CONFIGURATION-TO-PERFORMANCE SCALING LAW WITH NEURAL ANSATZ

Anonymous authors
Paper under double-blind review

ABSTRACT

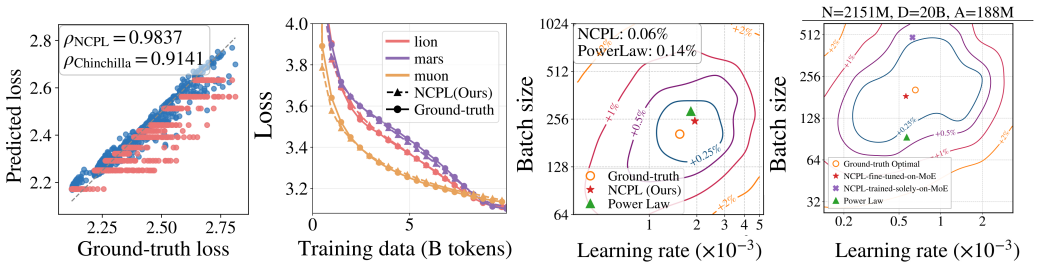
Researchers build scaling laws to forecast the training performance of expensive large-scale runs with larger model size N and data size D. These laws assume that other training hyperparameters are optimally chosen, which can require significant effort and, in some cases, be impossible due to external 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, and can be adapted to new training settings by fine-tuning on only a small amount of additional logs.

1 INTRODUCTION

As pretraining large language models is extremely costly (Liu et al., 2024a; Kimi Team et al., 2025), 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 pretraining 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 (Hoffmann et al., 2022; Porian et al., 2024). However, researchers often do not have the 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 how the performance metric depends on the full set of training 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.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.



(a) Final loss prediction (b) Loss curve prediction (c) Hyperparameter selection (d) Fine-tuning on MoE

Figure 1: **An overview of NCPL's performance across tasks. (a): NCPL predicts final loss more accurately than the Chinchilla law.** Predicted vs. ground-truth loss on the StepLaw OOD validation sets. The Chinchilla law yields configuration-agnostic prediction whereas NCPL takes in the full configuration as inputs and therefore achieves better prediction. **(b): 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=520M, D=10B). NCPL predicts optimizer-specific curve shapes accurately. **(c): NCPL enables hyperparameter tuning.** Optimal learning rate and batch size prediction in an OOD setup (StepLaw dataset; N=536M, D=28.4B). Configuration-dependent predictions naturally enable joint tuning over these two hyperparameters, achieving comparable performance to hand-designed functional form. **(d): NCPL adapts to new training settings with limited fine-tuning samples.** Starting from an NCPL trained on dense-model logs, fine-tuning on only 179 MoE samples yields substantially more accurate hyperparameter predictions than training from the same backbone using only MoE logs.

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 do not have sufficient data from expensive pretraining runs to build such a CPL. Fortunately, open-source 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 benefits from training on 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, training CPL is already feasible with current open pretraining logs and foundation models. We fine-tune 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$\times$ 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 Chinchilla law, which only takes N and D as inputs (Figure 1 (a)). 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 these two hyperparameters (Figure 1 (c)).
3. NCPL can 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 (b)).
4. NCPL exhibits broader capabilities enabled by learning from large-scale and diverse pretraining logs: it learns nuanced interactions between training configurations, such as the rarely noticed dependence between optimizer choice and weight-decay strength, and can be efficiently adapted

to new training settings (MoE architectures) by fine-tuning on only a small number of additional logs. (Figure 1 (d), Section 4.3)

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, 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. Our fine-tuning experiments provide preliminary evidence for this transfer: adapting an NCPL trained on dense-transformer logs to MoE training logs substantially improves over training from the same backbone using only MoE logs (Section 4.3).

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 (Hoffmann et al., 2022; Porian et al., 2024).

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; Li et al., 2025d; Zhou et al., 2026). For example, Li et al. (2025d) models the optimal learning rate (lr) and batch size (bs) as power-law functions of model size and 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. 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 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, specifying a closed-form law 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). We call the resulting model *Neural Configuration-to-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 prediction targets 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 Figure 2. In this work, we show that language models used as regressors can leverage these configuration features to make accurate, configuration-aware performance predictions.

In this paper, we consider two prediction targets: (i) the final pretraining loss and (ii) the pretraining loss at a specified intermediate training step. Predicting intermediate losses allows us to reconstruct the loss curve by querying losses at multiple intermediate steps.

```
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 (Eq. (3)). Full examples and additional details are provided in Section B.1.

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 4.4 and Table 5). Given the serialized input sequence x, the model embeds textual and numerical fields differently (Figure 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.3). 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] The baseline is fit using only the training set, so no

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

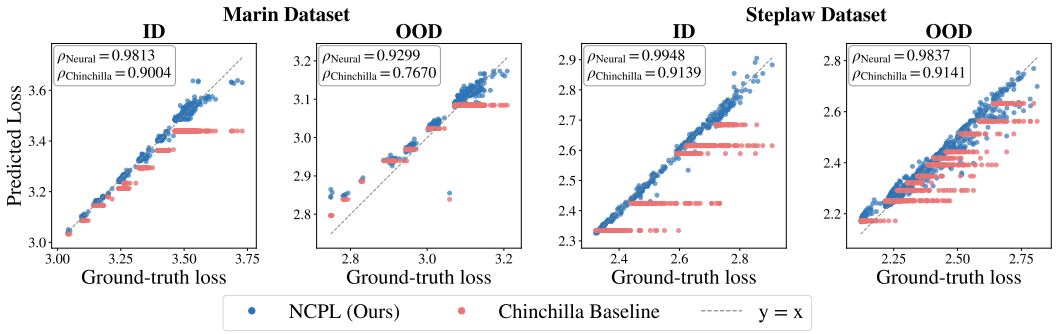


Figure 3: **Predicted loss vs. ground-truth loss.** Each point visualizes the predicted vs. ground-truth final pretraining loss of an individual run. NCPL uses the full training configuration as input, whereas the Chinchilla law only depends on (N, D) and therefore gives the same prediction for all runs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

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): Stage 1 update 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 with larger model sizes N. The pretraining runs we collected roughly follow the Chinchilla scaling law. 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 a language model as the regressor) *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. Looking ahead, the same Transformer backbone readily supports richer prediction targets (e.g., downstream-task performance prediction) and continual fine-tuning. We compare NCPL with strong tabular baselines in Section 4.4.*

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 both dense and MoE models, while fixing all other hyperparameters. For dense models, the sweeps cover model sizes from 215M to 1B

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

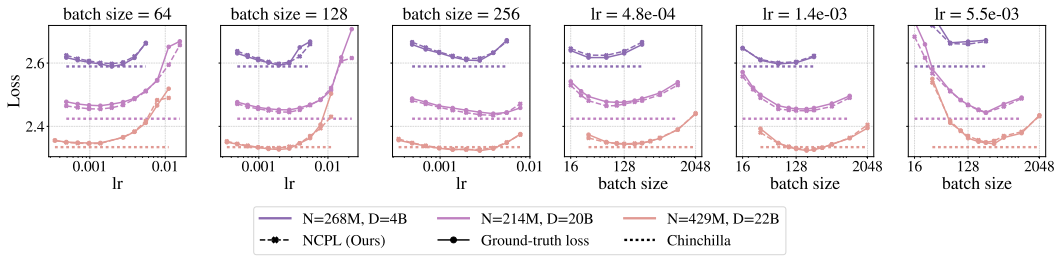


Figure 4: **Predicted vs. ground-truth loss across hyperparameters.** We compare predicted and ground-truth losses as lr and bs vary. NCPL accurately predicts how hyperparameters affect the final loss, whereas Chinchilla law yields a single configuration-agnostic prediction for each (N, D) pair.

parameters and data scales up to 56B tokens. For MoE models, the sweeps cover a total model size of 2B and data scales up to 20B tokens. [3] We use the dense-model logs, referred to as the *StepLaw* dataset, for training and evaluating NCPL in most of our experiments. We reserve the MoE logs specifically for the fine-tuning experiments in Section 4.3, and refer to them as *StepLaw-MoE*.

We exclude runs that are unstable, non-converged, or accidentally terminated (details in Section B.1) and obtain the Marin dataset and StepLaw dataset of 2,549 and 2,581 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 $(\text{optimizer}, N, D)$ tuple, and then randomly assign each group to either the training set or the ID validation set. Thus every run in the ID validation set does not have another run with the same optimizer, model size, and 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 training described in Section 3.2. More details are provided in Section B.2. [4]

4.2 Main Results

Configuration-dependent final loss prediction. In Figure 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 than the Chinchilla Law baseline on both ID and OOD validation sets, as summarized in Table 4. 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 profiles across hyperparameters, illustrating its ability to learn how training hyperparameters affect the final loss.

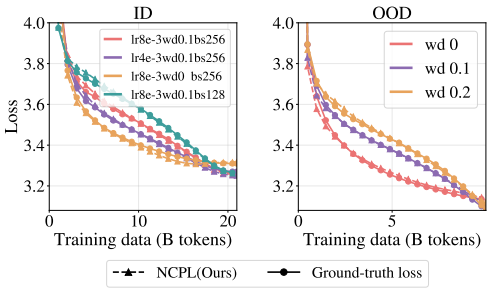
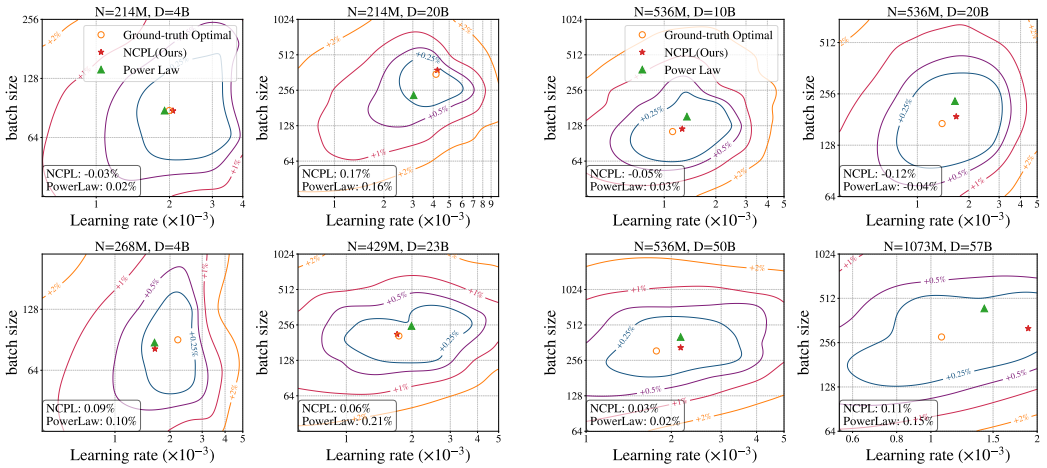


Figure 5: **Loss curve prediction.** Ground-truth and predicted loss curves under different hyperparameters. NCPL closely tracks the overall trajectories and learns hyperparameter-specific curve shapes. Setting: Marin Dataset. **Left:** N=130M, D=21B, AdamW optimizer with varying learning rate, weight decay, and batch size. **Right:** N=520M, D=10B, Muon optimizer with learning rate 0.008 and varying weight decay.

NCPL for loss curve prediction Beyond accepting the full training configuration 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. Specifically, we train NCPL to predict the pretraining loss at a specified intermediate training step, and reconstruct the loss curve by

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

[4] Our code and collected dataset are provided in the supplementary material.



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

Figure 6: **Hyperparameter selection on the StepLaw dataset.** Predicted optimal learning rates and batch sizes from NCPL and the power-law fitting baseline (Eq. (2)). The contour lines show the true loss landscape, with labels indicating relative losses to the minimum. The relative losses of the hyperparameters selected by NCPL and the power law are reported. NCPL's prediction aligns with the true optima, and achieves better or comparable losses to the power law.

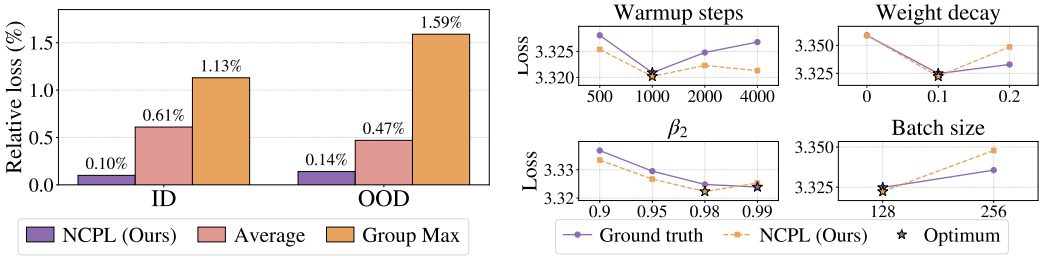


Figure 7: **Hyperparameter selection on the Marin dataset. Left:** The configurations selected by NCPL achieve losses that are on average only 0.10% and 0.14% above the observed minima in ID and OOD setups, respectively. **Right:** NCPL's predictions and ground-truth losses across hyperparameter sweeps. Results are shown for an ID validation setup with N=130M, D=10B and the MARS optimizer. NCPL identifies near-optimal values across different hyperparameters.

querying losses at multiple intermediate steps. Figure 5 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 Figure 1 and Figure 20.

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 the learned NCPL, one can sweep over candidate training configurations and use NCPL's estimated final loss as a proxy for the true pretraining loss, thereby identifying optimal hyperparameters without performing expensive training sweeps (see Section 3.3).

StepLaw dataset contains extensive joint sweeping of learning rate and batch size for different model and data sizes. On this dataset, we compare the optimal learning rate and batch size predicted by NCPL with the power-law scaling rule proposed in Li et al. (2025d) (Eq. (2)). As shown in Figure 6, NCPL's predictions closely match the true optima obtained from exhaustive sweeps, in both in-distribution and out-of-distribution settings. The relative losses are comparable to those of the fitted power-law baseline. Additional results are provided in Section C.2.

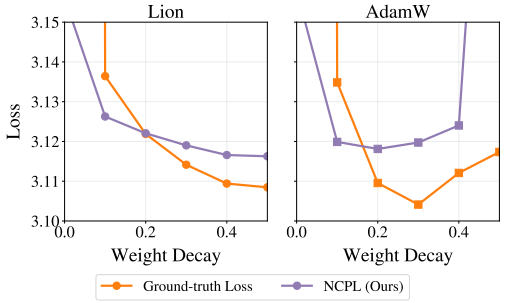


Figure 8: **NCPL learns interactions between weight decay and optimizer choice.** It's known that Lion requires substantially larger weight decay than AdamW (Chen et al., 2023; Loshchilov & Hutter, 2019; Wen et al., 2025). NCPL predicts this phenomenon on the OOD validation sets (Marin Dataset, N=520M, D=10B).

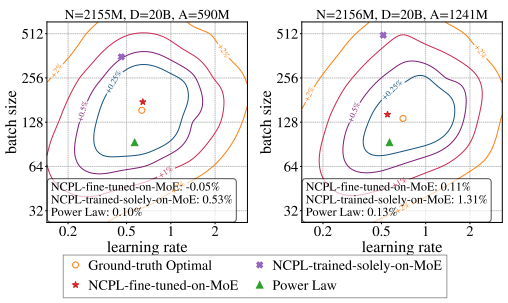


Figure 9: **Fine-tuning NCPL for MoE settings.** NCPL, previously trained on over 3k dense-transformer logs, is fine-tuned on only 179 MoE logs. The fine-tuned NCPL achieves substantially more accurate hyperparameter selection than training solely on MoE logs, showing that NCPL can transfer knowledge to new training settings.

The Marin dataset sweeps over multiple hyperparameters at small training scales, including learning rate, batch size, weight decay, β_1, β_2, warmup steps, etc. We consider training setups (N, D, and optimizer) with more than five evaluated hyperparameter configurations, and compare (1) the actual relative loss of the configuration with the lowest loss predicted by NCPL, (2) the average relative loss across all swept configurations, and (3) the maximum relative loss across these configurations. Relative loss is measured with respect to the minimum actual loss within the same setup. As shown in Figure 7 (Left), the configurations selected by NCPL achieve losses that are on average only 0.10% and 0.14% above the observed minima in ID and OOD setups, respectively. Figure 7 (Right) further shows that NCPL predicts near-optimal values for individual hyperparameters. These results show that NCPL remains effective at jointly selecting multiple hyperparameters.

4.3 Broader Capabilities of NCPL

We further demonstrate the broader promise of NCPL enabled by learning from a large collection of pretraining logs with diverse configuration inputs. Specifically, NCPL can learn meaningful interactions among training configurations, and it can adapt to new training settings by fine-tuning on a small number of target-domain logs, substantially fewer than those used in the original training.

Learning interactions between training configurations Different hyperparameters in the training configuration do not affect pretraining performance independently; instead, their effects interact in complicated ways. For example, different optimizers often prefer different hyperparameter choices (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 Figure 8, 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.

Fine-tuning NCPL for MoE settings with limited samples LLM pretraining may enter new training settings beyond those covered by the logs used to train NCPL, where model performance and optimal hyperparameters may shift. We show that NCPL can be adapted to such new settings by fine-tuning on only a small number of logs collected under the new setting. Specifically, we start from NCPL trained on over 3k dense-transformer logs from the Marin and StepLaw datasets, and fine-tune it using only 179 samples from StepLaw-MoE. These training samples cover four distinct activation sizes A and data sizes no larger than 8B, and we evaluate generalization to a larger data size of 20B. We use the same Chinchilla Law baseline fitted on the dense StepLaw logs for residual prediction, without refitting it on StepLaw-MoE. Details are provided in Section B.2. As shown in Figure 9, the fine-tuned NCPL achieves low prediction error for hyperparameter selection, with relative loss within 0.1% of the optimum, and substantially outperforms training from the same backbone using only StepLaw-MoE. This indicates that NCPL can transfer useful knowledge learned from large-scale pretraining logs and be efficiently adapted to new training settings with limited additional samples.

4.4 Ablation on the Backbone Architecture of NCPL

We compare different instantiations of CPL, including NCPL and strong tabular baselines: XGBoost (Chen & Guestrin, 2016), CatBoost (Prokhorenkova et al., 2018), and TabM (Gorishniy et al., 2025) (Table 4). For NCPL, we further compare fine-tuning (ft) and training from scratch (scratch) across two model sizes, using the Qwen3-1.7B (Yang et al., 2025) and SmolLM2-135M (Allal et al., 2025) architectures (Table 5).

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). For the StepLaw dataset, where the configurations vary only in a small set of continuous factors (N, D, lr, and bs), tabular baselines and NCPL (scratch) achieve comparable or slightly better performance to that of NCPL (ft). In contrast, NCPL (ft) offers a clear advantage on the Marin dataset, ranking first or a close second across evaluation metrics. Marin contains more diverse training configurations, spanning different optimizers and hyperparameter settings. Accurately capturing how these heterogeneous configurations affect training loss is crucial for learning CPL. Looking ahead, as we incorporate more data to train NCPL, the configurations will become even more diverse. Furthermore, NCPL demonstrates the ability to adapt to new training setups through fine-tuning. We therefore adopt NCPL (ft) as our main method. We note that the computational cost of fine-tuning a 1.7B model is negligible relative to that of the pretraining experiments themselves and thus should not be a concern.

5 Related work

Scaling Law Kaplan et al. (2020); Hoffmann et al. (2022) showed the pretraining loss of the Transformer-based LLMs follows 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 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.

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). Our work focuses on predicting the pretraining outcomes from full training configurations. To the best of our knowledge, this setting has not been systematically studied in prior work. We defer additional related works to Section A.

6 Conclusion and discussion

We propose to learn the mapping from full training configurations to training outcomes with generic neural networks by training on large-scale open-source pretraining logs. We instantiate this idea by fine-tuning a pretrained language model, resulting in the *Neural Configuration-to-Performance Scaling Law*. Empirically, NCPL accurately predicts how configurations affect pretraining performance, supports joint hyperparameter tuning and loss-curve prediction, reveals interactions among configurations, and can be efficiently fine-tuned to new training settings with limited additional logs. Looking ahead, we expect NCPL-style predictors to improve with the community's collective efforts to open-source pretraining experiments spanning diverse setups. Continuously incorporating new data should ultimately enable more reliable prediction for large-scale pretraining.

Reproducibility Statement

We provide implementation and experimental details in the appendix, including data sources, model configurations, and training setups. Our code is released as anonymized supplementary material to facilitate reproduction of the reported results.

AI Use Statement

We used generative AI tools to polish the writing and assist with writing experimental code. The experimental design was developed by the authors. We take responsibility for the final content of this work, including all AI-assisted text and code.

Ethics Statement

We investigate using language models to learn configuration-to-performance mappings from large-scale open-source pretraining logs. The proposed method aims to improve the predictability of pretraining experiments, which can help accelerate research iteration and reduce computational cost. All training logs used in this work are drawn from publicly available sources. We do not anticipate direct negative societal impacts.

References

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

Yash Akhauri, Bryan Lewandowski, Cheng-Hsi Lin, Adrian N. Reyes, Grant C. Forbes, Arissa Wongpanich, Bangding Yang, Mohamed S. Abdelfattah, Sagi Perel, and Xingyou Song. Performance prediction for large systems via text-to-text regression. *arXiv preprint 2506.21718*, 2025.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, et al. Smollm2: When smol goes big–data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Theodoros Athanasiadis, Steven Adriaensen, Samuel Müller, and Frank Hutter. Tune my adam, please! *arXiv preprint arXiv:2508.19733*, 2025.

Shane Bergsma, Nolan Dey, Gurpreet Gosal, Gavia Gray, Daria Soboleva, and Joel Hestness. Power lines: Scaling laws for weight decay and batch size in llm pre-training. In *Advances in Neural Information Processing Systems*, 2025.

Akshita Bhagia, Jiacheng Liu, Alexander Wettig, David Heineman, Oyvind Tafjord, Ananya Harsh Jha, Luca Soldaini, Noah A Smith, Dirk Groeneveld, Pang Wei Koh, et al. Establishing task scaling laws via compute-efficient model ladders. *arXiv preprint arXiv:2412.04403*, 2024.

Johan Bjorck, Alon Benhaim, Vishrav Chaudhary, Furu Wei, and Xia Song. Scaling optimal lr across token horizons. In *International Conference on Learning Representations*, 2025.

Charlie Blake, Constantin Eichenberg, Josef Dean, Lukas Balles, Luke Yuri Prince, Björn Deiseroth, Andres Felipe Cruz-Salinas, Carlo Luschi, Samuel Weinbach, and Douglas Orr. u-μ p: The unit-scaled maximal update parametrization. In *International Conference on Learning Representations*, 2025.

Blake Bordelon, Lorenzo Noci, Mufan Li, Boris Hanin, and Cengiz Pehlevan. Depthwise hyperparameter transfer in residual networks: Dynamics and scaling limit. In *International Conference on Learning Representations*, 2024.

Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. Broken neural scaling laws. In *International Conference on Learning Representations*, 2023.

Tianqi Chen and Carlos Guestrin. 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.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, et al. Symbolic discovery of optimization algorithms. *Advances in neural information processing systems*, 36:49205–49233, 2023.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

DeepSeek-AI, Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, Huazuo Gao, Kaige Gao, Wenjun Gao, Ruiqi Ge, Kang Guan, Daya Guo, Jianzhong Guo, Guangbo Hao, Zhewen Hao, Ying He, Wenjie Hu, Panpan Huang, Erhang Li, Guowei Li, Jiashi Li, Yao Li, Y. K. Li, Wenfeng Liang, Fangyun Lin, A. X. Liu, Bo Liu, Wen Liu, Xiaodong Liu, Xin Liu, Yiyuan Liu, Haoyu Lu, Shanghao Lu, Fuli Luo, Shirong Ma, Xiaotao Nie, Tian Pei, Yishi Piao, Junjie Qiu, Hui Qu, Tongzheng Ren, Zehui Ren, Chong Ruan, Zhangli Sha, Zhihong Shao, Junxiao Song, Xuecheng Su, Jingxiang Sun, Yaofeng Sun, Minghui Tang, Bingxuan Wang, Peiyi Wang, Shiyu Wang, Yaohui Wang, Yongji Wang, Tong Wu, Y. Wu, Xin Xie, Zhenda Xie, Ziwei Xie, Yiliang Xiong, Hanwei Xu, R. X. Xu, Yanhong Xu, Dejian Yang, Yuxiang You, Shuiping Yu, Xingkai Yu, B. Zhang, Haowei Zhang, Lecong Zhang, Liyue Zhang, Mingchuan Zhang, Minghua Zhang, Wentao Zhang, Yichao Zhang, Chenggang Zhao, Yao Zhao, Shangyan Zhou, Shunfeng Zhou, Qihao Zhu, and Yuheng Zou. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint 2401.02954*, 2024.

Nolan Dey, Bin Claire Zhang, Lorenzo Noci, Mufan Li, Blake Bordelon, Shane Bergsma, Cengiz Pehlevan, Boris Hanin, and Joel Hestness. Don't be lazy: Completep enables compute-efficient deep transformers. *arXiv preprint arXiv:2505.01618*, 2025.

Katie E Everett, Lechao Xiao, Mitchell Wortsman, Alexander A Alemi, Roman Novak, Peter J Liu, Izzeddin Gur, Jascha Sohl-Dickstein, Leslie Pack Kaelbling, Jaehoon Lee, et al. Scaling exponents across parameterizations and optimizers. In *International Conference on Machine Learning*, pp. 12666–12700. PMLR, 2024.

Zhiyuan Fan, Yifeng Liu, Qingyue Zhao, Angela Yuan, and Quanquan Gu. Robust layerwise scaling rules by proper weight decay tuning. *arXiv preprint arXiv:2510.15262*, 2025.

Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, Rui Xin, Marianna Nezhurina, Igor Vasiljevic, Luca Soldaini, Jenia Jitsev, Alex Dimakis, Gabriel Ilharco, Pang Wei Koh, Shuran Song, Thomas Kollar, Yair Carmon, Achal Dave, Reinhard Heckel, Niklas Muennighoff, and Ludwig Schmidt. 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.

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

Matilde Gargiani, Aaron Klein, Stefan Falkner, and Frank Hutter. Probabilistic rollouts for learning curve extrapolation across hyperparameter settings. *arXiv preprint arXiv:1910.04522*, 2019.

Yury Gorishniy, Akim Kotelnikov, and Artem Babenko. Tabm: Advancing tabular deep learning with parameter-efficient ensembling. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 77899–77935, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/c1ba41c694834aeef91ae161711d4939-Paper-Conference.pdf.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. In *Forty-first International Conference on Machine Learning*, 2024.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36:19622–19635, 2023.

David Hall, Ahmed Ahmed, Christopher Chou, Abhinav Garg, Rohith Kuditipudi, Will Held, Nikil Ravi, Herumb Shandilya, Jason Wang, Jason Bolton, Siddharth Karamcheti, Suhas Kotha, Tony Lee, Nelson Liu, Joel Niklaus, Ashwin Ramaswami, Kamyar Salahi, Kaiyue Wen, Chi Heem Wong, Sherry Yang, Ivan Zhou, and Percy Liang. Introducing marin: An open lab for building foundation models, 2025. URL `https://marin.community/blog/2025/05/19/announcement/`.

Tatsunori Hashimoto. Model performance scaling with multiple data sources. In Marina Meila and Tong Zhang (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`.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. Scaling laws for autoregressive generative modeling. *arXiv preprint 2010.14701*, 2020.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 2021.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems*, 2022.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

Michael Y Hu, Jane Pan, Ayush Rajesh Jhaveri, Nicholas Lourie, and Kyunghyun Cho. Neural neural scaling laws. *arXiv preprint arXiv:2601.19831*, 2026.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

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

Stanisław Jastrzębski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

Daniel P Jeong, Zachary Chase Lipton, and Pradeep Kumar Ravikumar. Llm-select: Feature selection with large language models. *Transactions on Machine Learning Research*, 2025.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

Arlind Kadra, Maciej Janowski, Martin Wistuba, and Josif Grabocka. Scaling laws for hyperparameter optimization. *Advances in Neural Information Processing Systems*, 36:47527–47553, 2023.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for llms. *arXiv preprint arXiv:2510.13786*, 2025.

Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijue Chen, Yanru Chen, Yuankun Chen, Yutian Chen, et al. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*, 2025.

Aaron Klein, Stefan Falkner, Jost Tobias Springenberg, and Frank Hutter. Learning curve prediction with bayesian neural networks. In *International Conference on Learning Representations*, 2017.

Atli Kosson, Jeremy Welborn, Yang Liu, Martin Jaggi, and Xi Chen. Weight decay may matter more than mup for learning rate transfer in practice. *arXiv preprint arXiv:2510.19093*, 2025.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations*, 2022.

Binghui Li, Fengling Chen, Zixun Huang, Lean Wang, and Lei Wu. Functional scaling laws in kernel regression: Loss dynamics and learning rate schedules. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025a.

Bingrui Li, Jiaxin Wen, Zhanpeng Zhou, Jun Zhu, and Jianfei Chen. Efficient hyperparameter tuning via trajectory invariance principle. *arXiv preprint arXiv:2509.25049*, 2025b.

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Zhenyu Ding, Haoying Wang, Zili Wang, Shijie Xuyang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part ii, farseer: A refined scaling law in large language models. *arXiv preprint 2506.10972*, 2025c.

Houyi Li, Wenzhen Zheng, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Zhenyu Ding, Haoying Wang, Ning Ding, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part i, step law – optimal hyperparameter scaling law in large language model pretraining. *arXiv preprint arXiv:2503.04715*, 2025d.

Haowei Lin, Haotian Ye, Wenzheng Feng, Quzhe Huang, Yujun Li, Hubert Lim, Zhengrui Li, Xiangyu Wang, Jianzhu Ma, James Zou, et al. Can language models discover scaling laws? *arXiv preprint arXiv:2507.21184*, 2025.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024a.

Chuan Liu, Huanran Chen, Yichi Zhang, Yinpeng Dong, and Jun Zhu. Scaling laws for black box adversarial attacks. *arXiv preprint arXiv:2411.16782*, 2024b.

Hejia Liu, Mochen Yang, and Gediminas Adomavicius. Robustness is important: Limitations of llms for data fitting. *arXiv preprint arXiv:2508.19563*, 2025a.

Jingyuan Liu, Jianlin Su, Xingcheng Yao, Zhejun Jiang, Guokun Lai, Yulun Du, Yidao Qin, Weixin Xu, Enzhe Lu, Junjie Yan, et al. Muon is scalable for llm training. *arXiv preprint arXiv:2502.16982*, 2025b.

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

Nicholas Lourie, Michael Y. Hu, and Kyunghyun Cho. Scaling laws are unreliable for downstream tasks: A reality check. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (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/.

Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, Marek Cygan, and Sebastian Jaszczur. Scaling laws for fine-grained mixture of experts. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (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.

Kairong Luo, Haodong Wen, Shengding Hu, Zhenbo Sun, Zhiyuan Liu, Maosong Sun, Kaifeng Lyu, and Wenguang Chen. A multi-power law for loss curve prediction across learning rate schedules. In *International Conference on Learning Representations*, 2025.

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

Martin Marek, Sanae Lotfi, Aditya Somasundaram, Andrew Gordon Wilson, and Micah Goldblum. 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.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

OLMo Team, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, et al. 2 olmo 2 furious. *arXiv preprint arXiv:2501.00656*, 2024.

Tomer Porian, Mitchell Wortsman, Jenia Jitsev, Ludwig Schmidt, and Yair Carmon. Resolving discrepancies in compute-optimal scaling of language models. *arXiv:2406.19146*, 2024.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/14491b756b3a51daac41c24863285549-Paper.pdf.

Shikai Qiu, Lechao Xiao, Andrew Gordon Wilson, Jeffrey Pennington, and Atish Agarwala. Scaling collapse reveals universal dynamics in compute-optimally trained neural networks. In *International Conference on Machine Learning*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Herilalaina Rakotoarison, Steven Adriaensen, Neeratyoy Mallik, Samir Garibov, Eddie Bergman, and Frank Hutter. In-context freeze-thaw bayesian optimization for hyperparameter optimization. In *International Conference on Machine Learning*, pp. 41982–42008. PMLR, 2024.

Yangjun Ruan, Chris J Maddison, and Tatsunori B Hashimoto. Observational scaling laws and the predictability of language model performance. *Advances in Neural Information Processing Systems*, 37:15841–15892, 2024.

Xian Shuai, Yiding Wang, Yimeng Wu, Xin Jiang, and Xiaozhe Ren. Scaling law for language models training considering batch size. *arXiv preprint arXiv:2412.01505*, 2024.

Xingyou Song, Oscar Li, Chansoo Lee, Bangding Yang, Daiyi Peng, Sagi Perel, and Yutian Chen. Omnipred: Language models as universal regressors. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=t9c3pfrR1X.

Howe Tissue, Venus Wang, and Lu Wang. Scaling law with learning rate annealing. *arXiv preprint arXiv:2408.11029*, 2024.

Robert Vacareanu, Vlad Andrei Negru, Vasile Suciu, and Mihai Surdeanu. 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.

Siqi Wang, Zhengyu Chen, Bei Li, Keqing He, Min Zhang, and Jingang Wang. Scaling laws across model architectures: A comparative analysis of dense and MoE models in large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (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/.

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

Kaiyue Wen, David Hall, Tengyu Ma, and Percy Liang. Fantastic pretraining optimizers and where to find them. *arXiv preprint arXiv:2509.02046*, 2025.

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

Xingyu Xie, Kuangyu Ding, Shuicheng Yan, Kim-Chuan Toh, and Tianwen Wei. Optimization hyper-parameter laws for large language models. *arXiv preprint arXiv:2409.04777*, 2024.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Ge Yang, Edward Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (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.

Greg Yang and Edward J. Hu. Tensor programs IV: feature learning in infinite-width neural networks. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research, pp. 11727–11737. PMLR, 2021. URL http://proceedings.mlr.press/v139/yang21c.html.

Hanlin Zhang, Depen Morwani, Nikhil Vyas, Jingfeng Wu, Difan Zou, Udaya Ghai, Dean Foster, and Sham M Kakade. How does critical batch size scale in pre-training? In *International Conference on Learning Representations*, 2025.

Yunhua Zhou, Shuhao Xing, Junhao Huang, Xipeng Qiu, and Qipeng Guo. How to set the learning rate for large-scale pre-training? *arXiv preprint arXiv:2601.05049*, 2026.

A Additional related work

More works on scaling laws Scaling law is also applied to broader domains and settings, including multimodal 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 extend 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). Recently, Lin et al. (2025) propose 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.

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; Yang & Hu, 2021). 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.

More Works on Foundation models as regressors 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).

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.

B Experimental details

B.1 Data processing

Prediction targets Since the per-step pretraining 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.

Table 1: 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	–

Training configuration used In Table 1, we list all training-configuration fields used as inputs to NCPL. Categorical fields are encoded using the backbone language model's 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 set in Marin's ablations, 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 when constructing inputs, so that different fields fall into a comparable range. The scaling factors are reported in Table 1.

Example training samples for final-loss prediction and loss curve prediction are shown in Figure 10.

B.2 Training setup and hyperparameters

Training NCPL for final loss prediction 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 and 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. We fine-tune the model using `float32` precision.

Training NCPL for loss curve prediction For each run we uniformly sample up to 30 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 (Figure 10 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.

Fine-tuning NCPL for MoE settings We construct a grouped MoE scaling-law training dataset from public training logs of Li et al. (2025d) with 4 groups, where each group is defined by a triplet of data size D, model size N, and activation size A. For training, we use four groups: (1) $D = 8$B, $N = 2150$M, $A = 187.974$M; (2) $D = 8$B, $N = 2150$M, $A = 232.579$M; (3) $D = 2$B, $N = 2155$M, $A = 590.436$M; and (4) $D = 4$B, $N = 2156$M, $A = 1241.270$M. These four groups contain 179 training samples in total. For fine-tuning, we find that training is more stable when mixing the grouped MoE training data with the original training data used to obtain the initialization checkpoint. Therefore, for fine-tuning we use a 50%/50% mixture of (i) the grouped MoE training dataset described above and (ii) the original training dataset used for training the NCPL predictor. For a fair comparison between fine-tuning and training solely on StepLaw-MoE dataset, we sweep hyperparameter sets for both settings. Specifically, we vary the second-stage learning rate over $\{10^{-5}, 6 \times 10^{-6}, 10^{-6}\}$ and training epochs in $\{100, 200, 400, 600\}$.

For the hyperparameter-selection analysis, we visualize the four $D = 20$B groups: (1) $D = 20$B, $N = 2150$M, $A = 187.974$M; (2) $D = 20$B, $N = 2150$M, $A = 232.579$M; (3) $D = 20$B, $N = 2155$M, $A = 590.436$M; and (4) $D = 20$B, $N = 2156$M, $A = 1241.270$M. For each model, we fit a quadratic surface over the predicted loss landscape in the $(\text{learning rate}, \text{batch size})$ plane to estimate the optimum. If the fitted minimum for the training-from-scratch model is invalid, e.g., when it falls outside the explored hyperparameter region, we fall back to the best hyperparameter point actually observed in the sweep. Additional results are provided in Figure 21.

Computation resources Training NCPL via fine-tuning Qwen3-1.7B (Yang et al., 2025) uses 8 H100 GPUs for approximately 5.5 hours

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 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 Loss Curve Prediction with Autoregressive Generation

In Section 4.2, we use NCPL to predict the entire loss curve from the training configuration by predicting the loss at each training step independently. We also considered an autoregressive loss-curve completion setting: the model is given the early part of the loss curve, together with the training configuration, and is asked to predict the remaining loss curve.

Specifically, we fine-tune SmolLM2-135M (Allal et al., 2025) on the Marin dataset for 20 epochs. Although the prediction error under teacher forcing is relatively low, the error becomes substantially larger when the model is rolled out autoregressively. This performance is significantly worse than the independent step-wise prediction approach used in the main text. The results are shown in Figure 11, Figure 12, Figure 13, and Table 2.

Table 2: The autoregressively trained loss-curve prediction model suffers from severe error accumulation during rollout. AR and TF denote autoregressive rollout and teacher forcing, respectively.

Split	AR MAE	AR RMSE	TF MAE	TF RMSE
Train	0.1004	0.1622	0.0390	0.0926
In-val	0.1109	0.2181	0.0417	0.1031
OOD-val	0.0919	0.1061	0.0398	0.0491

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)	**0.0120**	**0.0184**	**0.9777**	**0.0198**	**0.0270**	**0.9446**
	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)	**0.0103**	**0.0155**	0.9958	**0.0237**	**0.0339**	**0.9863**
	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

C.2 More results of fine-tuned NCPL

Final pretraining loss prediction results across learning rates and batch sizes on the StepLaw dataset are shown in Figure 14 (ID) and Figure 15 (OOD). Hyperparameter selection results for all held-out model–data size pairs are shown in Figure 16 and Figure 17. 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 more training logs. Loss-curve prediction results on the Marin dataset across different optimizers (ID and OOD) are presented in Figure 20. Figure 18 and Figure 19 show randomly sampled loss curve prediction results on ID and OOD runs, without cherry-picking.

C.3 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: predicting the final loss directly, rather than the residual with respect to the Chinchilla baseline, and tokenizing numerical fields with a standard tokenizer, rather than encoding them with a two-layer MLP. Results shown in Table 3 show consistent gains from both residual prediction and numerical token encoding.

C.4 Additional results of NCPL with training from scratch

Figure 22 and Figure 23 show the predicted vs. ground-truth final pretraining loss of NCPL trained from scratch, using 1.7B model and 135M model respectively.

Table 4: Comparison of NCPL against XGBoost, CatBoost, TabM, and Chinchilla-law baselines. NCPL, XGBoost, CatBoost, and TabM values are the mean over 5 random seeds; for variance, see Table 6, 7, 8. We report mean absolute error (MAE), root mean squared error (RMSE), and Spearman correlation (ρ). For final-loss prediction on the StepLaw dataset, where only the learning rate and batch size vary, tabular baselines are competitive. In contrast, NCPL achieves the best results or ranks a close second for final-loss prediction on the Marin dataset, which features more diverse configurations, and for loss-curve prediction on both datasets. **Bold** = best result; underline = second best.

Data	Method	Final-loss prediction						Loss-curve prediction			
		In-distribution			Out-of-distribution			In-distribution		Out-of-distribution	
		MAE↓	RMSE↓	ρ ↑	MAE↓	RMSE↓	ρ ↑	MAE↓	RMSE↓	MAE↓	RMSE↓
Marin	NCPL (Ours)	**0.0120**	**0.0184**	0.9777	**0.0198**	0.0270	0.9446	**0.0272**	**0.0473**	**0.0379**	**0.0642**
	XGBoost	0.0188	0.0282	0.9744	0.0328	0.0379	0.9162	0.0292	0.0549	0.0396	0.0693
	CatBoost	0.0140	0.0210	**0.9821**	0.0313	0.0352	**0.9473**	0.0296	0.0547	0.0868	0.1953
	TabM	0.0196	0.0291	0.9367	0.0208	**0.0269**	0.9132	0.0335	0.0558	0.1945	0.7745
	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 (Ours)	0.0103	0.0155	**0.9958**	0.0237	0.0339	0.9863	**0.0273**	**0.0992**	0.0422	0.1722
	XGBoost	**0.0096**	0.0163	0.9946	0.0255	0.0345	0.9852	0.0278	0.1095	0.0460	0.1821
	CatBoost	0.0118	0.0184	0.9930	0.0241	0.0322	0.9880	0.0291	0.1021	**0.0405**	**0.1717**
	TabM	0.0102	**0.0149**	0.9953	**0.0143**	**0.0242**	**0.9922**	0.0346	0.0995	0.0538	0.1756
	Chinchilla Law	0.0704	0.0949	0.9139	0.0440	0.0670	0.9141	–	–	–	–

Table 5: We ablate different versions of NCPL, by fine-tuning (*ft*) versus training from scratch (*scratch*) with two backbone sizes (1.7B and 135M). Results are mean over 5 random seeds; for variance, see Table 9, 10, 11. We report mean absolute error (MAE), root mean squared error (RMSE), and Spearman correlation (ρ). On StepLaw dataset, where only a small set of hyperparameters vary (learning rate and batch size), NCPL (scratch) performs comparably or slightly better than NCPL (ft); in contrast, on Marin dataset, which has more diverse configurations, NCPL (ft) provides a clear advantage over NCPL (scratch)

Data	Method	Final-loss prediction						Loss Curve Prediction			
		In-distribution			Out-of-distribution			In-distribution		Out-of-distribution	
		MAE↓	RMSE↓	ρ ↑	MAE↓	RMSE↓	ρ ↑	MAE↓	RMSE↓	MAE↓	RMSE↓
Marin	NCPL (ft, 1.7B)	**0.0120**	0.0184	0.9777	**0.0198**	**0.0270**	**0.9446**	0.0272	0.0473	**0.0379**	**0.0642**
	NCPL (scratch, 1.7B)	**0.0120**	**0.0175**	**0.9784**	0.0246	0.0332	0.9408	0.0280	0.0487	0.0390	0.0772
	NCPL (scratch, 135M)	0.0130	0.0198	0.9766	0.0220	0.0305	0.9368	**0.0270**	**0.0462**	0.0385	0.0858
StepLaw	NCPL (ft, 1.7B)	0.0103	0.0155	**0.9958**	0.0237	0.0339	0.9863	0.0273	0.0992	0.0422	0.1722
	NCPL (scratch, 1.7B)	**0.0085**	0.0142	0.9957	0.0236	0.0318	0.9894	**0.0251**	**0.0946**	0.0417	0.1718
	NCPL (scratch, 135M)	0.0086	**0.0142**	0.9957	**0.0219**	**0.0303**	**0.9902**	0.0259	0.0955	**0.0416**	**0.1712**

Table 6: Comparison of NCPL with XGBoost, CatBoost, TabM, and Chinchilla-law baselines for final-loss prediction on the ID validation set. Results for NCPL, XGBoost, CatBoost, and TabM are averaged over 5 random seeds. **Bold** = best result; underline = second best.

Data	Method	MAE↓	RMSE↓	ρ ↑
Marin	NCPL (Ours)	$\mathbf{0.0120}_{\pm 0.0010}$	$\mathbf{0.0184}_{\pm 0.0013}$	$0.9777_{\pm 0.0044}$
	XGBoost	$0.0188_{\pm 0.0010}$	$0.0282_{\pm 0.0014}$	$0.9744_{\pm 0.0013}$
	CatBoost	$0.0140_{\pm 0.0007}$	$0.0210_{\pm 0.0006}$	$\mathbf{0.9821}_{\pm 0.0016}$
	TabM	$0.0196_{\pm 0.0005}$	$0.0291_{\pm 0.0008}$	$0.9367_{\pm 0.0103}$
	Chinchilla Law	0.0566	0.0676	0.9004
	Per-optimizer Chinchilla	0.0416	0.0575	0.8268
StepLaw	NCPL (Ours)	$0.0103_{\pm 0.0006}$	$0.0155_{\pm 0.0007}$	$\mathbf{0.9958}_{\pm 0.0006}$
	XGBoost	$\mathbf{0.0096}_{\pm 0.0004}$	$0.0163_{\pm 0.0004}$	$0.9946_{\pm 0.0002}$
	CatBoost	$0.0118_{\pm 0.0006}$	$0.0184_{\pm 0.0009}$	$0.9930_{\pm 0.0006}$
	TabM	$0.0102_{\pm 0.0009}$	$\mathbf{0.0149}_{\pm 0.0014}$	$0.9953_{\pm 0.0006}$
	Chinchilla Law	0.0704	0.0949	0.9139

```
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: 0.01023
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 10: 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.

Table 7: Comparison of NCPL with XGBoost, CatBoost, TabM, and Chinchilla-law baselines for final-loss prediction on the OOD validation set. Results for NCPL, XGBoost, CatBoost, and TabM are averaged over 5 random seeds. **Bold** = best result; underline = second best.

Data	Method	MAE↓	RMSE↓	ρ ↑
Marin	NCPL (Ours)	**0.0198** ± 0.0046	0.0270 ± 0.0039	0.9446 ± 0.0084
	XGBoost	0.0328 ± 0.0025	0.0379 ± 0.0023	0.9162 ± 0.0051
	CatBoost	0.0313 ± 0.0025	0.0352 ± 0.0022	**0.9473** ± 0.0040
	TabM	0.0208 ± 0.0024	**0.0269** ± 0.0020	0.9132 ± 0.0143
	Chinchilla Law	0.0240	0.0326	0.7670
	Per-optimizer Chinchilla	0.0211	0.0278	0.8062
StepLaw	NCPL (Ours)	0.0237 ± 0.0013	0.0339 ± 0.0024	0.9863 ± 0.0029
	XGBoost	0.0255 ± 0.0006	0.0345 ± 0.0008	0.9852 ± 0.0007
	CatBoost	0.0241 ± 0.0011	0.0322 ± 0.0012	0.9880 ± 0.0008
	TabM	**0.0143** ± 0.0005	**0.0242** ± 0.0020	**0.9922** ± 0.0009
	Chinchilla Law	0.0440	0.0670	0.9141

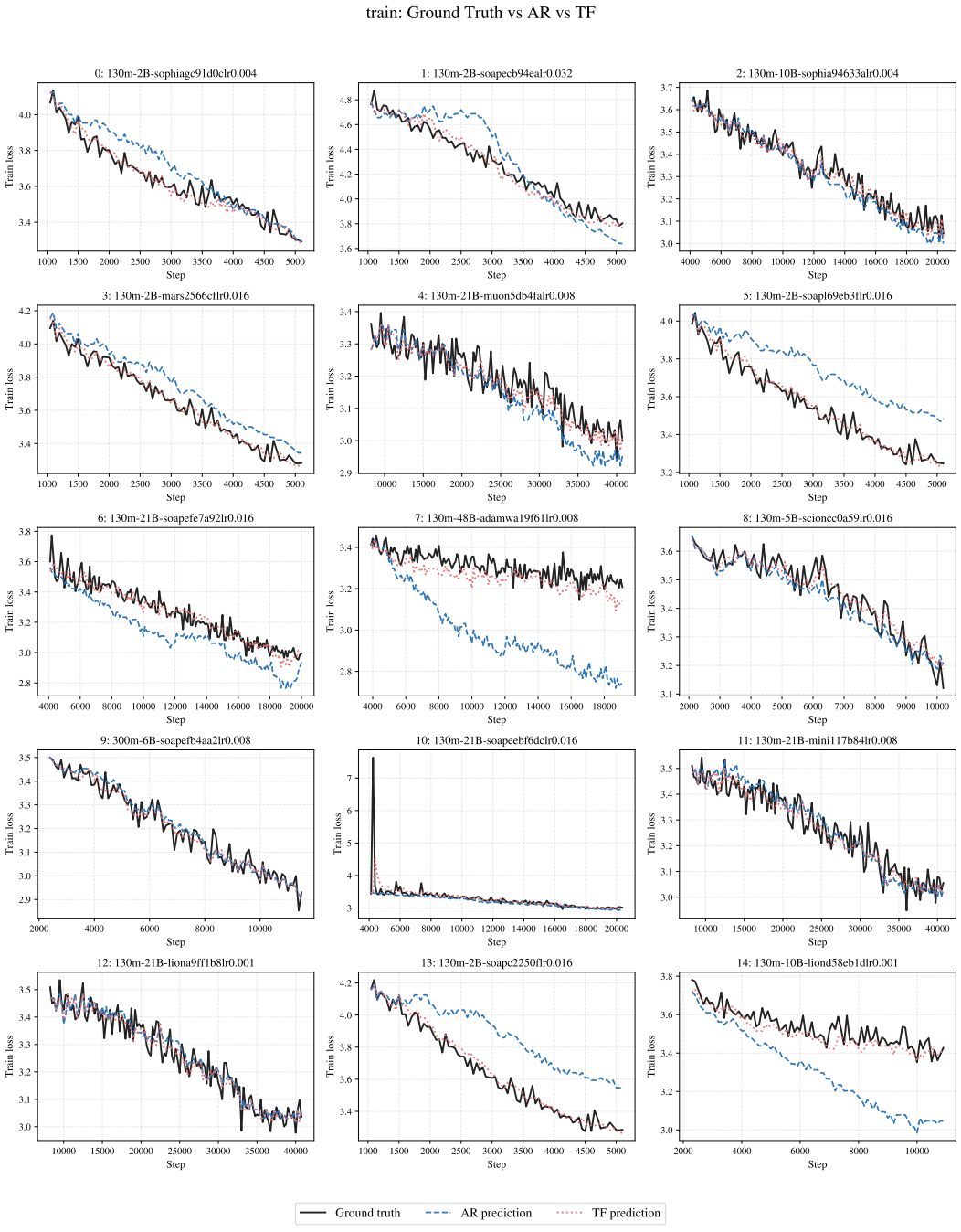


Figure 11: **autoregressive loss-curve prediction on the training set.** autoregressive rollout performs worse than independent step-wise loss prediction due to severe error accumulation.

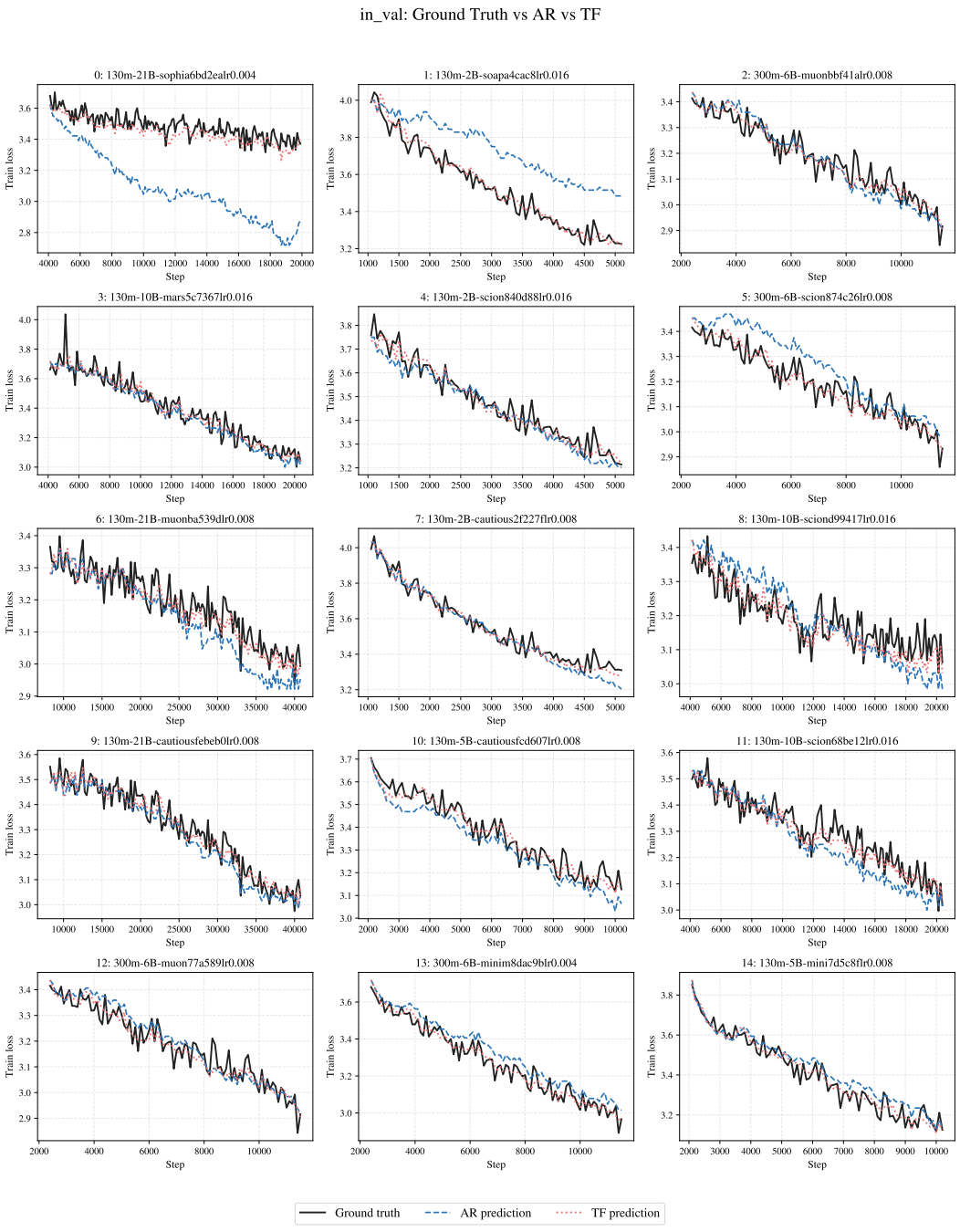


Figure 12: **autoregressive loss-curve prediction on the ID validation set.** autoregressive rollout performs worse than independent step-wise loss prediction due to severe error accumulation.

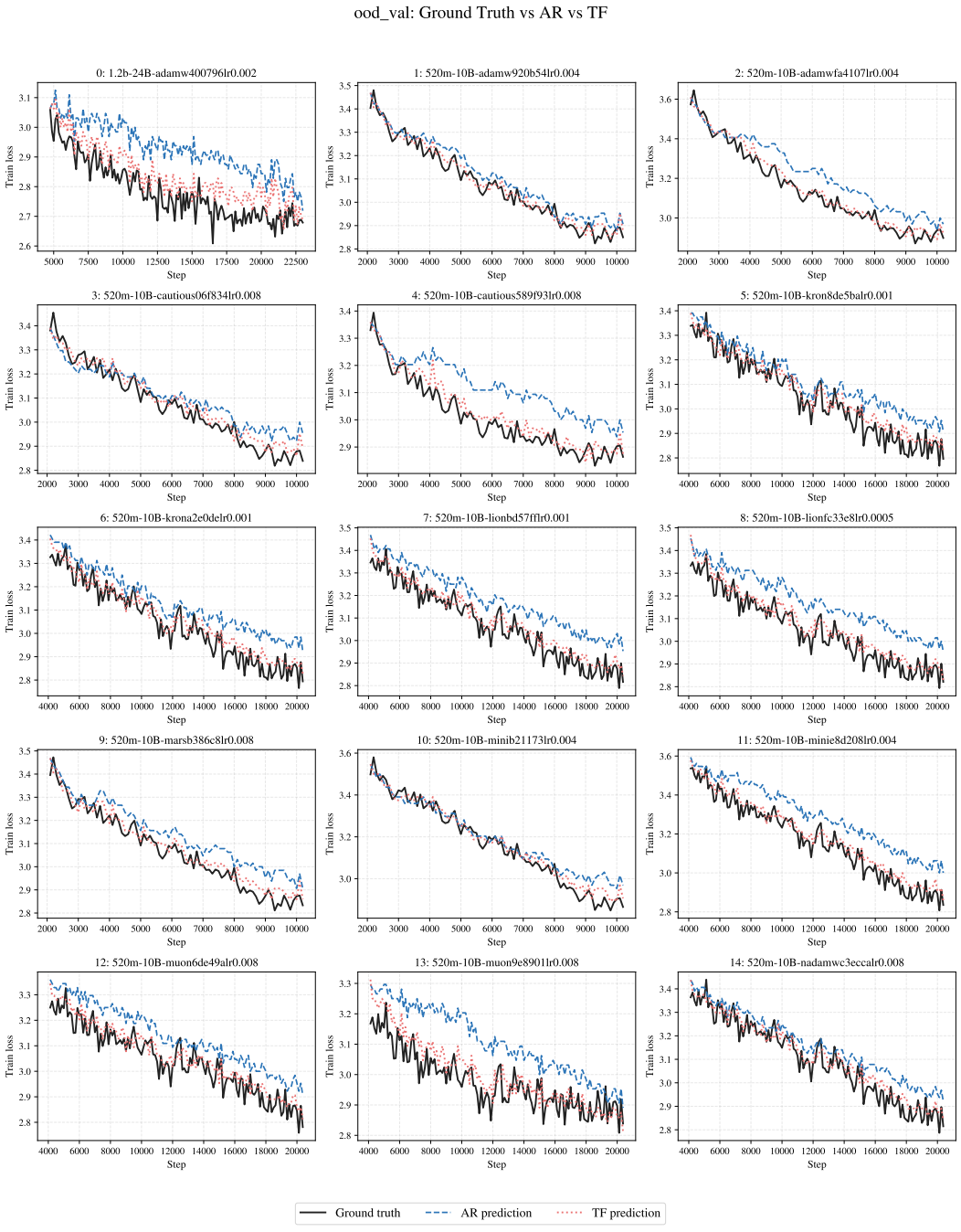


Figure 13: **autoregressive loss-curve prediction on the OOD validation set.** autoregressive rollout performs worse than independent step-wise loss prediction due to severe error accumulation.

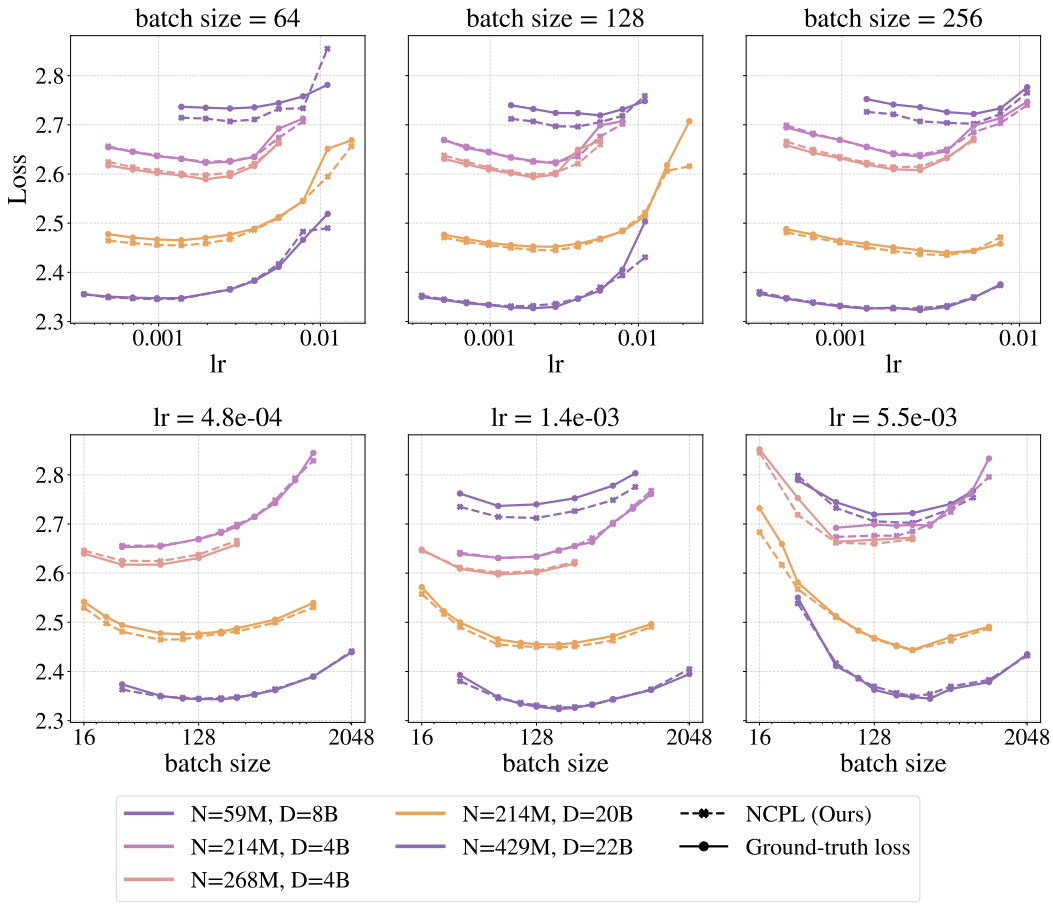


Figure 14: Final-loss prediction across learning rates and batch sizes for all 5 ID held-out (N, D) pairs.

Table 8: Comparison of NCPL with XGBoost, CatBoost, TabM, and Chinchilla-law baselines for loss-curve prediction. Results for NCPL, XGBoost, CatBoost, and TabM are averaged over 5 random seeds. **Bold** = best result; underline = second best.

Data	Method	In-distribution		Out-of-distribution	
		MAE↓	RMSE↓	MAE↓	RMSE↓
Marin	NCPL (Ours)	**0.0272** ±0.0046	**0.0473** ±0.0059	**0.0379** ±0.0056	**0.0642** ±0.0046
	XGBoost	0.0292 ±0.0011	0.0549 ±0.0008	0.0396 ±0.0016	0.0693 ±0.0033
	CatBoost	0.0296 ±0.0007	0.0547 ±0.0017	0.0868 ±0.0383	0.1953 ±0.0884
	TabM	0.0335 ±0.0014	0.0558 ±0.0047	0.1945 ±0.0909	0.7745 ±0.2987
StepLaw	NCPL (Ours)	**0.0273** ±0.0037	**0.0992** ±0.0022	0.0422 ±0.0044	0.1722 ±0.0014
	XGBoost	0.0278 ±0.0004	0.1095 ±0.0015	0.0460 ±0.0009	0.1821 ±0.0015
	CatBoost	0.0291 ±0.0006	0.1021 ±0.0010	**0.0405** ±0.0041	**0.1717** ±0.0013
	TabM	0.0346 ±0.0026	0.0995 ±0.0018	0.0538 ±0.0034	0.1756 ±0.0034

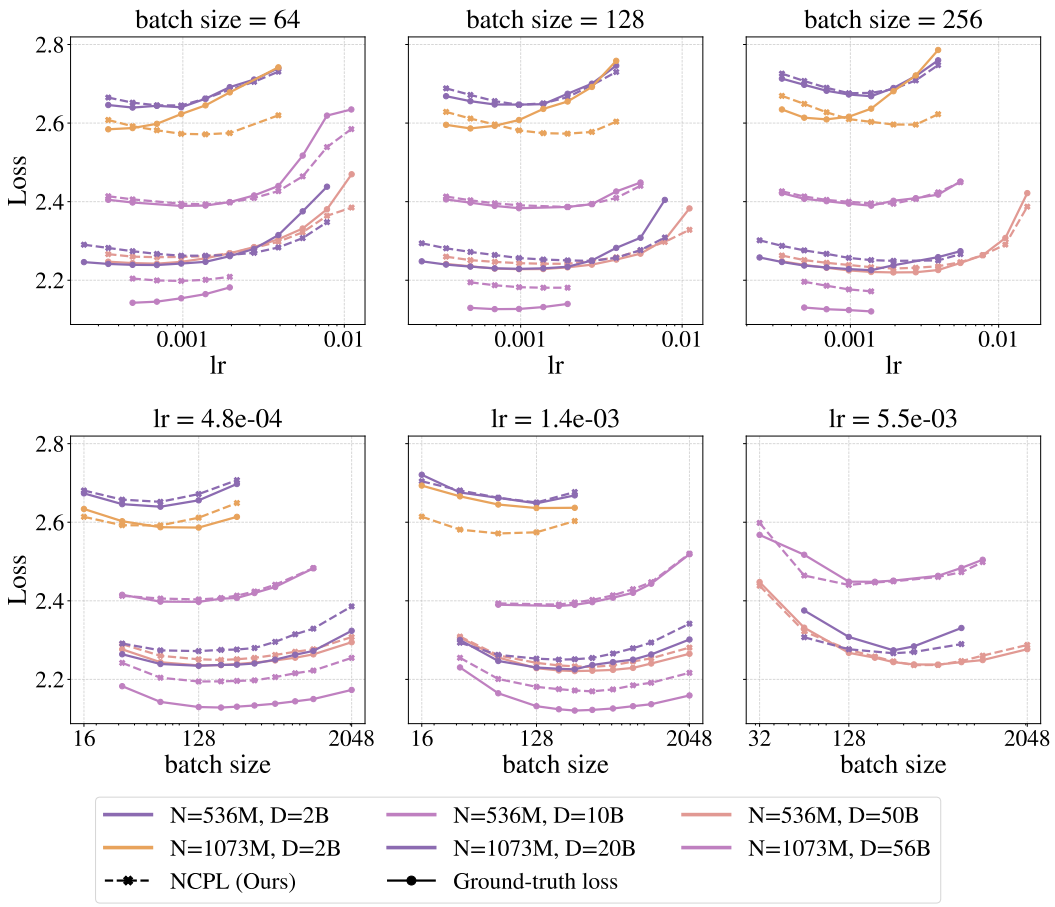


Figure 15: Final-loss prediction across learning rates and batch sizes for 6 OOD held-out (N, D) pairs.

Table 9: Ablation of different versions of NCPL for final-loss prediction on the ID validation set. Mean $\pm$ std over 5 random seeds.

Data	Method	MAE↓	RMSE↓	ρ ↑
Marin	NCPL (ft, 1.7B)	**0.0120** $\pm$0.0010	0.0184 $\pm$0.0013	0.9777 $\pm$0.0044
	NCPL (scratch, 1.7B)	**0.0120** $\pm$0.0006	**0.0175** $\pm$0.0004	**0.9784** $\pm$0.0040
	NCPL (scratch, 135M)	0.0130 $\pm$0.0006	0.0198 $\pm$0.0011	0.9766 $\pm$0.0023
StepLaw	NCPL (ft, 1.7B)	0.0103 $\pm$0.0006	0.0155 $\pm$0.0007	**0.9958** $\pm$0.0006
	NCPL (scratch, 1.7B)	**0.0085** $\pm$0.0006	0.0142 $\pm$0.0005	0.9957 $\pm$0.0002
	NCPL (scratch, 135M)	0.0086 $\pm$0.0006	**0.0142** $\pm$0.0008	0.9957 $\pm$0.0006

Table 10: Ablation of different versions of NCPL for final-loss prediction on the OOD validation set. Mean $\pm$ std over 5 random seeds.

Data	Method	MAE↓	RMSE↓	ρ ↑
Marin	NCPL (ft, 1.7B)	**0.0198** $\pm$0.0046	**0.0270** $\pm$0.0039	**0.9446** $\pm$0.0084
	NCPL (scratch, 1.7B)	0.0246 $\pm$0.0024	0.0332 $\pm$0.0022	0.9408 $\pm$0.0054
	NCPL (scratch, 135M)	0.0220 $\pm$0.0058	0.0305 $\pm$0.0049	0.9368 $\pm$0.0121
StepLaw	NCPL (ft, 1.7B)	0.0237 $\pm$0.0013	0.0339 $\pm$0.0024	0.9863 $\pm$0.0029
	NCPL (scratch, 1.7B)	0.0236 $\pm$0.0013	0.0318 $\pm$0.0010	0.9894 $\pm$0.0015
	NCPL (scratch, 135M)	**0.0219** $\pm$0.0021	**0.0303** $\pm$0.0025	**0.9902** $\pm$0.0024

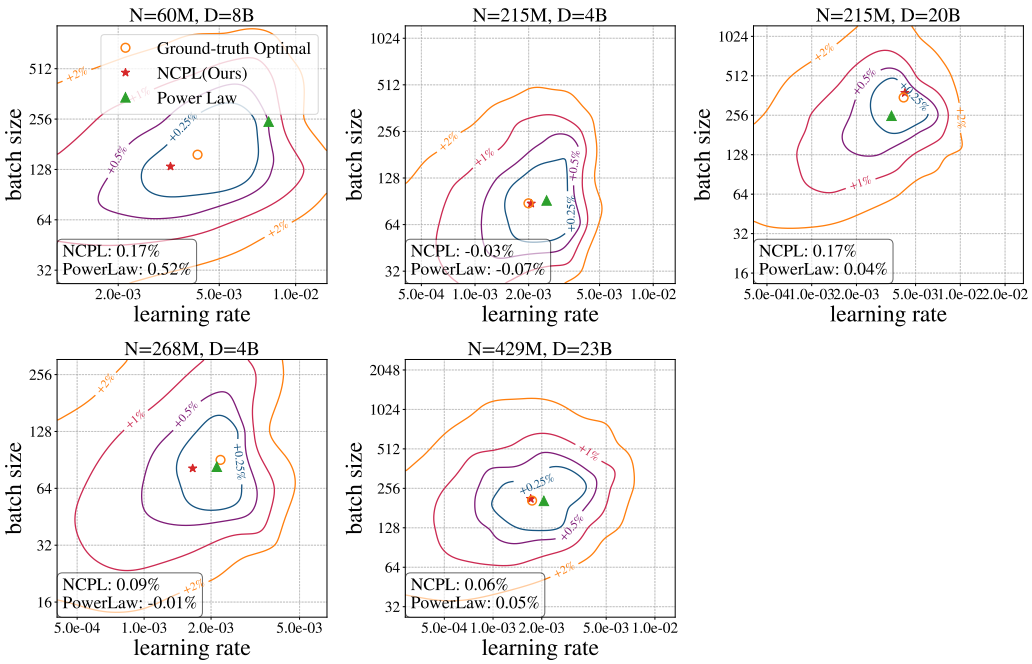


Figure 16: Predicted optimal learning rates and batch sizes for NCPL and the power law baseline on all 5 held-out ID (N, D) pairs, together with their relative losses.

Table 11: Ablation of different versions of NCPL for loss-curve prediction on the OOD validation set. Mean $\pm$ std over 5 random seeds.

Data	**Method**	In-distribution		Out-of-distribution	
		MAE↓	RMSE↓	MAE↓	RMSE↓
Marin	NCPL (ft, 1.7B)	<u>0.0272</u> $\pm$0.0046	<u>0.0473</u> $\pm$0.0059	**0.0379** $\pm$0.0056	**0.0642** $\pm$0.0046
	NCPL (scratch, 1.7B)	0.0280 $\pm$0.0028	0.0487 $\pm$0.0054	0.0390 $\pm$0.0010	<u>0.0772</u> $\pm$0.0198
	NCPL (scratch, 135M)	**0.0270** $\pm$0.0035	**0.0462** $\pm$0.0048	<u>0.0385</u> $\pm$0.0062	0.0858 $\pm$0.0278
StepLaw	NCPL (ft, 1.7B)	0.0273 $\pm$0.0037	0.0992 $\pm$0.0022	0.0422 $\pm$0.0044	0.1722 $\pm$0.0014
	NCPL (scratch, 1.7B)	**0.0251** $\pm$0.0015	**0.0946** $\pm$0.0008	<u>0.0417</u> $\pm$0.0037	<u>0.1718</u> $\pm$0.0018
	NCPL (scratch, 135M)	<u>0.0259</u> $\pm$0.0046	<u>0.0955</u> $\pm$0.0013	**0.0416** $\pm$0.0054	**0.1712** $\pm$0.0032

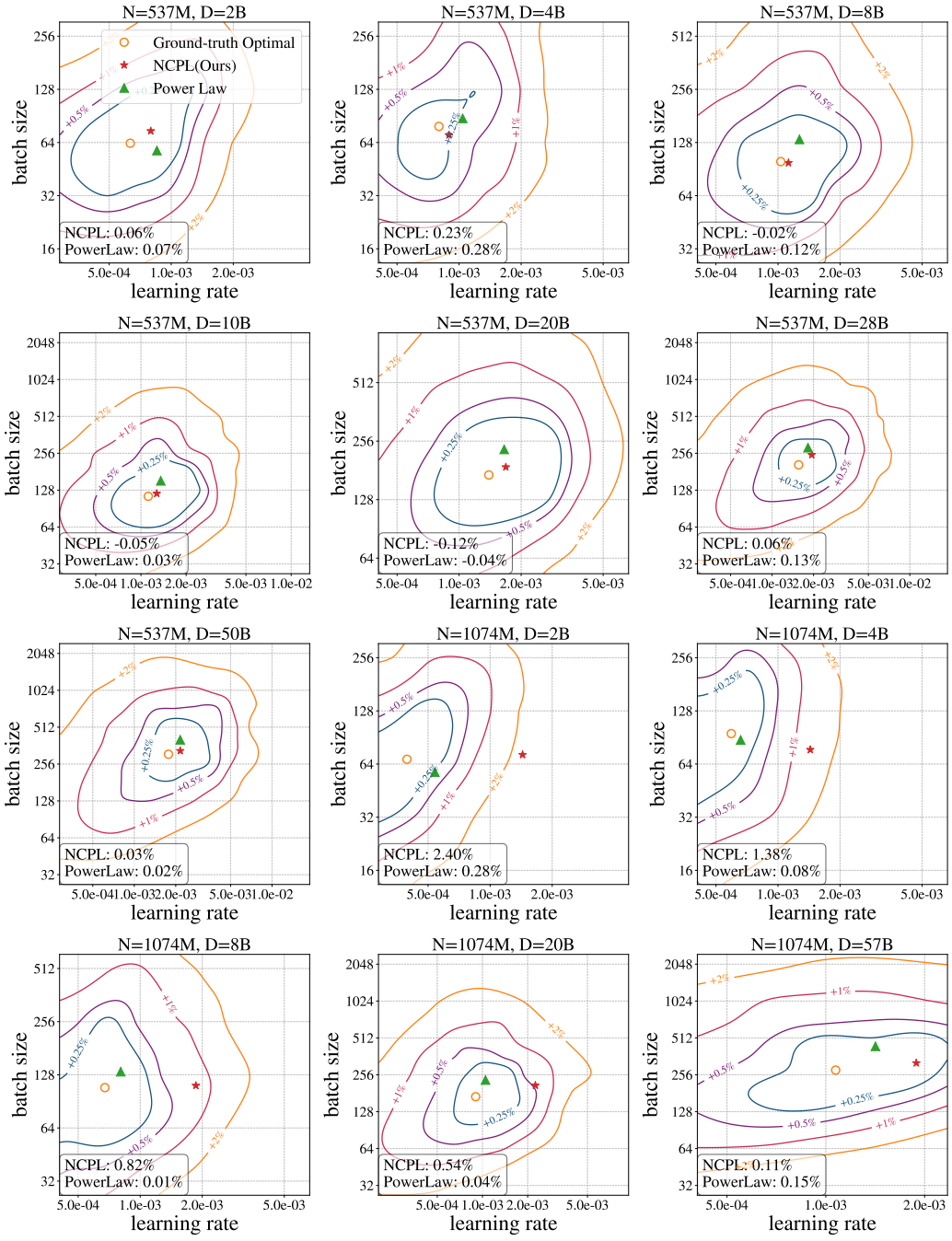


Figure 17: Predicted optimal learning rates and batch sizes for NCPL and the power law baseline on all 12 held-out OOD (N, D) pairs, together with their relative losses.

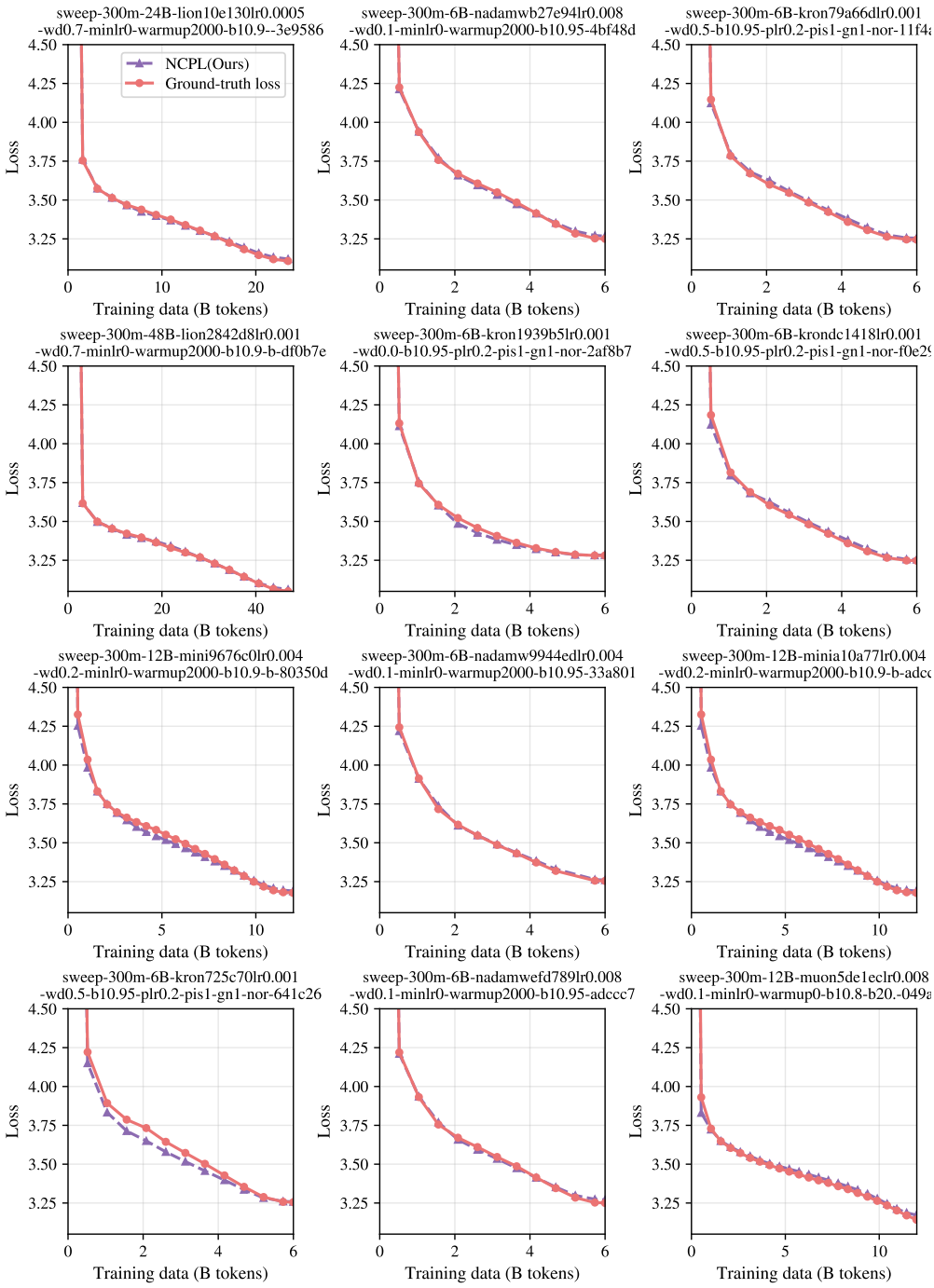


Figure 18: Loss-curve prediction on 12 randomly sampled ID runs (Marin, 300M). Each subplot title is the corresponding run name from the original Wandb project (Wen et al., 2025).

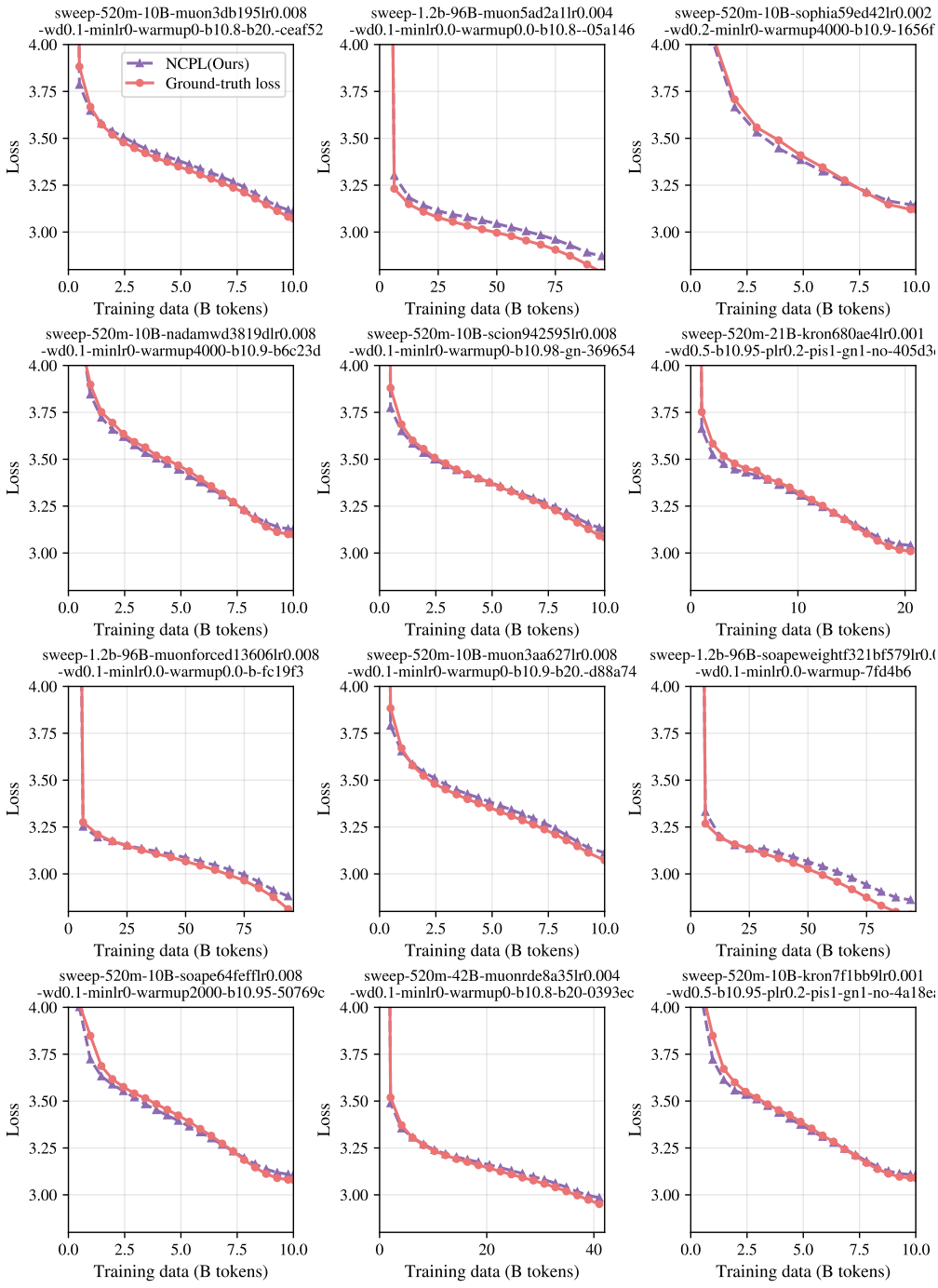


Figure 19: Loss-curve prediction on 12 randomly sampled OOD runs (Marin). Each subplot title is the corresponding run name from the original Wandb project (Wen et al., 2025).

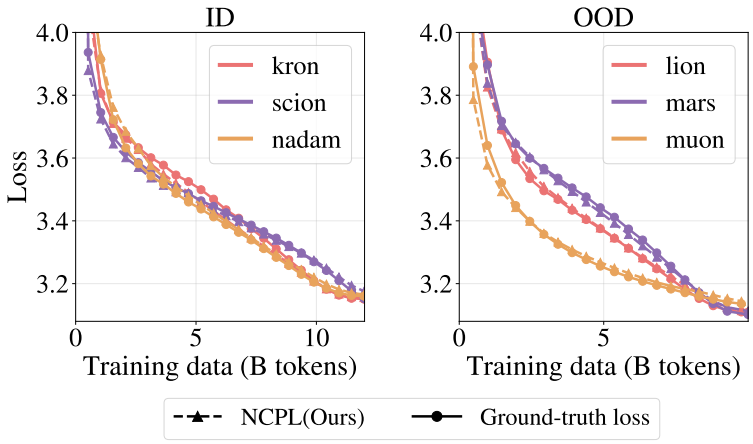


Figure 20: **Loss curve prediction.** Ground-truth and predicted pretraining loss curves under different optimizer settings on the ID and OOD validation sets. NCPL closely tracks the overall trajectories and captures optimizer-specific curve shapes. Setting: Marin Dataset. **Left:** ID validation, $N = 300$M, $D = 12$B, optimizers Kron/Scion/Nadam. **Right:** OOD validation, $N = 520$M, $D = 10$B, optimizers Lion/Mars/Muon. Results of loss curves under different hyperparameters are shown in Figure 5.

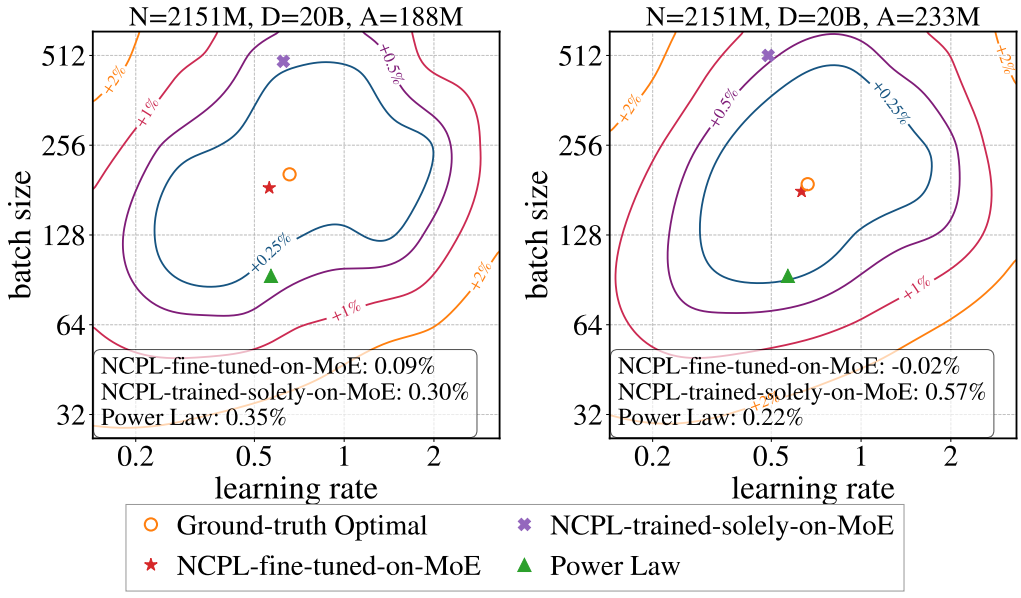


Figure 21: Fine-tuning NCPL for MoE settings with two additional activated-parameter sizes.

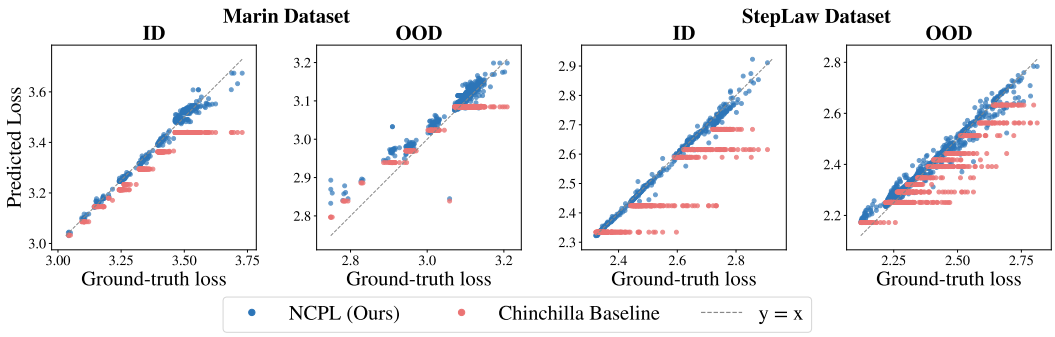


Figure 22: Predicted vs. ground-truth final pretraining loss of NCPL trained from scratch (1.7B model).

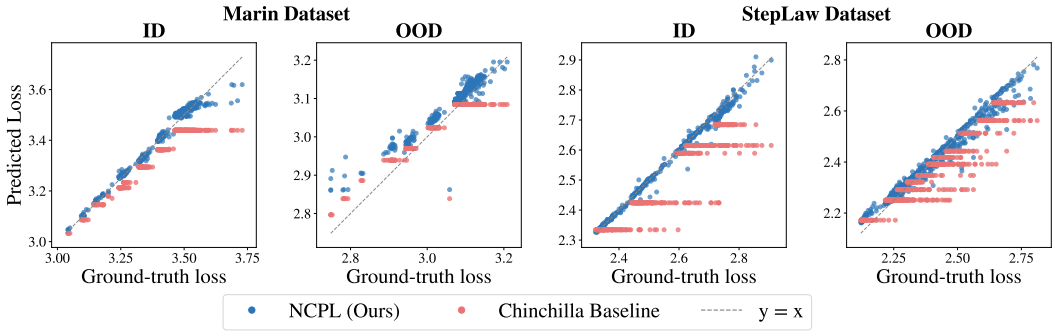


Figure 23: Predicted vs. ground-truth final pretraining loss of NCPL trained from scratch (135M model).