via a two-layer MLP. For (β_1, β_2), ϵ, and the preconditioner learning rate of the Kron optimizer, since these values lie in a small discrete range, we treat them as categorical and encode them with the standard tokenizer. For ϵ, we additionally apply a negative $\log$ transform.

Scaling. Numerical configuration values can vary by orders of magnitude, yet they are all processed by the same numerical encoder. We therefore apply a fixed scaling to each numerical field

```
source:  steplaw
data size:  25.0
model size:  268.0
num layers:  8
num heads:  16
hidden dim:  9552
optimizer:  adamw
learning rate:  0.000977
lr schedule:  cosine
warmup:  2000
weight decay:  0.1
batch size:  960
beta1:  0.9
beta2:  0.95
epsilon:  8
max_grad_norm:  1.0
minlr ratio:  0.00102
minlr:  1e-05
max step:  127155
final loss:  0.0235
```

```
source:  marin
data size:  26.8
model size:  134.0
num layers:  32
num heads:  8
hidden dim:  512
optimizer:  soap
learning rate:  0.0016
lr schedule:  cosine
warmup:  1000
weight decay:  0.1
batch size:  1280
beta1:  0.95
beta2:  0.98
epsilon:  15
max_grad_norm:  1.0
minlr ratio:  0
minlr:  0
block_size:  256
max step:  127155
frac:  0.3907
final loss:  0.6609
```

Figure 8: Example training samples. Left: an input for final loss prediction. Right: an input for loss curve prediction. Numerical values are embedded with a two-layer MLP, while other text is embedded using standard token embeddings. The values 0.0235 and 0.6609 denote target labels and are not part of the input.

when constructing inputs, so that different fields fall into a comparable range. The scaling factors are reported in table 2.

Example training samples for final-loss prediction and loss curve prediction are shown in fig. 8.

B.2 Training setup and hyperparameters

As described in section 4.1, we adopt a two-stage training pipeline for stability. We use Qwen3-1.7B as the base model (Yang et al., 2025). In the first stage, we freeze the backbone and update only the two-layer MLP encoder for numerical fields together with the linear prediction head. We train for 20 epochs with learning rate 5×10^{-5} and a warmup ratio of 0.1 of total steps. In the second stage, we fine-tune all model parameters for 1000 epochs using learning rate 1×10^{-5} with a 1000-step warmup. We reset the optimizer state between the two stages. In both stages, we use AdamW with linear learning-rate decay, weight decay 0.01, and batch size 480.

Training NCPL for loss curve prediction. For each run we uniformly sample up to 40 intermediate checkpoints, append a scalar field indicating the fraction of total training steps completed, and train the model to predict the difference between the current loss and the Chinchilla baseline (fig. 8 Right). We use the same two-stage procedure and hyperparameters, but train for 10 epochs in the first stage and 400 epochs in the second stage.

B.3 Evaluations

Hyperparameter selection. The contour plots are generated by interpolating the scattered ground-truth loss values in log-scaled learning-rate/batch-size space using a smooth RBF interpolant, followed by light Gaussian smoothing; we then draw iso-loss contour lines on the resulting surface. The "minimum" markers for both ground truth and NCPL are obtained by fitting a quadratic surface

Table 3: Ablations on final-loss prediction on both ID and OOD splits. We compare NCPL (Ours) with two ablations: removing residual prediction and removing numerical tokens. We report mean absolute error (MAE), root mean squared error (RMSE), and Spearman correlation (ρ).

Data	Method	Final-loss prediction					
		In-distribution			Out-of-distribution		
		MAE↓	RMSE↓	$\rho\uparrow$	MAE↓	RMSE↓	$\rho\uparrow$
Marin	NCPL (Ours)	**0.0109**	**0.0169**	**0.9813**	**0.0168**	**0.0239**	**0.9299**
	No Residual Prediction	0.0133	0.0195	0.9745	0.1503	0.1576	0.9264
	No Numerical Tokens	0.0123	0.0243	0.9665	0.0217	0.0288	0.9267
Steplaw	NCPL (Ours)	**0.0097**	**0.0158**	0.9948	**0.0223**	**0.0345**	**0.9837**
	No Residual Prediction	0.0208	0.0301	**0.9968**	0.0988	0.1134	0.9607
	No Numerical Tokens	0.0123	0.0212	0.9906	0.0357	0.0428	0.9763

in log space using only near-optimal points whose loss is within 1% of the minimum predicted loss, and taking the minimizer of the fitted quadratic.

C Additional results

C.1 More results of Fine-tuned NCPL.

Final pretraining loss prediction results across learning rates and batch sizes on the Steplaw dataset are shown in fig. 9 (ID) and fig. 10 (OOD). Hyperparameter selection results for all held-out model–data size pairs are shown in fig. 11 and fig. 12. At the largest extrapolation scale ($N = 1073$M), NCPL's predicted pretraining loss tends to be higher than the ground-truth loss. This is partly because the Chinchilla baseline itself overpredicts at this scale; adding our predicted residual on top of an inflated baseline can further increase the final prediction. In addition, the model predicts a larger optimal learning rate for larger N when the D/N ratio is small. This trend may reflect limited data diversity in the training set, and could be alleviated by incorporating additional training logs. Loss-curve prediction results on the Marin dataset across different optimizers (ID and OOD) are presented in fig. 15. Fig. 13 and fig. 14 show randomly sampled loss curve prediction results on ID and OOD runs, without cherry-picking.

C.2 Ablation on NCPL design choices

We study two key design choices in NCPL: (i) predicting residuals relative to a Chinchilla-law baseline, and (ii) encoding scalar configuration values using numerical tokens via a two-layer MLP. We consider two ablated variants: replacing residual prediction with respect to the Chinchilla baseline with direct loss prediction, and replacing the two-layer MLP encoder with standard tokenization for numerical fields. Results shown in table 3 show consistent gains from both residual prediction and numerical tokens.

We study two key design choices in NCPL: (i) predicting residuals relative to a Chinchilla-law baseline, and (ii) encoding scalar configuration values using numerical tokens. Specifically, we consider two ablations: (a) predicting the final loss directly, rather than the residual w.r.t. the Chinchilla baseline; and (b) tokenizing numerical fields with a standard tokenizer, rather than encoding them with a two-layer MLP. As shown in table 3, both residual prediction and numerical-token encoding contribute substantially to performance.

C.3 Additional results of NCPL with training from scratch.

Fig. 16 and fig. 17 show the predicted vs. ground-truth final pretraining loss of NCPL trained from scratch, using 1.7B model and 135M model respectively.

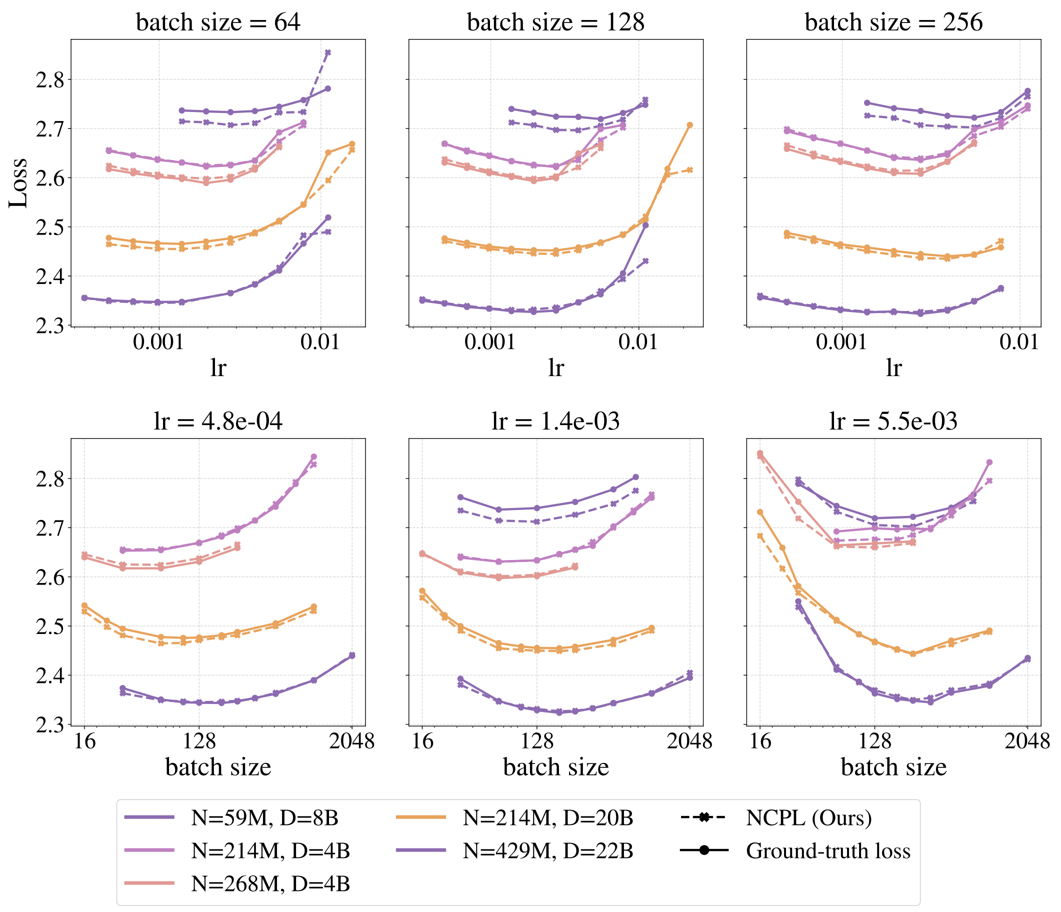


Figure 9: Final-loss prediction across learning rates and batch sizes for all 5 ID held-out (N, D) pairs.

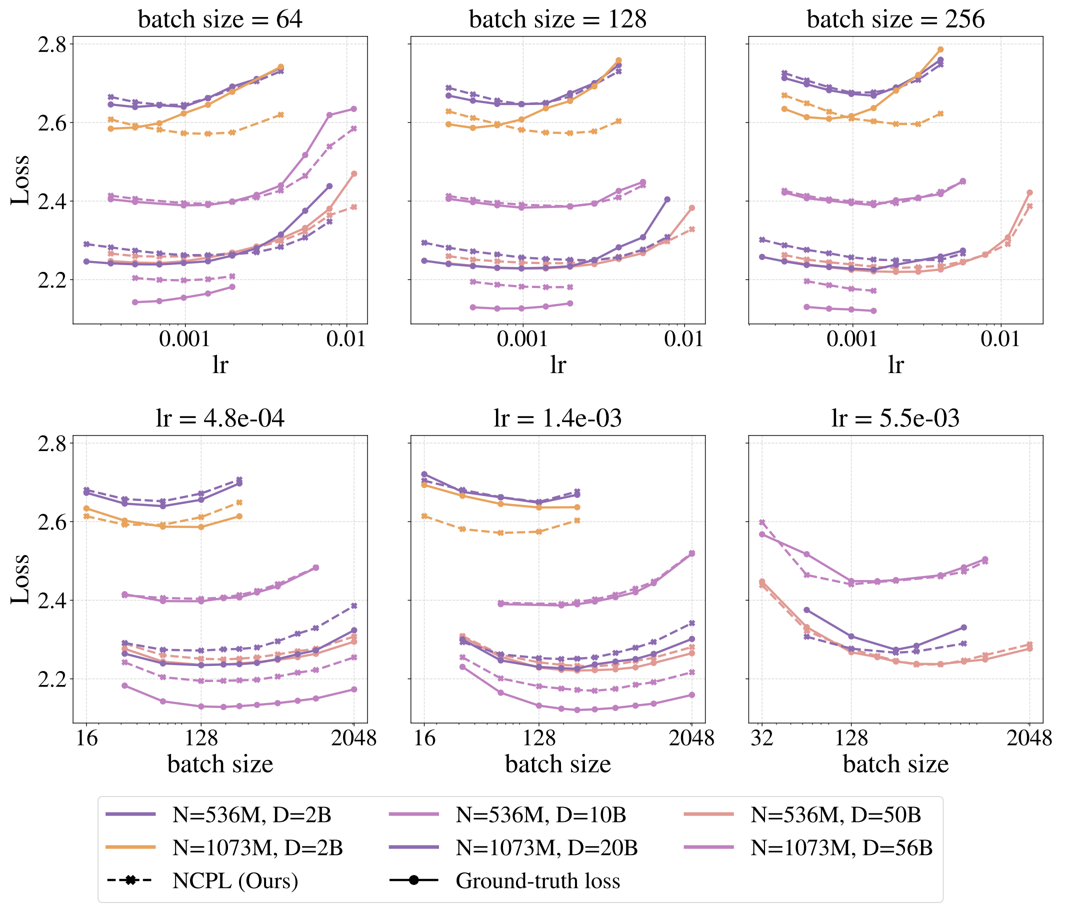


Figure 10: Final-loss prediction across learning rates and batch sizes for 6 OOD held-out (N, D) pairs.

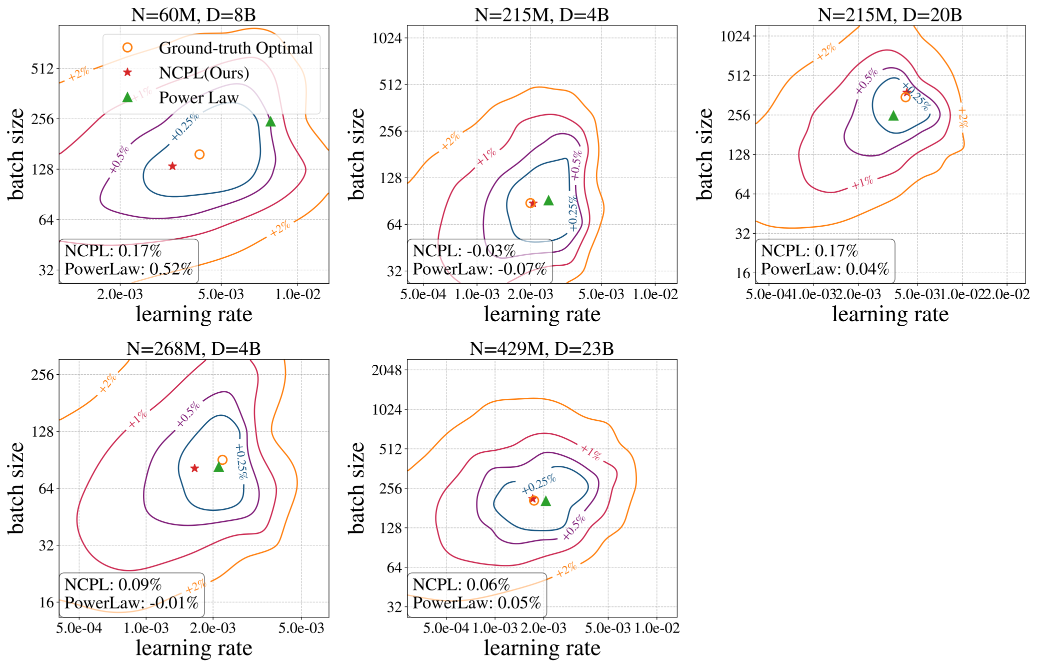


Figure 11: Predicted optimal learning rates and batch sizes for NCPL and the power law baseline on all 5 held-out ID (N, D) pairs, together with their relative loss.

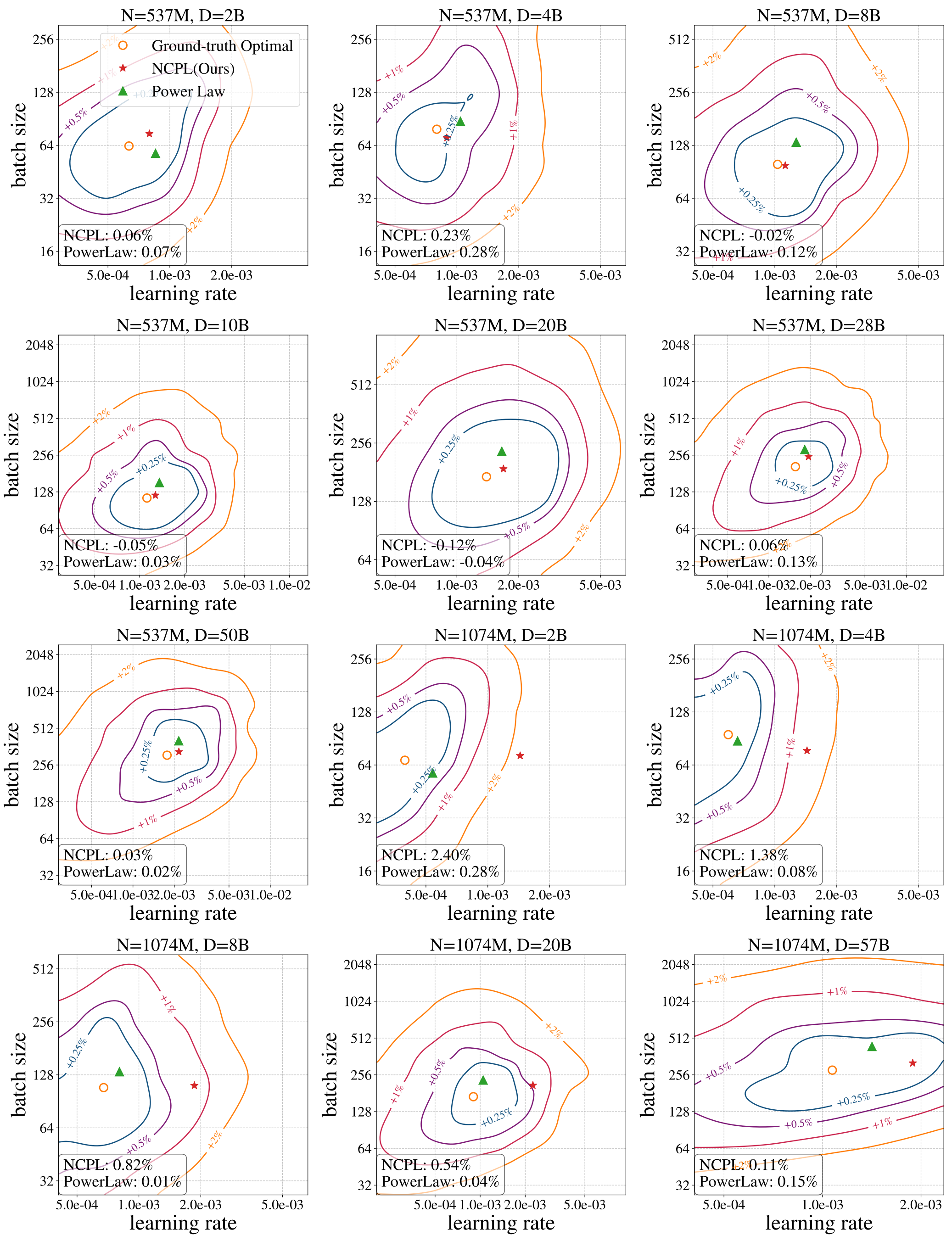


Figure 12: Predicted optimal learning rates and batch sizes for NCPL and the power law baseline on all 12 held-out OOD (N, D) pairs, together with their relative loss.

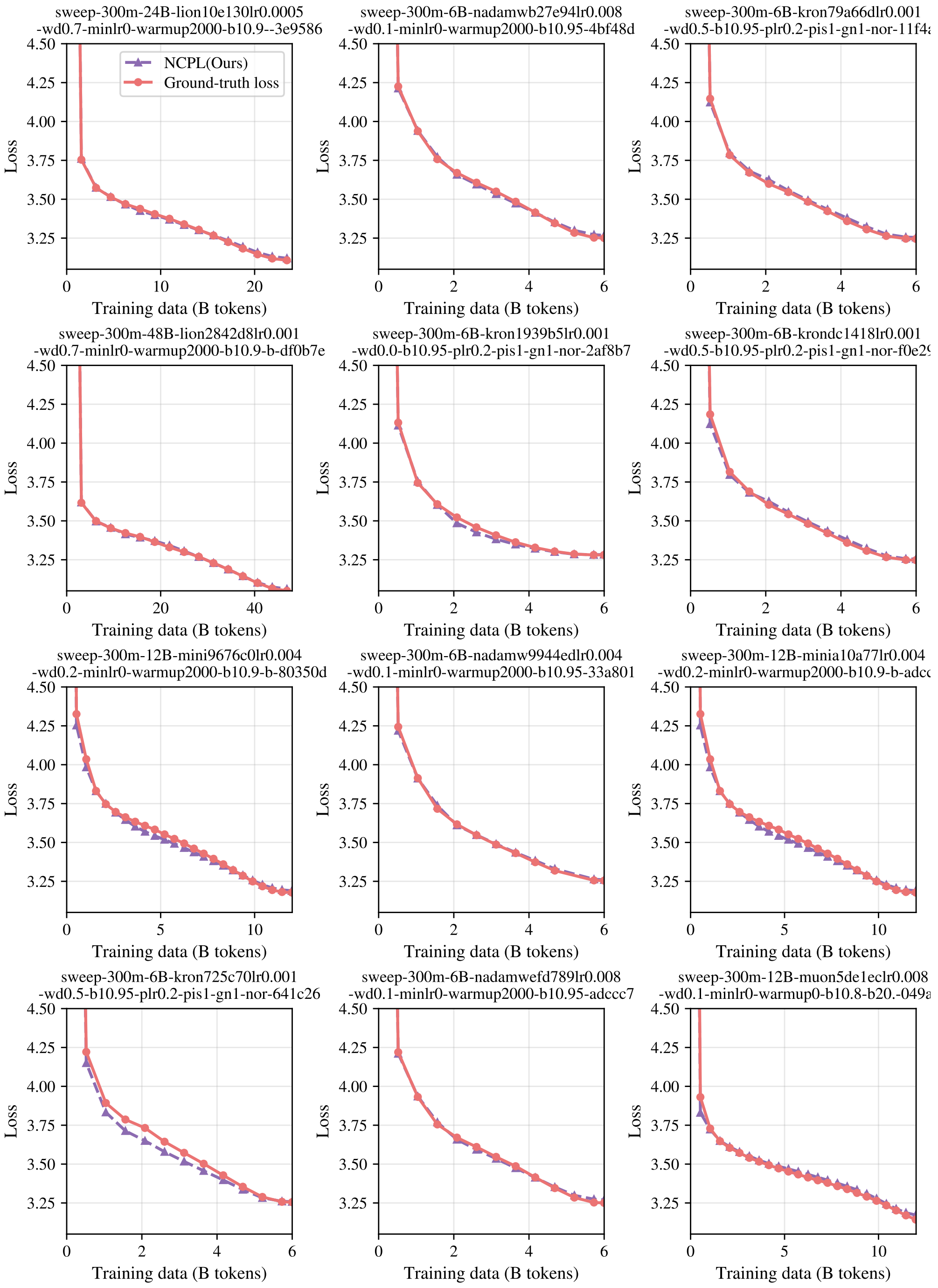


Figure 13: Loss-curve prediction on 12 randomly sampled ID runs (Marin, 300M). Each subplot title is the corresponding run name from the original Wandb project (Wen et al., 2025).

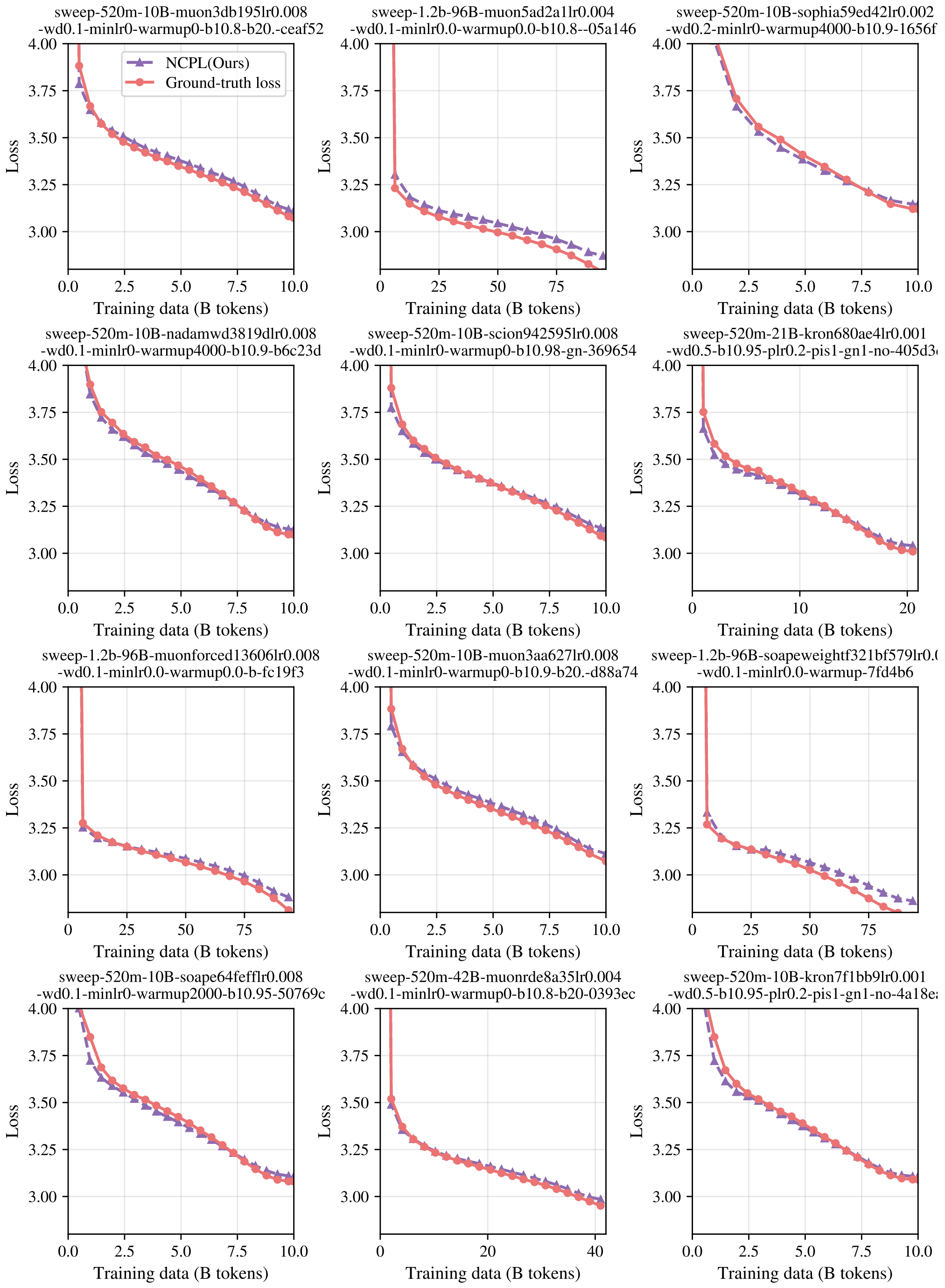


Figure 14: Loss-curve prediction on 12 randomly sampled OOD runs (Marin). Each subplot title is the corresponding run name from the original Wandb project (Wen et al., 2025).

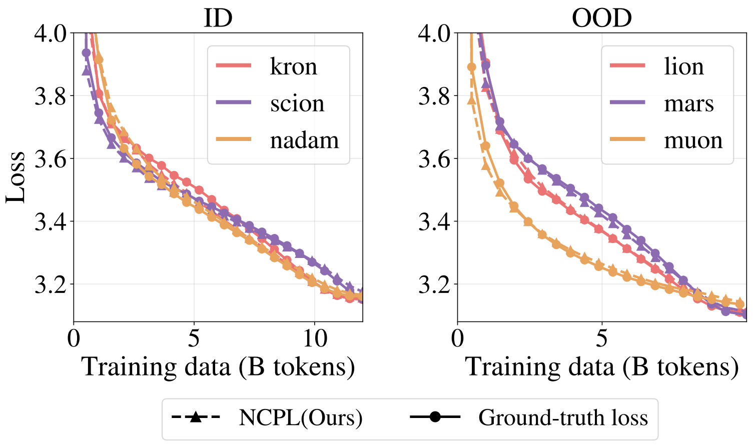


Figure 15: **Loss curve prediction.** Ground-truth and predicted pretraining loss curves under different optimizer settings on the ID and OOD validation sets. NCPL closely tracks the overall trajectories and captures optimizer-specific curve shapes. Setting: Marin Dataset. **Left:** ID validation, $N = 300\text{M}$, $D = 12\text{B}$, optimizers Kron/Scion/Nadam. **Right:** OOD validation, $N = 520\text{M}$, $D = 10\text{B}$, optimizers Lion/Mars/Muon. Results of loss curves under different hyperparameter are shown in fig. 7.

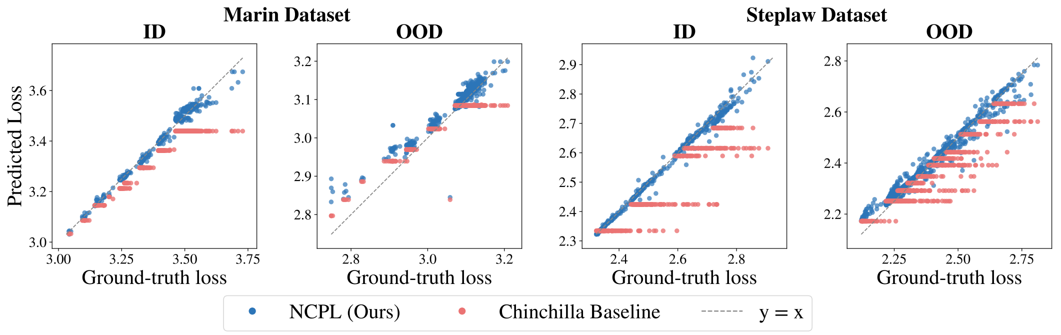


Figure 16: Predicted vs. ground-truth final pretraining loss of NCPL trained from scratch (1.7B model).

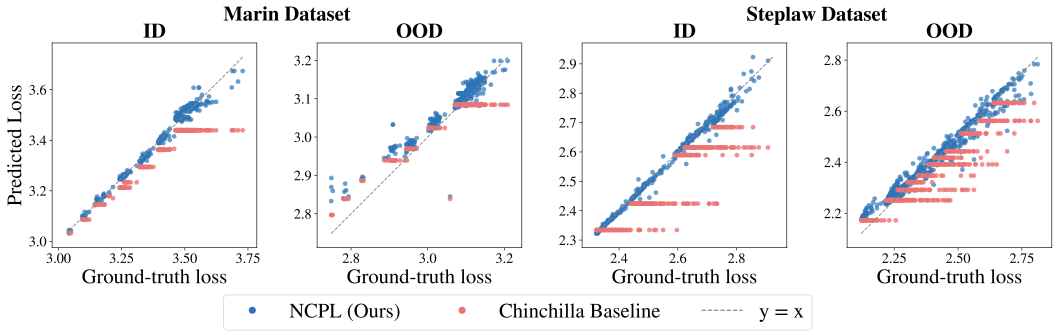


Figure 17: Predicted vs. ground-truth final pretraining loss of NCPL traineds from scratch (135M model).