Model Balancing Helps Low-data Training and Fine-tuning

Zihang Liu[*1,2], **Yuanzhe Hu**[*1,3], **Tianyu Pang**[1], **Yefan Zhou**[1],
Pu Ren[4], **Yaoqing Yang**[1]
[1]Dartmouth College
[2]University of California, Berkeley
[3]University of California, San Diego
[4]Lawrence Berkeley National Lab
{zihang.liu, yuanzhe.hu, yefan.zhou.gr,
yaoqing.yang}@dartmouth.edu, tianyupang628@gmail.com,
pren@lbl.gov

Abstract

Recent advances in foundation models have emphasized the need to align pre-trained models with specialized domains using small, curated datasets. Studies on these foundation models underscore the importance of low-data training and fine-tuning. This topic, well-known in natural language processing (NLP), has also gained increasing attention in the emerging field of scientific machine learning (SciML). To address the limitations of low-data training and fine-tuning, we draw inspiration from Heavy-Tailed Self-Regularization (HT-SR) theory, analyzing the shape of empirical spectral densities (ESDs) and revealing an imbalance in training quality across different model layers. To mitigate this issue, we adapt a recently proposed layer-wise learning rate scheduler, `TempBalance`, which effectively balances training quality across layers and enhances low-data training and fine-tuning for both NLP and SciML tasks. Notably, `TempBalance` demonstrates increasing performance gains as the amount of available tuning data decreases. Comparative analyses further highlight the effectiveness of `TempBalance` and its adaptability as an "add-on" method for improving model performance.

1 Introduction

Recent surges in foundation models (FMs) have stimulated research on aligning pre-trained models with specialized domains using small-sized datasets. This "pre-train and fine-tune" paradigm is prevalent in natural language processing (NLP) tasks (Wang et al., 2019, 2020; Rajpurkar et al., 2016; Lu et al., 2022). It is also gaining popularity in other machine learning (ML) fields, such as scientific machine learning (SciML) (Subramanian et al., 2024; Lanusse et al., 2023; McCabe et al., 2023; Wu et al., 2023; Hao et al., 2024; Chen et al., 2024). From a practical perspective, the challenge of fine-tuning often lies in curating high-quality datasets (possibly with labeled examples) to achieve alignment with the new domain. In SciML, people often use FMs for training on different types of partial differential equations (PDEs) (McCabe et al., 2023; Wu et al., 2023; Hao et al., 2024) and fine-tuning it on a certain domain when accessible scientific data from that domain is limited. As a concrete example, turbulence simulations at extremely high Reynolds numbers are computationally intensive and time-consuming, often leading to only a few available trajectories. Therefore, training SciML FMs on trajectories with different Reynolds numbers and fine-tuning it on trajectories simulated at extremely high ones is beneficial for solving the problem of poor training performance caused by insufficient data volume. Using SciML FMs, researchers can train these models to generalize across a wider range of downstream tasks, thereby enhancing their applicability and efficiency in diverse scientific scenarios. Prior research has shown that strong performance can indeed be achieved by fine-tuning with a few carefully selected examples (Zhou et al., 2023), but training with low data can still lead to unstable performance (Zhang et al., 2021). Therefore, finding fine-tuning algorithms that improve performance in low-data settings, especially few-shot alignment, becomes crucial.

In this work, we draw inspiration from Heavy-Tailed Self-Regularization (HT-SR) theory (Martin and Mahoney, 2021; Martin et al., 2021), to improve model performance in low-data regimes. HT-SR theory proposes that well-trained neural network (NN) models exhibit strong correlations in weights, resulting in a Heavy-Tail (HT) structure in the Empirical Spectral Density (ESD, usually represented by a histogram of eigenvalue distribution) of each layers' weight matrix. To quantify the HT structure, we can fit a power law (PL) distribution to the HT part of the ESD and extract

[*]Equal contribution. Work completed during an internship at Dartmouth College.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 1311–1331
November 12-16, 2024

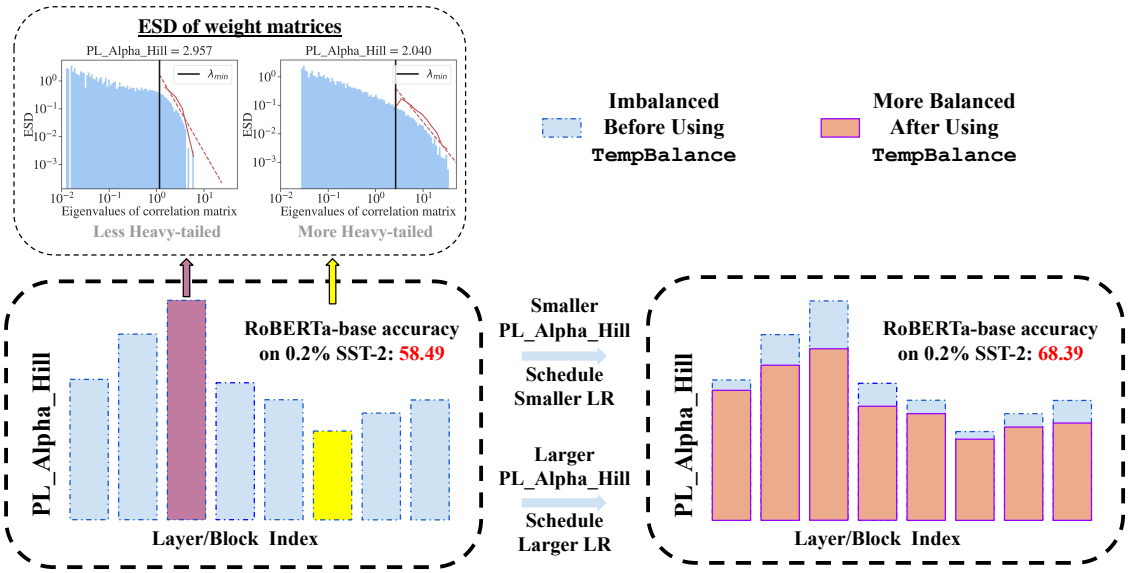


Figure 1: **Heavy-tail ESD analysis and `TempBalance` learning rate schedule.** To characterize the heavy-tailed structure of ESD, we fit a power-law exponent `PL_Alpha_Hill` on the tail part of the ESDs (blue histograms at top left), shown as the red dashed line on the histogram. Given the imbalanced layer-wise `PL_Alpha_Hill` (bottom left), `TempBalance` assigns lower learning rate to layers with lower `PL_Alpha_Hill` (more heavy-tailed), and assign higher learning rate to layers with higher `PL_Alpha_Hill` (less heavy-tailed). `TempBalance` aims to balance the `PL_Alpha_Hill` distribution across layers in low-data regimes (bottom right).

its exponent, namely `PL_Alpha_Hill` (see Figure 1). HT-SR theory suggests that a more HT ESD (lower `PL_Alpha_Hill`) represents better training quality, and vice versa. This estimation of model and layer quality has been shown to be effective in recent work on model selection (Martin et al., 2021; Martin and Mahoney, 2020, 2022; Yang et al., 2023), layer-wise hyperparameter tuning (Zhou et al., 2024), and pruning of large language models (LLMs) (Lu et al., 2024).

Using HT-SR theory, we analyze the limitations of model training in low-data regimes by measuring the layer-wise distribution of `PL_Alpha_Hill` (discussed in 4.2). Our main finding is that when we train with sufficient data, `PL_Alpha_Hill` becomes more evenly distributed across layers, resulting in better layer-wise balance; in this case, high performance can be achieved without layer-specific manipulations. However, when we reduce the number of training data samples, test performance decreases, and the standard deviation (STD) of `PL_Alpha_Hill` across layers tends to increase (see Figure 2), indicating that `PL_Alpha_Hill` is more unevenly distributed when training with fewer data, resulting in worse layer-wise balance. This finding indicates that different layers' training quality becomes more poorly aligned as we reduce training data. Therefore, layer-wise balancing is beneficial to balance under-trained layers and over-trained layers in low data regimes.

Motivated by this observation, we incorporate the variance of `PL_Alpha_Hill` across layers with the recently proposed layer-wise learning rate scheduling algorithm `TempBalance` (Zhou et al., 2024), to design a novel method to balance the training quality across layers. To evaluate its empirical performance, we use `TempBalance` in curated low-data regime in LLM fine-tuning and SciML tasks. We compare `TempBalance` with commonly used baseline methods and demonstrate that `TempBalance` not only achieves superior performance in low-data training and fine-tuning, but also can be used as a plug-in method on top of existing optimization methods to achieve even better test performance and stability, such as SAM (Foret et al., 2021) and AdaFactor (Shazeer and Stern, 2018). Furthermore, in our analysis, we reveal that `TempBalance` successfully balances training quality across all layers during training from the HT-SR point of view. We show that `TempBalance` balances the training quality of each layer by reducing the STD of `PL_Alpha_Hill` of all layers. We summarize our contributions as follows [1]:

[1]In order that our results can be reproduced and extended, we have open-sourced our code. https://github.com/ZihangHLiu/ModelBalancing.

- We find that low-data fine-tuning is a crucial training paradigm that can lead to imbalanced training quality across different layers of the model, measured by the large STD of `PL_Alpha_Hill` values across layers.

- We focus on low-data training scenarios and demonstrate the effectiveness of using `TempBalance` to balance layers and improve the performance of both NLP and SciML models. For example, we show that `TempBalance` can improve RoBERTa-base trained on SST2 dataset by at most 9.9% and increase the test accuracy of LLaMA-7B on ScienceQA dataset by at most 1.97%, and reduce the normalized root-mean-squared-error (nRMSE) of FNO trained on 2D Compressible Navier-Stokes(CFD)[2] dataset by 14.47%. Furthermore, we show that `TempBalance` achieves gradually increased performance gains as the number of data points decreases.

- In LM fine-tuning tasks, we demonstrate that `TempBalance` can achieve better fine-tuning performance compared to baselines (including SAM (Foret et al., 2021) and AdaFactor (Shazeer and Stern, 2018)) and can be used as an add-on method to combine with these existing optimization methods to achieve further improvements.

2 Related Work

Heavy-tailed Phenomenon. Recently, several studies have observed that a well-trained deep NN exhibits HT spectra in its weight matrices. Many papers focus on investigating the cause of the emergence of HT, and they have attributed HT spectra (or limiting HT distributions of weights) to strong correlation in weight elements (Martin and Mahoney, 2021; Martin et al., 2021), feature learning (Wang et al., 2024b; Kothapalli et al., 2024), the Kesten–Goldie mechanism (Hodgkinson and Mahoney, 2021; Gurbuzbalaban et al., 2021), α-stable Lévy process (Gurbuzbalaban et al., 2021; Simsekli et al., 2020), and the maximum-entropy principle (Xie et al., 2024). More importantly, several studies have shown that the heavytailness of the weight spectra is strongly correlated with the quality of neural networks. For example, Martin and Mahoney (2021) proposed HT-SR theory, demonstrating that the degree of HT in the ESD of each layer can be used to predict model quality: the heavier the tail of the ESD, the better the quality of the model. In addition, Simsekli et al. (2020); Hodgkinson et al. (2022); Wang et al. (2024a) proved generalization bounds dependent on the HT distributions in either model weights or the ESDs of the weight matrices, which are validated through extensive experiments. Motivated by these studies, some efforts have begun to leverage the degree of HT for model training (Zhou et al., 2024; Li et al., 2024; Qing et al., 2024), model selection (Agrawal et al., 2022; Yang et al., 2023), and model compression (Barsbey et al., 2021; Lu et al., 2024), as well as to enhance model robustness (Nassar et al., 2020).

Resource-constrained Fine-tuning. The pre-training and fine-tuning paradigm has been a primary method for adapting foundation models to downstream tasks for resource-limited users. When adapting very large models, people often resort to the Low-Rank Adaptation method (LoRA) (Hu et al., 2021), which is also considered in this paper. Our primary focus is on low-data fine-tuning, an increasingly studied paradigm where the emphasis is often on careful data selection (Zhou et al., 2023). Furthermore, when training models in a few-shot fashion, such as *in-context learning* (Brown et al., 2020; Logan IV et al., 2021; Zhang et al., 2022), data selection plays a crucial role in improving model performance. Our paper, however, explores layer-balancing schemes to improve model performance.

Data-constrainted Training and Fine-tuning in SciML. There has been an increasing interest in the use of ML methods to solve scientific problems (Raissi et al., 2019; Li et al., 2020; Karniadakis et al., 2021; Wang et al., 2023). One representative line of work is on neural operators (Li et al., 2020; Lu et al., 2021; Hao et al., 2023; Raonic et al., 2024). These operators have demonstrated their effectiveness in scientific modeling. However, they require extensive scientific datasets. Generating high-fidelity numerical datasets is computationally demanding. Hence, to mitigate the costs associated with simulation, self-supervised pretraining has been introduced for operator learning (Chen et al., 2024). Additionally, in low-data regimes, researchers also propose to incorporate physical laws into ML models to facilitate the learning of the underlying governing equations, often through soft regularization constraints (Raissi et al., 2019). Nevertheless, the physics-constrained ML strategy is limited to specific PDE scenarios (e.g., fixed

[2]CFD means compressible fluid dynamics or, equivalently, the compressible Navier-Stokes equations.

PDE coefficients) (Ye et al., 2024), which poses challenges to generalization.

3 Methodology

In this section, we first revisit HT-SR theory and important HT-SR metrics related to model performance. Then, we discuss `TempBalance` (Zhou et al., 2024), which works well on different model architectures based on "shape metrics" from HT-SR Theory.

3.1 HT-SR Theory

HT-SR theory (Martin and Mahoney, 2021) demonstrates the empirical fact that very well-trained models tend to exhibit strong correlations in weights, resulting in HT structure in the ESD of each layer. Its underlying motivation stems from random matrix theory and statistical physics, as well as the observation that HT ESDs are ubiquitous in well-trained NN models.

Obtaining the ESD of Weight Matrices. To obtain the ESDs of a model, we take an NN with L layers and the corresponding weight matrices $\mathbf{W}_1, \mathbf{W}_2, \cdots, \mathbf{W}_L$. For the i-th layer, we calculate the eigenvalues of its correlation matrix $\mathbf{X}_i = \mathbf{W}_i^\top \mathbf{W}_i$. Then, we plot the ESD for that layer, which is the empirical distribution of these eigenvalues. During training, the ESD will typically gradually change to have an HT structure. There are many metrics that have been proposed to study the properties of ESDs, among which shape metrics (metrics that depict the shape of ESD) have been shown to predict the training quality of each layer (Yang et al., 2023).

Analyzing ESDs with PL Fitting. To obtain robust shape metrics that predict layer quality, we fit a PL distribution to the heavy-tailed part of the ESD within an interval $(\lambda_{\text{min}}, \lambda_{\text{max}})$. The PL fit has the following formula:

$$p(\lambda) \propto \lambda^{-\alpha}, \lambda_{\text{min}} < \lambda < \lambda_{\text{max}}. \tag{1}$$

We then extract its exponent α as an empirical metric. To fit a PL distribution to the ESD, we use the Hill Estimator (Hill, 1975; Zhou et al., 2024): for the i-th layer, suppose the weight matrix is $\mathbf{W}_i$ and the correlation matrix $\mathbf{W}_i^\top \mathbf{W}_i$ has ascending eigenvalues $\{\lambda_i\}_{i=1}^n$. The Hill estimator calculates `PL_Alpha_Hill` as:

$$\texttt{PL\_Alpha\_Hill} = 1 + \frac{k}{(\sum_{i=1}^{k} \ln \frac{\lambda_{n-i+1}}{\lambda_{n-k}})}, \tag{2}$$

where k is an adjustable parameter.

`PL_Alpha_Hill` Distribution and Model Quality. When using `PL_Alpha_Hill` to analyze model performance, related works suggest that a layer with smaller `PL_Alpha_Hill` tends to be relatively "overtrained" (compared to other layers in the model), while layers with higher `PL_Alpha_Hill` are relatively "undertrained." (Zhou et al., 2024) find that in CV tasks, models trained with optimized hyperparameter scheduling outperform baseline methods and yield a more concentrated `PL_Alpha_Hill` distribution across layers, suggesting that a more uniformly distributed `PL_Alpha_Hill` has more balanced training quality across layers, leading to better overall quality of the model.

3.2 `TempBalance` Algorithm

Prior research (Martin and Mahoney, 2021) has shown that temperature-like parameters significantly influence the HT structure of individual layers' ESDs. Therefore, to balance the shape of ESDs across layers, we propose to adapt the `TempBalance` algorithm, which dynamically tunes the learning rate on a layer-wise basis, as the learning rate is the most important temperature parameter. Smaller learning rates are assigned to layers with more heavy-tailed ESDs to slow down the training, while larger learning rates are assigned to those with more light-tailed ESDs to accelerate the training. We propose a novel method to map the `PL_Alpha_Hill` of each layer to the layer-wise learning rate. We first calculate their difference with the mean `PL_Alpha_Hill` value across all layers, then rescale the difference using a sigmoid-like function. Finally, we use the rescaled value as the exponent to assign the new learning rate $f_t(i)$ for the layer. We refer to this scheduling algorithm as `TB_Sigmoid`. The equations are as follows:

$$f_t(i) = \eta_t \cdot 10^{\phi}, \tag{3}$$

$$\phi = s \cdot \left(\frac{1}{1 + e^{-\tau \cdot (\alpha_i - \overline{\alpha})}} - 0.5 \right), \tag{4}$$

where η_t is the base learning rate at step t, α_i is the `PL_Alpha_Hill` of layer i, and $\overline{\alpha}$ is the mean `PL_Alpha_Hill` across all layers. Note that s and τ are tunable hyperparameters in experiments, and we often obtain the best results when we set $\tau = 10$. In `TempBalance`, if a layer's `PL_Alpha_Hill` is higher than the mean, a learning rate higher than the base learning rate is assigned, and if it is lower, a lower

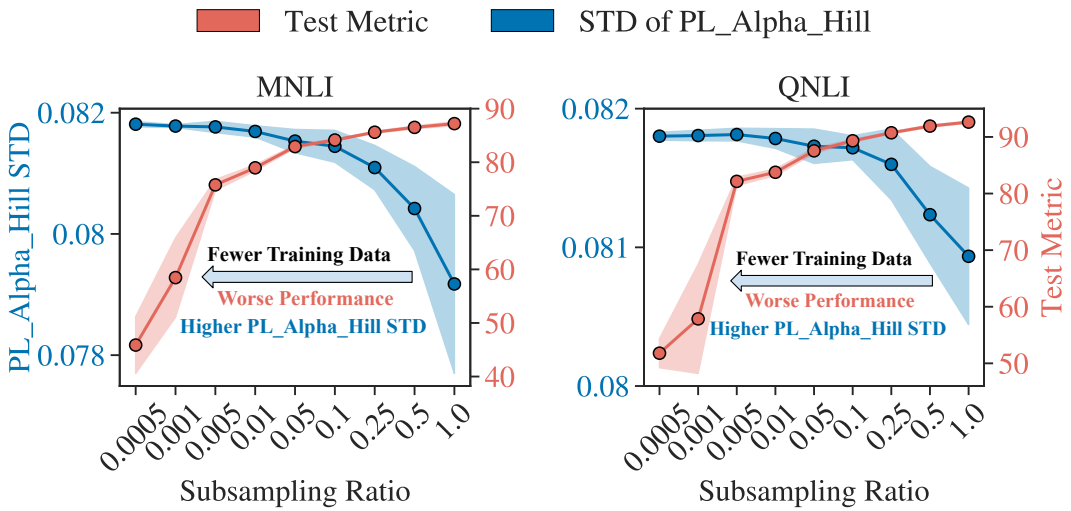


Figure 2: Test performance and STD of `PL_Alpha_Hill` across all layers of RoBERTa-base model trained on MNLI (Accuracy↑) and QNLI (Accuracy↑) under different subsampling ratios.

learning rate is assigned. Furthermore, layers with `PL_Alpha_Hill` significantly different from the mean receive more substantial adjustments, while those closer to the mean receive minimal changes. The intuition of this scheduling function is that it not only controls `PL_Alpha_Hill` by adjusting the learning rate based on its value, but also takes the difference of `PL_Alpha_Hill` to the mean into account to reduce the variance of `PL_Alpha_Hill` across layers by assigning learning rate changes proportional to the difference, finally balancing the training quality. In Table 4, we empirically show that `TB_Sigmoid` works better than other layer-wise learning rate scheduling methods.

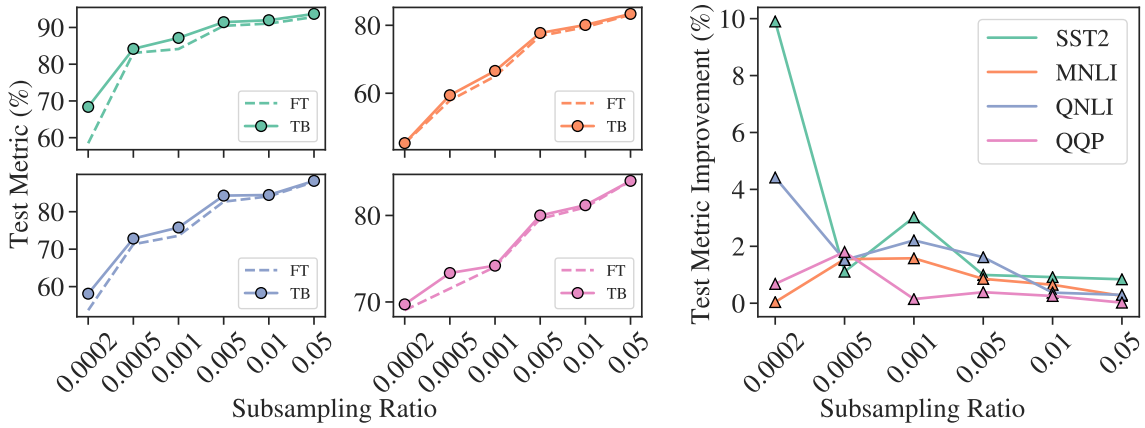
Using `TempBalance` on Transformers. For Transformer-based architectures, we note each Transformer block consists of different types of layers (such as Query, Output, and Down Projection) with different matrix sizes, resulting in distinct ESD shapes. Therefore, we explore a more favorable scheduling method to eliminate unfair comparison of `PL_Alpha_Hill` of different ESD shapes. We reschedule each blocks' learning rate by averaging the `PL_Alpha_Hill` across all layers within the block, while in each block we use the same learning rate across all layers. In Table 5 in Appendix B, we show that the per-block scheduling method consistently outperforms the per-layer method in different low-data regimes. Given such a design, we note that a "layer" used in this work when discussing Transformer-based models refers to a Transformer block.

4 Empirical Results

In this section, we employ HT metrics to diagnose model performance in data-limited regimes and demonstrate the effectiveness of `TempBalance` in addressing data limitation in two fields: NLP and SciML. In Section 4.1, we describe our experimental setup. In Section 4.2, we study the correlation between ESD behaviors and model performance with limited training data. Then, in Section 4.3, we evaluate `TempBalance` in our experimental setup. In Section 4.4, we compare our methods with other optimization baselines. We analyze the experimental results in Section 4.6. Finally, we perform ablation studies in Section 4.7.

4.1 Experimental Setup

Models and Evaluation. For NLP, we evaluate `TempBalance` with two widely-used fine-tuning methods: Full fine-tuning (FT) and LoRA fine-tuning (Hu et al., 2021) using the Huggingface framework (Wolf et al., 2020). We select two models with distinct sizes: RoBERTa-base (Liu et al., 2019) and LLaMA2-7b (Touvron et al., 2023). We train the models on subsampled common fine-tuning datasets, including GLUE (Wang et al., 2019), SuperGLUE (Wang et al., 2020), SQuAD (Rajpurkar et al., 2016), and ScienceQA (Lu et al., 2022). We train with sampling ratios ranging from 0.02% to 50% to evaluate our method. We also evaluate `TempBalance` on low-resource datasets from three specialized domains: BioMed, CS, and News. We choose five datasets from these domains: RCT with 500 samples (Dernoncourt and Lee, 2017), SciCite (Co-



a RoBERTa-base on GLUE datasets

b Trend of improvement

Figure 3: **(Main Results on LLM Fine-tuning).** `TempBalance` (TB) achieves better test metric (↑) than baseline Full Fine-tuning (FT) on GLUE tasks, especially if training data is small. 3a compares test performances of baseline FT (Full Fine-tuning) and `TempBalance` to train RoBERTa-base model on four larger GLUE datasets (color-coded as in 3b). 3b shows the trend of performance improvement of `TempBalance`.

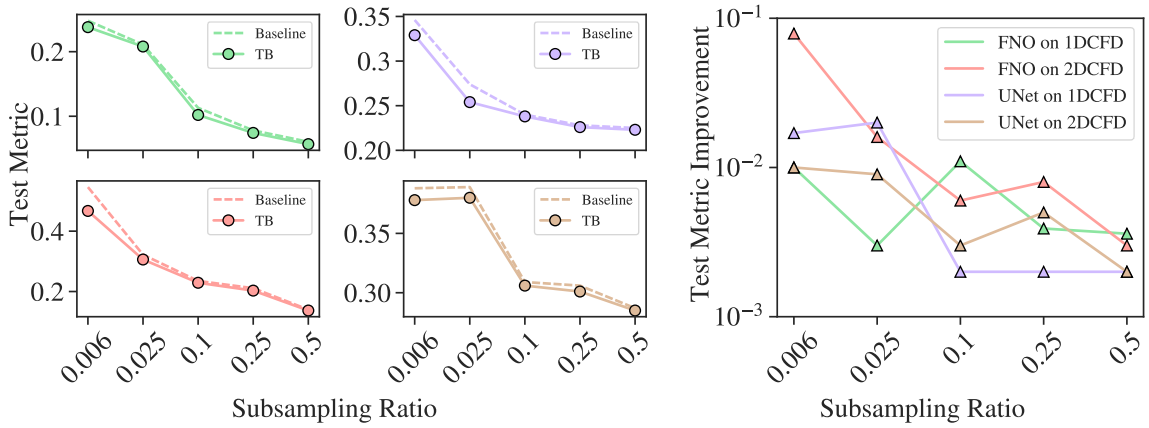
han et al., 2019), ChemProt (Kringelum et al., 2016), SciERC (Luan et al., 2018), and Hyperpartisan News (Kiesel et al., 2019), and we train the RoBERTa-base model with entire datasets. For SciML, we evaluate `TempBalance` by training or fine-tuning neural PDE solvers to learn PDEs. We use previously studied SciML models, including FNO (Li et al., 2020), UNet (Ronneberger et al., 2015) and DPOT (Hao et al., 2024). We train the models on simulated solutions of PDEs: one time-independent PDE (DarcyFlow) and two time-dependent PDEs (1D and 2D CFD), with a sampling ratio from 0.6% to 100%.

Baselines. To ensure fair comparison, we use publically available pre-trained checkpoints for training, and adopt training configurations from previous works to reproduce their results. For NLP tasks, we use FT and LoRA to train the RoBERTa-base (125M) model, and we use the Adam optimizer (Kingma and Ba, 2014) with linear learning rate decay with warmup; for SciML tasks, we refer the experiments settings from (Takamoto et al., 2022), use the Adam optimizer and schedule the base learning rate by step-wise learning rate decay. To obtain a proper hyperparameter setup, we perform grid searches on temperature parameters (learning rate, batch size). For other training configurations, we refer to existing works (Liu et al., 2019; Hu et al., 2021; Yang and Osher, 2024), and find the best hyperparameters. See Appendix C and D for details on dataset subsampling and hyperparameter configurations, respectively.

4.2 Diagnosing Layer Imbalance Using HT Metrics when Training with Limited Data

To analyze the performance of models trained in low-data settings, we employ HT-SR theory and examine the distribution of `PL_Alpha_Hill` across different layers. Our findings reveal a strong correlation between the trend of `PL_Alpha_Hill` distribution and test performance. We use checkpoints of the RoBERTa-base model trained with subsampling ratios ranging from 0.05% to 100% on MNLI and QNLI dataset, and we plot the trend of test performance and block-wise STD of `PL_Alpha_Hill`, as shown in Figure 2. As test performance decreases with training data samples, we observe that the STD of `PL_Alpha_Hill` across layers increases, suggesting a more unevenly distributed `PL_Alpha_Hill` across different layers. Similar trends are also present in SciML tasks (Figure 6).

Given that `PL_Alpha_Hill` is a robust predictor of model and layer quality (Yang et al., 2023; Zhou et al., 2024), we propose that models trained on fewer data samples have more unevenly distributed layer qualities, this layer-wise balance becomes worse as we reduce the number of training data points. Training with more data points, on the other hand, can make the distribution of `PL_Alpha_Hill` more balanced. Therefore, when training with limited data, layer balancing is necessary for balancing the training quality of different layers.



a FNO and UNet on 1D and 2D CFD datasets

b Trend of improvement

Figure 4: **(Main Results on PDE Learning).** `TempBalance` (TB) achieves lower nRMSE(↓) than baseline method on CFD tasks, especially if subsampling ratio is small. 4a compares test performances of baseline trained and `TempBalance` trained FNO and UNet models on 1D and 2D CFD datasets (color-coded as in 4b). 4b demonstrates the trend of performance improvement brought by `TempBalance`.

4.3 Improving Low-Data Training Using `TempBalance`

Natural Language Understanding. In Figure 3, we report the evaluation result of fine-tuning the RoBERTa-base model with four larger GLUE datasets. We compare `TempBalance` (shown as "TB") with Full Fine-tuning (shown as "FT") with different subsampling ratios. We also show the results on smaller GLUE tasks in Table 18. We can see that `TempBalance` consistently demonstrates performance improvement in all low-data regimes. For example, when fine-tuning on the larger SST2 dataset, `TempBalance` significantly outperforms the baseline with 9.9% improvement in test accuracy with 0.02% subsampling ratio. Regarding the smaller RTE dataset with 50% training data, `TempBalance` can improve test accuracy by 3.13%. The detailed results of all GLUE tasks are shown in Table 17 and 18, in Appendix E.1.

Domain-specific Language Modeling. In Figure 5, we report the results of `TempBalance` on five domain-specific low-resource datasets. We show that when fine-tuned on these datasets in low-data settings, `TempBalance` continues to yield better test performance than the baseline method. Specifically on Hyperpartisan News dataset, `TempBalance` outperforms baseline FT by 5.13%. This indicates that `TempBalance` brings significant improvement when applying to specialized language modeling domains with low resources.

Neural PDE Solver Training. In Figure 4, we report the results of training the FNO and UNet model on the 1D and 2D CFD (compressible fluid dynamics) dataset with a subsampling ratio ranging from 0.6% to 100%, evaluated by Normalized Root Mean Squared Error (nRMSE). The detailed results are shown in Table 19, Appendix E.4. We find that `TempBalance` achieves lower nRMSE compared to the baseline on all subsampling ratios. Specifically, `TempBalance` reduces the nRMSE of the FNO model trained on 10.0% of the 1DCFD dataset significantly by 9.73% and improves the nRMSE of UNet on 2.5% by 7.30%. Furthermore, `TempBalance` can achieve comparable performance gain to increasing the number of training data samples. For example, when solving 2DCFD problem using the UNet model with 10% data, applying `TempBalance` yields comparable performance gain to increasing the subsampling ratio to 25%.

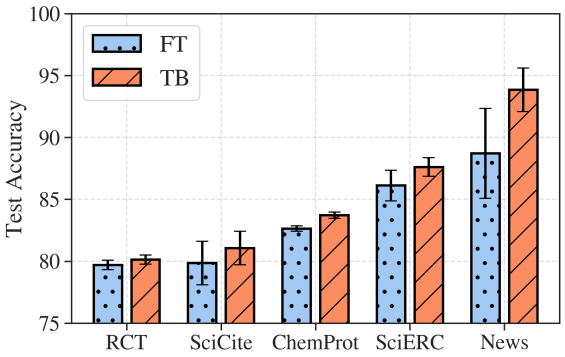


Figure 5: **Domain Specific Language Modeling.** `TempBalance` demonstrates significant performance gain when training the RoBERTa-base model on five low-resource domain-specific datasets.

Complementary Results. To further demonstrate the generalizability of `TempBalance`, we pro-

Ratio	1%	0.5%	0.1%	0.05%
FT	84.09±0.36	82.68±0.43	73.57±0.90	71.31±1.29
SAM	**85.10±0.55**	83.35±0.61	73.38±1.48	71.18±1.29
TB	84.47±0.55	**84.30±0.46**	**75.67±1.17**	**72.65±1.10**

Ratio	1%	0.5%	0.1%	0.05%
FT	84.09±0.36	82.68± 0.43	73.57±0.90	71.31±1.29
TB	84.47±0.55	83.40±0.46	75.67±1.17	72.65±1.10
AdaFactor	84.79±0.37	83.29±0.23	76.73±0.95	74.09±1.29
AdaFactor+TB	**84.81±0.25**	**84.00±0.46**	**77.75±0.38**	**76.04±1.10**

Table 1: Comparing `TempBalance` with Sharpness-Aware Minimization (SAM) and AdaFactor on RoBERTa-base model trained with QNLI dataset. For SAM, we choose hyperparameter ρ in the range of {0.5, 0.25, 0.1, 0.05}

vide supplementary results on a broader range of settings in Appendix E. We first evaluate `TempBalance` on more full fine-tuning and LoRA fine-tuning tasks of RoBERTa-base and LLaMA-7B, then we explore more SciML settings by training the FNO and UNet to solve CFD PDEs. We also provide statistical testing to verity the significance of our results.

4.4 Comparison with Other Methods

Recent works have proposed optimization methods that efficiently improve low-data training especially on LLMs. For example, Sharpness-Aware Minimization (SAM) (Foret et al., 2021) has been shown to effectively improve fine-tuning performance when training data is limited, by encouraging convergence to flatter local minima (Bahri et al., 2022). Also, AdaFactor is a memory-efficient optimizer suitable for training large models (Shazeer and Stern, 2018). We show that `TempBalance` not only outperforms these methods in most low-data regimes, but can be used as an "add-on" method to further enhance model performance.

We compare `TempBalance` with SAM and AdaFactor using RoBERTa-base model trained with QNLI on four subsampling ratios, as shown in Table 1. We can see that when we have fewer data points, SAM achieves worse results than baseline FT. Meanwhile, `TempBalance` consistently outperforms baseline FT, and achieves better results than SAM in almost all cases. For the AdaFactor optimizer, we can see that it outperforms baseline and `TempBalance` in most cases. Still, when we combine `TempBalance` with AdaFactor, we can achieve the best results across all low-data regimes, with at most 1.95% test accuracy increase higher than AdaFactor alone.

4.5 Neural PDE Fine-tuning

To explore diverse scenarios in SciML, we conduct experiments on low-data fine-tuning using the 2DCFD dataset with DPOT-Tiny and DPOT-Small models. In solving PDEs, we utilize foundational models pre-trained on various fluid dynamics datasets, which are then fine-tuned on another specific physical scenario. In Table 2, we show that `TempBalance` (TB) offers better improvements compared to the baseline FT under different subsampling ratios.

The experimental settings for SciML tasks are as follows: For `TempBalance` (TB) and FT, we train the models for 500 epochs with a batch size of 160 for the Tiny model and 64 for the Small model, and a dropout rate of 1e-6. We test initial learning rates among {0.001, 0.0005, 0.00025, 0.0001, 0.00005}. We use the Adam optimizer, and decay the learning rate by $\gamma = 0.5$ every 50 epochs. The mean and standard deviation of nRMSE across 3 random seeds on the test set are reported.

Subsampling Ratio	Method	DPOT-Tiny	DPOT-Small
5%	FT	1.863e-2±1.067e-5	1.546e-2±3.346e-5
	TB	**1.856e-2±3.646e-5**	**1.539e-2±1.328e-5**
10%	FT	1.747e-2±1.502e-5	1.426e-2±1.157e-5
	TB	**1.730e-2±1.173e-5**	**1.415e-2±1.890e-5**
25%	FT	1.543e-2±4.008e-5	1.226e-2±2.094e-5
	TB	**1.517e-2±2.807e-5**	**1.203e-2±1.313e-5**
50%	FT	1.309e-2±2.356e-5	1.025e-2±2.063e-5
	TB	**1.283e-2±2.494e-5**	**1.005e-2±8.860e-6**
100%	FT	1.096e-2±3.875e-5	8.400e-3±1.030e-5
	TB	**1.078e-2±4.527e-5**	**8.193e-3±1.509e-5**

Table 2: `TempBalance` achieves lower nRMSE(↓) than baseline method on SciML fine-tuning task.

4.6 Analysis

Following section 4.2, we study the effectiveness of `TempBalance` in overcoming low-data limitations. First, we look into the trend of improvement brought by `TempBalance`, and demonstrate that layer-wise tuning like `TempBalance` brings more significant improvement as we train with fewer data. Second, we investigate the distribution of `PL_Alpha_Hill` across layers, and show that `TempBalance` successfully balances layer-wise training quality, resulting in a more uniform `PL_Alpha_Hill` distribution compared to the baseline method.

Analyzing Performance Gain of `TempBalance`.

As we have shown in our main results, we note that TempBalance achieves greater performance gain as the subsampling ratio becomes lower. This trend suggests that TempBalance is more effective as we train the model with fewer data. This trend suggests that when training data is large, model training quality is high without specific manipulations. However, if we only have a few samples, the layer-wise balancing method becomes increasingly beneficial and can significantly improve model performance.

Analyzing PL_Alpha_Hill Distribution. We compare the distribution of PL_Alpha_Hill between baseline FT and TempBalance. As observed in Figure 7, TempBalance consistently shows lower PL_Alpha_Hill variance on RoBERTa-base trained on QNLI under various subsampling ratios. Furthermore, in SciML tasks, we can see a similar trend that is more significant when we train the model from scratch (Figure 8).

Following the trend shown previously in Figure 2, this finding suggests that as layer-wise training quality becomes more unevenly distributed as we train with fewer data, TempBalance effectively balances training quality across different layers (estimated by the variance of PL_Alpha_Hill).

4.7 Ablation study

Temperature Balancing with Different ESD Metrics. Recent theoretical works have proposed several metrics that measure the shape of the ESD (Martin and Mahoney, 2021; Martin et al., 2021; Yang et al., 2023), and we compare their performance with PL_Alpha_Hill in assigning layer-wise learning rates. We mainly consider two shape metrics: Spectral_Norm and Stable_Rank. Results are presented in Table 3. We can see that in all subsampling ratios, PL_Alpha_Hill continues to outperform other metrics, while other metrics may perform worse than baseline Full FT. We can conclude that PL_Alpha_Hill have more robust performance than other shape metrics in assigning layer-wise learning rates.

Different Learning Rate Scheduling functions. In the TempBalance algorithm, we choose TB_Sigmoid equation as our layer-wise scheduling function. To verify the superiority of TB_Sigmoid function, we evaluate another scheduling function TB_Linear_Map, which is proven to have great performance on image classification tasks (Zhou et al., 2024). The results are shown in Table 4. We can see that TB_Sigmoid function outperforms TB_Linear_Map in almost all subsampling ratios.

Ratio	1%	0.5%	0.1%	0.05%
FT	84.09±0.36	82.68± 0.43	73.57±0.90	71.31±1.29
Spectral_Norm	83.18±0.41	81.68±0.23	70.52±5.18	65.79±0.85
Stable_Rank	83.22±0.15	82.29±0.36	71.87±1.57	67.18±3.71
PL_Alpha_Hill	**84.47±0.55**	**84.30±0.46**	**75.78±0.47**	**72.83±1.65**

Table 3: Comparing different ESD metrics used to schedule layer-wise learning rate trained with RoBERTa-base model on QNLI task. We choose Spectral_Norm and Stable_Rank to compare with PL_Alpha_Hill that we use in the TempBalance algorithm.

Ratio	1%	0.5%	0.1%	0.05%
FT	84.09±0.36	82.68±0.43	73.57±0.90	71.31±1.29
TB_Linear_Map	**84.60±0.07**	83.87±0.61	73.49±2.92	72.76±1.54
TB_Sigmoid	84.47±0.55	**84.30±0.46**	**75.78±0.47**	**72.83±1.65**

Table 4: Comparing different Temperature Balancing scheduling algorithm on RoBERTa-base model trained with QNLI dataset.

For more ablation study results on SciML tasks, please refer to Appendix G.1.

5 Conclusions

In this work, we leverage HT-SR theory to diagnose the limitations of low-data training and improve the learning rate scheduling algorithm TempBalance to balance the training quality of different layers in low-data regimes. Our extensive experiments demonstrate that TempBalance effectively balances layer-wise training quality and improves performance in NLP fine-tuning and SciML training. Our analysis reveals that TempBalance achieves greater performance gain as we train with fewer data. Furthermore, the compatibility of TempBalance makes it possible to add TempBalance to existing optimization methods, bringing further performance improvements. We show that HT-SR theory brings useful guidance in low-data training and fine-tuning, and we expect it to be a more generalized toolbox for diagnosing model performance in more training scenarios.

Acknowledgments. This work is supported by DOE under Award Number DE-SC0025584, DARPA under Agreement number HR00112490441, and Dartmouth College.

Limitations

Despite achieving improvements in NLP and SciML tasks, `TempBalance` has some potential limitations.

For computational costs, since `TempBalance` dynamically reschedules learning rates during training, frequent calculations of ESD of weight matrices are required. In our work, the computation overhead of `TempBalance` during training the RoBERTa-base model can take up to 25% of the total training time: when training on 0.02% SST2 dataset, the total training time is 265.73 seconds, in which `TempBalance` takes up 65.40 seconds. This computational cost could scale up as the model size becomes larger. Since the calculation of ESD contributes to most of the computation cost (the SVD process), we will focus on improving the efficiency of measuring the Heavy-Tail structure of the ESD.

In addition, we only discuss the scheduling of the learning rate in this work, whereas other temperature-like parameters can also influence the structure of ESD during training, such as batch size or weight decay. Therefore it would be of interest to explore how HT-SR theory can assist in acquiring a comprehensive set of hyperparameter tuning tools.

Ethics Statement

This paper leverages HT-SR theory to design a layer-wise fine-tuning scheme for LLMs and SciML models. Our study in itself does not pose any negative societal risks or ethical concerns. On the contrary, it improves our understanding of the inner mechanisms of training NNs which can potentially aid in optimizing the amount of compute resources spent on training large NNs for wide societal use.

References

Kumar Krishna Agrawal, Arnab Kumar Mondal, Arna Ghosh, and Blake Aaron Richards. 2022. α-req : Assessing {\bf Re}presentation {\bf Q}uality in self-supervised learning by measuring eigenspectrum decay. In *Advances in Neural Information Processing Systems*.

Dara Bahri, Hossein Mobahi, and Yi Tay. 2022. Sharpness-aware minimization improves language model generalization. *Preprint*, arXiv:2110.08529.

Melih Barsbey, Milad Sefidgaran, Murat A Erdogdu, Gael Richard, and Umut Simsekli. 2021. Heavy tails in sgd and compressibility of overparametrized neural networks. *Advances in neural information processing systems*, 34:29364–29378.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wuyang Chen, Jialin Song, Pu Ren, Shashank Subramanian, Dmitriy Morozov, and Michael W Mahoney. 2024. Data-efficient operator learning via unsupervised pretraining and in-context learning. *Advances in Neural Information Processing Systems*.

Arman Cohan, Waleed Ammar, Madeleine Van Zuylen, and Field Cady. 2019. Structural scaffolds for citation intent classification in scientific publications. In *NAACL*.

Franck Dernoncourt and Ji Young Lee. 2017. Pubmed 200k rct: a dataset for sequential sentence classification in medical abstracts. *arXiv preprint arXiv:1710.06071*.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. 2021. Sharpness-aware minimization for efficiently improving generalization. *Preprint*, arXiv:2010.01412.

Mert Gurbuzbalaban, Umut Simsekli, and Lingjiong Zhu. 2021. The heavy-tail phenomenon in sgd. In *International Conference on Machine Learning*, pages 3964–3975. PMLR.

Zhongkai Hao, Chang Su, Songming Liu, Julius Berner, Chengyang Ying, Hang Su, Anima Anandkumar, Jian Song, and Jun Zhu. 2024. Dpot: Auto-regressive denoising operator transformer for large-scale pde pre-training. *arXiv preprint arXiv:2403.03542*.

Zhongkai Hao, Zhengyi Wang, Hang Su, Chengyang Ying, Yinpeng Dong, Songming Liu, Ze Cheng, Jian Song, and Jun Zhu. 2023. Gnot: A general neural operator transformer for operator learning. In *International Conference on Machine Learning*, pages 12556–12569. PMLR.

Bruce M. Hill. 1975. A Simple General Approach to Inference About the Tail of a Distribution. *The Annals of Statistics*, 3(5):1163 – 1174.

Liam Hodgkinson and Michael Mahoney. 2021. Multiplicative noise and heavy tails in stochastic optimization. In *International Conference on Machine Learning*, pages 4262–4274. PMLR.

Liam Hodgkinson, Umut Simsekli, Rajiv Khanna, and Michael Mahoney. 2022. Generalization bounds using lower tail exponents in stochastic optimizers. In *International Conference on Machine Learning*, pages 8774–8795. PMLR.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. 2021. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440.

Johannes Kiesel, Maria Mestre, Rishabh Shukla, Emmanuel Vincent, Payam Adineh, David Corney, Benno Stein, and Martin Potthast. 2019. Semeval-2019 task 4: Hyperpartisan news detection. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 829–839.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Vignesh Kothapalli, Tianyu Pang, Shenyang Deng, Zongmin Liu, and Yaoqing Yang. 2024. Crafting heavy-tails in weight matrix spectrum without gradient noise. *Preprint*, arXiv:2406.04657.

Jens Kringelum, Sonny Kim Kjaerulff, Søren Brunak, Ole Lund, Tudor I Oprea, and Olivier Taboureau. 2016. Chemprot-3.0: a global chemical biology diseases mapping. *Database*, 2016:bav123.

Francois Lanusse, Liam Parker, Siavash Golkar, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Geraud Krawezik, Michael McCabe, Ruben Ohana, Mariel Pettee, et al. 2023. Astroclip: Cross-modal pre-training for astronomical foundation models. *arXiv preprint arXiv:2310.03024*.

Pengxiang Li, Lu Yin, Xiaowei Gao, and Shiwei Liu. 2024. Owlore: Outlier-weighed layerwise sampled low-rank projection for memory-efficient llm finetuning. *arXiv preprint arXiv:2405.18380*.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. 2020. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Robert L Logan IV, Ivana Balažević, Eric Wallace, Fabio Petroni, Sameer Singh, and Sebastian Riedel. 2021. Cutting down on prompts and parameters: Simple few-shot learning with language models. *arXiv preprint arXiv:2106.13353*.

Haiquan Lu, Yefan Zhou, Shiwei Liu, Zhangyang Wang, Michael W. Mahoney, and Yaoqing Yang. 2024. Alphapruning: Using heavy-tailed self regularization theory for improved layer-wise pruning of large language models. *Advances in Neural Information Processing Systems*.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. 2021. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. *arXiv preprint arXiv:1808.09602*.

Charles H Martin and Michael W Mahoney. 2020. Heavy-tailed universality predicts trends in test accuracies for very large pre-trained deep neural networks. In *SIAM International Conference on Data Mining*.

Charles H Martin and Michael W Mahoney. 2021. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *Journal of Machine Learning Research*, 22(165):1–73.

Charles H. Martin and Michael W. Mahoney. 2022. Post-mortem on a deep learning contest: a simpson's paradox and the complementary roles of scale metrics versus shape metrics. *Preprint*, arXiv:2106.00734.

Charles H Martin, Tongsu Peng, and Michael W Mahoney. 2021. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications*, 12(1):4122.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Holden Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, et al. 2023. Multiple physics pretraining for physical surrogate models. *arXiv preprint arXiv:2310.02994*.

Josue Nassar, Piotr Sokol, SueYeon Chung, Kenneth D Harris, and Il Memming Park. 2020. On 1/n neural representation and robustness. *Advances in Neural Information Processing Systems*, 33:6211–6222.

Peijun Qing, Chongyang Gao, Yefan Zhou, Xingjian Diao, Yaoqing Yang, and Vosoughi Soroush. 2024. Alphaexpert: Assigning lora experts based on layer training quality. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel de Bézenac. 2024. Convolutional neural operators for robust and accurate learning of pdes. *Advances in Neural Information Processing Systems*, 36.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. *Preprint*, arXiv:1505.04597.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *Preprint*, arXiv:1804.04235.

Umut Simsekli, Ozan Sener, George Deligiannidis, and Murat A Erdogdu. 2020. Hausdorff dimension, heavy tails, and generalization in neural networks. *Advances in Neural Information Processing Systems*, 33:5138–5151.

Shashank Subramanian, Peter Harrington, Kurt Keutzer, Wahid Bhimji, Dmitriy Morozov, Michael W Mahoney, and Amir Gholami. 2024. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. *Advances in Neural Information Processing Systems*, 36.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. 2022. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Preprint*, arXiv:2307.09288.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2020. Superglue: A stickier benchmark for general-purpose language understanding systems. *Preprint*, arXiv:1905.00537.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. *Preprint*, arXiv:1804.07461.

Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. 2023. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60.

Yutong Wang, Rishi Sonthalia, and Wei Hu. 2024a. Near-interpolators: Rapid norm growth and the trade-off between interpolation and generalization. In *International Conference on Artificial Intelligence and Statistics*, pages 4483–4491. PMLR.

Zhichao Wang, Andrew Engel, Anand D Sarwate, Ioana Dumitriu, and Tony Chiang. 2024b. Spectral evolution and invariance in linear-width neural networks. *Advances in Neural Information Processing Systems*, 36.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Chaoyi Wu, Xiaoman Zhang, Ya Zhang, Yanfeng Wang, and Weidi Xie. 2023. Towards generalist foundation model for radiology. *arXiv preprint arXiv:2308.02463*.

Zeke Xie, Qian-Yuan Tang, Mingming Sun, and Ping Li. 2024. On the overlooked structure of stochastic gradients. *Advances in Neural Information Processing Systems*, 36.

Liu Yang and Stanley J Osher. 2024. Pde generalization of in-context operator networks: A study on 1d scalar nonlinear conservation laws. *arXiv preprint arXiv:2401.07364*.

Yaoqing Yang, Ryan Theisen, Liam Hodgkinson, Joseph E Gonzalez, Kannan Ramchandran, Charles H Martin, and Michael W Mahoney. 2023. Test accuracy vs. generalization gap: Model selection in nlp without accessing training or testing data. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3011–3021.

Zhanhong Ye, Xiang Huang, Leheng Chen, Hongsheng Liu, Zidong Wang, and Bin Dong. 2024.

Pdeformer: Towards a foundation model for one-dimensional partial differential equations. *arXiv preprint arXiv:2402.12652*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2021. Revisiting few-sample {bert} fine-tuning. In *International Conference on Learning Representations*.

Yiming Zhang, Shi Feng, and Chenhao Tan. 2022. Active example selection for in-context learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9134–9148, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. *Preprint*, arXiv:2305.11206.

Yefan Zhou, Tianyu Pang, Keqin Liu, Michael W Mahoney, Yaoqing Yang, et al. 2024. Temperature balancing, layer-wise weight analysis, and neural network training. *Advances in Neural Information Processing Systems*, 36.

Appendix

A Potential Risks

Our work leverages HT-SR theory as a model diagnosis tool to analyze the limitations of low-data training and fine-tuning, and help the design of an improved learning rate scheduling algorithm. We do not see any immediate negative societal impacts or ethics issues stemming from the algorithm itself. In addition, our analysis could inspire future research on diagnosing performance limitations in different scenarios, securing the safe use of LLMs.

B Ablation study on granularity of Learning Rate Scheduling: Per-block vs. Per-layer.

Following the discussion on scheduling method for Transformer-based models in Section 3.2, here we compare the performance of block-wise and layer-wise scheduling in RoBERTa-base model trained on QNLI dataset. Table 5 shows that the block-wise method generally outperforms the per-layer method in different subsampling ratios. The results suggest that block-wise learning rate scheduling is a more favorable method than layer-wise scheduling when we use `TempBalance` on Transformer-based models.

Ratio	5%	1%	0.5%	0.1%	0.05%
baseline	87.54±0.20	84.09±0.36	82.68± 0.43	73.57±0.90	71.31±1.29
Layer-wise	87.83±0.23	**84.81±0.07**	83.78±0.17	75.30±1.72	70.99±1.86
Block-wise	**88.24±0.08**	84.47±0.55	**84.30±0.46**	**75.78±0.47**	**72.83±1.65**

Table 5: Comparing layer-wise and block-wise learning rate schedule trained with RoBERTa-base model on QNLI task. We choose.

C Data Subsampling

To create low-data regimes, we design sets of subsampling ratios based on the size of different training datasets (see Table 6 and 7). For GLUE fine-tuning, we partition the datasets in GLUE into two groups: larger datasets (SST-2, MNLI, QNLI and QQP), and smaller datasets (CoLA, MRPC, STS-B and RTE). For larger datasets, we choose subsampling ratio from {0.02% 0.05%, 0.1%, 0.5%, 1%, 5%}, and for smaller datasets, we choose subsampling ratios from {10% 20%, 50%}. For PDE solving tasks, we use the datasets from PDEBench (Takamoto et al., 2022) and choose different data ratios considering the training difficulty in different datasets. For DarcyFlow dataset, the range of subsampling ratio is {0.6%, 2.5%, 5.0%, 10.0%, 100.0%}. For training the FNO and UNet on 1D and 2D CFD dataset, the range of subsampling ratio is {0.6%, 2.5%, 10.0%, 25.0%, 50.0%, 100.0%}.

Dataset	**SST-2**	**MNLI**	**QNLI**	**QQP**	**CoLA**	**MRPC**	**STS-B**	**RTE**
# of Data	67K	393K	105K	364K	8.5K	3.7K	7K	2.5K

Table 6: Number of training data samples of each GLUE tasks

Dataset	**DarcyFlow**	**1D CFD**	**2D CFD**
# of Data	9K	9K	9K
Parameter	$\beta = 100$	$\eta = \zeta = 0.01$, Rand periodic	$M = 0.1, \eta = \zeta = 0.01$, Rand periodic

Table 7: Number of training data samples and parameter of PDE Datasets.

D Hyperparameter Settings

In this section, we provide detailed hyperparameter settings to reproduce the experimental results.

D.1 Full Fine-tuning on GLUE and SuperGLUE Datasets

For full-finetuning, we choose to fine-tune RoBERTa-base model on GLUE and SuperGLUE datasets. For each subsampling ratio, we train using the Adam optimizer with a linear learning rate decay schedule for 10 epochs. We choose the sequence length of 128, and grid search learning rate and batch size to obtain the best results. When training on four smaller GLUE datasets (CoLA, MRPC, STSB, RTE) and SuperGLUE datasets, we search learning rate across {1e-5, 2e-5, 3e-5} and batch size across{16, 32}; when training on four larger GLUE datasets (SST2, MNLI, QNLI, QQP), the search range of learning rate and batch size are shown in Table 8 and 9 respectfully. For other hyperparameters and model configurations, we use the same settings following Liu et al. (Liu et al., 2019). We report the mean over 3 random seeds for each setting, where the results for each run are taken from the best epoch.

Dataset	**SST-2**	**MNLI**	**QNLI**	**QQP**
5%	{1e-5, 2e-5, 3e-5}			
1%	{1e-5, 2e-5, 3e-5}			
0.5%	{1e-5, 2e-5, 3e-5}			
0.1%	{1e-5, 2e-5, 3e-5}			
0.05%	{1e-5, 2e-5, 3e-5}			
0.02%	{1e-5, 2e-5, 3e-5, 5e-5}	{1e-5, 2e-5, 3e-5}		

Table 8: Learning rate range of training RoBERTa-base model on subsets of SST2, MNLI, QNLI and QQP datasets.

Dataset	SST-2	MNLI	QNLI	QQP
5%	{16, 32}			
1%	{16, 32}			
0.5%	{16, 32}			
0.1%	{4, 8, 16, 32}	{16, 32}		
0.05%	{4, 8, 16, 32}		{16, 32}	
0.02%	{4, 8, 16, 32}			

Table 9: Batch size range of training RoBERTa-base model on subsets of SST2, MNLI, QNLI and QQP datasets.

In addition to standard training configurations, we report the hyperparameters of `TempBalance` corresponding to the best results. Specifically, we report hyperparameters s. Note that during hyperparameter search, we find that assigning different s values to layers with `PL_Alpha_Hill` higher or lower than the mean `PL_Alpha_Hill` across all layers can achieve better results, and in the tables, we show them as a pair (s_1, s_2), often (2, 1).

Dataset	SST2	MNLI	QNLI	QQP
5%	(2, 1)	1.25	(2, 1)	1.25
1%	1.25	1.25	1	1
0.5%	1	1.25	1	1.25
0.1%	(2, 1)	1	1.25	1.25
0.05%	1.25	0.5	1.25	1
0.02%	1.25	1.25	0.25	1.25

Table 10: Best hyperparameter s for `TempBalance` of training RoBERTa-base model on subsets of SST2, MNLI, QNLI and QQP datasets.

Dataset	CoLA	MRPC	STSB	RTE
50%	1.25	1.25	0.75	1.25
20%	1	1.25	0.5	0.5
10%	1	1	1	1.25

Table 11: Best hyperparameter s for `TempBalance` of training RoBERTa-base model on subsets of CoLA, MRPC, STSB and RTE datasets.

Domain-specific Fine-tuning. For fine-tuning on domain-specific datasets, we train the RoBERTa-base models for 10 epochs with a batch size of 16 and an initial learning rate of 3e-5. We use the AdamW optimizer and apply linear learning rate decay with a 0.06 warmup ratio. The mean and standard deviation of test accuracy across 3 random seeds on the test set are reported.

D.2 LoRA Fine-tuning

For LoRA fine-tuning, we adopt the training configurations from previous works and perform a line search around the base learning rate. For training RoBERTa-base model on GLUE datasets, we follow Hu et al (Hu et al., 2021). and evaluate learning rate at 2e-4 and 6e-4 around the base learning rate (4e-4 or 5e-5). For LLaMA-7B on ScienceQA, we trained with AdamW optimizer for 50 epochs, and search the best learning rate in the range of {2e-4, 3e-4, 4e-4}. We set the cutoff length as 256 and batch size as 128. For LoRA adapters, we set the rank to 8, LoRA alpha to 16, and LoRA dropout to 0.05.

D.3 Neural PDE Solving

For SciML, we referred to PDEBench(Takamoto et al., 2022) for the hyperparameter settings and selected the appropriate learning rate, weight decay and batch size using a grid search method to make baseline models achieve good performances. For each subsampling ratio, we train the models with the Adam optimizer, scheduling the base learning rate by decaying the learning rate by $\gamma = 0.5$ every 100 epochs. We chose to train the models for enough epochs to ensure that the trained models were close to a convergent state. For the hyperparameter s in `TempBalance`, we choose from the range $\{0.125, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5\}$.

For training the FNO and UNet on DarcyFlow ($\beta = 100$), the search range of leanring rate and the selected weight decay are displayed in Table 12 and the batch size is 50.

Model	FNO		UNet	
Hyperparameters	Learning Rate	Weight Decay	Learning Rate	Weight Decay
100%	{5e-3, 1e-2, 1.5e-2}	1e-6	{2.5e-4, 5e-4, 1e-3}	1e-7
10.0%	{1.5e-2, 2.5e-2, 5e-2}	1e-4	{5e-3, 1e-2, 2.5e-2}	1e-4
5.0%	{1.5e-2, 2.5e-2, 5e-2}	1e-3	{5e-3, 1e-2, 2.5e-2}	1e-3
2.5%	{1.5e-2, 2.5e-2, 5e-2}	1e-3	{1.5e-2, 2.5e-2, 5e-2}	1e-3
0.6%	{1.5e-2, 2.5e-2, 5e-2}	1e-2	{2.5e-2, 5e-2, 1e-1}	1e-3

Table 12: Learning rate range and the selected weight decay of training FNO and UNet model on subsets of DarcyFlow($\beta = 100.0$) dataset.

When training the FNO on 1D and 2D CFD datasets, the search range of learning rate and the selected weight decay is shown in Table 13. The batch size for the subsampling ratio {100%, 50.0%, 25.0%, 10.0%} in training on 1DCFD is 25 and 10 for {2.5%, 0.6%}, while on the 2DCFD dataset the batch size is 20.

Dataset	1DCFD		2DCFD	
Hyperparameters	Learning Rate	Weight Decay	Learning Rate	Weight Decay
100%	{2.5e-3, 5e-3, 1e-2}	1e-2	{1e-3, 2.5e-3, 5e-3}	1e-4
50.0%	{2.5e-3, 5e-3, 1e-2}	1e-2	{1e-3, 2.5e-3, 5e-3}	1e-4
25.0%	{2.5e-3, 5e-3, 1e-2}	1e-2	{1e-3, 2.5e-3, 5e-3}	1e-4
10.0%	{2.5e-3, 5e-3, 1e-2}	1e-2	{1e-3, 2.5e-3, 5e-3}	1e-4
2.5%	{2.5e-3, 5e-3, 1e-2}	1e-1	{1e-3, 2.5e-3, 5e-3}	1e-4
0.6%	{1e-3, 2.5e-3, 5e-3}	1e-1	{2.5e-3, 5e-3, 1e-2}	1e-4

Table 13: Learning rate range and the selected weight decay of training FNO model on subsets of 1D and 2D CFD datasets.

Table 14 demonstrates the properly chosen weight decay and the learning rate range of training UNet on 1D and 2D CFD datasets. The batch size for the subsampling ratio {100%, 50.0%, 25.0%} in training on 1DCFD is 100, for {10.0%, 2.5%} is 50, and for{0.6%} is 25, while on the 2DCFD dataset the batch size is 20.

Dataset	1DCFD		2DCFD	
Hyperparameters	Learning Rate	Weight Decay	Learning Rate	Weight Decay
100%	{5e-3, 1e-2, 2.5e-2}	1e-5	{1e-2, 2.5e-2, 5e-2}	1e-3
50.0%	{5e-3, 1e-2, 2.5e-2}	1e-1	{2.5e-3, 5e-3, 1e-2}	1e-1
25.0%	{5e-3, 1e-2, 2.5e-2}	1e-1	{2.5e-3, 5e-3, 1e-2}	1e-1
10.0%	{5e-3, 1e-2, 2.5e-2}	1e-1	{2.5e-3, 5e-3, 1e-2}	1e-1
2.5%	{2.5e-2, 5e-2, 1e-1}	1e-1	{5e-3, 1e-2, 2.5e-2}	1e-1
0.6%	{2.5e-2, 5e-2, 1e-1}	1e-1	{2.5e-2, 5e-2, 1e-1}	1e-1

Table 14: Learning rate range and the selected weight decay of training UNet model on subsets of 1D and 2D CFD datasets.

E Complementary Results

In this section, we first provide detailed results discussed in Section 4.3 in the paper, then further evaluate `TempBalance` on NLP and SciML training tasks. Also in Section E.2, we provide statistical testing results to demonstrate the significance of improvement brought by `TempBalance`. First, in E.1 and E.4 we show detailed results of GLUE full fine-tuning and two time-dependent PDEs discussed in Section 4.3. Second, we present complementary results of `TempBalance` on fine-tuning RoBERTa-base model on SuperGLUE and SQuAD datasets in E.3. Then, we apply `TempBalance` to LoRA fine-tuning, and show the results of LoRA fine-tuning of RoBERTa-base model on GLUE tasks in E.5, and LLaMA-7B model on ScienceQA in E.6. Afterwards, we evaluate `TempBalance` on solving DarcyFlow PDEs with FNO and UNet model in E.7.

E.1 Detailed Fine-tuning Results on GLUE Datasets

In Table 17 and 18, we show the full results of fine-tuning RoBERTa-base model on GLUE datasets, corresponding to Figure 3 and the discussions in Section 4.3.

E.2 Statistical Testing on the Significance of Improvement

We perform statistical testing to verify the effectiveness of our algorithm compared to baseline methods. We define the Null Hypothesis (H0) as "There is no significant difference in performance between our algorithm and the baseline", and the Alternative Hypothesis (H1) as "Our algorithm performs significantly better than the baseline". We run experiments of training RoBERTa-base on SST-2 with different subsampling ratios for 10 random seeds and perform t-tests on the results. We present the results in the table below:

Ratio	0.02%	0.1%	0.5%	1%	5%
P-value	$3.85e^{-9}$	0.13	0.003	0.003	$4.06e^{-5}$

Table 15: Statistical testing results on RoBERTa-base model trained with different subsampling ratios of the SST-2 dataset.

Subsampling Ratio	1%	5%	10%	20%	50%
FT	45.84$_{\pm 2.26}$	79.49$_{\pm 0.22}$	86.88$_{\pm 0.12}$	88.56$_{\pm 0.14}$	90.97$_{\pm 0.15}$
TB	**48.91**$_{\pm 1.27}$	**81.18**$_{\pm 0.07}$	**88.08**$_{\pm 0.05}$	**89.49**$_{\pm 0.20}$	**91.16**$_{\pm 0.03}$

Table 16: Test accuracy (%) on SQuAD v1.1 dataset of ROBERTa-base model trained with different subsampled training sets.

E.3 Full Fine-tuning on SuperGLUE and SQuAD

SuperGLUE. In Table 20, we present the results of applying `TempBalance` on training RoBERTa-base model on SuperGLUE tasks. The tasks and their corresponding evaluation metrics are: BoolQ (Accuracy), RTE (Accuracy), CB (Accuracy and F1), WiC (Accuracy), MultiRC (F1 and Exact Match (EM)), COPA (Accuracy). We can see that `TempBalance` effectively increases test performance in most cases, and archives significant overall improvement. Specifically, `TempBalance` achieves 7.14% performance gain when training on 50% CB dataset. `TempBalance` can also improve the overall mean performance by 1.65% when trained with 50% data.

SQuAD. In Table 16, we present the results of applying `TempBalance` on training RoBERTa-base model on SQuAD (v1.1) dataset across five subsampling ratios: 1%, 5%, 10%, 20%, 50%. We train the model for 10 epochs with learning rate 2e-5

Subsampling Ratio	Method	SST-2	MNLI	QNLI	QQP	Avg.
0.02%	FT	58.49±10.96	45.28±0.62	53.69±0.44	69.04±0.19	56.63
	TB	**68.39**±3.21	**45.32**±1.31	**58.11**±6.29	**69.72**±0.70	**60.39(↑3.76)**
0.05%	FT	83.07±0.66	57.87±1.14	71.31±1.29	71.55±1.25	70.95
	TB	**84.17**±0.25	**59.42**±1.90	**72.83**±1.65	**73.35**±1.43	**72.44(↑1.49)**
0.1%	FT	84.13±1.97	64.99±2.39	73.57±0.90	74.05±0.94	74.19
	TB	**87.16**±0.81	**66.57**±2.51	**75.78**±0.47	**74.20**±0.61	**75.93(↑1.74)**
0.5%	FT	90.44±0.46	76.88±0.33	82.68±0.43	79.61±0.24	82.40
	TB	**91.44**±0.42	**77.73**±0.47	**84.30**±0.46	**80.00**±0.21	**83.37(↑0.97)**
1%	FT	91.06±0.16	79.45±0.22	84.09±0.36	80.93±0.31	83.88
	TB	**91.97**±0.48	**80.10**±0.25	**84.47**±0.55	**81.18**±0.22	**84.43(↑0.55)**
5%	FT	92.85±0.24	83.10±0.02	87.94±0.08	83.98±0.04	86.97
	TB	**93.69**±0.16	**83.36**±0.15	**88.24**±0.08	**84.00**±0.15	**87.32(↑0.35)**

Table 17: Evaluation results of RoBERTa-base model trained on larger GLUE tasks. We compare `TempBalance` (TB) with Full Fine-tuning (FT) trained with Adam optimizer and linear learning rate decay. The tasks and their corresponding evaluation metrics are: SST-2 (accuracy, ↑), MNLI (accuracy, ↑), QNLI (accuracy, ↑) and QQP (combined score of F1 score and accuracy, ↑)

Subsampling Ratio	Method	CoLA	MRPC	STSB	RTE	Avg.
10%	FT	49.01±1.63	81.29±1.61	84.36±1.03	59.69±0.45	68.59
	TB	**50.34**±0.91	**81.70**±1.61	**86.04**±0.80	**60.53**±1.78	**69.65(↑1.06)**
20%	FT	49.50±2.08	84.64±0.50	87.45±0.25	66.07±0.88	71.92
	TB	**51.28**±0.73	**85.86**±0.61	**88.39**±0.55	**67.27**±0.34	**73.13(↑1.21)**
50%	FT	56.78±1.96	87.66±0.42	90.12±0.20	71.48±1.35	76.51
	TB	**58.60**±0.74	**88.40**±0.42	**90.24**±0.06	**74.85**±1.78	**78.02(↑1.51)**

Table 18: Evaluation results of RoBERTa-base trained on smaller GLUE tasks using full fine-tuning. We compare `TempBalance` with baseline FT (Full Fine-tuning) on: CoLA (Matthews Correlation, ↑), MRPC (combined score of F1 score and accuracy, ↑), STS-B (combined score of Pearson and Spearman Rank, ↑), and RTE (Accuracy, ↑)

and a batch size of 24 using the AdamW optimizer with a warmup rate of 0.06 and linear learning rate decay. We follow the detailed hyperparameter settings from (Liu et al., 2019). The mean and standard deviation of test accuracy across 3 random seeds on the test set are reported. We observe that `TempBalance` continues to achieve better test performance than baseline FT, and significantly outperforms baseline FT in low-data regimes: when trained on 1% data of SQuAD, `TempBalance` increases the test accuracy by 3.07%.

E.4 Detailed Results on 1D and 2D CFD Datasets

In Table 19, we present the detailed results of training FNO and UNet model on 1D and 2D CFD datasets, corresponding to Figure 4 and the discussions in Section 4.3.

E.5 LoRA Fine-tuning on GLUE

Measuring the ESD of LoRA Adapters. Some models are too large to fine-tune fully, so one often needs to use LoRA. In this case, LoRA adapters are added to selected layers in the model, and only these adapters are trained during fine-tuning, while the original weight matrix remains fixed. For a layer with weight matrix $\mathbf{W} \in \mathbb{R}^{d\times k}$ and LoRA adapters $\mathbf{B} \in \mathbb{R}^{d\times r}$ and $\mathbf{A} \in \mathbb{R}^{r\times k}$, we cannot simply calculate ESD of the product between adapters $\mathbf{B} \times \mathbf{A}$, since the rank of the adapters $r \leq \min(d, k)$ are low-rank matrices, which would result in a poor ESD landscape. Therefore, for layers with LoRA adapters, we calculate the sum of the product of LoRA adapters and the weight matrix $\mathbf{W}' = \mathbf{W} + \mathbf{B} \times \mathbf{A}$, and then calculate the ESD of its correlation matrix $\mathbf{X} = \mathbf{W}'^{\top}\mathbf{W}'$.

We present the results of applying `TempBalance` on LoRA Adapters in Table 21. We can see that `TempBalance` consistently

Subsampling Ratio	Model	FNO		UNet	
	Dataset	1DCFD	2DCFD	1DCFD	2DCFD
100%	Baseline	5.02e-02±4.43e-03	1.23e-01±7.44e-03	2.08e-01±1.71e-02	2.96e-01±7.05e-03
	TB	**4.74e-02**±6.57e-04	**1.16e-01**±4.29e-03	**1.91e-01**±1.59e-02	**2.90e-01**±1.94e-03
	Error Reduced	5.58%	5.69%	8.17%	2.03%
50.0%	Baseline	6.04e-02±3.17e-03	1.40e-01±4.68e-03	2.25e-01±2.24e-03	2.87e-01±6.49e-03
	TB	**5.68e-02**±2.28e-03	**1.37e-01**±3.53e-03	**2.23e-01**±1.24e-03	**2.85e-01**±5.64e-04
	Error Reduced	5.96%	2.14%	0.89%	0.70%
25.0%	Baseline	7.81e-02±3.79e-03	2.11e-01±3.27e-03	2.28e-01±1.79e-03	3.06e-01±1.77e-03
	TB	**7.42e-02**±1.87e-03	**2.03e-01**±5.54e-03	**2.26e-01**±1.52e-03	**3.01e-01**±1.63e-03
	Error Reduced	4.99%	3.79%	0.88%	1.97%
10.0%	Baseline	1.13e-01±4.79e-03	2.35e-01±1.61e-03	2.40e-01±2.42e-03	3.09e-01±1.92e-03
	TB	**1.02e-01**±1.88e-03	**2.29e-01**±1.41e-03	**2.38e-01**±2.00e-04	**3.06e-01**±2.96e-03
	Error Reduced	9.73%	2.55%	0.83%	0.97%
2.5%	Baseline	2.11e-01±2.79e-03	3.22e-01±5.37e-03	2.74e-01±2.88e-02	3.89e-01±3.77e-02
	TB	**2.08e-01**±5.25e-03	**3.06e-01**±1.15e-02	**2.54e-01**±4.61e-03	**3.80e-01**±1.76e-02
	Error Reduced	1.42%	4.97%	7.30%	2.31%
0.6%	Baseline	2.48e-01±3.35e-03	5.46e-01±2.20e-02	3.46e-01±4.15e-03	3.88e-01±2.15e-02
	TB	**2.38e-01**±2.84e-03	**4.67e-01**±2.85e-02	**3.29e-01**±1.87e-02	**3.78e-01**±2.78e-02
	Error Reduced	4.03%	14.47%	4.91%	2.58%

Table 19: Evaluation results of FNO and UNet model trained on 1D and 2D CFD datasets. We compare our method (TB) with the baseline. The evaluation metric is nRMSE (↓).

Subsampling Ratio	Method	BoolQ	RTE	CB	WiC	MultiRC	COPA	Avg.
10%	FT	64.97±2.58	62.57±1.68	68.45±2.23	62.80±3.00	32.95±0.33	54.67±0.47	57.73
	TB	**65.95**±2.17	**62.69**±1.19	**69.64**±1.46	**63.43**±1.90	**33.22**±0.47	**58.33**±2.62	**58.88(↑1.15)**
20%	FT	69.93±2.01	67.87±1.64	72.61±0.84	**67.14**±0.98	34.92±0.88	57.00±2.16	61.58
	TB	**71.80**±1.92	**70.04**±1.35	72.61±0.84	66.67±1.74	**35.00**±0.16	**59.33**±6.13	**62.58(↑1.00)**
50%	FT	76.73±0.49	74.84±0.90	77.38±2.23	68.44±2.50	35.77±0.92	58.67±1.25	65.29
	TB	**76.85**±0.13	74.84±1.62	**84.52**±0.03	**70.32**±1.10	**36.44**±0.59	58.67±2.87	**66.94(↑1.65)**

Table 20: Evaluation results of RoBERTa-base model trained on SuperGLUE tasks using full fine-tuning.

achieves higher test results than LoRA alone. We note that our method can at most improve the test accuracy of 3.29% on 0.02% SST2 dataset, indicating a significant improvement. From average improvement increases across different tasks, we can see that as we reduce the subsampling ratio, the average improvement of `TempBalance` on all tasks continues to increase. This observation aligns with the discussion in Section 4.6, that `TempBalance` achieves gradually increased gains in fine-tuning performance as the number of tuning data points decreases, further proving the effectiveness of `TempBalance` in achieving model alignment in low-data regimes.

E.6 Question Answering

To draw more robust conclusions, we evaluate the empirical performance of `TempBalance` on LLM fine-tuning. We choose to fine-tune LLaMA-7B model with LoRA adapters on the ScienceQA dataset (Lu et al., 2022). In Table 22 we report the test accuracy of LoRA and `TempBalance` under different subsampling ratios on ScienceQA dataset. We can see that `TempBalance` continues to yield better test performance on low-data regimes.

E.7 Training FNO and UNet Model on DarcyFlow Dataset

In Table 23 we show the test results of training the FNO and UNet model on the DarcyFLow dataset with a subsampling ratio ranging from 0.6% to 100%, evaluated by Normalized Root Mean Squared Error (nRMSE). We show that `TempBalance` achieves lower nRMSE compared to the baseline on all subsampling ratios. Specifically, `TempBalance` reduces the nRMSE of the UNet model trained on 2.5% of the DarcyFlow dataset by a significant 10.89%, and improve the nRMSE of FNO on 0.6% by 9.71%.

F Compute Resources

We conduct our experiments on Quadro RTX 6000, NVIDIA L40(40GB), and NVIDIA RTX A6000 GPU clusters. Specifically, we run every full fine-tuning of RoBERTa-base on GLUE and SuperGLUE datasets using one Quadro RTX 6000 GPU

Subsampling Ratio	Method	SST-2	MNLI	QNLI	QQP	Avg.
0.02%	LoRA	66.82±0.81	37.93±0.89	51.58±0.29	61.18±2.72	54.38
	LoRA+TB	**70.11**±0.84	**39.39**±1.84	**51.93**±0.41	**63.77**±0.99	**56.3(↑1.92)**
0.05%	LoRA	**82.03**±1.33	54.74±0.57	54.91±0.41	67.80±0.62	64.87
	LoRA+TB	81.77±1.97	**55.19**±0.97	**59.93**±1.07	**68.75**±0.30	**66.41(↑1.54)**
0.1%	LoRA	87.42±1.08	66.43±0.41	69.05±4.27	70.83±0.97	73.43
	LoRA+TB	**88.34**±0.52	**66.79**±0.73	**69.72**±3.36	**71.21**±0.94	**74.02(↑0.59)**
0.5%	LoRA	90.82±0.09	76.77±0.31	81.79±0.82	**78.69**±0.54	82.02
	LoRA+TB	**91.09**±0.54	**77.09**±0.46	**82.02**±0.41	78.45±0.25	**82.16(↑0.14)**
1%	LoRA	92.69±0.14	79.26±0.29	84.29±0.13	80.34±0.13	84.14
	LoRA+TB	**93.04**±0.10	**79.43**±0.07	**84.34**±0.44	**80.51**±0.16	**84.33(↑0.19)**

Table 21: Evaluation results of RoBERTa-base model trained on four larger GLUE tasks. We compare our method (TB) with Low-Rank Adaptation training (LoRA) fine-tuning. The tasks and their corresponding evaluation metrics are: SST-2 (accuracy), MNLI (accuracy), QNLI (accuracy) and QQP (combined score of F1 score and accuracy)

Subsampling Ratio	1%	5%	10%
LoRA	51.12±0.87	65.24±1.04	73.40±0.39
LoRA+TB	**53.09**±1.64	**65.96**±1.21	**73.70**±0.80

Table 22: Test accuracy (%) on ScienceQA dataset of LLaMA-7B model trained with different subsampled training set.

per job. For each of the LoRA fine-tuning of RoBERTa-base on GLUE tasks, we utilize a single NVIDIA RTX A6000 GPU to train the model. For LLaMA-7B LoRA fine-tuning experiments, we use 4 NVIDIA RTX A6000 GPUs for one job. For all Neural PDE experiments, we use a single NVIDIA L40(40GB) GPU for each job.

G More Ablation Study Results

G.1 Different ESD metrics and scheduling functions in using `TempBalance` in SciML.

We compare the performance of using different ESD measuring metrics and scheduling functions of `TempBalance` on SciML tasks. Table 24 reports the results of different `TempBalance` settings in training the FNO model on solving the 1DCFD task. We can see that `TempBalance` outperforms the baseline method at every subsampling ratio, and our proposed scaling function `TB_Sigmoid` achieves more stable performance than `TB_Linear_Map`. At most subsampling ratios, using `PL_Alpha_Hill` we can achieve results that are comparable to or even better than those obtained with other metrics.

Subsampling Ratio	Method	FNO	UNet
100%	Baseline	2.58e-03±2.69e-05	5.27e-03±3.27e-05
	TB	**2.52e-03**±5.68e-05	**5.07e-03**±1.41e-05
	Error Reduced	2.33%	3.80%
10.0%	Baseline	1.04e-02±4.11e-04	1.43e-02±1.21e-03
	TB	**1.01e-02**±1.30e-04	**1.34e-02**±9.50e-04
	Error Reduced	2.88%	6.29%
5.0%	Baseline	1.76e-02±5.17e-04	1.98e-02±1.79e-03
	TB	**1.62e-02**±2.19e-04	**1.81e-02**±1.35e-03
	Error Reduced	7.95%	8.59%
2.5%	Baseline	2.88e-02±9.79e-04	2.57e-02±9.89e-04
	TB	**2.64e-02**±5.72e-04	**2.29e-02**±1.94e-03
	Error Reduced	8.33%	10.89%
0.6%	Baseline	6.28e-02±1.78e-03	4.59e-02±3.10e-03
	TB	**5.67e-02**±1.62e-03	**4.45e-02**±1.48e-03
	Error Reduced	9.71%	3.05%

Table 23: Evaluation results of FNO and UNet model trained on DarcyFlow ($\beta = 100$) dataset. We compare our method (TB) with the baseline. The evaluation metric is nRMSE (↓).

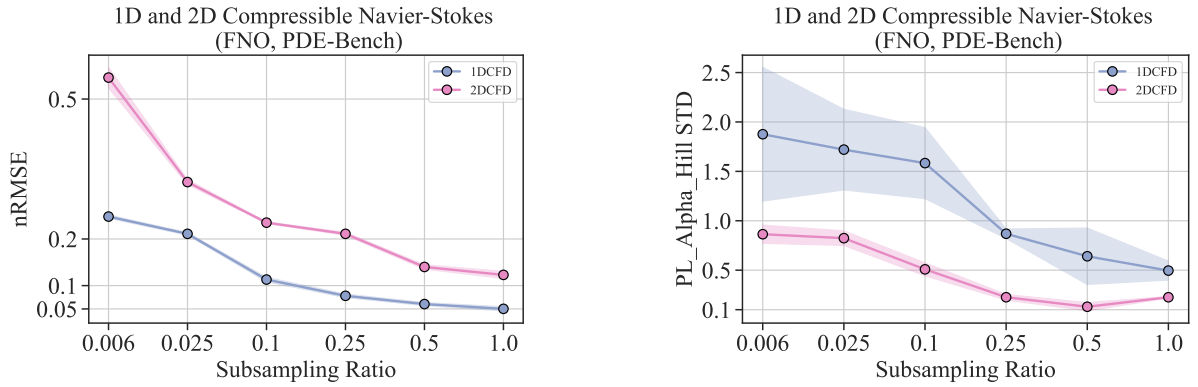
H More Analysis Results

H.1 Diagnosing the Data Limitation Using HT Metrics

Following Section 4.2, here we further analyzed FNO model's test performance using Alpha-related metrics as the training data size decreases. Figure 6 demonstrates that the change of the STD of `PL_Alpha_Hill` corresponds very closely with the variations in the model's performance. We observe that as the subsampling ratio decreases, the nRMSE on the 1D and 2D CFD PDEs solving increases, indicating a deterioration in model's performance. Simultaneously, the STD of `PL_Alpha_Hill` becomes larger, suggesting that the training across the model layers is becoming increasingly uneven. Therefore, the STD of `PL_Alpha_Hill` effectively captures

Ratio	100%	50.0%	25.0%	10.0%	2.5%	0.6%
Baseline	5.02e-02±4.43e-03	6.04e-02±3.17e-03	7.81e-02±3.79e-03	1.13e-01±4.79e-03	2.11e-01±2.79e-03	2.48e-01±3.35e-03
TB_Linear_Map	4.95e-02±3.49e-03	5.70e-02±5.52e-04	**7.26e-02±1.02e-03**	1.02e-01±3.00e-03	2.05e-01±4.77e-03	2.40e-01±7.47e-03
TB_Sigmoid(PL_Alpha_Hill)	**4.74e-02±6.57e-04**	**5.68e-02±2.28e-03**	7.42e-02±1.87e-03	**1.02e-01±1.88e-03**	2.08e-01±5.25e-03	2.38e-01±2.84e-03
TB_Sigmoid(Stable_Rank)	4.89e-02±2.03e-03	6.03e-02±7.47e-04	7.32e-02±1.73e-03	1.06e-01±4.85e-03	2.07e-01±1.36e-03	2.45e-01±6.11e-03
TB_Sigmoid(Spectral_Norm)	4.84e-02±2.86e-03	5.77e-02±1.48e-03	7.50e-02±5.70e-03	1.03e-01±4.66e-03	**1.91e-01±1.05e-02**	**2.34e-01±1.12e-03**

Table 24: Comparing different Temperature Balancing scheduling algorithm and ESD metrics on FNO model trained with 1DCFD dataset. The TempBalance series functions can help models achieve lower test nRMSE among all subsampling ratios, and the TB_Sigmoid outperform the original TB_Linear_Map function.



a nRMSE (↓)

b STD of layer-wise PL_Alpha_Hill

Figure 6: Predicting model performance under different training data using the variance of layer-wise PL_Alpha_Hill. 6a shows the trend of test performance of FNO model on 1D and 2D CFD datasets. 6b shows the trend of standard deviation of PL_Alpha_Hill across different FNO layers in different training data.

the model's performance variations in response to changes in the amount of training data, which aligns closely with the results obtained in our previous experiments in Figure 7.

H.2 More Analysis Study Results in the STD of PL_Alpha_Hill

In Figure 7 and 8, we compare the STD of the PL_Alpha_Hill between the baseline and TempBalance on fine-tuned LLM and trained FNO models at different subsampling ratios. When the subsampling ratio is relatively large, the STD of PL_Alpha_Hill of models is smaller, and the impact of the TempBalance method on this metric is also minimal. However, when the subsampling ratio is relatively small, the opposite is true: the TempBalance method makes the distribution of PL_Alpha_Hill across each layer of the model more uniform.

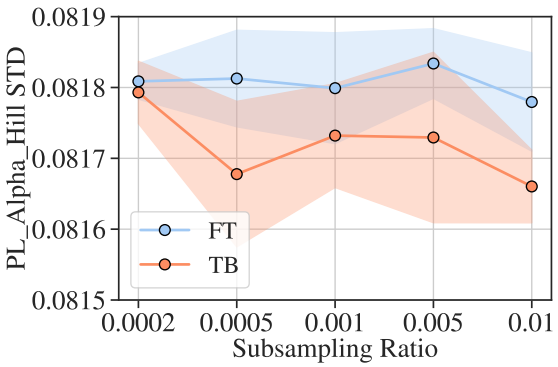
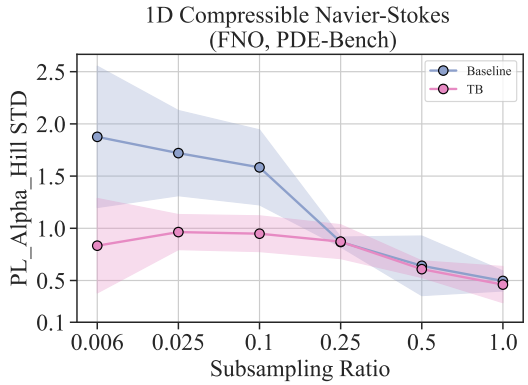
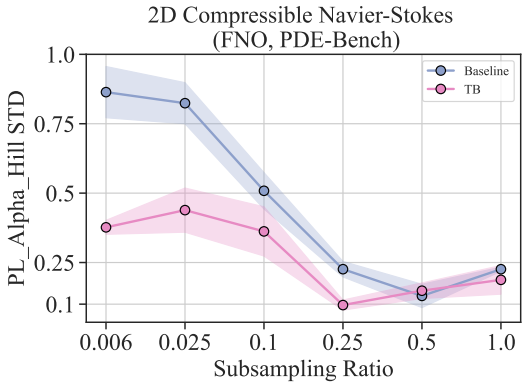


Figure 7: Analyzing the distribution of PL_Alpha_Hill of baseline FT and TempBalance on RoBERTa-base model trained on QNLI across different subsampling ratios. We observe that TempBalance continues to show lower STD of PL_Alpha_Hill, suggesting a more evenly distributed PL_Alpha_Hill.



a STD of layer-wise `PL_Alpha_Hill` in training FNO on 1DCFD



b STD of layer-wise `PL_Alpha_Hill` in training FNO on 2DCFD

Figure 8: Comparing the STD of layer-wise `PL_Alpha_Hill` measured in using baseline method and `TempBalance` training FNO model on 1D and 2D CFD datasets. The results demonstrate that `TempBalance` can reduce the STD, and this effect is more significant when the subsampling ratio is smaller, indicating that our approach helps ensure more uniform training across each layer of the model.