Monotonic NMF: Accurate and Data-efficient Prediction of Language Model Fine-tuning Loss

Abstract

We present **Monotonic Non-negative Matrix Factorization (M-NMF)**, which decomposes loss curves into a set of smooth, monotonically non-increasing additive basis. Crucially, we discover that fine-tuning loss curves inherently exhibit **low-rank structure**: only three basis functions are sufficient to reconstruct loss curves with minimal error. Leveraging this insight, we apply M-NMF to the task of predicting complete fine-tuning loss curves from early training observations. Existing methods rely on extrapolating analytical forms, generally some variant of the power law. In contrast, our approach, which utilizes basis functions directly learned from historical data, achieves significantly improved predictive accuracy. Across all experiment domains, M-NMF requires **40% fewer training examples** compared to standard power law extrapolation to consistently predict within 0.05 of the final normalized loss. Most notably, in the challenging out-of-distribution setting, M-NMF achieves this prediction accuracy using just **9,5656 examples—less than 14% of the full dataset and 65% less data than power law extrapolation**. Finally, we provide an in-depth analysis explaining why M-NMF identifies nearly optimal basis functions and explore how these learned basis functions differ across training hyperparameter choices.

1 Introduction

Language models have rapidly become foundational tools across diverse applications, driven primarily by their remarkable capability to generalize learned linguistic patterns to new, specialized domains. Supervised fine-tuning emerged as one of the the predominant approaches to optimize language model performance on specific tasks. However, despite fine-tuning's widespread adoption, understanding and effectively predicting the dynamics of fine-tuning loss remains challenging. Fine-tuning loss curves often exhibit complex, non-linear behavior, typically characterized by three distinct phases: an initial "pre-power" where loss is relatively flat, followed by the more widely studied "power-law" phase of linear improvement in log-log scale, and concluding with a "convergence" phase, where performance gains diminish significantly. The precise form of these phases—and the transitions between them—varies greatly depending on the data, model architecture, and hyperparameter choices, making generalizable analytical modeling extremely difficult.

Fine-tuning requires collecting high quality data, which is a time consuming and expensive process. In an ideal scenario, practitioners could accurately forecast the fine-tuning loss of language models using only a small initial dataset. Such predictive capability would enable informed decisions about whether and how much additional data collection is necessary to reach desired performance targets, thereby optimizing resource allocation and significantly reducing computational and data acquisition costs. Additionally, predicting loss curves is core to many research methods. For example, Jiang et al. [2024] use a power law form to predict loss for specific data domains, and use it to inform dataset curation. However, they note a consistent overestimate of loss with the power law. Qin et al. [2025] use a rectified power law to predict language model fine-tuning error on synthetic data, and use the prediction to inform data generation and the maximum possible performance of the models.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

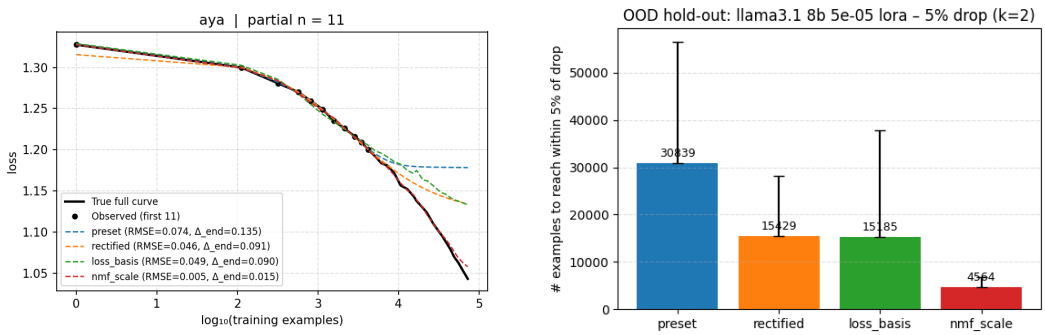


Figure 1: (a) A loss curve prediction on the Aya dataset, using just 4608 examples. (b) Training examples required to reach an end loss prediction error at 72k training examples within 0.05 of the final normalized loss subsection 5.1 on 8 datasets.

To address the limitations of these analytical approaches, we propose directly learning training dynamics from historical data using Monotonic Non-negative Matrix Factorization (M-NMF) subsection 3.1. Our novel M-NMF algorithm decomposes loss curves into smooth, monotonically non-increasing basis functions, each capturing characteristic behaviors common across fine-tuning trajectories. This data-driven approach significantly enhances predictive accuracy when extrapolating full loss curves from partial observations. Using these basis functions, we're able to predict the end loss to within 0.05 of the final normalized loss using less than 14% of the dataset in the most realistic experiment domain, over 14,000 fewer examples than the next best method. The basis learned from M-NMF can even be transferred across model size training configurations. Furthermore, the interpretability of M-NMF basis functions enables researchers and practitioners to quantitatively analyze training dynamics across different models and training conditions, providing deeper insights into the fine-tuning process.

In this paper, we first show that loss curves exhibit a low rank structure with respect to these learned basis functions in section 3. Specifically, we find that just three basis functions are sufficient to accurately reconstruct a wide variety of loss curves across different model architectures, training configurations, and datasets. We then analyze the predictive performance of the basis functions from M-NMF's in section 5 through three different experiment domains: 1) In-distribution analysis, where basis functions are learned from fine-tuning curves with identical training configurations; (2) Model ladder evaluation, where basis functions learned from smaller models within a family are applied to larger models; and (3) Out-of-distribution evaluation, involving scenarios where loss curves available for training do not match the model family or fine-tuning method of the target. Crucially, for the in-distribution and out-of-distribution domains, we estimate M-NMF's performance on held-out tasks to demonstrate its practicality. Finally, we perform an analysis of how basis functions change with different training factors, discuss the optimality of M-NMF and other matrix factorizations compared to using analytical forms, and show that M-NMF can be transferred across different training horizons (and training schedules?) in section 6.

2 Background

2.1 Problem Formulation

Let a fine-tuning procedure consist of T total validation checks, producing losses

$$\mathcal{L}_{\text{full}} = \big(\ell_1, \ell_2, \ldots, \ell_T\big),$$

where ℓ_i is the validation loss at step i. We observe only the first t points, yielding the *partial* loss curve

$$\mathcal{L}_{\text{partial}} = \big(\ell_1, \ell_2, \ldots, \ell_t\big), \quad t < T.$$

Our goal is twofold:

1. Extrapolate $\mathcal{L}_{\text{partial}}$ to estimate the complete trajectory $\hat{\mathcal{L}}_{\text{full}} = \big(\hat{\ell}_1, \hat{\ell}_2, \ldots, \hat{\ell}_T\big)$.

2. Accurately predict the final converged loss ℓ_T.

We define the *normalized loss* as follows, which rescales the loss curve to be between 0 and 1:

$$\tilde{\ell}_i = \frac{\ell_i - \ell_T}{\ell_1 - \ell_T}$$

In all experiments, our primary evaluation metric is the quantity of data required by each method to achieve predictions within 0.05 of the final normalized loss. This criterion is equivalent to predicting the final loss with an error no greater than 5% of the total improvement observed during fine-tuning. By adopting this normalized metric, we ensure fair comparisons across training runs exhibiting significantly different scales of loss improvement.

2.2 Rectified Scaling Law

The most common method for predicting $\hat{\mathcal{L}}_{\text{full}}$ consists of extrapolating an analytical function based on the partial loss observation $\mathcal{L}_{\text{partial}}$. Lin et al. [2024] model fine-tuning loss after training on dataset size D using their rectified power law. The rectified power law is a simplified version of the scaling law from Kaplan et al. [2020], with an added term D_l to model pre-learned data size, or how much of the downstream data a model learned through pretraining.

$$\hat{\mathcal{L}}(D) = \frac{B}{D_l + D^{\beta}} + E \tag{1}$$

We fit the coefficients $\xi = \{B, D_l, \beta, E\}$ in Equation 1 to $\mathcal{L}_{\text{partial}}$ using the standard L-BFGS regression and Huber Loss and use the fit coefficients to predict $\mathcal{L}_{\text{full}}$, with full details in subsection C.1.

2.3 Preset Basis Functions

An alternative approach treats loss curves as a convex combination of a predefined set of analytical basis functions. Originally introduced by Domhan et al. [2015] to model pretraining validation accuracy, we select a new set of analytical functions f_k better suited to predict finetuning loss. We take the maximum likelihood estimate of Equation 9, to predict $\mathcal{L}_{\text{partial}}$, which serves as another baseline against which we compare our method.

$$\hat{\mathcal{L}}(t \mid \xi) = \sum_{k=1}^{K} w_k \, f_k\big(t \mid \theta_k\big) + v, \; \xi = \big(w_1, \ldots, w_K, v, \, \theta_1, \ldots, \theta_K\big) \tag{2}$$

Here, θ_k are the parameters for the kth analytical basis function. We provide our selected basis functions for finetuning loss and optimization routine in subsection C.2.

3 Learning Basis Functions

Instead of relying on preset analytical basis functions, our method employs M-NMF to learn a set of basis functions and more accurately predict $\mathcal{L}_{\text{full}}$ given a partial loss curve observation. Our method involves two parts. First, we use M-NMF to find an optimal set of basis functions from a corpus of loss curves. Then, we fit weights w_k and horizontal scale parameters s_k for each basis function to $\mathcal{L}_{\text{partial}}$, and use them for the full loss curve prediction. After introducing our method, we show the interesting result that loss curves are low rank with respect to these basis functions - with just three basis functions, we can reconstruct any loss curve with minimal reconstruction error. Using this low rank property, we analyze the shape of the basis functions to explain their behavior and ability to reconstruct loss curves.

3.1 Monotonic Non-negative Matrix Factorization (M-NMF)

Given a set of m fine-tuning loss curves, we construct an $m \times n$ data matrix X, where n denotes the number of validation loss observations during fine-tuning. Non-negative Matrix Factorization

(NMF) Lee and Seung [1999] approximates this matrix as a product of two non-negative matrices W and H. Specifically, NMF seeks to solve the optimization problem:

$$\min_{W\geq 0,\, H\geq 0} \|X - WH\|_F^2, \quad \text{where} \quad W \in \mathbb{R}_{\geq 0}^{m\times b}, \quad H \in \mathbb{R}_{\geq 0}^{b\times n},$$

with $\|\cdot\|_F$ representing the Frobenius norm, and b corresponding to the number of basis functions. Notably, NMF finds an additive set of basis and does not enforce orthogonality, thereby providing both an interpretable and efficient representation of the fine-tuning dynamics.

To reflect the natural properties of well-behaved loss curves—monotonic decrease and smoothness—we introduce a monotonicity constraint into the factorization process and apply smoothing to the resulting basis functions.

1. **Monotonicity:** We enforce monotonic decrease within each basis vector (row of H) by projecting updated vectors onto the monotone non-negative cone after each gradient step using isotonic regression Ayer et al. [1955]. Formally, for each row h in H, we solve the projection:

$$\min_{h_j \geq h_{j+1}} \|h - \tilde{h}\|_2^2,$$

where $\tilde{h}$ is the unconstrained update from gradient descent. We detail the full optimization process in the appendix subsection A.1.

2. **Smoothness:** After obtaining basis functions from the M-NMF, we apply Gaussian smoothing to smooth out spikes in loss curves that may be reflected in the basis functions.

3.2 Loss Curve Prediction

Given basis functions $\{H_b(t)\}_{b=1}^B$ from M-NMF, we allow each basis to be scaled horizontally with $\alpha_b \geq 1$ when reconstructing $\mathcal{L}_{\text{partial}} = \{(t_i, \ell_i)\}_{i=1}^t$. Using linear interpolation, the resulting scaled basis functions defined as:

$$\widetilde{H}_b(t) = H_b\left(\frac{t}{\alpha_b}\right).$$

Then, our reconstruction of the partial loss curve is

$$\hat{\mathcal{L}}_{\text{partial}} = v + \sum_{b=1}^{B} w_b\, \widetilde{H}_b(t).$$

The parameters $\{w_b\}$, v, and scale factors $\{\alpha_b\}$ are estimated by minimizing the squared reconstruction error over the partial observation:

$$\min_{\substack{w_b\geq 0,\, v\geq 0,\\ \alpha_b \geq 1}} \sum_{i=1}^{M} \left(\ell_i - \left(v + \sum_{b=1}^{B} w_b\, H_b\left(\frac{t_i}{\alpha_b}\right)\right)\right)^2 \tag{3}$$

The predicted full loss curve is thus:

$$\hat{\mathcal{L}}_{\text{full}} = \hat{v} + \sum_{b=1}^{B} \hat{w}_b H_b\left(\frac{t_i}{\hat{\alpha}_b}\right), \quad i = t+1, \ldots, T. \tag{4}$$

We term this prediction method *mnmf-scale* for all future references, and detail the full fitting process in subsection A.2.

3.3 Loss Curves are Low Rank

A key finding from our analysis is that fine-tuning loss curves naturally exhibit a low-rank structure when decomposed via Monotonic Non-negative Matrix Factorization (M-NMF). To validate this observation, we took 96 distinct fine-tuning loss curves on models ranging from 1B to 32B parameters,

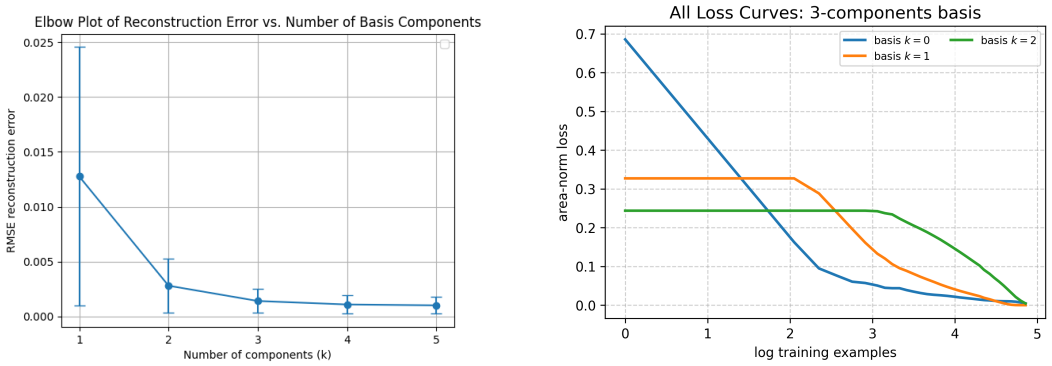


Figure 2: Visualization of the low-rank structure in fine-tuning loss curves. (a) Reconstruction error saturation (elbow plot) demonstrating diminishing returns beyond three basis functions. (b) The three basis functions recovered with M-NMF

trained over different learning rates and using QLoRA, LoRA, and full SFT. We performed M-NMF on these loss curves, subsequently reconstructing them using the nmf-scale approach. By varying the number of basis functions (b) from 1 to 5, we measured the L1 reconstruction error across all curves. Figure Figure 2 indicates that with only three basis functions, reconstruction error has essentially flatlined, with an average difference between the true loss curve and reconstruction of just 0.002. Given these results, for all further loss curve prediction experiments, we use $b = 3$ basis functions when performing M-NMF.

We analyze the shape of the three basis functions in Figure Figure 2 after normalizing their Area Under Curve. Notably, different basis control the behavior of different portions of loss curves, which we discuss below:

1. **Basis–0 (Blue)** This basis controls the behavior of the first 100 training examples, dictating whether or not there is a pre-power phase. The larger w we have for this basis, the more quickly the model learns the finetuning task and loss starts to decrease.
2. **Basis–1 (Orange)** This basis functions exhibits all three phases, and can be seen as a relatively standard loss curve. Isolating this basis and the Green Basis, we control the pre-power phase and the rate of loss decreasing during the convergence phase.
3. **Basis–2 (Green)** This basis has a negative second derivative, and helps to control the behavior after 1000 training examples. If this basis has more weight, the loss curve is still in the earlier parts of the power law phase. The other two basis are approaching the converge phase after around 1000 examples, and a lack of weight for this basis corresponds with the model starting to saturate on the task.

4 Experimental Setup

In this section, we outline the experimental framework we employ to rigorously evaluate the predictive capabilities and generalization properties of M-NMF basis. We first detail our training configurations, followed by a description of the three experimental domains we test mnmf-scale on.

4.1 Models and Training Configurations

We investigate fine-tuning dynamics across various models, datasets, and hyperparameter configurations. Specifically, our experiments span multiple fine-tuning setups involving the following:

Datasets: We employ 8 diverse instruction-following and language datasets: Aya, CodeInstruct, Chain of Thought, DailyMail, Dolphin, Numinamath, Tulu, and WMT-19 (English-Chinese), trained on 72k examples.

	7.3k Examples		12.7k Examples		<0.05 Final Normalized Error	
Method	RMSE	FLPE	RMSE	FLPE	Data Needed	Win % vs Rectified
Rectified Power Law	0.022	0.037	0.016	0.026	27403	–
Preset Basis Functions	0.023	0.039	0.022	0.036	24143	65.9
Loss Basis Functions	0.024	0.033	0.021	0.029	30406.	52.3
MNMF-Scale (ours)	**0.017**	**0.027**	**0.013**	**0.020**	**16076**	**72.2**

Table 1: In-Distribution Results: we report average RMSE and FLPE (final loss prediction error) predicting loss up to 72k examples. M-NMF shows consistently stronger RMSE and FLPE, meaning it finds a better overall curve fit and predicts end loss more accurately. Additionally, M-NMF requires significantly less data to consistently predict within 0.05 of the final normalized loss, and gets to this threshold faster than the Rectified Scaling Law 72.2% of the time.

	Qwen2.5 1.5B/3B QLoRA, $1.0 * 10^{-4}$ **LR**		**Qwen2.5 1.5B/3B QLoRA,** $1.0/2.0 * 10^{-4}$ **LRs**	
Method	Data Needed	Win % vs Rectified	Data Needed	Win % vs Rectified
Rectified Power Law	28113	–	28113	–
Preset Basis Functions	19297	57.1	19297	57.1
Loss Basis Functions	45666	37.5	39687	37.5
M-NMF	**17453**	**75.0**	**6305**	**87.5**

Table 2: Model Ladder Results: we use training curves from Qwen2.5 1.5B and 3B to learn basis functions. Left, we use a total of one learning rate for two smaller Qwen models, and right two learning rates for each model. We see a 4.7k and 15.8k decrease in data needed vs the In-Distribution setting using two and four training configurations from the smaller models, respectively.

	Qwen2.5 32B QLoRA		**Llama3.2 1b Full SFT**		**All Averaged**	
Method	Data Needed	Win Rate %	Data Needed	Win Rate %	Data Needed	Win Rate %
Rectified Power Law	28113	–	40440	–	27403	–
Preset Basis Functions	19297	57.1	50993	28.7	24143	65.9
Loss Basis Functions	41659	37.5	**24342**	75.0	25459	60.9
MNMF-Scale	**8535**	**87.5**	26834	**85.7**	**9566**	**91.6**

Table 3: Out-of-Distribution Results: We highlight predicting a model four times larger than any in the train corpus (Qwen2.5 32B) and predicting the only training configuration with SFT and a cosine learning rate (Llama3.2 1B). Overall, mnmf-scale significantly outperforms all other methods in this realistic experiment domain.

Models: Our experiments involve Mistral v0.3 (7B), Llama 3.1 (8B), Llama 3.2 (1B), and Qwen 2.5 models (1.5B, 3B, and 32B).

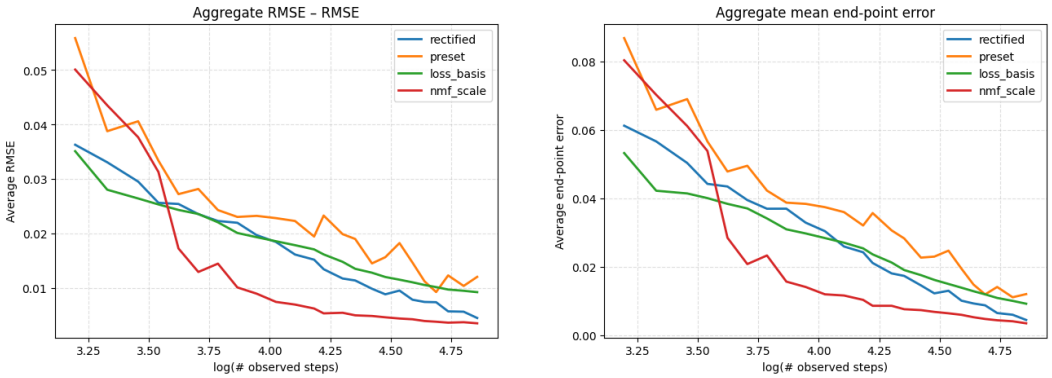
Training Methods: We apply multiple supervised fine-tuning methods, including full model fine-tuning (SFT), LoRA, and QLoRA.

Validation Loss Observations: We evaluate each model on the validation set using a front-loaded evaluation schedule during finetuning, meaning we run more evaluations early in the training run to provide enough signal to predict the full loss curve. To ensure fairness in our results, we first ensure that this front-loaded validation schedule does not significantly degrade performance compared to an evenly spaced schedule when evaluating the Rectified Power Law loss curve prediction Appendix B.

Detailed configurations (exact evaluation schedule, model specifics, learning rates, and fine-tuning methods) for each experiment are provided in the appendix Appendix D.

4.2 Experimental Domains

1. In-Distribution Analysis: This setup examines predictive performance within a training configuration (model architecture, hyperparameters, training method, learning rate). For each of the 8 datasets, we designate it as the test curve, using the remaining 7 to learn basis functions.



(a) Average RMSE in the Out-of-Distribution Setting.

(b) End Loss Prediction (Absolute) Error

Figure 3: RMSE and End Loss Prediction Absolute Error are consistently lower with MNMF-Scale.

2. Model Ladder Evaluation: We test M-NMF's ability to learn to learn basis functions on smaller models from the same model family, which are cheaper to finetune and evaluate on. Specifically, we use smaller Qwen models (Qwen2.5 1.5B and Qwen2.5 3B) to predict the fine-tuning loss trajectories of the significantly larger Qwen2.5 32B model.

3. Out-of-Distribution Evaluation: We investigate generalization across completely distinct model sizes and training paradigms. For each model configuration, we only take loss curves from models in different families. When evaluating a dataset, we also hold out all loss curves from that dataset, simulating an environment where you have a new model and dataset and want to predict finetuning loss. We specifically highlight two of the out-of-distribution experiments that evaluate the generalizability of m-nmf.:

- Predicting fine-tuning loss curves for **Llama 3.2 1B**, the only model trained with full model SFT and a cosine learning rate schedule, using basis functions learned exclusively from other models trained via a constant learning rate and LoRA or QLoRA.
- Predicting fine-tuning loss curves for **Qwen2.5 32B** while holding out all other Qwen family models. This tests M-NMF's ability to generalize to a model 4 times larger than any model in the train corpus.

When testing each dataset, we remove all instances of any loss curve obtained via a different model finetuned on the same dataset to make it truly out-of-distribution.

4.3 Other Baselines

In addition to using the Rectified Power Law subsection 2.2 and Preset Basis Functions subsection 2.3, we additionally test a method we call Loss Basis Functions. Given the same loss curves that we use for the loss curve corpos to learn basis functions for mnmf, Loss Basis Functions fits a weighted linear combination of the train loss curves and a horizontal offset to $\mathcal{L}_{\text{partial}}$ to predict $\hat{\mathcal{L}}_{\text{full}}$. See subsection C.3 for full details.

5 Results

In this section, we present the results of our experiments designed to evaluate the predictive performance and generalization capabilities of the proposed Monotonic Non-negative Matrix Factorization (M-NMF) method compared to several baseline methods.

5.1 Quantitative Evaluation

We quantitatively evaluate the predictive performance of our proposed M-NMF method against three baseline methods—Rectified Power Law, Preset Basis Functions, and Loss Basis Functions—across three experimental domains.

To assess predictive efficiency and effectiveness, we introduce two practical evaluation metrics. First, we measure the number of training examples required (*data needed*) to consistently predict within 0.05 of the final normalized loss. Specifically, given a sequence of validation loss observations, we consider predictions made using the first n and the first $n + 1$ observations. If both predictions fall within the specified threshold, we designate the number of training examples at observation n as the amount of data needed. For this section, we term this objective as two consecutive successful predictions. Second, we compute the *win rate* relative to the Rectified Power Law baseline, indicating which method first achieves this prediction accuracy. We exclude cases where both methods reach the target simultaneously.

In-Distribution Results (Table Table 1). Each in-distribution experiment contains just 7 loss curves from which we can learn the three basis functions. However, even with this limited data, we require over 8000 fewer finetuning examples than next best method to reach two consecutive successful predictions.

Model Ladder Results (Table Table 2). An interesting property from M-NMF is that basis functions can be learned from a smaller model and applied to a much larger model in the same family Table 2. Using Qwen2.5 1.5B and 3B with a learning rate of $1.0 * 10^{-4}$, we see a 21% decrease in the data needed to reach two consecutive successful predictions. Using four configurations, with learning rates of $1.0 * 10^{-4}, 2.0 * 10^{-4}$ with the smaller models, we need 71% less data. While more experiments need to be conducted validating this observation on different model families and scales, M-NMF provides a promising direction towards efficiently predicting finetuning loss by using a smaller model's loss curves.

Out-of-Distribution Results (Table Table 3). What if you want to use M-NMF out of the box, without requiring other finetuning loss curves trained using the same model, or other loss curves from the same dataset? We show in table Table 3 that, given a large corpus of out of distribution loss curves, M-NMF succeeds in this scenario. We're able to reach the three successful prediction threshold on Qwen2.5 32b QLoRA, on average, after finetuning on less than 12% of the entire dataset. Additionally, we evaluate this scenario with Llama3.2 1b Full-SFT, where we outperform rectified power law extrapolation 85.7% if the time. In this case, our train corpus contains no models trained with either Full-SFT or a Cosine Learning Rate Schedule - all other configurations use QLoRA or LoRA and a constant learning rate. We expect that, given other curves trained with full SFT or a cosine learning rate, we'd see even better performance in this setting. Over all training configurations in this setting, mnmf-scale requires 60% less data than the next best method to reach two consecutive successful predictions.

Additionally, mnmf-scale exhibits consistently lower RMSE Figure 3a when we sweep the amount of data observed in $\hat{\mathcal{L}}_{\text{partial}}$ once we have enough loss observations, signifying mnmf-scale is also accurate at predicting the shape of the overall loss curve.

6 Analysis

6.1 Qualitative comparison of the learned basis functions

Figure Figure 4 panels (a)–(d) juxtapose the *normalized* M–NMF bases that summarise how finetuning dynamics change when varying model size, learning rate, and finetuning method. Across all settings we observe a consistent structural decomposition:

Basis–0 represents a finetuning curve immediately in the power phase, and also controls the convergence phase. **Basis–1** (dashed) typically exhibits a flat pre-power phase, and then the linear power phase. **Basis–2** (solid lines) adds to the late–training *convergence* regime, helping to determine if we are in the power law or convergence phase.

Each basis functions is normalized to have the same Area Under curve, focusing our analysis on the shape of the basis functions. We now evaluate how different training hyperparameter choices infleunce the basis functions:

Learning-rate sensitivity (Fig. Figure 4a). Increasing the step size from 5×10^{-5} to 2×10^{-4} leaves Basis–1 and Basis–2 relatively unchanged. The primary difference is in Basis–0 - as expected,

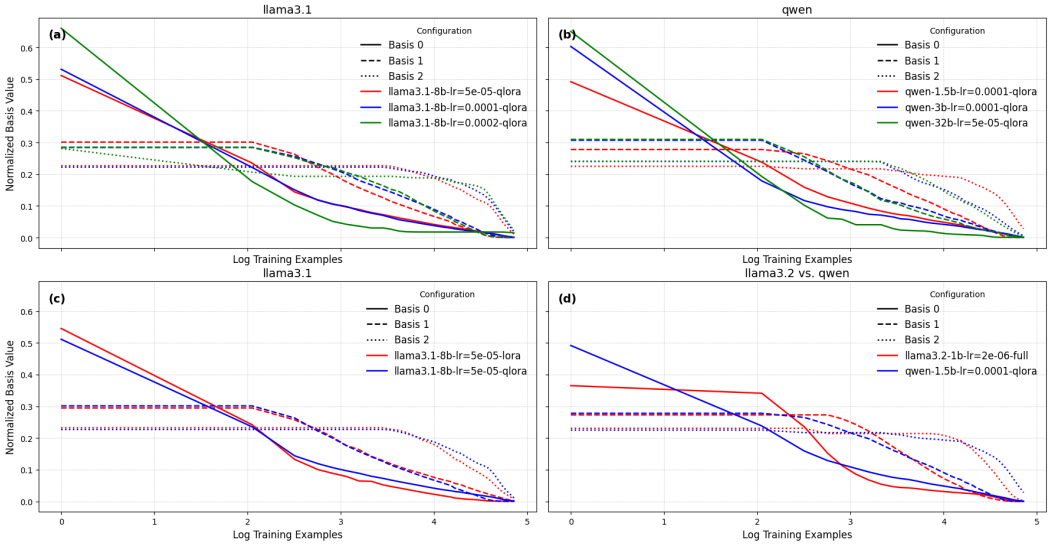


Figure 4: Basis function comparison across different learning rates for Llama3.1-8B (QLoRA).

increasing the learning rate increases the rate of loss decrease during the initial in 1000 training examples.

Model Size effects (Fig. Figure 4b). We compare Qwen models spanning 1.5B, 3B and 32B parameters trained with QLoRA. We see a similar effect here to scaling learning-rate, where Basis–0 has a larger peak to trough drop in loss with the 32B model. Additionally, Qwen2.5 1.5B exhibits later inflections points in loss across Basis–1 and Basis–2, suggesting it typically enters the power phase later in training.

LoRA versus QLoRA (Fig. Figure 4c). We see very little difference between the basis functions when comparing QLoRA vs LoRA - Basis–0 exhibits a slightly stronger decrease, but otherwise the basis are very similar. This is in line with the training results when training with QLoRA vs LoRA, where there is typically almost no difference in the end loss between the two methods.

Full-parameter fine-tuning vs. QLoRA (Fig. Figure 4d). This is where we see large differences in the basis functions and overall loss curve shape due to the cosine learning rate schedule and warmup steps. Loss curves trained with full SFT and a cosine learning rate exhibit a longer, flatter pre-power phase, which is reflected in these basis. Additionally, Basis–0 and Basis–1 show a shorter power phase and a steeper decrease in loss before reaching flat convergence phases to due to the decay in learning rate.

6.2 How does MNMF-Scale compare to similar Matrix Factorizations

We ablate our algorithm decisions for *mnmf-scale* by comparing it against basis functions found using various other methods:

- PCA-Basis, for which the basis functions and weights are not non-negative.
- Standard Non-negative Matrix Factorization (NMF).
- kmeans, which clusters loss curves from the train set into three clusters based on shape using KMeans, and takes the average of each to form the basis for prediction.
- nmf-scale, which takes the basis functions found from standard NMF and applies the same horizontal scaling as in Equation 3.

We evaluate all methods in the out-of-distribution setting, and report the averaged results in Table 4. Notably, every single method we test that learns representations (clustering or factorizations) from

Method	Data Needed
Rectified	27403
PCA Basis	15299
KMeans Basis	17751
NMF	17909
NMF-Scale	15678
MNMF-Scale (ours)	**9566**

Table 4: Data needed top reach two consecutive loss predictions within 0.05 of the final normalized loss. Every single method we test that learns basis from historical data significantly outperforms the rectified power law.

other loss curves significantly outperforms rectified power law extrapolation. For M-NMF, we see empirically that applying the horizontal scale factor, and especially forcing basis functions to be monotonic, lead to significantly improvements in loss prediction capability.

6.3 Evaluating Basis Functions over Varying Temporal Horizons

Thus far, all our experiments have been conducted with a set training horizon of 72k examples. What happens when you plan to train for a shorter length, and would still like to predict $\mathcal{L}_{\text{full}}$? Similar to the out of distribution setting, we start with a corpus of loss curves from finetuning runs for models from other families, and learn basis functions using mnmf. Given we expect well behaved, hyperparameter optimized loss curves to follow the three distinct loss regimes, we then rescale our loss curves using the same horizontal scale as in subsection 3.2 to match the length of the prediction loss curve.

We evaluate our method on Dolly15k, MedicalReasoning, Alpaca, and OpenThoughts, trained to 12.7k, 22.5k, 48.7k, and 34.4k examples, respectively. We focus on cases where we have multiple learning rate runs for the same model, and only evaluate on the configuration that achieved the lowest final loss. Full experiment details in subsection D.6.

We use the same evaluations schedule for these experiments as the 72k training datasets, and note that MNMF-Scale generally requires at least 8-10 loss observations to be most effective. Even with this caveat, we find that MNMF-Scale outperforms the other baseline methods on three of the four datasets Table 5.

We are unable to test MNMF-Scale when attempting to predict loss curves with a longer training horizon that the training corpus, as we don't have enough loss curves optimized for a consistent, shorter training horizon.

6.4 Limitations

All our finetuning loss runs use a batch size of 8 and finetune for only one epoch, training to 72k examples. Additionally, most experiments focus on performance efficient finetuning (PEFT) methods, with a constant learning rate. We do not expect that any training configuration choices leading to sharp spikes or drops in validation loss, such as a learning rate schedule with restarts or training for multiple epochs, can be accurately predicted with M-NMF unless these behaviors have been sufficiently observed in the training corpus.

Additionally, mnmf-scale performs quite poorly when there are fewer than 8 loss observations available, and generally works best when you have at least 10 observations.

More research needs to be done with Full-SFT and larger models to ensure that M-NMF generalizes well in these domains. However, we feel confident that in resource constrained settings, M-NMF provides a strong method to predict the full finetuning curve using minimal data.

7 Related Work

Extrapolating Loss Curves Predicting and extrapolating finetuning loss curves have been explored more in recent years with the rise in popularity of language model post-training. Lin et al.

Method	Alpaca	Dolly	MedicalReasoning	OpenThoughts
Rectified Power Law	4480	**2656**	12366	6574
Preset Basis Functions	9712	3972	7808	4220
MNMF-Scale	**4192**	3646	**5652**	**3464**

Table 5: RMSE comparison across methods using 6.7k training examples. MNMF-Scale achieves the lowest RMSE (bold).

[2024] use the rectified power law Equation 1, and propose the Accept then Stop *(AtS)* algorithm to predict the fine-tuning loss curves. *AtS* starts by fine-tuning the model on an initial subset of the dataset S_{sub}, then iteratively halves the dataset size, fine-tuning again at each step. The algorithm continues this halving procedure until the observed points deviate significantly from linearity, signaling the end of the linear power-law phase. The identified linear regime to extrapolate and predict the full fine-tuning loss curve. However, *AtS* does not account for the subsequent convergence phase observed after the power phase.

Jain et al. [2023] use a piecewise power law to model finetuning loss, with a random forest regressor that predicts the inflection between pre-power and power phases, but also does not capture the convergence phase. Kuramoto and Suzuki [2025] also use a piecewise function to model validation accuracy, training the function on small dataset sizes and extrapolating to a much larger dataset. They focus solely on classification tasks, finetuning smaller BERT models.

Hyperparameter Optimization Extrapolating performance curves was initially explored in the hyperparameter optimization domain. Domhan et al. [2015] used a set of analytical basis functions to model validation accuracy, and subsequent work uses regression forests and neural networks to predict pretraining performance from a partial pretraining curve and/or hyperparameter configuration (Klein et al. [2017], Gargiani et al. [2019], Kadra et al. [2023]). However, such neural-based approaches require thousands of historical performance curves, limiting their feasibility in many fine-tuning scenarios. These predictions typically include a confidence interval around their prediction, such that if you are 95% the current training run won't improve on the best configuration's performance, you can terminate early and save compute. We briefly explore the MCMC sampling routine introduced by Domhan et al. [2015] to model prediction confidence intervals, but observe significant overconfidence and leave this extension for future work.

Scaling Laws Scaling laws research aims to relate model performance to different training factors such as model parameters and dataset size. Kaplan et al. [2020], Hoffmann et al. [2022] fit power laws on various model and dataset sizes to determine the optimal compute allocation between the two. Zhang et al. [2024] evaluate how various training decisions influence finetuning performance with a joint multiplicative power law. Scaling law approaches require learning the performance behavior through cheaper experiments, such as sequentially smaller models, in order to predict the behavior of the largest model. Our work renders this training ladder behavior unnecessary - we can accurately predict finetuning performance from a general loss curve corpus.

8 Conclusion

We introduced Monotonic Non-negative Matrix Factorization (M-NMF) to model fine-tuning loss curves, and MNMF-Scale to predict $\mathcal{L}_{\text{full}}$ from a partial loss observation. By decomposing curves into smooth, monotonically decreasing basis functions, M-NMF acurately captures low-rank structure shared across tasks and model sizes. Leveraging these basis functions, MNMF-Scale predicts full loss trajectories using far fewer observations than power-law baselines. In the most realistic out-of-distribution setting, MNMF-Scale achieves accurate end loss prediction using 65% fewer data than the rectified power law. Beyond predictive accuracy, the learned bases offer insights into fine-tuning dynamics and highlight consistent patterns across training hyperparameters. We hope MNMF-Scale provides a practical approach to informing data collection and finetuning loss prediction that is both accessible and accurate.

References

Miriam Ayer, H. D. Brunk, G. M. Ewing, W. T. Reid, and Edward Silverman. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 26(4):641–647, December 1955. doi: 10.1214/aoms/1177728423.

T. Domhan, J. T. Springenberg, and F. Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 2015.

M. Gargiani, A. Klein, S. Falkner, and F. Hutter. Probabilistic rollouts for learning curve extrapolation across hyperparameter settings. *arXiv preprint arXiv:1910.04522v1 [cs.LG]*, Oct 2019.

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. *arXiv preprint arXiv:2203.15556v1 [cs.CL]*, Mar 2022.

Achin Jain, Gurumurthy Swaminathan, Paolo Favaro, Hao Yang, Avinash Ravichandran, Hrayr Harutyunyan, Alessandro Achille, Onkar Dabeer, Bernt Schiele, Ashwin Swaminathan, and Stefano Soatto. A meta-learning approach to predicting performance and data requirements. *arXiv preprint arXiv:2303.01598v1 [cs.CV]*, Mar 2023.

Yiding Jiang, Allan Zhou, Zhili Feng, Sadhika Malladi, and J. Zico Kolter. Adaptive data optimization: Dynamic sample selection with scaling laws. *arXiv preprint arXiv:2410.11820v1*, 2024.

Arlind Kadra, Maciej Janowski, Martin Wistuba, and Josif Grabocka. Scaling laws for hyperparameter optimization. *arXiv preprint arXiv:2302.00441v3 [cs.LG]*, Oct 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:2001.08361v1 [cs.LG]*, Jan 2020.

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

Toshiki Kuramoto and Jun Suzuki. Predicting fine-tuned performance on larger datasets before creating them. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pages 204–212, Jan 2025.

Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

Haowei Lin, Baizhou Huang, Haotian Ye, Qinyu Chen, Zihao Wang, Sujian Li, Jianzhu Ma, Xiaojun Wan, James Zou, and Yitao Liang. Selecting large language model to fine-tune via rectified scaling law. *arXiv preprint arXiv:2402.02314*, 2024.

Zeyu Qin, Qingxiu Dong, Xingxing Zhang, Li Dong, Xiaolong Huang, Ziyi Yang, Mahmoud Khademi, Dongdong Zhang, Hany Hassan Awadalla, Yi R. Fung, Weizhu Chen, Minhao Cheng, and Furu Wei. Scaling laws of synthetic data for language models. *arXiv preprint arXiv:2503.12811v1*, 2025.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, Dillon Cournapeau, Eric Burovski, Pauli Peterson, Wes Weckesser, Jonathan Bright, Stéfan van der Walt, Matthew Brett, Josh Wilson, K. Jarrod Millman, Noah R. Nelson, Eric Jones, Robert Kern, Erik Larson, Christoph Carey, İbrahim Polat, Yuxuan Feng, Eric W. Moore, Jessica J. VanderPlas, David Laxalde, José A. Quintero, Charles R. Harris, Alban M. Archibald, Antonio H. Ribeiro, Felipe Pedregosa, Peter van Mulbregt, and the SciPy 1.0 Contributors. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020.

Biao Zhang, Zhongtao Liu, Colin Cherry, and Orhan Firat. When scaling meets llm finetuning: The effect of data, model and finetuning method. *arXiv preprint arXiv:2402.17193v1 [cs.CL]*, Feb 2024.

A M-NMF Optimization Routines

A.1 M-NMF Optimization

We perform a non-negative matrix factorization (NMF) of a given matrix $X \in \mathbb{R}_{\geq 0}^{n \times t}$ into two matrices $W \in \mathbb{R}_{\geq 0}^{n \times r}$ and $H \in \mathbb{R}_{\geq 0}^{r \times t}$, subject to the additional constraint that each row of H is monotonically decreasing:

$$X \approx WH, \quad W \geq 0,\ H \geq 0,\ H_{i,0} \geq H_{i,1} \geq \cdots \geq H_{i,t-1}, \quad \forall i.$$

To achieve this factorization, we use an alternating optimization strategy with the following steps:

Initialization. We initialize W and H randomly from a non-negative distribution, ensuring initial monotonicity by projecting each row of H onto the monotone non-negative cone. Specifically, for each row H_i, we apply isotonic regression via the Pool Adjacent Violators Algorithm (PAVA):

$$H_i \leftarrow \text{isotonic\_regression}(H_i, \text{increasing=False})$$

This ensures each row of H is monotonically decreasing from the start.

Optimization Procedure. The optimization iteratively alternates between updates of W and H:

1. **Updating** W:
   Given fixed H, we update each column $W_{:,k}$ using Hierarchical Alternating Least Squares (HALS):

   $$W_{:,k} \leftarrow \max\left(0,\ \frac{XH_{k,:}^{\top} - W(HH^{\top})_{:,k} + W_{:,k}\|H_{k,:}\|^2}{\|H_{k,:}\|^2 + \gamma_W}\right), \quad k = 1, \ldots, r.$$

   Here, γ_W is a regularization parameter ensuring numerical stability.
2. **Updating** H:
   Given fixed W, we perform projected gradient descent to update H:

   $$H \leftarrow H - \eta_H \nabla_H \|X - WH\|_F^2,$$

   where the gradient is computed as:

   $$\nabla_H \|X - WH\|_F^2 = -2W^{\top}(X - WH).$$

   After each gradient step, we project each row of H onto the monotone non-negative cone using isotonic regression:

   $$H_i \leftarrow \text{isotonic\_regression}\left(\max(H_i, 0), \text{increasing=False}\right), \quad \forall i.$$

Smoothing Post-processing. Finally, we smooth the resulting matrix H along the feature axis (time dimension) using a Gaussian filter:

$$H_{\text{smooth}} \leftarrow \text{GaussianFilter1D}(H,\ \sigma = 1.0),$$

preserving the first 10 columns without smoothing to maintain the fidelity of initial conditions. This smoothing step ensures smoother factorized curves, beneficial for interpretability and downstream modeling tasks.

A.2 MNMF-Scale Optimization

Given the set of B basis functions $\{H_b(t)\}_{b=1}^{B}$ obtained from M-NMF, we allow each basis to stretch horizontally by a factor $\alpha_b \geq 1$ using linear interpolation when fitting the observed partial loss curve.

The parameters $\{w_b\}_{b=1}^{B}$, v, and $\{\alpha_b\}_{b=1}^{B}$ are found by solving

$$\min_{\substack{w_b \geq 0,\ v \geq 0,\\ \alpha_b \geq 1}} \underbrace{\sum_{i=1}^{M} \Big(\ell_i - v - \sum_{b=1}^{B} w_b\, H_b\big(t_i/\alpha_b\big)\Big)^2}_{\text{data fit}} \;+\; \underbrace{\lambda \sum_{b=1}^{B} w_b^2}_{\ell_2\text{-penalty}} \tag{5}$$

via an alternating-optimization scheme:

1. **Weight & offset update (fix scales α):**
   With all α_b held constant, solve use non-negative least squares to solve
   $$\min_{w_b \geq 0,\, v \geq 0} \sum_{i=1}^{M} \Big(\ell_i - v - \sum_{b=1}^{B} w_b\, H_b\big(t_i/\alpha_b\big)\Big)^2 \;+\; \lambda \sum_{b=1}^{B} w_b^2$$
2. **Scale update (fix weights w, offset v):**
   With $\{w_b\}$ and v fixed, optimize each $\alpha_b \in [1, \alpha_{\max}]$ by
   $$\min_{\alpha_b \in [1, \alpha_{\max}]} \sum_{i=1}^{M} \Big(\ell_i - v - \sum_{b=1}^{B} w_b\, H_b\big(t_i/\alpha_b\big)\Big)^2$$
   using L-BFGS-B. (The regularization term is independent of α_b.)

After convergence, the full-curve prediction for $t = 1, \ldots, T$ is

$$\hat{\mathcal{L}}_{\text{full}}(t) \;=\; v^* \;+\; \sum_{b=1}^{B} w_b^*\, H_b\big(t/\alpha_b^*\big), \tag{6}$$

where (w_b^*, v^*, α_b^*) are the minimizers of (5).

When fitting mnmf-scale, we use $\lambda = 0.1$ when using fewer than 10 loss observations, and don't use any L2 regularization when we have at least 10 points to fit in mnmf-scale.

B Linear vs. Front-Loaded Evaluation Schedule

For all training runs, we take validation loss after training for n training examples, at the below schedule. Note, we use a subset of these evaluation steps for experiments, see subsection D.1):

> [0, 8, 24, 48, 80, 112, 144, 184, 224, 272, 320, 376, 432, 496, 568, 640, 720, 816, 912, 1024, 1144, 1272, 1416, 1568, 1736, 1928, 2128, 2352, 2592, 2856, 3152, 3464, 3816, 4192, 4608, 5064, 5560, 6104, 6696, 7344, 8048, 8824, 9672, 10600, 11608, 12712, 13928, 15248, 16696, 18272, 20000, 22544, 26448, 30032, 34368, 38904, 43608, 48712, 54616, 63296, 72000]

We first want to ensure that such a validation loss schedule does not significantly degrade the predictive performance of the Rectified Power Law and lead to a biased comparison in performance between it and MNMF-Scale. We are unable to compare an exact linear vs front-loaded evaluation schedule. However, we use the below to create optimal linear schedules given our observations:

We first split these evaluation steps into a **prefix** and **suffix** based on a threshold e_{max}:

$$\mathbf{e}^{\text{prefix}} = \{e_i \mid e_i \leq e_{\text{max}}\},$$
$$\mathbf{e}^{\text{suffix}} = \{e_i \mid e_i > e_{\text{max}}\}.$$

The suffix remains unchanged, while the prefix is resampled using two distinct methods:

Linear Resampling We select n evenly spaced points between the first and last points of the prefix. Each selected point is snapped to the closest original evaluation step.

Power-law Resampling We sample n points according to a power-law distribution with exponent $\alpha = 1.5$, biasing towards earlier steps. This results in schedules very similar to what we use in subsection D.1. Specifically, for uniformly spaced points $u \in [0, 1]$, we apply:

$$\mathbf{e}_{\text{power}}^{\text{ideal}} = e_1^{\text{prefix}} + \left(e_M^{\text{prefix}} - e_1^{\text{prefix}}\right) \cdot u^{1.5},$$

and again snap each point to the nearest original evaluation step.

We systematically vary two parameters to evaluate the sensitivity of the Rectified Power Law to these resampling strategies:

- **Maximum prefix step** $e_{\text{max}} \in \{3000, 5000, 7000, 10000, 15000, 20000\}$.
- **Number of resampled points** $n \in \{6, 8, 10, 12, 15, 20\}$.

We evaluate the Rectified Power Law's predictive performance for each configuration using two metrics:

- **RMSE**: Root-mean-square error over suffix predictions.
- **EndErr**: Absolute error at the final evaluation step.

Table 6: Aggregate performance under linear and power-law schedules (mean $\pm$ std across all curves).

Schedule	RMSE	EndErr
Linear	0.0137 ± 0.0154	0.0260 ± 0.0309
Power ($\alpha = 1.5$)	0.0149 ± 0.0154	0.0278 ± 0.0304

As shown in Table Table 6, both linear and power-law schedules yield comparable performance for the Rectified Power Law, suggesting that using the front-loaded evaluation schedule in subsection D.1 is not the reason for MNMF-Scale's superior performance compared to the rectified scaling law.

C Baseline Methods

C.1 Rectified Power Law

We use the rectified power law from Lin et al. [2024] to characterize fine-tuning loss curves:

$$\hat{L}(D) = \frac{B}{D_\ell + D^\beta} + E, \tag{7}$$

where D represents the amount of training data, D_ℓ denotes the effective size of data already learned by the model, β quantifies the learning difficulty, B adjusts the initial test loss scale, and E indicates the asymptotic minimal loss achievable by the model.

To estimate the model-specific parameters B, E, D_ℓ, β, we solve the following optimization problem:

$$\min_{B,E,D_\ell,\beta} \sum_i \text{Huber}\delta \left(\text{LSE}(\log B - \log(D\ell + D_i^\beta), \log E) - \log L_i\right), \tag{8}$$

where L_i denotes the observed test loss at training data size D_i, LSE refers to the numerically-stable log-sum-exp operator, and $\text{Huber}\delta$ indicates the Huber loss with robustness parameter $\delta = 0.001$.

Optimization is carried out using the L-BFGS-B algorithm implemented in the standard Python library `scipy` Virtanen et al. [2020]. We employ a grid search over 250 parameter initializations with $B \in \{1.0, 5.0, 25.0, 125.0, 625.0\}, E \in \{0.5, 1.0\}, \beta \in \{1.0, 5.0, 25.0, 125.0, 625.0\}, D_\ell \in \{1.0, 5.0, 25.0, 125.0, 625.0\}$ and take the best fit over $\mathcal{L}_{\text{partial}}$ to predict $\hat{\mathcal{L}}_{\text{full}}$.

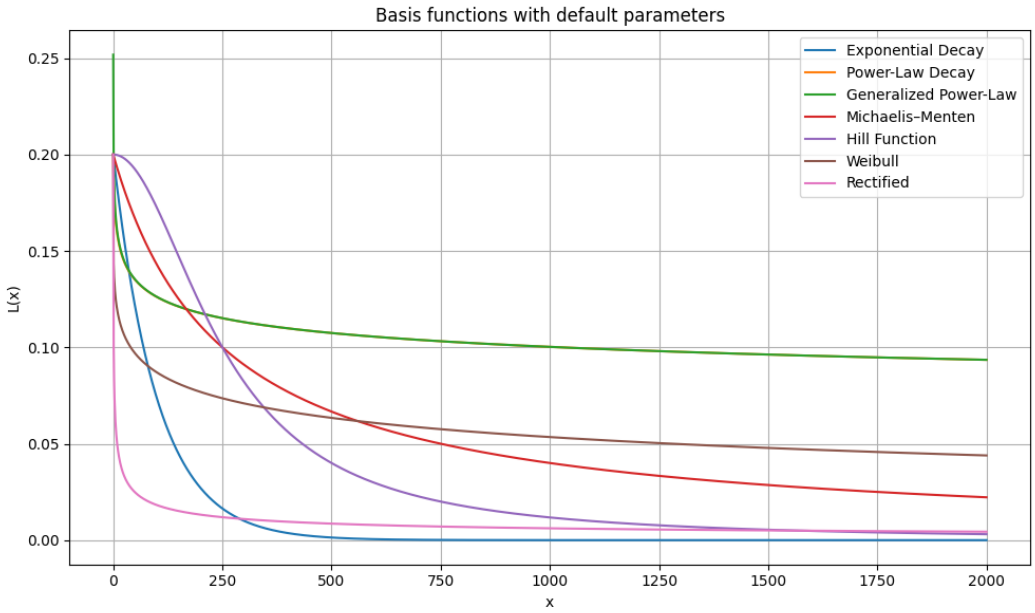


Figure 5: Visualization of the preset basis functions used to model fine-tuning loss curves.

C.2 Preset Basis Functions

We select a set of analytical basis functions to represent fine-tuning loss curves, each capturing distinct variations of the power law decay generally observed in loss curves. The set of functions used is summarized in Table 7.

Name	**Function Form**
Exponential Decay	$L_0 \exp(-kt)$
Power-Law Decay	$L_0(t+c)^{-b}$
Generalized Power-Law Decay	$L_0\left(\frac{t+c}{d}\right)^{-b}$
Michaelis–Menten Decay	$\frac{L_0}{1+t/K}$
Hill Function Decay	$\frac{L_0}{1+(t/K)^n}$
Weibull Decay	$L_0 \exp\left[-(t/\lambda)^k\right]$
Rectified Decay	$\frac{L_0}{D_l+t^\beta}$

Table 7: Preset analytical basis functions used to model loss curves.

We visualize each analytical function, with its default parameters Figure 5.

These preset basis functions are combined in a weighted sum to model the loss curve as:

$$\hat{\mathcal{L}}(t \mid \xi) = \sum_{k=1}^{K} w_k \, f_k\big(t \mid \theta_k\big) + v, \quad \xi = \big(w_1, \ldots, w_K, v, \theta_1, \ldots, \theta_K\big) \tag{9}$$

where w_k are non-negative weights determining the contribution of each basis function f_k, θ_k represents the parameters specific to each basis function as defined in Table 7, and v is the vertical offset.

Similar to Domhan et al. [2015], we fit all model parameters using non-linear least squares optimization, and enforce non-negativity on all parameters including weights and offset.

C.3 Loss Basis Functions

The Loss Basis Functions approach utilizes the same corpus of training loss curves as MNMF-Scale for all experiments. However, rather than constructing a low-rank basis through MNMF, we directly use the observed set of training curves themselves as basis functions. Formally, let $\{L_b(t)\}_{b=1}^{B}$ denote the corpus of B observed loss curves, where each $L_b : \mathbb{R}_{\geq 0} \to \mathbb{R}$ represents an individual training loss trajectory.

To approximate a target loss curve $\mathcal{L}(t)$, the method identifies an optimal linear combination of these observed curves, each adjusted by a global vertical offset v:

$$\{v^*, \mathbf{w}^*\} = \arg \min_{v \in \mathbb{R},\, \mathbf{w} \in \mathbb{R}^B_{\geq 0}} \sum_{i=1}^{T} \left(\mathcal{L}(t_i) - v - \sum_{b=1}^{B} w_b \, L_b(t_i) \right)^2 \tag{10}$$

Here, v represents the global vertical offset, and $\mathbf{w} = [w_1, w_2, \ldots, w_B]$ are non-negative weights determining the linear combination of observed loss curves. We use non-negative least squares to solve the optimization in Equation 10.

D Experiment Details

D.1 Training schedule and data

We evaluate all fine-tuning runs on a front-loaded schedule of 30 validation checks. The evaluation steps in training iterations are:

> [0, 112, 320, 568, 816, 1144, 1568, 2128, 2856, 3464, 4192, 5064, 6104, 7344, 8824, 10600, 12712, 15248, 16696, 20000 22544, 26448, 30032, 34368, 38904, 43608, 48712, 54616, 63296, 72000]

All primary experiments use the eight full-length text-generation datasets *AYA*, *CodeInstruct*, *CoT*, *NuminaMath*, *Dolphin*, *DailyMail*, *Tulu*, and *WMT*. Short-horizon experiments additionally employ *Dolly*, *MedicalReasoning*, *Alpaca* and *OpenThoughts*.

D.2 Training Configurations

We evaluate all 8 loss primary datasets on the following training configurations, which we use for all experiments:

- Llama3.1 8B LoRA, learning rate 5×10^{-5};
- Llama3.1 8B QLoRA, learning rates 5×10^{-5}, 1×10^{-4}, and 2×10^{-4};
- Llama3.2 1B full fine-tuning with learning rate 2×10^{-6};
- Mistral 7B QLoRA, learning rates 5×10^{-5} and 1×10^{-4};
- Qwen2.5 1.5B QLoRA, learning rates 1×10^{-4} and 2×10^{-4};
- Qwen2.5 3B QLoRA, learning rates 1×10^{-4} and 2×10^{-4};
- Qwen2.5 32B QLoRA, learning rate 5×10^{-5}.

For each configuration, we report the In-Distribution experiment results in Table 8:

D.3 Model Ladder configuration

We train basis functions on smaller Qwen models and predict the curves of Qwen2.5 32B. The training sets include:

1. four curves (1.5B at 1×10^{-4} and 2×10^{-4}, 3B at 1×10^{-4} and 2×10^{-4});
2. two curves (1.5B at 1×10^{-4} and 3B at 1×10^{-4});
3. one curve (1.5B at 1×10^{-4});

Training Configuration	Rectified	Preset Basis		Loss Basis		M - NMF	
	Data Needed	D.N.	Win %	D.N.	Win %	D.N.	Win %
Llama3.1 8b LoRA, 5×10^{-5}	15429	30839	37.5	25773	50.0	**11024**	**62.5**
Llama3.1 8b QLoRA, 5×10^{-5}	29585	23418	71.4	31839	57.1	**13355**	**100.0**
Llama3.1 8b QLoRA, 1×10^{-4}	22588	19703	75.0	36345	37.5	**10632**	**87.5**
Llama3.1 8b QLoRA, 2×10^{-4}	**20027**	21065	**57.1**	33852	**57.1**	25917	42.9
Llama3.2 1B Full SFT, 2×10^{-6}	40440	50993	28.6	36785	57.1	**16536**	**85.7**
Mistral 7B QLoRA, 5×10^{-5}	25056	29370	50.0	32419	37.5	**17381**	**83.3**
Mistral 7B QLoRA, 1×10^{-4}	25478	21982	**85.7**	36578	42.9	**20878**	50.0
Qwen2.5 1.5B QLoRA, 1×10^{-4}	35851	24693	**75.0**	23264	71.4	**16160**	**75.0**
Qwen2.5 1.5B QLoRA, 2×10^{-4}	30753	17306	**87.5**	23355	57.1	**14784**	71.4
Qwen2.5 3B QLoRA, 1×10^{-4}	30021	15332	75.0	24119	66.7	**4935**	**87.5**
Qwen2.5 3B QLoRA, 2×10^{-4}	25496	**15722**	**87.5**	22496	62.5	19180	62.5
Qwen2.5 32B QLoRA, 5×10^{-5}	28113	**19297**	57.1	38049	37.5	22130	**62.5**

Table 8: Average `needed_frac_5` "Data Needed" and win rate (percentage of runs beating the rectified baseline) for each method across the listed training configurations. Bold values denote the best (lowest) data–needed score per row and all win rates that are at least 50 %.

4. one curve (1.5B at 2×10^{-4});
5. one curve (3B at 1×10^{-4});
6. one curve (3B at 2×10^{-4}).

For each model ladder configuration, we report the prediction results in Table 9:

Model Ladder Configuration	Rectified	Preset Basis		Loss Basis		M - NMF	
	Data Needed	D.N.	Win %	D.N.	Win %	D.N.	Win %
4 curves	28113	19297	57.1	39687	37.5	**6305**	**87.5**
2 curves	28113	19297	57.1	45666	37.5	**17453**	**75.0**
Qwen2.5 1.5B QLoRA, 1×10^{-4}	28113	19297	57.1	41708	37.5	**8329**	**100.0**
Qwen2.5 1.5B QLoRA, 2×10^{-4}	28113	19297	57.1	36007	50.0	**12080**	**87.5**
Qwen2.5 3B QLoRA, 1×10^{-4}	28113	19297	57.1	44829	37.5	**13097**	**85.7**
Qwen2.5 3B QLoRA, 2×10^{-4}	28113	**19297**	**57.1**	39834	37.5	22137	50.0

Table 9: Model-ladder experiments: `needed_frac_5` "Data Needed" and win rate versus the rectified baseline. Bold values denote (i) the lowest data-needed score among the three basis methods and (ii) the single highest win rate in each row.

D.4 Out-of-Distribution configuration

We evaluate each training configuration in the out-of-distribution setting, and report the results in Table 10:

D.5 Prediction method ablation

We ablate four other methods — PCA, standard NMF, k-means clustering, and NMF Scale — and compare them to MNMF-Scale. Each method leverages a corpus of observed training loss curves $\{L_b(t)\}_{b=1}^{B}$, where each curve $L_b : \mathbb{R}_{\geq 0} \to \mathbb{R}$ represents an individual training loss trajectory.

PCA-Basis We use PCA with 3 basis, which finds orthogonal basis functions that are not guaranteed to be non-negative.

Standard NMF We solve the standard NMF:

$$\min_{W\geq 0, H\geq 0} \|C_{\text{pos}} - WH\|_F^2,$$

Training Configuration	Rectified	Preset Basis		Loss Basis		M - NMF	
	Data Needed	D.N.	Win %	D.N.	Win %	D.N.	Win %
Llama3.1 8b LoRA, 5×10^{-5}	15429	30839	37.5	15185	62.5	**4564**	**87.5**
Llama3.1 8b QLoRA, 5×10^{-5}	29585	23418	71.4	12399	**100.0**	**5534**	**100.0**
Llama3.1 8b QLoRA, 1×10^{-4}	22588	19703	75.0	15057	75.0	**7672**	**100.0**
Llama3.1 8b QLoRA, 2×10^{-4}	20027	21065	57.1	23758	50.0	**19496**	**75.0**
Llama3.2 1B Full SFT, 2×10^{-6}	40440	50993	28.6	**24342**	75.0	26834	**85.7**
Mistral 7B QLoRA, 5×10^{-5}	25056	29370	50.0	16250	66.7	**6224**	**87.5**
Mistral 7B QLoRA, 1×10^{-4}	25478	21982	85.7	21615	57.1	**6187**	**87.5**
Qwen2.5 1.5B QLoRA, 1×10^{-4}	35851	24693	75.0	34417	57.1	**8091**	**100.0**
Qwen2.5 1.5B QLoRA, 2×10^{-4}	30753	17306	87.5	35149	42.9	**7364**	**100.0**
Qwen2.5 3B QLoRA, 1×10^{-4}	30021	15332	75.0	32563	57.1	**5765**	**87.5**
Qwen2.5 3B QLoRA, 2×10^{-4}	25496	15722	87.5	33116	50.0	**8523**	**100.0**
Qwen2.5 32B QLoRA, 5×10^{-5}	28113	19297	57.1	41659	37.5	**8535**	**87.5**

Table 10: Out-of-distribution evaluation: average `needed_frac_5` "Data Needed" and win rate versus the rectified baseline. Bold numbers denote the lowest data-needed score (among the three basis methods) and the single highest win rate per row.

obtaining three additive non-negative basis functions H.

KMeans-Clustering Basis Here, we use standard kmeans with $k = 3$ to find clusters for our finetuning loss curves, then take the three cluster centroids as basis functions.

NMF-Scale We take the basis functions derived from standard NMF, but then apply the same horizontal scaling to the basis functions as MNMF-Scale subsection A.2.

For PCA and KMeans Basis, we use ordinary least squares to compute the optimal basis weights and horizontal offset for $\mathcal{L}_{\text{partial}}$. For standard NMF, we use non-negative least squares. We report the results for these 5 methods on all training configurations in Table 11

Training Configuration	nmf_scale	mnmf_scale	pca	nmf	kmeans
Llama3.1 8b LoRA, 5×10^{-5}	22689	**4564**	6537	12900	11566
Llama3.1 8b QLoRA, 5×10^{-5}	14517	**5534**	6602	14845	11244
Llama3.1 8b QLoRA, 1×10^{-4}	13435	**7672**	12964	17525	11545
Llama3.1 8b QLoRA, 2×10^{-4}	27015	19496	20861	28996	**13043**
Llama3.2 1B Full SFT, 2×10^{-6}	43586	**26834**	40565	52524	59660
Mistral 7B QLoRA, 5×10^{-5}	12807	**6224**	12313	14424	11650
Mistral 7B QLoRA, 1×10^{-4}	14278	**6187**	12548	18180	12891
Qwen2.5 1.5B QLoRA, 1×10^{-4}	**7184**	8091	14658	11378	18456
Qwen2.5 1.5B QLoRA, 2×10^{-4}	**7297**	7364	13241	11234	17760
Qwen2.5 3B QLoRA, 1×10^{-4}	7800	**5765**	12847	13874	13134
Qwen2.5 3B QLoRA, 2×10^{-4}	**6928**	8523	17496	11540	15054
Qwen2.5 32B QLoRA, 5×10^{-5}	10597	8535	12953	**7484**	17003

Table 11: Average `needed_frac_5` "Data Needed" (lower is better) for each prediction method across the ordered training configurations. The bolded figure in each row is the lowest (best) value.

D.6 Shorter training horizons

We investigate how well MNMF-Scale can predict the final loss ($\mathcal{L}_{\text{full}}$) when the available loss observations are from training runs with shorter horizons than those used to fit the basis functions. Specifically, we train Alpaca, Dolly15k, MedicalReasoning, and OpenThoughts to 12.7k, 22.5k, 48.7k, and 34.4k examples, respectively, and use MNMF-Scale to extrapolate their loss curves beyond these observed horizons.

For each dataset, we select the loss curve corresponding to the learning rate that achieves the lowest final loss. The selected model configurations and learning rates for this evaluation are as follows:

- Llama3.1 8B LoRA, 5×10^{-5} — Alpaca, Dolly15k;
- Llama3.1 8B LoRA, 1×10^{-4} — MedicalReasoning;
- Llama3.1 8B LoRA, 2×10^{-4} — OpenThoughts;
- Mistral 7B QLoRA, 5×10^{-5} — Alpaca, Dolly15k;
- Mistral 7B QLoRA, 1×10^{-4} — MedicalReasoning, OpenThoughts;
- Qwen2.5 1.5B QLoRA, 1×10^{-4} — Alpaca;
- Qwen2.5 1.5B QLoRA, 2×10^{-4} — MedicalReasoning, OpenThoughts;
- Qwen2.5 3B QLoRA, 1×10^{-4} — Alpaca;
- Qwen2.5 3B QLoRA, 2×10^{-4} — MedicalReasoning, OpenThoughts.

To handle the discrepancy between the length of the learned basis functions and the shorter test curves, we proceed as follows:

We first fit MNMF to a corpus of loss curves obtained from models in other families with longer training horizons, producing a set of basis functions $H_{\text{mnmf}} \in \mathbb{R}^{K\times T_{\text{train}}}$, where K is the number of components, and T_{train} is the temporal horizon of the basis curves, in our case always 72k.

Given that our basis functions were learned on loss curves with a longer training horizon (T_{train}) than the test curves we wish to predict (T_{test}), we rescale each basis function to match the shorter horizon. Formally, we map each original time step $t \in \{1, \ldots, T_{\text{train}}\}$ to a new time step $t' = t \times \frac{T_{\text{test}}}{T_{\text{train}}}$. We then linearly interpolate these compressed basis functions at the integer time steps $\{1, \ldots, T_{\text{test}}\}$. After this interpolation, the scaled basis functions can be directly applied to predict loss curves observed over the shorter test horizon.

After interpolation, the length-adjusted basis set $\hat{H} \in \mathbb{R}^{K\times T_{\text{test}}}$ is used in the MNMF-Scale prediction step. The weights and scaling parameters are then solved via non-negative least squares as described in subsection 3.2, allowing us to predict the full loss curve and extrapolate the final loss for the shorter training curve.

We do not test the opposite direction, where we predict a test loss curve longer than the loss curves in the training corpus, since we do not have enough shorter loss curves trained the same horizon to use as the training corpus.