MD tree: a model-diagnostic tree grown on loss landscape

Yefan Zhou [*1] **Jianlong Chen** [*2] **Qinxue Cao** [3] **Konstantin Schürholt** [4] **Yaoqing Yang** [1]

Abstract

This paper considers "model diagnosis", which we formulate as a classification problem. Given a pre-trained neural network (NN), the goal is to predict the source of failure from a set of failure modes (such as a wrong hyperparameter, inadequate model size, and insufficient data) without knowing the training configuration of the pre-trained NN. The conventional diagnosis approach uses training and validation errors to determine whether the model is underfitting or overfitting. However, we show that rich information about NN performance is encoded in the optimization loss landscape, which provides more actionable insights than validation-based measurements. Therefore, we propose a diagnosis method called `MD tree` based on loss landscape metrics and experimentally demonstrate its advantage over classical validation-based approaches. We verify the effectiveness of `MD tree` in multiple practical scenarios: (1) use several models trained on one dataset to diagnose a model trained on another dataset, essentially a few-shot dataset transfer problem; (2) use small models (or models trained with small data) to diagnose big models (or models trained with big data), essentially a scale transfer problem. In a dataset transfer task, `MD tree` achieves an accuracy of 87.7%, outperforming validation-based approaches by 14.88%. Our code is available at https://github.com/YefanZhou/ModelDiagnosis.

1. Introduction

There is a notable gap in the literature in systematically diagnosing the reason for the underperformance of a pre-trained neural network (NN). For example, it is often difficult to access proprietary datasets, detailed configuration parameters, and complete training methodologies to develop these models. When a trained model underperforms, the conventional approach to troubleshooting through retraining with alternative settings thus becomes impractical. Therefore, we should develop diagnostic methods that do not involve complete datasets, retraining processes, and explicit configuration details. Can we say anything about why the model is underperforming in this setting? Answering this question requires a comprehensive analysis of the individual-trained model to uncover the root causes of failure and identify targeted strategies for improvement. The critical question in this context is the following:

How can we methodically diagnose the root causes of the underperformance of a model without the need for detailed configuration specifics, retraining processes, and extensive reliance on the dataset?

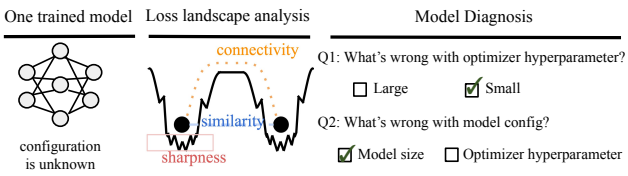


Figure 1. **(Overview of our model-diagnostic framework using `MD tree`).** This framework is designed to analyze and diagnose NNs where the training configuration is unknown. By examining the loss landscape structure of a given trained model, `MD tree` can identify potential failure sources of suboptimal performance.

Canonical Problems in Model Diagnosis. In this paper, we define the problem of "model diagnosis" and present a systematic framework for evaluating and approaching it. The goal is to identify the causes of underperformance in trained models without relying on configuration specifics, retraining processes, or extensive use of validation datasets. See Figure 1 for an overview of the framework. We categorize the sources that lead to model failure into four: (1) small optimizer hyperparameter (e.g., small batch size); (2) large optimizer hyperparameter; (3) inadequate model size; and (4) insufficient data amount. The four sources of failure are inspired by the *statistical physics* viewpoint of learning (Seung et al., 1992; Watkin et al., 1993; Haussler et al., 1994; Engel & den Broeck, 2001; Martin & Mahoney, 2017; Yang et al., 2021; Zhou et al., 2023). According to prior works in this area (Martin & Mahoney, 2017; Yang et al.,

[*]Equal contribution [1]Department of Computer Science, Dartmouth College, NH, USA [2]Zhejiang University, Zhejiang, China [3]University of Illinois Urbana-Champaign, IL, USA [4]University of St. Gallen, SUI. Correspondence to: Yefan Zhou <yefan.zhou.gr@dartmouth.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

2021), the most important optimizer hyperparameters are the *temperature-like parameters* that characterize the magnitude of noise introduced in stochastic training. These can be as common as the learning rate and batch size. Similarly, these works use *load-like parameters* to refer to the size of the data relative to the size of the model. They use temperature-like and load-like parameters together to model the peculiar generalization properties of deep learning, an approach known as the *very simple deep-learning model* (Martin & Mahoney, 2017; Yang et al., 2021). Therefore, our framework considers temperature-like hyperparameters (mainly batch size in this paper) and load-like parameters (data amount and model size) [1]. Controlling these parameters is also considered important in recent large-scale studies, such as the neural scaling law (Kaplan et al., 2020).

More concretely, we decompose the four sources into two canonical binary classification problems. That is, given a trained model, without knowing the training configurations, `MD tree` aims to answer the following questions:

- Q1 Can we determine whether the optimizer hyperparameter (e.g. batch size) used to train the model is large or small, compared to the optimal hyperparameter?
- Q2 Can we determine the more severe failure source between inadequate model size and wrong optimizer hyperparameter? Which one should be addressed first?

In Section 2, we discuss why our approach to defining and addressing model diagnosis is orthogonal or complementary to previous work.

Good Features for Model Diagnosis. The key to answering these questions is to extract informative features from trained models associated with different failure sources. Conventional metrics, as discussed in Raschka (2018), typically diagnose a trained model by analyzing training and validation error (loss), which can be used to (roughly) determine whether the model is overfitting or underfitting. However, if we want a fine-grained model diagnosis, such as answering questions like Q1 and Q2, the validation-based metrics have several limitations. First, we show that validation-based metrics provide limited information on the root causes of underperformance. For example, while large training errors may indicate underfitting, they do not distinguish between insufficient model size and suboptimal optimizer hyperparameters. Second, a detailed visualization of the model diagnosis shows that validation-based metrics exhibit complex relationships with failure sources, resulting in complicated nonlinear classification boundaries in the binary classification problems Q1 and Q2. These boundaries not only lead to inaccurate diagnoses but also transfer poorly across different groups of models.

Again, inspired by the statistical physics viewpoint of learning, this research uses metrics from the NN loss landscape to extract useful features for model diagnosis. Prior works (Yang et al., 2021; Zhou et al., 2023) identify several useful loss landscape metrics, including sharpness (Yao et al., 2020), connectivity (Garipov et al., 2018; Nguyen et al., 2021) and similarity (Kornblith et al., 2019), in determining the distinct *regimes* to characterize the test accuracy of NNs, where each regime represents a range of configurations where the loss landscape properties remain homogeneous. Our work, however, takes one step further and uses these features for model diagnosis, aiming to answer fine-grained questions such as Q1 and Q2, as shown in Figure 1.

A Simple Model for Model Diagnosis. Our framework employs an interpretable tree-based method called `MD tree` to address Q1 and Q2. This method uses a divide-and-conquer way to partition the NN configuration space into distinct regimes using loss landscape metrics. Each identified regime is then associated with a particular source of failure, allowing for interpretable diagnostic predictions. Unlike traditional decision trees (DTs) that rely on information gain for branching, `MD tree` adopts a hierarchy inspired by the statistical physics viewpoint. That is, we prioritize the metrics shown in previous work to capture the sharp transitions of the NN regimes by placing these metrics at shallower levels of the tree.

We evaluate `MD tree` in few-shot learning scenarios. Few-shot means that a few trained NNs are used as a "training set" to develop a function that maps loss landscape metrics to diagnostic predictions. We evaluate the diagnostic ability of this classifier under two transfer learning scenarios: *dataset transfer*, where the NNs to be diagnosed are models trained on different datasets, possibly with label noise, and *scale transfer*, where the training involves diagnosing small models, while the testing involves diagnosing large ones.

Main Findings. Our empirical analysis uses datasets of pre-trained models with 1690 different configurations, including different model sizes, data amounts, and optimization hyperparameters. Our analysis shows that `MD tree`, which uses loss landscape metrics, can effectively diagnose the sources of model failures and significantly outperform the validation-based method. Our key observations to support this main claim include:

- **O1** `MD tree` uses loss landscape metrics to distinguish the "regime" of the models: the models within the same regime are homogeneous and share the same failure source (e.g., the failure is caused by a small optimizer hyperparameter value). Also, most of the boundaries between the regimes are sharp, so the models with different failure sources are linearly separable by applying simple thresholding to the loss landscape metrics.
- **O2** The boundaries determined by `MD tree` have high

[1] Note that large data/model size can also hurt (Nakkiran et al., 2021a) due to double descent (Belkin et al., 2019; Bartlett et al., 2020; Hastie et al., 2022; Mei & Montanari, 2022), but we do not consider large data/model size as primary failure sources here, as appropriate regularization can often mitigate double descent (Nakkiran et al., 2021b).

transferability across different groups of models (e.g., transferring from models trained on clean data to those with label noise). For example, `MD tree` trained with small-scale models (e.g., models with 0.01M parameters) can be used to diagnose failures in models with up to 44.66M parameters and can achieve an accuracy of 82.56%. Therefore, `MD tree`'s decision thresholds determined by a few known sample models can be effectively transferred to diagnose other unseen models.

O3 The tree hierarchy of `MD tree`, inspired by recent studies from Yang et al. (2021), is beneficial to our model diagnosis problem. The effectiveness of the hierarchy is demonstrated by `MD tree`'s better performance compared to standard DTs using the same set of loss landscape metrics. Specifically, in a few-shot setting with 12 training samples, `MD tree` outperforms standard DT by 12.71%.

In summary, we introduce `MD tree`, a useful model diagnostic tool that uses previous research on loss landscape metrics to provide actionable insights on model failures. Our key contributions are as follows:

- We introduce a model diagnosis method called `MD tree`, which can identify the specific failure source within ML pipelines (optimizer, model size, data amount) that affects a model. In cases where multiple failure sources are at play, `MD tree` can determine which failure source is most detrimental to performance.
- We show that `MD tree` can be applied to a range of practical transfer learning cases, including (i) diagnosing models trained with noisy data given models trained with clean data; (ii) diagnosing models trained with large-scale parameter/data given trials on small-scale models; (iii) diagnosing models evaluated on out-of-distribution (OOD) test data (Hendrycks & Dietterich, 2019) or trained on class-imbalanced training data (Cui et al., 2019) given normal models; (iv) diagnosing Transformer models given ResNet models.
- Moreover, `MD tree` outperforms validation-based methods in diagnostic accuracy by a significant margin of 14.88%. In Appendix A, we apply `MD tree` to a novel task of determining a "one-step configuration change" to improve test performance. We show that `MD tree` can lead to superior CIFAR-10 test accuracy improvement compared to validation-based methods.

2. Related Work

2.1. Model Diagnosis

Model diagnostics are widely studied for linear regression models (Weisberg, 2005; Chatterjee & Hadi, 2009). These approaches typically validate several assumptions, such as the normality of the residuals, the linearity of the relationship between the explanatory and response variables, and the presence of outliers or influential data points. These diagnostics provide "actionable" insights and allow us to choose to remove or handle outlier points, transform features, or include or remove features to better explain the data. However, for NNs, applying linear regression diagnostics is not straightforward. Nonetheless, model diagnostics for deep NNs is a growing field with ongoing efforts to understand model failures and behaviors. This section discusses these efforts and shows that our unique perspective on model diagnosis is orthogonal to previous studies.

The work most closely related to ours in the literature uses theoretically principled approaches to measure NNs. One such approach is *Heavy-tailed Self-regularization (HT-SR) theory* (Martin & Mahoney, 2017; 2021b; Martin et al., 2021; Martin & Mahoney, 2021a; Yang et al., 2023; Zhou et al., 2024), which analyzes weight matrices to predict and explain model performance. This line of work also connects to *generalization metrics* (McAllester, 1999; Keskar et al., 2017; Bartlett et al., 2017; Jiang et al., 2019; Dziugaite et al., 2020; Baek et al., 2022; Jiang et al., 2022; Kim et al., 2023; Andriushchenko et al., 2023). Previous research on generalization metrics has primarily focused on predicting test performance trends and designing regularizers during training. In contrast, our study develops metrics to diagnose sources of model failure and predict optimal ways to improve model configuration.

Recent research on LLMs often uses the *self-diagnosis* ability to evaluate output quality (Schick et al., 2021; Chen et al., 2023; Fu et al., 2023; Ji et al., 2023; Zheng et al., 2024). Although these methods provide useful information, many lack transparency, making their diagnostic processes difficult to interpret. In contrast, our work uses theoretically motivated metrics and easy-to-interpret tree models for diagnostics. Research in *mechanistic interpretability* aims to understand the learned circuits in models (Wang et al., 2023; Conmy et al., 2023; Chughtai et al., 2023) and to probe the learned features or concepts (Gurnee et al., 2023). Other studies (Ilyas et al., 2022) focus on *data attribution*, attributing model behavior to training samples, and extending this approach to large-scale models (Park et al., 2023) and analyzing training algorithms (Shah et al., 2023). Instead of concentrating on specific model components (e.g., mechanistic interpretability) or individual training data samples (e.g., data attribution), our work aims to diagnose failures arising from generic sources in machine learning pipelines, such as training hyperparameters, data quantity, and model size. While *model editing* (De Cao et al., 2021; Mitchell et al., 2022; Meng et al., 2022) is also related to diagnosis, it primarily focuses on downstream control and knowledge editing, which is beyond the scope of our diagnostic approach.

2.2. Learning Curve Prediction

Learning curves, as seen in Banko & Brill (2001); Hestness et al. (2017); Sun et al. (2017); Mohr & van Rijn (2022); Viering & Loog (2023), measure a model's generalization

performance relative to training data size. Recent research has focused on neural scaling laws (NSLs) (Kaplan et al., 2020; Alabdulmohsin et al., 2022; Hoffmann et al., 2022; Caballero et al., 2023; Muennighoff et al., 2024), particularly in LLMs. These laws indicate that language model performance scales with model size, dataset size, and training computation in a power-law manner. However, NSLs are typically used to predict optimal resource allocation before or during the early stages of training, assuming known training configurations to generate initial points on the curves. In contrast, our diagnosis framework focuses on post-training analysis, where a single pre-trained model is available, and the exact training configuration is unknown. Our paper also focuses on developing transferable metrics that enable robust model diagnostics when hyperparameter tuning on the original training data is not feasible. Furthermore, the simple power-law relationship in NSLs may overlook complex behaviors that arise from the joint influence of load and temperature parameters. For example, recent work shows that a model's performance exhibits such complex behaviors when both load and temperature change simultaneously, potentially leading to multi-regime patterns (Yang et al., 2021; Zhou et al., 2023) and double descent (Bartlett et al., 2020; Nakkiran et al., 2021a; Mei & Montanari, 2022; Hastie et al., 2022; Caballero et al., 2023), beyond simple power-law-like patterns. Our work incorporates load and temperature parameters to handle these multi-regime patterns effectively.

2.3. Statistical Physics of Learning

Our framework builds on the statistical physics of learning (Seung et al., 1992; Watkin et al., 1993; Engel, 2001; Zdeborová & Krzakala, 2016; Bahri et al., 2020) and considers factors such as data amount and model size. From the perspective of statistical physics, load-like parameters characterize the quantity and/or quality of data relative to model size. Temperature-like parameters, on the other hand, characterize the magnitude of noise introduced during stochastic training. Several recent papers (Martin & Mahoney, 2017; Yang et al., 2021; Zhou et al., 2023) use load and temperature parameters to analyze multi-regime patterns in NNs.

3. Preliminaries and Background

NN Training. We consider training a NN f, with trainable parameters $\boldsymbol{\theta} \in \mathbb{R}^p$, on a training dataset comprising n datapoint/label pairs, using an optimizer parameterized by a hyperparameter t. This paper mostly considers *temperature-like* hyperparameters t, such as the learning rate or batch size. We denote the error of NNs (evaluated on a particular dataset) by $\mathcal{E}$ and the loss by $\mathcal{L}$. The error evaluated using the training and validation sets is thus denoted as $\mathcal{E}_{\text{tr}}$ and $\mathcal{E}_{\text{val}}$, respectively, while the loss is denoted by $\mathcal{L}_{\text{tr}}$ and $\mathcal{L}_{\text{val}}$.

Loss Landscape Metrics. Three types of metrics were introduced in Yang et al. (2021) to quantify the local and global geometric structures of loss landscapes, and they were used to study "phase transitions" in the hyperparameter space. In particular, these metrics are: "connectivity" metrics such as mode connectivity (Garipov et al., 2018; Draxler et al., 2018), "similarity" metrics such as Centered Kernel Alignment (Kornblith et al., 2019), and "local sharpness" metrics such as the Hessian trace and largest Hessian eigenvalue (Yao et al., 2020). The mode connectivity ($\mathcal{C}$) quantifies how well different local minima are connected to each other in the loss landscape. The Hessian trace ($\mathcal{H}_t$) and largest Hessian eigenvalue ($\mathcal{H}_e$) contain the curvature information of the loss landscape, which could be used to quantify the local sharpness. Since $\mathcal{H}_t$ and $\mathcal{H}_e$ have been shown to provide similar information in determining the regime (Yang et al., 2021), we mainly use $\mathcal{H}_t$ to measure sharpness. CKA similarity ($\mathcal{S}$) captures the similarity between the outputs of trained models. These metrics are precisely defined in the Appendix B.

4. Constructing Model Diagnosis Framework

In this section, we build the framework of `MD tree`. Section 4.1 presents definitions of the diagnosis problems. Section 4.2 presents `MD tree` method. Section 4.3 presents different model metrics and the few-shot diagnosis method. Section 4.4 presents the evaluation setup, including the diagnostic datasets and transfer setups.

4.1. Defining the Diagnosis Problem

Given a training algorithm $\mathcal{A}$, a NN f_0 trained with a specific setting of p parameters, n data samples, and optimizer hyperparameter t can be represented as $f_0 = \mathcal{A}(p, n, t)$. In the context of real-world constraints, we set the maximum allowable values for data amount, model size, and the range for optimizer hyperparameters as $n_{\max}$, $p_{\max}$ and $[t_{\min}, t_{\max}]$. Our goal is to identify the "failure source" m of a trained model f_0, which is the hyperparameter primarily responsible for its suboptimal performance. The impact of the failure source is measured using a metric we call *room for improvement* (RFI).

Failure Sources. We consider four types of failure sources:

- Failure $m_t^{\downarrow}$: the optimizer hyperparameter (t) is smaller than the optimal choice.
- Failure $m_t^{\uparrow}$: the optimizer hyperparameter (t) is larger than the optimal choice.
- Failure m_p: the number of parameters (p) is too small.
- Failure m_n: the amount of data (n) is too small.

The above taxonomy defines the set of failure sources considered in this paper: $\mathcal{M} = \{m_t^{\downarrow}, m_t^{\uparrow}, m_p, m_n\}$.

Room for Improvement (RFI). The RFI of a failure source m is defined as the gap (in validation error) between the current configuration (p, n, t) and the optimal configuration

when changing *the single hyperparameter* in (p, n, t) that corresponds to the failure source m. For example, the RFI of failure source m_p is defined as

$$\begin{aligned} \text{RFI}(m_p, f_0) &= \mathcal{E}_{\text{val}}(f_0) - \mathcal{E}_{\text{val}}(f^*), \\ \text{where} \quad f^* &= \mathcal{A}(q^*, n, t), \\ q^* &= \underset{q \in [p, p_{\max}]}{\arg\min}\ \mathcal{E}_{\text{val}}(\mathcal{A}(q, n, t)). \end{aligned} \tag{1}$$

where the last two lines indicate finding the optimal number of model parameters larger than or equal to the current value p but smaller than or equal to the maximum allowable model size $p_{\max}$. The RFI for other failure sources are defined similarly and presented in Appendix C.1.

Canonical Diagnosis Questions. Using RFI, we define the diagnosis questions Q1 and Q2. We define the *RFI gap* as the RFI difference between two failure sources, denoted as G. For example, the RFI gap between large and small optimizer hyperparameters is $G(m_t^{\uparrow}, m_t^{\downarrow}) = \text{RFI}(m_t^{\uparrow}) - \text{RFI}(m_t^{\downarrow})$. Then, Q1 can be defined as the following binary classification problem:

$$\text{Q1} : f_0 \to \{G(m_t^{\uparrow}, m_t^{\downarrow}) > 0, G(m_t^{\uparrow}, m_t^{\downarrow}) < 0\}. \tag{2}$$

Similarly, Q2 can be defined as

$$\text{Q2} : f_0 \to \{G(m_p, m_t) > 0, G(m_p, m_t) < 0\}, \tag{3}$$

where $G(m_p, m_t) = \text{RFI}(m_p) - \text{RFI}(m_t)$ and $\text{RFI}(m_t) = \max\{\text{RFI}(m_t^{\uparrow}), \text{RFI}(m_t^{\downarrow})\}$. We write $G(m, m')$ simply as G when the two failure sources m, m' are clear. In Appendix C.2, we explain why $G = 0$ is not considered as a third class in the problem.

Diagnosis Objective. Given a sample (which is a pre-trained model f_0), one can use equation (1) to define the RFI and then define the binary label using equation (2) or equation (3). The goal is to learn a function that maps a pre-trained model f_0 to the binary label in $\{G(m, m') > 0, G(m, m') < 0\}$, using only features extracted from f_0, without calculating $G(m, m')$, and no retraining is allowed.

4.2. Defining `MD tree`

We now introduce `MD tree`, which is a tree-based method to predict whether $G(m, m')$ is greater than 0. The main idea of `MD tree` is to use a divide-and-conquer way to diagnose failure sources using loss landscape metrics. See Figure 2 for the tree structure. We aim to partition the hyperparameter space into multiple regimes, where the models in the same regime roughly share the same failure source. Yang et al. (2021) use these loss landscape metrics to determine the NN regimes and predict test performance. Our tree construction borrows the regime partition from this work to grow a partial tree and then completes it using more experimentally useful branch partitions. The tree is constructed hierarchically, prioritizing metrics that have been shown

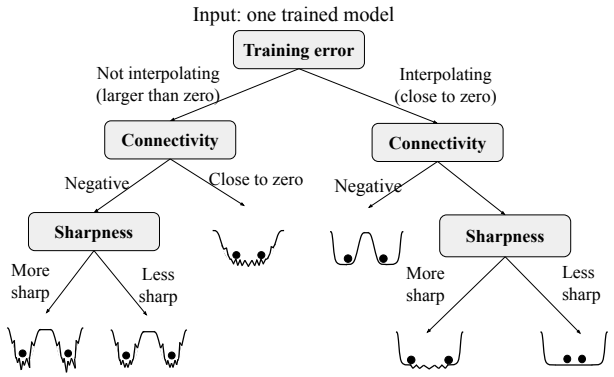


Figure 2. **(MD tree based on loss landscape structure of trained models).** A DT using loss landscape metrics to determine distinct regimes of model configurations such that each regime has the same root cause of failure. The tree hierarchy is fixed, while the decision threshold is trained in a few-shot manner. Part of the tree hierarchy is selected using ideas from Yang et al. (2021), which suggests a "multi-regime" structure of the hyperparameter space.

to lead to sharp transitions in the hyperparameter space, such as training error and mode connectivity. The similarity measure tends to have a smooth transition and is given a lower priority. Therefore, the tree is constructed in the following order: training error at the root, connectivity at the first depth level, and finally sharpness or similarity. This order emphasizes the importance of the interpolation threshold (Nakkiran et al., 2021a) and connectivity (Yang et al., 2021; Zhou et al., 2023) because they can be used to measure sharp and clear transitions in NN behaviors. Thus, the tree structure is fixed. `MD tree` only optimizes the thresholds of the metrics at each internal node sequentially from top to bottom. Initial values and search ranges are provided for each threshold, and the bounded Brent method (Brent, 1973) is used to optimize these thresholds to maximize training accuracy. The hyperparameters are provided in Appendix D.

4.3. Defining Baseline Methods

We introduce baseline diagnosis methods for solving Q1 and Q2, which cover the model metrics that are used as features and the diagnosis functions.

Model Metrics. To predict failure sources based on a trained model f_0, we define a feature vector $\mathbf{P}$, which comprises several metrics derived from f_0. We study loss landscape metrics and two conventional baselines:

- **Loss landscape**: combining the local and global loss landscape metrics defined in Section 3 with training error. The components of the feature vector are $\mathbf{P} = [\mathcal{E}_{\text{tr}}, \mathcal{C}, \mathcal{H}_t, \mathcal{S}]$.
- **Validation**: training error/loss, and validation error/loss, formally $\mathbf{P} = [\mathcal{E}_{\text{val}}, \mathcal{E}_{\text{tr}}, \mathcal{L}_{\text{val}}, \mathcal{L}_{\text{tr}}]$.
- **Hyperparameter**: the hyperparameters of trained NN that are not related to the diagnosis question. For example, when predicting whether t is large or small ($m_t^{\uparrow}$ vs $m_t^{\downarrow}$), t is not considered a metric. However, the parameter

amount p and the data amount n are considered.

Diagnosis Function. We adopt a standard DT as a baseline classifier to map the feature vector $\mathbf{P}$ to the binary decision $\{G > 0, G < 0\}$. Recall that G denotes the RFI gap between two failure sources. We adopt the standard implementation of DT in Hastie et al. (2001), with details provided in Appendix D.2.

4.4. Experimental Setup

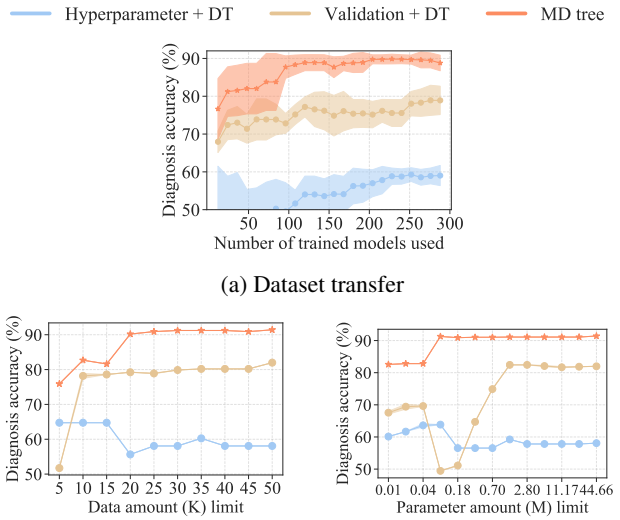
Collections of Pre-trained Models. We introduce the datasets used to evaluate `MD tree`, primarily obtained from the collections of pre-trained models studied by Yang et al. (2021). We release these collections for future research on model diagnosis[2]. The collection of models, denoted as $\mathcal{F}$, includes various ResNet models trained on CIFAR-10 with differing number of parameters (p), data amounts (n), and optimizer hyperparameters (t, batch size). This collection comprises a total of 1690 configurations, each with five runs (models) using different random seeds. Additionally, we have another set, $\mathcal{F}'$, which includes 1690 configurations trained with the same varying parameters but trained with 10% label noise. For details, see Appendix C.3.

Few-shot Setup. We consider a few practical model diagnostic problems, where the training set for `MD tree` is small (few-shot). We elaborate on how the corresponding training sets are constructed here, with more details about the training sample labeling provided in Appendix C.4.

- **w/ dataset transfer**: For Q1 (which is to determine whether the hyperparameter t is large or small), the training set consists of models randomly sampled from $\mathcal{F}$ for a fixed parameter count and data amount (p, n). For Q2 (determining model size versus optimizer), the training set consists of models randomly sampled from $\mathcal{F}$ for a fixed data amount. For both cases, the test set is $\mathcal{F}'$.
- **w/ scale transfer**: The training set includes models trained with data amounts below certain thresholds (e.g., 5K data points) or with parameter counts below specific limits (e.g., 0.04M). The test set is $\mathcal{F}'$, the same as dataset transfer. Note that using $\mathcal{F}'$ as the test set in the scale transfer study inherently includes dataset transfer. This design ensures consistency in our test set $\mathcal{F}'$ across the different transfer studies, allowing for comparability of results and ensuring that the test set does not include any training samples from $\mathcal{F}$.

In addition to the two primary cases, we consider three additional transfer scenarios. We first examine OOD generalization, where pre-trained models handle distribution shifts during testing. Second, we investigate class-imbalanced training, where pre-trained models are initially trained on long-tailed datasets. Finally, we explore diagnosing unseen model architectures, specifically focusing on Vision Transformers, while the training set includes ResNet models. The details are presented in Appendix E.1.

[2]https://github.com/YefanZhou/ModelDiagnosis



(a) Dataset transfer

(b) Scale transfer: limiting model's training data amount

(c) Scale transfer: limiting model's parameter amount

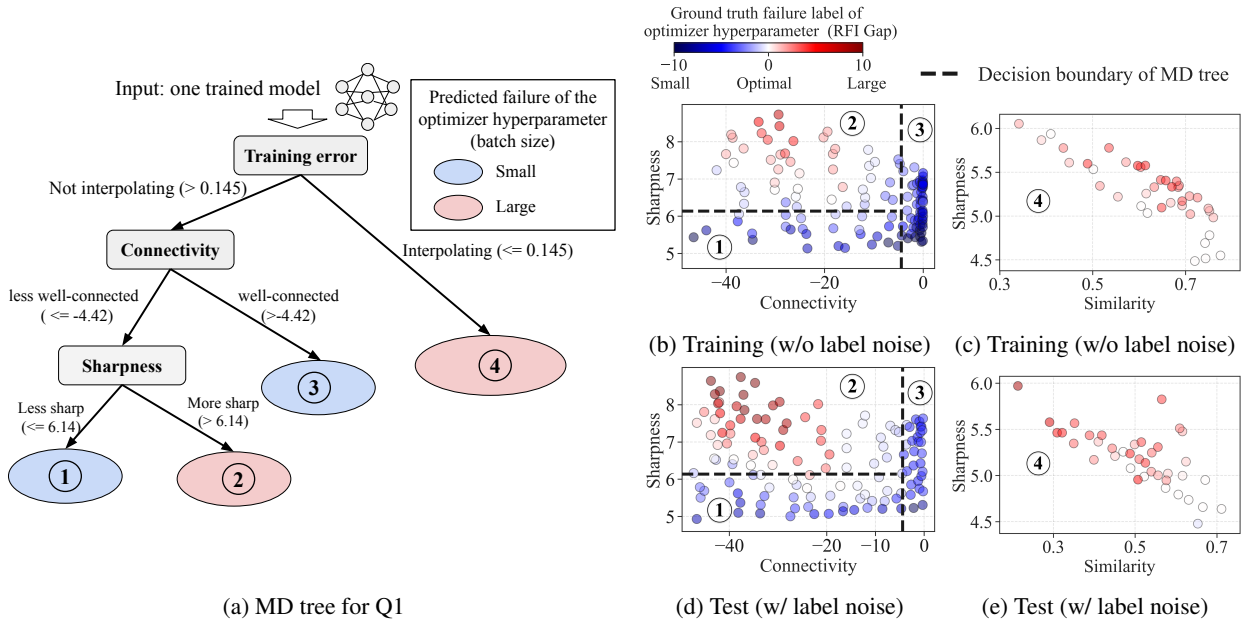
Figure 3. **(Comparing `MD tree` to baseline methods on Q1 tasks with dataset and scale transfer).** The y-axis indicates the diagnosis accuracy. (a) The x-axis indicates the number of pre-trained models used for building the training set. (b) The x-axis indicates the maximum amount of training (image) data for training models in the training set. (c) The x-axis indicates the maximum number of parameters of the models in the training set.

5. Is the Optimizer Hyperparameter Large or Small

In this section, we focus on Q1, which is to determine whether an optimizer hyperparameter (e.g., batch size) is too large or too small. This is formulated as a binary classification problem defined in equation (2). In Section 5.1, we evaluate `MD tree` on the dataset transfer and the scale transfer tasks. In Appendix A.2, we further study `MD tree` on a one-step configuration change task. In Appendix E.2, we study three additional transfer scenarios.

5.1. Evaluating Diagnosis Methods

We evaluate `MD tree` by comparing it with two baseline methods: normal DT with **Validation** or **Hyperparameter** as features, defined in Sections 4.3. In Figure 3, we report the diagnosis accuracy in the few-shot setting on two tasks: dataset transfer and scale transfer. We repeat all experiments in Figure 3 for five runs and report the mean and standard deviation. Figure 3(a) shows the diagnosis accuracy versus the size of the training set, for which the training data is randomly subsampled, which leads to a large standard deviation. Figure 3(b) and Figure 3(c), on the other hand, show the results of model diagnosis when the training set is restricted to the subset of models trained with limited



(a) MD tree for Q1 (b) Training (w/o label noise) (c) Training (w/o label noise) (d) Test (w/ label noise) (e) Test (w/ label noise)

Figure 4. **(Visualizing MD tree and its diagnosis results for Q1).** *Left*: structure of the tree defined in Section 4.2. The color of the leaf node indicates the predicted class by `MD tree`. The threshold values are learned from the training set. *Right*: The first row represents training samples, and the second row represents test samples. Each colored circle represents one sample (which is one pre-trained model configuration), and the color represents the ground-truth label: blue means the hyperparameter is too small, while red means too large. The black dashed line indicates the decision boundary of `MD tree`. Each numbered regime on the right corresponds to the leaf node with the same number on the tree. The samples in 4(b) and those in 4(c) are separated by training error. The same applies to 4(d) and 4(e).

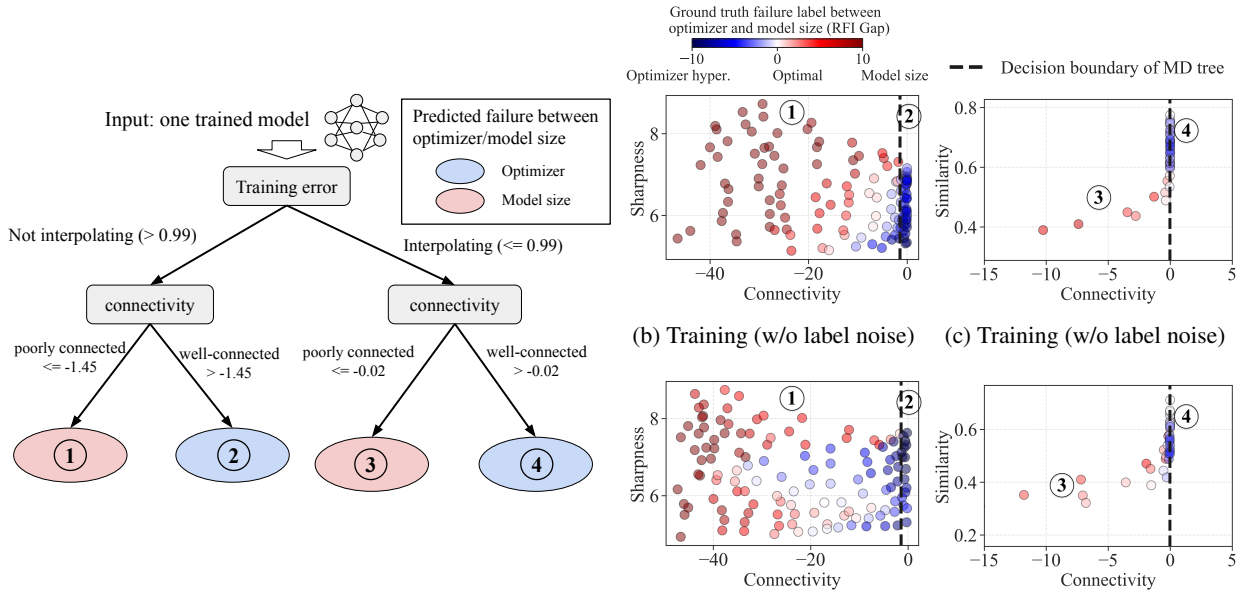
data or model size. We do not perform any subsampling in this subset. Therefore, the training set is deterministic, and the diagnostic accuracy has low standard deviations. In Appendix F.1, we conduct an ablation study comparing our method with baselines using normal DT with combined Validation and Hyperparameter features.

`MD tree` provides accurate and interpretable diagnosis. Figure 3(a) shows that `MD tree` reaches around 87.7% accuracy when trained using 96 samples. When trained with 96 samples, `MD tree` outperforms normal DTs with Validation metrics by 14.88%. When trained with 12 samples, `MD tree` can still reach an accuracy of 76.64%.

Further, in Figure 4, we analyze `MD tree` by visualizing how the samples are categorized into each leaf node. On the left side, we show the `MD tree` structure along with the metric thresholds used to split each internal node. The color of each leaf node represents the predicted labels. On the right side, we illustrate the classification boundaries (black dashed lines) established by `MD tree`. Each regime corresponds to a leaf node on the left side, and the color of each circle represents the actual label of the failure source. In Figures 4(b) and 4(c), we see that the regimes (numbered from ① to ④), separated by `MD tree`'s black boundaries, mainly consist of circles of uniform color, matching the color of the corresponding leaf node in Figure 4(a). The same observation holds for test samples shown in Figure 4(d) and 4(e). This indicates that `MD tree` accurately classifies samples into the correct categories. More importantly, the diagnosis results from `MD tree` are interpretable. For example, it can predict that a model's batch size is too large because that leads to poor connectivity and sharper local minima (regime ②), which is indeed an issue with large-batch training discussed in Keskar et al. (2017).

Loss landscape metrics are more effective than validation metrics in diagnosing model failures. Comparing the visualization of `MD tree` in Figure 4 and validation metrics in Figure 19 further explains why `MD tree` outperforms the latter in diagnosis accuracy. Validation-based metrics result in nonlinear classification boundaries between failure sources, with significant discrepancies between models in the training set (Figure 19(a), w/o label noise) and models in the test set (Figure 19(c)). On the contrary, in Figure 4, representing pre-trained models using loss landscape metrics as features makes them piecewise linearly separable. In addition, the threshold (or boundaries) learned during training (top) can be generalized to the test set (bottom).

`MD tree` has high transferability from small-scale to large-scale models. As evidenced in Figures 3(b) and 3(c), `MD tree` demonstrates strong transferability from small-scale to large-scale models. Recall that the test set F' comprises 1560 pre-trained models, including large-scale models trained with up to 50K data points and 44.66M parameters. From Figure 3(b), we see that `MD tree` achieves 75.90% accuracy in predicting across the entire test set



(a) MD tree for Q2 (b) Training (w/o label noise) (c) Training (w/o label noise) (d) Test (w/ label noise) (e) Test (w/ label noise)

Figure 5. **(Visualizing MD tree and its diagnosis results for Q2).** *Left*: structure of the tree defined in Section 4.2. The color of the leaf node indicates the predicted class by `MD tree`. The threshold values are learned from the training set. *Right*: The first row represents training samples, and the second row represents test samples. Each colored circle represents one sample (which is one pre-trained model configuration), and the color represents the ground-truth label: blue means the failure source is the optimizer, while red means model size. The black dashed line indicates the decision boundary of `MD tree`. Each numbered regime on the right corresponds to the leaf node with the same number on the tree. The samples in 5(b) and those in 5(c) are separated by training error. The same applies to 5(d) and 5(e).

$\mathcal{F}'$ when trained on small-scale models with only 5K data points. In Figure 3(c), `MD tree` achieves 82.56% accuracy on $\mathcal{F}'$ when trained on models with fewer than 0.01M parameters. This high transferability is attributed to the universal multi-regime pattern in the space of model configurations (as illustrated in Yang et al. (2021) using extensive experiments), which emerges early even if we only consider small-scale models. The multi-regime pattern is further illustrated in Figure 21, which shows that the decision boundaries established by `MD tree` for models trained with small data amounts closely resemble those of models trained with large data amounts. This observation underscores the practicality of our method, as `MD tree` can be trained in few-shot experiments using small-scale models. In turn, `MD tree` can be used to diagnose failures in large-scale models.

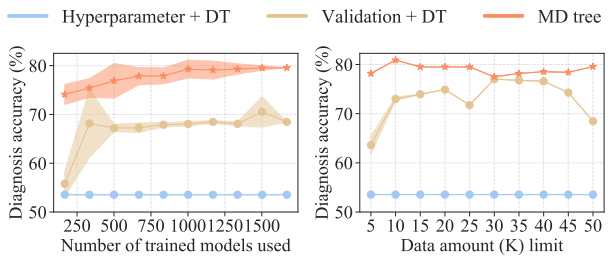
A notable transition of diagnosis performance is observed in scale transfer. A significant transition in diagnostic performance occurs during scale transfer. In Figure 3(c), we note a sudden increase in `MD tree`'s accuracy from 82.82% to 91.28% when the training set begins to include models with 0.10M parameters. This jump in accuracy can be attributed to including models that represent a more complete range of loss landscape regimes. Figure 23 in the Appendix provides the details of this transition. Models smaller than 0.10M parameters are often limited to a certain range of loss landscape metric values, such as consistently negative mode connectivity, making it challenging to establish accurate and generalizable diagnostic thresholds. This highlights the challenges in extrapolating diagnostic insights from very small-scale models to larger ones, a phenomenon discussed by Zhou et al. (2023), where the optimal hyperparameters differ significantly between small and large models. Interestingly, this also suggests that, for an effective diagnosis of extremely large models, extremely large models are not necessary; medium-sized models (those with adequate mode connectivity, as defined by Zhou et al. (2023)), are sufficient.

6. Which Hyperparameter Leads to Failure

We now turn to Q2, which considers situations where, despite the possibility of altering two hyperparameters for improved performance, budget constraints require prioritizing the search for only one of them. Specifically, we compare two factors: the optimizer hyperparameter (t) and the model size (p), defined in equation (3). Similar to Q1, we also study a one-step configuration change task in Appendix A.3. We will evaluate `MD tree` using the accuracy of the diagnosis. In addition, we study a variation of Q2 that focuses on the amount of data versus the optimizer in Appendix G. All findings demonstrate the effectiveness of `MD tree`.

6.1. Evaluating Diagnosis Methods

Similar to Section 5.1, we compare `MD tree` with the baseline methods and provide an interpretable visualization



(a) Dataset transfer (b) Scale transfer

Figure 6. **(Comparing `MD tree` to baseline methods on Q2 in dataset and scale transfer).** y-axis indicate the diagnosis accuracy. (a) x-axis indicates the number of pre-trained models in the training set. (b) x-axis indicates the maximum amount of training (image) data for the models in the training set.

in Figure 5. The diagnosis accuracy is reported in Figure 6.

`MD tree` can provide interpretable visualization and transfer well from small-scale to large-scale models. In Figure 6(a), `MD tree` reaches 74.11% with 160 trained models, significantly outperforming the validation-based method by 18.31%. The visualizations in Figure 5 highlight `MD tree`'s ability to use loss landscape metrics to distinctly separate models. In contrast, the validation metrics shown in Figure 20 present complex and less interpretable boundaries.

Figure 6(b) demonstrates `MD tree`'s effective transferability, maintaining high performance (78.17%) even when trained on small-scale models with 5K data points. This is supported by Figure 22, which shows that the connectivity-based decision boundary established with limited data can be successfully applied to models trained with larger data amount. However, `MD tree` exhibits some limitations, such as the misclassification of some test models in Figure 5(d), numbered regime ①. This limitation results in `MD tree` achieving a maximum accuracy of 79.56% in Q2, which is lower than the 89.85% accuracy in Q1. This could be attributed to shifts in the model's distribution caused by training with label noise. Future research could explore whether fine-tuning in the target domain enhances `MD tree`'s generalization.

Comparing `MD tree` with validation-based methods. We provide insights into the advantages of using loss landscape metrics in `MD tree`. We find that loss landscape metrics are more sensitive to hyperparameter changes than validation metrics. This is demonstrated by the case study in Figure 7, which addresses Q2 in a *zero-shot scenario*. Two pretrained models, A and B, need to be diagnosed. Validation metrics only show both models with high training errors, indicating underfitting, but fail to distinguish the different failure sources (optimizer vs. model size) that lead to the underfitting. However, loss landscape metrics reveal that Model A, with poor connectivity, suffers from model size issues, whereas Model B, with good connectivity, is affected

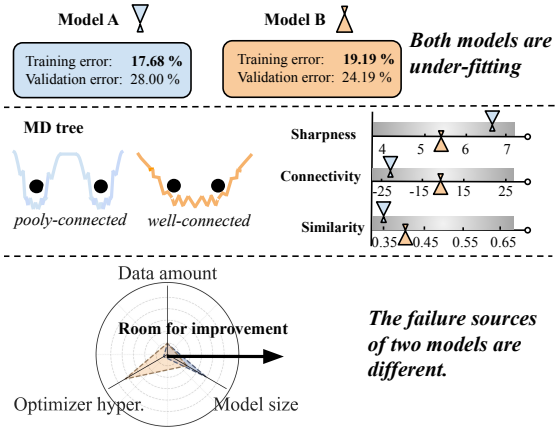


Figure 7. **(Case Study: `MD tree` vs. validation method for Q2).** *Top*: Validation metrics provide limited diagnosis: both models have the same underfitting issues. *Middle*: Loss landscape metrics distinguish the models: Model A has poor connectivity, while Model B has good connectivity. *Bottom*: Model A suffers from a size issue, and Model B has a training hyperparameter issue.

by optimizer settings. This is further confirmed by the radar plot, where Model A shows a higher RFI in model size, and Model B has a higher RFI in optimizer hyperparameters. In Figure 26, we show a similar zero-shot analysis on two overfitting models.

The importance of `MD tree`'s tree hierarchy. Lastly, we evaluate the role of `MD tree`'s fixed tree structure, which partitions data according to a specific sequence of metrics: training error, connectivity, and then sharpness or similarity. This structure is inspired by Yang et al. (2021), which highlights the usefulness of encoding known inductive biases in model diagnosis. We evaluate a conventional DT employing the same loss landscape metrics $\{\mathcal{E}_{\text{tr}}, \mathcal{C}, \mathcal{H}_t, \mathcal{S}\}$ but without adhering to this precise order. As shown in Figure 17, `MD tree` consistently outperforms standard DT in Q1 and Q2, especially with a limited number of trained models. This shows that this inductive bias imposes a beneficial regularization on `MD tree` when data is limited. Details are provided in Appendix F.2.

7. Conclusion

In conclusion, `MD tree` introduces an innovative framework for diagnosing the underperformance of trained NN models without retraining, meeting the urgent need for effective diagnostics with low computational cost. Using loss landscape metrics, `MD tree` outperforms conventional validation-based methods by providing more accurate and generalizable diagnoses. Through quantitative few-shot classifier predictions and qualitative visualization, `MD tree` proves its efficacy in identifying critical failure sources, such as inappropriate optimizer hyperparameters or inadequate model sizes. This has been demonstrated in various scenarios, including dataset and scale transfers, without relying on detailed training configurations.

Impact Statement

Our research focuses on creating a diagnostic method for pre-trained models. Although `MD tree` can be used in various applications, including those with potential adverse effects, the algorithm itself does not present immediate negative social impacts. On the contrary, `MD tree` has significant social value, particularly in helping universities and researchers with limited computing resources to effectively use pre-trained models.

Acknowledgment

We would like to thank Caleb Geniesse and Tiankai Xie for helping with the implementation of loss landscape metrics. We would like to thank Vignesh Kothapalli, Tianyu Pang, Lei Hsiung, and Michael Mahoney for their valuable feedback.

References

Alabdulmohsin, I. M., Neyshabur, B., and Zhai, X. Revisiting neural scaling laws in language and vision. *Advances in Neural Information Processing Systems*, 35:22300–22312, 2022.

Andriushchenko, M., Croce, F., Müller, M., Hein, M., and Flammarion, N. A modern look at the relationship between sharpness and generalization. In *International Conference on Machine Learning*, 2023.

Avron, H. and Toledo, S. Randomized algorithms for estimating the trace of an implicit symmetric positive semidefinite matrix. *J. ACM*, 2011.

Baek, C., Jiang, Y., Raghunathan, A., and Kolter, J. Z. Agreement-on-the-line: Predicting the performance of neural networks under distribution shift. *Advances in Neural Information Processing Systems*, 35:19274–19289, 2022.

Bahri, Y., Kadmon, J., Pennington, J., Schoenholz, S. S., Sohl-Dickstein, J., and Ganguli, S. Statistical mechanics of deep learning. *Annual Review of Condensed Matter Physics*, 11:501–528, 2020.

Banko, M. and Brill, E. Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, 2001.

Bartlett, P. L., Foster, D. J., and Telgarsky, M. J. Spectrally-normalized margin bounds for neural networks. *Advances in Neural Information Processing Systems*, 30, 2017.

Bartlett, P. L., Long, P. M., Lugosi, G., and Tsigler, A. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Brent, R. P. *Algorithms for Minimization without Derivatives*. 1973.

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

Chatterjee, S. and Hadi, A. S. *Sensitivity analysis in linear regression*. John Wiley & Sons, 2009.

Chen, Y., Wang, R., Jiang, H., Shi, S., and Xu, R. Exploring the use of large language models for reference-free text quality evaluation: A preliminary empirical study. In *Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023 (Findings)*, pp. 361–374, 2023.

Chughtai, B., Chan, L., and Nanda, N. A toy model of universality: Reverse engineering how networks learn group operations. In *International Conference on Machine Learning*, 2023.

Conmy, A., Mavor-Parker, A., Lynch, A., Heimersheim, S., and Garriga-Alonso, A. Towards automated circuit discovery for mechanistic interpretability. *Advances in Neural Information Processing Systems*, 36:16318–16352, 2023.

Cui, Y., Jia, M., Lin, T.-Y., Song, Y., and Belongie, S. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9268–9277, 2019.

De Cao, N., Aziz, W., and Titov, I. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6491–6506, 2021.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, 2017.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning*, 2018.

Dziugaite, G. K., Drouin, A., Neal, B., Rajkumar, N., Caballero, E., Wang, L., Mitliagkas, I., and Roy, D. M. In search of robust measures of generalization. *Advances in Neural Information Processing Systems*, 33:11723–11733, 2020.

d'Ascoli, S., Touvron, H., Leavitt, M. L., Morcos, A. S., Biroli, G., and Sagun, L. Convit: Improving vision transformers with soft convolutional inductive biases. In *International Conference on Machine Learning*, 2021.

Engel, A. *Statistical mechanics of learning*. Cambridge University Press, 2001.

Engel, A. and den Broeck, C. V. *Statistical mechanics of learning*. Cambridge University Press, 2001.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

Fu, J., Ng, S.-K., Jiang, Z., and Liu, P. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*, 2023.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in Neural Information Processing Systems*, 31, 2018.

Gurnee, W., Nanda, N., Pauly, M., Harvey, K., Troitskii, D., and Bertsimas, D. Finding neurons in a haystack: Case studies with sparse probing. *Transactions on Machine Learning Research*, 2023.

Hastie, T., Tibshirani, R., and Friedman, J. *The Elements of Statistical Learning*. Springer Series in Statistics. 2001.

Hastie, T., Montanari, A., Rosset, S., and Tibshirani, R. J. Surprises in high-dimensional ridgeless least squares interpolation. *Annals of statistics*, 50(2):949, 2022.

Haussler, D., Seung, H. S., Kearns, M., and Tishby, N. Rigorous learning curve bounds from statistical mechanics. In *Proceedings of the Seventh Annual Conference on Computational Learning Theory*, pp. 76–87, 1994.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Vinyals, O., Rae, J., and Sifre, L. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, 2022.

Ilyas, A., Park, S. M., Engstrom, L., Leclerc, G., and Madry, A. Datamodels: Understanding predictions with data and data with predictions. In *International Conference on Machine Learning*, 2022.

Ji, Y., Gong, Y., Peng, Y., Ni, C., Sun, P., Pan, D., Ma, B., and Li, X. Exploring chatgpt's ability to rank content: A preliminary study on consistency with human preferences. *arXiv preprint arXiv:2303.07610*, 2023.

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2019.

Jiang, Y., Nagarajan, V., Baek, C., and Kolter, J. Z. Assessing generalization of SGD via disagreement. In *International Conference on Learning Representations*, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Keskar, N. S., Nocedal, J., Tang, P. T. P., Mudigere, D., and Smelyanskiy, M. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017.

Kim, H., Park, J., Choi, Y., and Lee, J. Fantastic robustness measures: The secrets of robust generalization. In *Advances in Neural Information Processing Systems*, volume 36, pp. 48793–48818, 2023.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, 2019.

Martin, C. H. and Mahoney, M. W. Rethinking generalization requires revisiting old ideas: statistical mechanics approaches and complex learning behavior. *arXiv preprint arXiv:1710.09553*, 2017.

Martin, C. H. and Mahoney, M. W. Post-mortem on a deep learning contest: a Simpson's paradox and the complementary roles of scale metrics versus shape metrics. Technical Report Preprint: arXiv:2106.00734, 2021a.

Martin, C. H. and Mahoney, M. W. 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, 2021b.

Martin, C. H., Peng, T. S., and Mahoney, M. W. Predicting trends in the quality of state-of-the-art neural networks without access to training or testing data. *Nature Communications*, 12(1):1–13, 2021.

McAllester, D. A. Pac-bayesian model averaging. In *Annual Conference Computational Learning Theory*, 1999.

Mei, S. and Montanari, A. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022.

Meng, K., Bau, D., Andonian, A., and Belinkov, Y. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.

Miller, J. P., Taori, R., Raghunathan, A., Sagawa, S., Koh, P. W., Shankar, V., Liang, P., Carmon, Y., and Schmidt, L. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In *International Conference on Machine Learning*, 2021.

Mitchell, E., Lin, C., Bosselut, A., Finn, C., and Manning, C. D. Fast model editing at scale. In *International Conference on Learning Representations*, 2022.

Mohr, F. and van Rijn, J. N. Learning curves for decision making in supervised machine learning–a survey. *arXiv preprint arXiv:2201.12150*, 2022.

Muennighoff, N., Rush, A., Barak, B., Le Scao, T., Tazi, N., Piktus, A., Pyysalo, S., Wolf, T., and Raffel, C. A. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021a.

Nakkiran, P., Venkat, P., Kakade, S. M., and Ma, T. Optimal regularization can mitigate double descent. In *International Conference on Learning Representations*, 2021b.

Nguyen, T., Raghu, M., and Kornblith, S. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. In *International Conference on Learning Representations*, 2021.

Park, S. M., Georgiev, K., Ilyas, A., Leclerc, G., and Madry, A. TRAK: Attributing model behavior at scale. In *International Conference on Machine Learning*, 2023.

Raghu, M., Unterthiner, T., Kornblith, S., Zhang, C., and Dosovitskiy, A. Do vision transformers see like convolutional neural networks? *Advances in Neural Information Processing Systems*, 34:12116–12128, 2021.

Raschka, S. Model evaluation, model selection, and algorithm selection in machine learning. *arXiv preprint arXiv:1811.12808*, 2018.

Schick, T., Udupa, S., and Schütze, H. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp. *Transactions of the Association for Computational Linguistics*, 9:1408–1424, 2021.

Seung, H. S., Sompolinsky, H., and Tishby, N. Statistical mechanics of learning from examples. *Physical review A*, 45(8):6056, 1992.

Shah, H., Park, S. M., Ilyas, A., and Madry, A. Modeldiff: A framework for comparing learning algorithms. In *International Conference on Machine Learning*, 2023.

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 843–852, 2017.

Viering, T. and Loog, M. The shape of learning curves: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):7799–7819, 2023.

Wang, K. R., Variengien, A., Conmy, A., Shlegeris, B., and Steinhardt, J. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*, 2023.

Watkin, T. L. H., Rau, A., and Biehl, M. The statistical mechanics of learning a rule. *Rev. Mod. Phys.*, 65:499–556, Apr 1993.

Weisberg, S. *Applied linear regression*, volume 528. John Wiley & Sons, 2005.

Yang, Y., Hodgkinson, L., Theisen, R., Zou, J., Gonzalez, J. E., Ramchandran, K., and Mahoney, M. W. Taxonomizing local versus global structure in neural network loss landscapes. *Advances in Neural Information Processing Systems*, 34:18722–18733, 2021.

Yang, Y., Theisen, R., Hodgkinson, L., Gonzalez, J. E., Ramchandran, K., Martin, C. H., and Mahoney, M. W. 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*, pp. 3011–3021, 2023.

Yao, Z., Gholami, A., Lei, Q., Keutzer, K., and Mahoney, M. W. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems*, volume 31, pp. 4949–4959, 2018.

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE International Conference on Big Data*, pp. 581–590. IEEE, 2020.

Zdeborová, L. and Krzakala, F. Statistical physics of inference: Thresholds and algorithms. *Advances in Physics*, 65(5):453–552, 2016.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhou, Y., Yang, Y., Chang, A., and Michael, M. W. A three-regime model of network pruning. In *International Conference on Machine Learning*, 2023.

Zhou, Y., Pang, T., Liu, K., Mahoney, M. W., Yang, Y., et al. Temperature balancing, layer-wise weight analysis, and neural network training. *Advances in Neural Information Processing Systems*, 36, 2024.

Appendix

A. Applying Diagnosis to One-step Configuration Change

A.1. Experimental Setup

Here we elaborate on the experimental setup for the one-step configuration change task. The one-step change includes 1) determining the change direction, and 2) determining the size of the adjustment step. The first one, for Q1, means determining whether to increase or decrease the optimizer hyperparameter. For Q2, it means first determining whether to increase the model size or adjust the optimizer hyperparameter, if the first decision chooses the latter one, then the second decision has to be made to decide whether to increase or decrease. This change direction is determined by the diagnosis method such as `MD tree`, e.g. if `MD tree` finds that the failure source is the optimizer hyperparameter is too large ($m_t^{\uparrow}$), then we choose to decrease it.

In addition to comparing `MD tree` with the diagnosis methods **Validation** + DT and **Hyperparameter** + DT, we also compare it with Random diagnosis and Optimal diagnosis. Random diagnosis, which selects the change direction randomly between two options, represents the lower bound of performance for diagnosis-based hyperparameter changes. Optimal diagnosis, using the ground-truth label to choose the best direction, represents the upper bound of performance, equivalent to a diagnosis method achieving 100% test accuracy. We examine three approaches for determining the change steps: "fixed (one) step size", "random step size", and "optimal step size". The "fixed" refers to taking one tuning step, "random" refers to taking a random step size along the determined direction, and "optimal" refers to taking the step size that can reach the best performance along the chosen direction, representing the upper bound of performance improvement of a specific diagnosis method.

A.2. One-step Change on Optimizer Hyperparameter (Q1)

We extend the practicality of `MD tree`, by applying it to a task involving a one-step change in the model configuration. These methods predict the direction–either an increase or decrease–of an adjustment in an optimizer hyperparameter (e.g. batch size here). This adjustment is executed using one of three different types of step size: a fixed (one) step size, a random step size, or the optimal step size. We then evaluate the impact of this adjustment by measuring the test accuracy improvement of the adjusted model on CIFAR-10. Diagnosis methods are built using the training set $\mathcal{F}$ and evaluated on the test set $\mathcal{F}'$. More information on the setup and evaluation can be found in Appendix A.1.

Table 1 shows that `MD tree` enhances task performance, outperforming three baselines, including the **Validation** approach, across three different step size settings. The task performance gains correlate with the diagnostic accuracy shown in Figure 3, with `MD tree` leading, followed by the **Validation** method, and then the **Hyperparameter** approaches. An important takeaway is the impact of accurate diagnostics on improving task performance through a single hyperparameter adjustment. Specifically, an accurate diagnosis like `MD tree` can boost performance by 5.52% with the optimal step size, markedly better than the Random diagnosis (3.50%). This highlights the importance of precise diagnostic tools for effective tuning. Future work will explore predicting the optimal step size for hyperparameter adjustments using loss landscape metrics.

Table 1. **(Q1: task performance improvement (CIFAR-10 accuracy (↑, %)) of performing one-step change only on the optimizer hyperparameter).** The diagnosis methods decide whether to increase or decrease the given trained model's hyperparameter and then the decision is combined with three types of step sizes. We trained using 96 configurations randomly sampled from $\mathcal{F}$ and evaluated the average improvement across 1560 sub-optimal model configurations from $\mathcal{F}'$ for testing. Each experiment was repeated with five random seeds, and we reported the mean and standard deviation.

Diagnosis method	w/ fixed step	w/ random step	w/ optimal step
Optimal diagnosis	1.48 ± 0.00	2.92 ± 0.03	5.83 ± 0.00
Random diagnosis	0.92 ± 0.03	1.78 ± 0.03	3.50 ± 0.10
Hyperparameter + DT	0.92 ± 0.07	1.84 ± 0.26	3.52 ± 0.50
Validation + DT	1.24 ± 0.12	2.31 ± 0.17	4.63 ± 0.23
`MD tree`	$\mathbf{1.34 \pm 0.06}$	$\mathbf{2.74 \pm 0.13}$	$\mathbf{5.52 \pm 0.15}$

A.3. One-step Change on Optimizer Hyperparameter or Model Size (Q2)

We applied the diagnosis methods for addressing Q2 to a one-step configuration change task, with the setup detailed in Appendix A.1. The diagnosis methods decide whether to adjust the optimizer hyperparameter or increase the model size.

If the former option is selected, the diagnosis method for Q1 will further determine whether to increase or decrease the hyperparameter, and all decisions are combined with three types of adjustment step sizes. Table 2 demonstrates that `MD tree` surpasses three baseline methods in enhancing task performance, affirming its better diagnostic accuracy. This experiment highlights the importance of precise hyperparameter adjustment for task improvement, a goal that `MD tree` facilitates.

Table 2. **(Q2: task performance improvement (CIFAR-10 accuracy (↑, %)) of performing one-step hyperparameter change on the optimizer hyperparameter and model size).** The diagnosis method decides whether to adjust the optimizer hyperparameter or increase the model size. We used 156 configurations randomly sampled from $\mathcal{F}$ for training and evaluated the average improvement on 1560 sub-optimal configurations from $\mathcal{F}'$. Each experiment was repeated with five random seeds, and we reported the mean and standard deviation.

Method	w/ fixed step	w/ random step	w/ optimal step
Optimal diagnosis	1.92 ± 0.00	5.47 ± 0.05	8.83 ± 0.00
Random diagnosis	1.19 ± 0.03	3.09 ± 0.11	5.03 ± 0.10
Hyperparameter + DT	1.48 ± 0.00	4.36 ± 0.09	6.59 ± 0.00
Validation + DT	1.42 ± 0.12	4.09 ± 0.59	6.33 ± 0.61
`MD tree`	$\mathbf{1.73 \pm 0.04}$	$\mathbf{4.79 \pm 0.16}$	$\mathbf{8.10 \pm 0.22}$

B. Loss Landscape Metrics

In this section, we define loss landscape metrics, then we discuss the related work on sharpness. Lastly, we discuss the computational efficiency of these metrics.

Connectivity Metrics. For two models with two parameter configurations $\boldsymbol{\theta}, \boldsymbol{\theta}'$, we can learn a curve $\gamma(t), t \in [0,1]$ connecting $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ such that $\gamma(0) = \boldsymbol{\theta}$, $\gamma(1) = \boldsymbol{\theta}'$, and the training error (loss) evaluated on $\gamma(t)$ for any $t \in [0,1]$ is minimal. One approach to parameterizing $\gamma(t)$ is to use a Bezier curve (Garipov et al., 2018) with $k+1$ bends, $\gamma_\phi(t) = \sum_{j=0}^{k} \binom{k}{j} (1-t)^{k-j} t^j \boldsymbol{\theta}_j, t \in [0,1]$, where $\boldsymbol{\theta}_0 = \boldsymbol{\theta}$, $\boldsymbol{\theta}_k = \boldsymbol{\theta}'$, and $\phi = \{\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_{k-1}\}$ are trainable parameters of additional models. In this study, we use Bezier curves with three bends ($k = 2$). We use the peak of the curve to quantify whether there exists a barrier:

$$\begin{aligned} &\mathcal{C}(\boldsymbol{\theta}, \boldsymbol{\theta}') = -\mathcal{E}_{\text{tr}}(\gamma(t^*)), \\ &\text{where, } t^* = \arg\max_t \left| \frac{1}{2} \left(\mathcal{E}_{\text{tr}}(\boldsymbol{\theta}) + \mathcal{E}_{\text{tr}}(\boldsymbol{\theta}') \right) - \mathcal{E}_{\text{tr}}(\gamma_\phi(t)) \right| \end{aligned} \tag{4}$$

Similarity Metrics. CKA similarity measures the difference in features between two parameter configurations $\boldsymbol{\theta}, \boldsymbol{\theta}'$. Let $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_s\}$ be randomly sampled data points, and let $F_{\boldsymbol{\theta}} = [f_{\boldsymbol{\theta}}(\mathbf{x}_1) \ldots f_{\boldsymbol{\theta}}(\mathbf{x}_s)]^\top \in \mathbb{R}^{s \times d_{\text{out}}}$ be the output of the network. Then, the CKA similarity between two parameter configurations $\boldsymbol{\theta}, \boldsymbol{\theta}'$ is given by:

$$\mathcal{S}(\boldsymbol{\theta}, \boldsymbol{\theta}') = \frac{\text{Cov}(F_{\boldsymbol{\theta}}, F_{\boldsymbol{\theta}'})}{\sqrt{\text{Cov}(F_{\boldsymbol{\theta}}, F_{\boldsymbol{\theta}}) \, \text{Cov}(F_{\boldsymbol{\theta}'}, F_{\boldsymbol{\theta}'})}}. \tag{5}$$

where $\text{Cov}(\mathbf{X}, \mathbf{Y}) = (s-1)^{-2} \operatorname{tr}\left(\mathbf{X}\mathbf{X}^\top \mathbf{H}_s \mathbf{Y}\mathbf{Y}^\top \mathbf{H}_s\right)$, and $\mathbf{H}_s = \mathbf{I}_s - s^{-1}\mathbf{1}\mathbf{1}^\top$ is a centering matrix.

Measuring the connectivity and similarity metrics necessitates two distinct sets of weights, $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, each corresponding to the same configuration trained with different random seeds. In the diagnostic dataset, each data sample is a training configuration instead of a single trained model. This is because each configuration contains five pre-trained models trained with different random seeds. For each training configuration, we evaluate a single similarity/connectivity score.

Sharpness Metrics. The Hessian matrix at a given point $\boldsymbol{\theta}_0$ in the parameter space can be represented as $\nabla^2_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_0)$. We report the leading eigenvalue $\mathcal{H}_e = \lambda_{\max}\left(\nabla^2_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_0)\right)$ and the trace $\mathcal{H}_t = \operatorname{tr}\left(\nabla^2_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_0)\right)$ to summarize the local curvature properties in a single value.

Here, we discuss related work on sharpness. We elaborate the connection between Yang et al. (2021) and Dinh et al. (2017); Keskar et al. (2017); Foret et al. (2021); Yao et al. (2018). The connection is that prior work finds that in some cases flat minima have better generalization (Keskar et al., 2017; Foret et al., 2021), but there exist some cases such as adversarial training and reduction of ℓ_2 regularization that sharp minima generalize better (Dinh et al., 2017; Yao et al., 2018). Yang et al. (2021) studies why this happens, and the explanation was that prior work neglects global loss landscape properties

such as connectivity. Furthermore, Yang et al. (2021) shows that combining the local and global metrics leads to a more accurate prediction of generalization performance. This motivates us to combine connectivity and sharpness in our MD tree to predict the failure sources.

Computational Efficiency. To demonstrate the low computational burden of loss landscape measurements, we provide details on calculating sharpness and connectivity, along with additional runtime results. For measuring sharpness, we follow the implementation by Yao et al. (2020), using the Hessian trace to represent sharpness. The Hessian trace is computed using Hutchinson's method (Avron & Toledo, 2011) from RandNLA. Let $\mathbf{H}$ denote the Hessian matrix. This method approximates the Hessian trace using $\mathbb{E}[\mathbf{v}^T\mathbf{H}\mathbf{v}]$, where $\mathbf{v}$ is a random vector with components i.i.d. sampled from the Rademacher distribution. The computational cost of a Hessian matrix-vector multiplication is equivalent to a single gradient backpropagation. We estimate the expectation by drawing multiple random samples until convergence or until 100 iterations are reached. Thus, the cost of measuring sharpness for a trained model is at most 100 backpropagations. For measuring connectivity, we adopt the implementations of Yang et al. (2021) and Garipov et al. (2018) to compute mode connectivity. This process involves 50 epochs of training to search for a three-bend Bezier curve and five inferences on the training set to evaluate the curve's barrier.

In Table 3, we compare the runtime of training a single model configuration with the time taken to evaluate two loss landscape metrics on the same configuration. The model configuration involves training ResNet18 on CIFAR-10 with a batch size of 128, using five random seeds for 150 epochs. Our measurements indicate that, on average over 10 runs, each training epoch takes 23.32 seconds. We also calculate the percentage increase in time due to measuring loss landscape metrics compared to model training. Our results show that measuring sharpness and connectivity increases the total training time by only 0.244% and 7.04%, respectively, indicating that the computational burden is reasonable. The testing platform used was a Quadro RTX 6000 GPU paired with an Intel Xeon Gold 6248 CPU.

Table 3. Comparison of runtime for model training versus sharpness and connectivity measurements used in `MD tree`.

	Model training	Sharpness	Connectivity
Runtime (seconds)	17492.25	42.73	1232.76
Increase in runtime compared to model training (%)	-	0.244%	7.04%

C. Model Diagnosis Framework

C.1. Room for Improvement (RFI) of Failure Sources

Here, we provide the complete definitions of RFI for the failure sources. We define four failure sources $\mathcal{M} = \{m_t^{\downarrow}, m_t^{\uparrow}, m_p, m_n\}$, and RFI of m_p is defined by equation (1) in Section 4.1.

The RFI of failure source $m_t^{\uparrow}$ (the optimizer hyperparameter is larger than the optimal choice) is defined as

$$\begin{aligned} \text{RFI}(m_t^{\uparrow}, f_0) &= \mathcal{E}_{\text{val}}(f_0) - \mathcal{E}_{\text{val}}(f^*), \\ \text{where} \quad f^* &= \mathcal{A}(p, n, q^*), \\ q^* &= \underset{q \in [t_{\min}, t]}{\arg\min}\, \mathcal{E}_{\text{val}}(\mathcal{A}(p, n, q)). \end{aligned} \tag{6}$$

The RFI of failure source $m_t^{\downarrow}$ (the optimizer hyperparameter is smaller than the optimal choice) is defined as

$$\begin{aligned} \text{RFI}(m_t^{\downarrow}, f_0) &= \mathcal{E}_{\text{val}}(f_0) - \mathcal{E}_{\text{val}}(f^*), \\ \text{where} \quad f^* &= \mathcal{A}(p, n, q^*), \\ q^* &= \underset{q \in [t, t_{\max}]}{\arg\min}\, \mathcal{E}_{\text{val}}(\mathcal{A}(p, n, q)). \end{aligned} \tag{7}$$

The RFI of failure source m_n (the amount of data n is too small) is defined as

$$\begin{aligned} \text{RFI}(m_n, f_0) &= \mathcal{E}_{\text{val}}(f_0) - \mathcal{E}_{\text{val}}(f^*), \\ \text{where} \quad f^* &= \mathcal{A}(p, q^*, t), \\ q^* &= \underset{q \in [n, n_{\max}]}{\arg\min}\, \mathcal{E}_{\text{val}}(\mathcal{A}(p, q, t)). \end{aligned} \tag{8}$$

C.2. Rationale for Considering a Binary Classification Problem

We consider a binary classification problem that predicts $\{G > 0, G < 0\}$, rather than including near-optimal cases ($G = 0$) to avoid a three-class classification. First, the number of near-optimal configurations is relatively low compared to the other two classes, leading to a class imbalance in the training set. For example, when predicting whether an optimizer hyperparameter is large or small, only 130 out of 1690 configurations are "optimal" (given our current sampling granularity), while the remaining 1560 configurations fall into the other two classes. Second, this imbalance will not disappear even if we increase the sampling granularity because, technically speaking, the "optimal" configurations form the set of critical points in the hyperparameter space and thus should have a lower dimensionality than the hyperparameter space itself. That is, even if we increase the granularity by $100\times$, the "optimal" configurations will only occupy an even lower ratio. Therefore, we exclude these optimal configurations from the training and test sets and do not consider the third class in the problem.

C.3. Collections of Trained Models

In our few-shot studies, we use models sampled from $\mathcal{F}$ as the *training set* to train the `MD tree` classifier; then we use another model collection $\mathcal{F}'$ as the test set. $\mathcal{F}'$ comprises models trained with the same varying factors, but each model is trained with 10% label noise. Both $\mathcal{F}$ and $\mathcal{F}'$ include various ResNet models trained on the CIFAR-10 dataset from scratch, and each differs in three factors: 1) the number of parameters (p) is varied by changing the width of ResNet-18 among $\{2, 3, 4, 6, 8, 11, 16, 23, 32, 45, 64, 91, 128\}$, 2) the optimizer hyperparameter (t) is varied by changing the batch size among $\{16, 24, 32, 44, 64, 92, 128, 180, 256, 364, 512, 724, 1024\}$, 3) the amount of data samples (n) is varied via subsampling the subset from CIFAR-10 training set between 10% to 100%, with increments of 10%. The combination of these three factors provides 1690 configurations, each comprising 5 runs with different random seeds.

C.4. Setup of Dataset/Scale Transfer

In the few-shot transfer learning setup, we detail the procedure for labeling RFI and G for each sample in the training set. The training/test set refers to the collections of pre-trained models. To label a training sample, additional models are required to compute RFI and G. We build the training set by sampling $\mathcal{F}$ in a structured manner. For example, to diagnose whether the optimizer hyperparameter of model f_0 is too large or too small, we sample $\mathcal{F}$ by fixing the parameter and data amount (p, n). We then obtain all the possible optimizer hyperparameters within the range $[t_{\min}, t]$ and $[t, t_{\max}]$, where t is the optimizer hyperparameter used to train f_0, and $t_{\min}$ and $t_{\max}$ are the minimum and maximum allowable values. Using this sampling, we can label $\text{RFI}(m_t^{\uparrow})$ using equation (6), $\text{RFI}(m_t^{\downarrow})$ using equation (7), and compute $G = \text{RFI}(m_t^{\uparrow}) - \text{RFI}(m_t^{\downarrow})$. In all experiments, when we report the number of pre-trained models, that includes those used for labeling G and RFI.

D. Diagnosis Methods

D.1. Tree Construction

In addition to Section 4.2, we provide further details on constructing `MD tree`. We prioritize connectivity over sharpness at the shallower levels of the tree due to its effectiveness in distinguishing dichotomous phenomena in the NN hyperparameter space. Prior work (Yang et al., 2021; Zhou et al., 2023) has demonstrated that the trends of sharpness, similarity, and model performance, influenced by temperature-like parameters, can vary significantly depending on the connectivity value, even exhibiting opposite behaviors. We treat sharpness and similarity as alternatives at the same level, using sharpness in the main experiments. An ablation study in Appendix H demonstrates that using similarity instead of sharpness provides comparable results.

For Q1 and Q2, we use different fixed subtree structures derived from the complete tree shown in Figure 2. We ensure the tree structures for each task remain consistent across all evaluations. While the tree structures are fixed, the metric thresholds at each internal node are learned from the training set. Future work could explore developing an algorithm that dynamically terminates the branching.

D.2. Hyperparameters

Hyperparameters for Baseline Methods. Our baseline methods include **Hyperparameter** + DT, **Validation** + DT, and **Loss landscape** + DT. We implement the standard DT method with the Gini impurity criterion and the best splitting strategy. We set the maximum depth of the tree to be 4 and the minimum number of samples required to split an internal node to be 2. We run all the methods for five runs with random seeds $\{42, 90, 38, 18, 72\}$.

Hyperparameters for `MD tree`. We provide the hyperparameters of our method in Table 4. We run `MD tree` for five runs with random seeds {42, 90, 38, 18, 72}.

Table 4. Initial values and search ranges of loss landscape metric threshold used in `MD tree`.

	Training error	Connectivity	Sharpness	Similarity
Initial value	0.5	-10	{5, 7}	0.5
Bound	[0, 1]	[-30, 0]	[4, 9]	[0.2, 0.8]

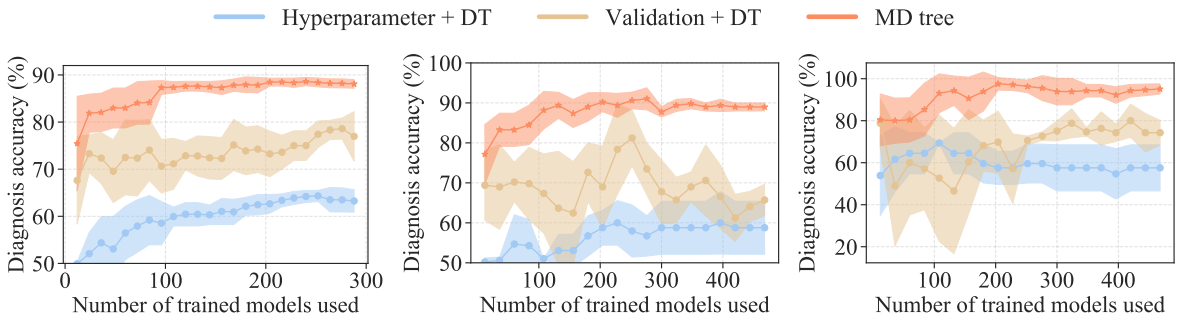
E. Broader Transfer Scenarios of `MD tree`

In this section, we demonstrate the robustness of the proposed `MD tree` in diagnosing failure sources across three common scenarios. The first scenario involves out-of-distribution (OOD) generalization (Hendrycks & Dietterich, 2019), where pre-trained models handle distribution shifts during testing. The second scenario considers class-imbalanced training, in which pre-trained models are initially trained on long-tailed datasets. The third scenario involves diagnosing unseen model architectures, specifically when the target models are Vision Transformer (Dosovitskiy et al., 2021), while `MD tree` is trained on ResNet architecture models.

E.1. Experimental Setup of Additional Transfer Scenarios

In the three additional studies, we focus on the first diagnostic task (Q1), which assesses whether the optimizer hyperparameter is large or small. The training set for `MD tree` in these studies remains consistent with the original setup described in Section 4.4, using pre-trained ResNet-18 models sampled from $\mathcal{F}$, trained on a class-balanced, in-distribution CIFAR-10 dataset. The deviation from the original setup is the creation of new test sets to evaluate the diagnostic methods under three scenarios:

- OOD generalization: test models are generated as in the original setup, sampled from $\mathcal{F}'$, but are evaluated on CIFAR-10C (Hendrycks & Dietterich, 2019) to obtain corruption accuracy. Failure source labels are then annotated based on the corruption accuracy.
- Class-imbalanced training: the test set consists of models trained on CIFAR-10-LT with an imbalance ratio of 10, evaluated using the standard CIFAR-10 test set. To get enough test models to evaluate `MD tree`, we varied model widths in the range {4, 6, 8, 16, 32, 45, 64} and training batch sizes within the range {8, 16, 32, 64, 128, 256, 512, 1024} in the test set. Each configuration was trained for 50 epochs using three random seeds.
- Transformer architecture: the test set comprises ViT-tiny models trained on CIFAR-10, evaluated using the standard CIFAR-10 test set. We vary the model configuration by changing the embedding size over {6, 12, 24, 48, 60, 96, 192}, and changing batch size over {8, 16, 32, 64, 128, 256, 364, 512}. We train each configuration for 150 epochs using three random seeds.

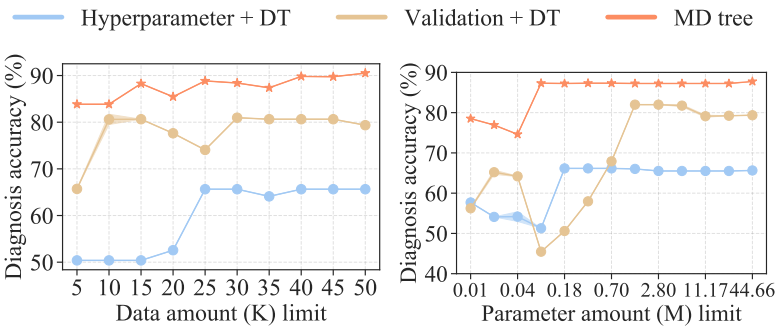


(a) OOD generalization (b) Class-imbalanced training (c) Transformer architecture

Figure 8. **(Comparing `MD tree` to baseline methods in diagnosing trained models under three additional transfer scenarios).** The y-axis represents diagnosis accuracy, while the x-axis shows the number of pre-trained models used for the training set. (a) Models in the test set are evaluated on OOD CIFAR-10C test sets instead of the original ID test sets. (b) Models in the test set are trained on class-imbalanced CIFAR-10-LT datasets. (c) Models in the test set use transformer architectures.

E.2. Empirical Evaluation

We compared `MD tree` to two baseline methods across three transfer scenarios. Figure 8 and Figure 9 show that `MD tree` consistently outperforms the baselines, demonstrating its robust generalizability. Visualizations interpreting these methods are available in Appendix E.3. Our observations indicate that in challenging transfer scenarios, validation-based methods perform worse due to increased distribution shifts between the training set's pre-trained models and the new test sets. For example, in the class-imbalanced training scenario, the validation errors of CIFAR-10-LT models differ markedly from those of CIFAR-10 models, causing decision trees built on this feature to fail in generalizing. However, the loss landscape regimes remain transferable, providing interpretable diagnostics.



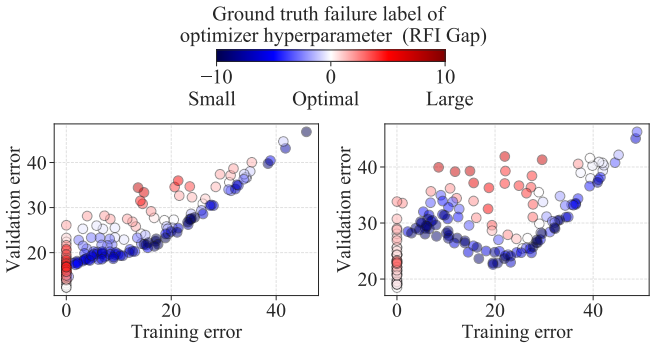
(a) Scale transfer (b) Scale transfer

Figure 9. **(Diagnosing trained models tested on out-of-distribution data (CIFAR-10-C)).** The y-axis indicates the diagnosis accuracy. (a) The x-axis indicates the maximum amount of training (image) data for training models in the training set. (b) The x-axis indicates the maximum number of parameters of the models in the training set.

E.3. Visualization

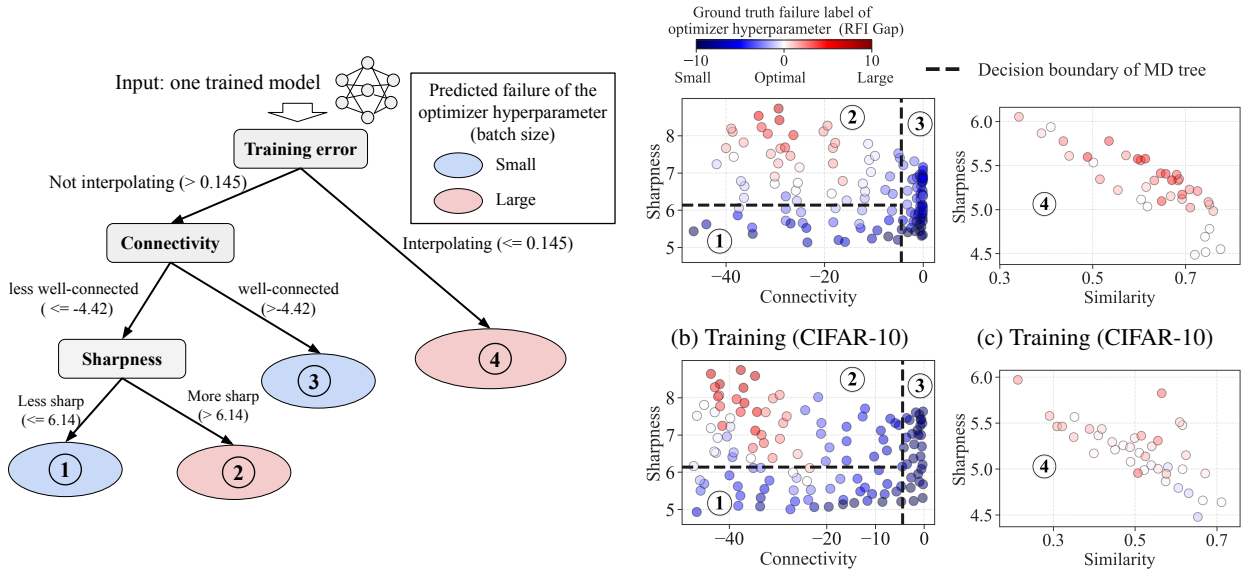
We present additional results of the studies on diagnosis under the three transfer scenarios, with the experimental setup shown in Section E.1.

OOD Generalization. Figure 9 illustrates the results of combining scale transfer and OOD generalization, confirming the effectiveness of the proposed methods. The visualizations of the validation-based method and our method are shown in Figures 10 and 11, respectively. Figure 10 highlights the issues associated with using validation metrics, showing a significant distribution shift between the training and test sets. Figure 11(a) presents the `MD tree` structure and threshold. Comparing Figures 11(b) and 11(d), we observe that pre-trained models generally exhibit similar ID and OOD failure sources, which are effectively classified by our method's decision boundary. This observation aligns with the findings of Miller et al. (2021), which indicate a positive correlation between ID and OOD performance, suggesting that models often share the same ID and OOD failure sources.



(a) Training (CIFAR-10) (b) Test (CIFAR-10-C)

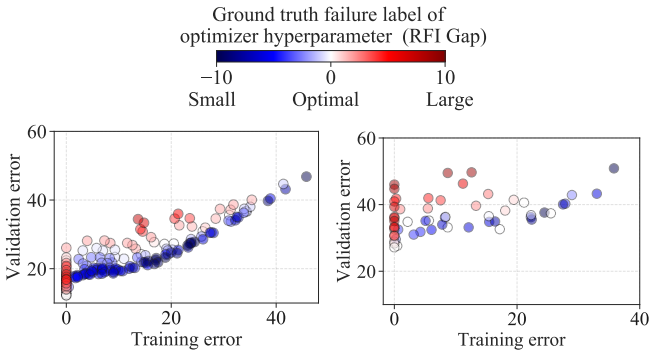
Figure 10. **(Applying validation metrics to diagnose the models tested on out-of-distribution data (CIFAR-10-C)).** (a) models in the training set are evaluated on ID CIFAR-10 test data, (b) models in the test set are evaluated on OOD CIFAR-10-C test data.



(a) MD tree (b) Training (CIFAR-10) (c) Training (CIFAR-10) (d) Test (CIFAR-10-C) (e) Test (CIFAR-10-C)

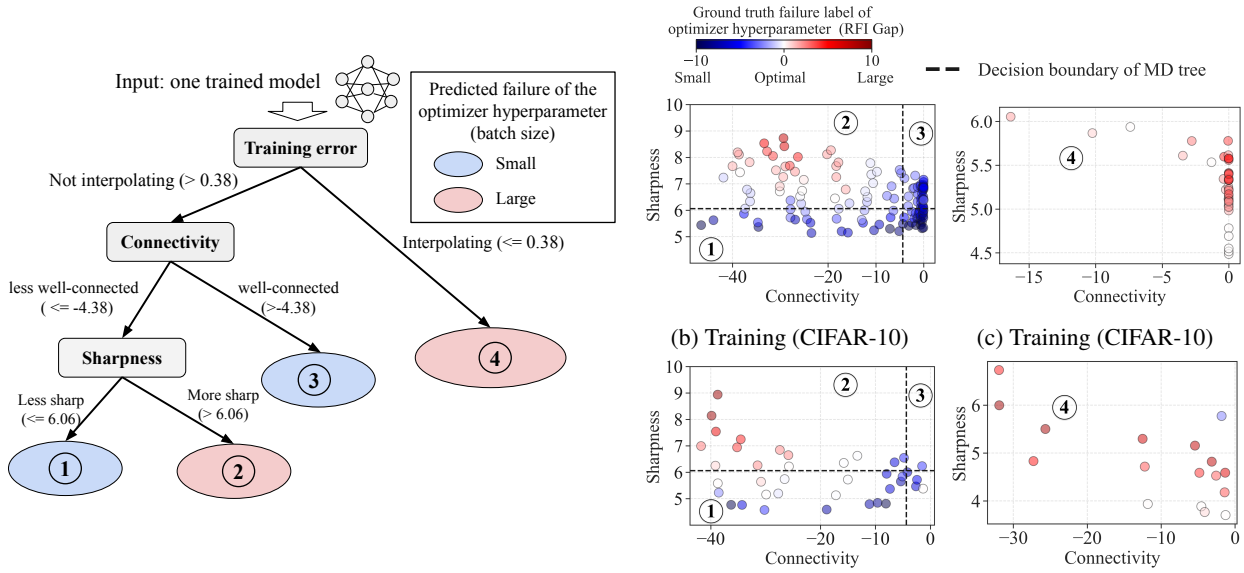
Figure 11. **(Visualizing MD tree and its diagnosis results on models tested on out-of-distribution data (CIFAR-10-C)).** *Left side*: structure of the MD tree. The color of the leaf node indicates the predicted class by `MD tree`. The threshold values are learned from the training set. *Right side*: The first row represents training samples, and the second row represents test samples. Each colored circle represents one sample (which is one pre-trained model configuration), and the color represents the ground-truth label: blue means the hyperparameter is too large, while red means small. The black dashed line indicates the decision boundary of `MD tree`. Each numbered regime on the right corresponds to the leaf node with the same number on the tree. The samples in 11(b) and those in 11(c) are separated by training error. The same applies to 11(d) and 11(e).

Class-imbalanced Training. This subsection presents the visualization results for the class-imbalanced training scenario. Figure 12 illustrates the substantial distribution shift in validation metrics between CIFAR-10 and CIFAR-10-LT models. Notably, the validation error for CIFAR-10-LT models is predominantly above 25%, whereas most CIFAR-10 models exhibit validation errors below this threshold. Figure 13 demonstrates that the decision boundaries learned by the MD tree from CIFAR-10 models effectively transfer to CIFAR-10-LT models, highlighting the robustness of the MD tree's loss landscape metrics in handling class imbalance.



(a) Training (CIFAR-10) (b) Test (CIFAR-10-LT)

Figure 12. **(Applying validation metrics to diagnose the models trained with imbalanced data (CIFAR-10-LT)).** (a) training set comprises ResNet18 models trained with CIFAR-10, (b) test set comprises ResNet18 models trained with CIFAR-10-LT.

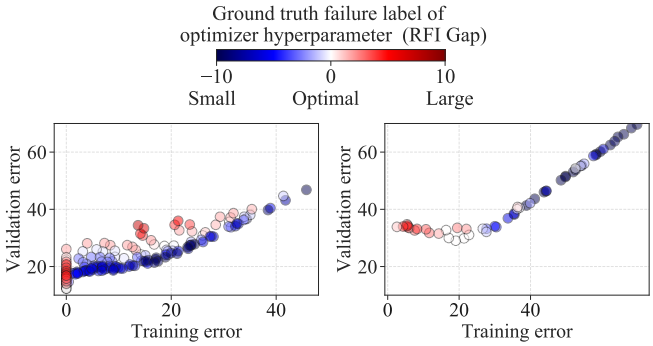


(a) MD tree (b) Training (CIFAR-10) (c) Training (CIFAR-10) (d) Test (CIFAR-10-LT) (e) Test (CIFAR-10-LT)

Figure 13. **(Visualizing MD tree and its diagnosis results on models trained on imbalanced data (CIFAR-10-LT)).** *Left side*: structure of the MD tree. The color of the leaf node indicates the predicted class by `MD tree`. The threshold values are learned from the training set. *Right side*: The first row represents training samples, and the second row represents test samples. Each colored circle represents one sample (which is one pre-trained model configuration), and the color represents the ground-truth label: blue means the hyperparameter is too large, while red means small. The black dashed line indicates the decision boundary of `MD tree`. Each numbered regime on the right corresponds to the leaf node with the same number on the tree. The samples in 13(b) and those in 13(c) are separated by training error. The same applies to 13(d) and 13(e).

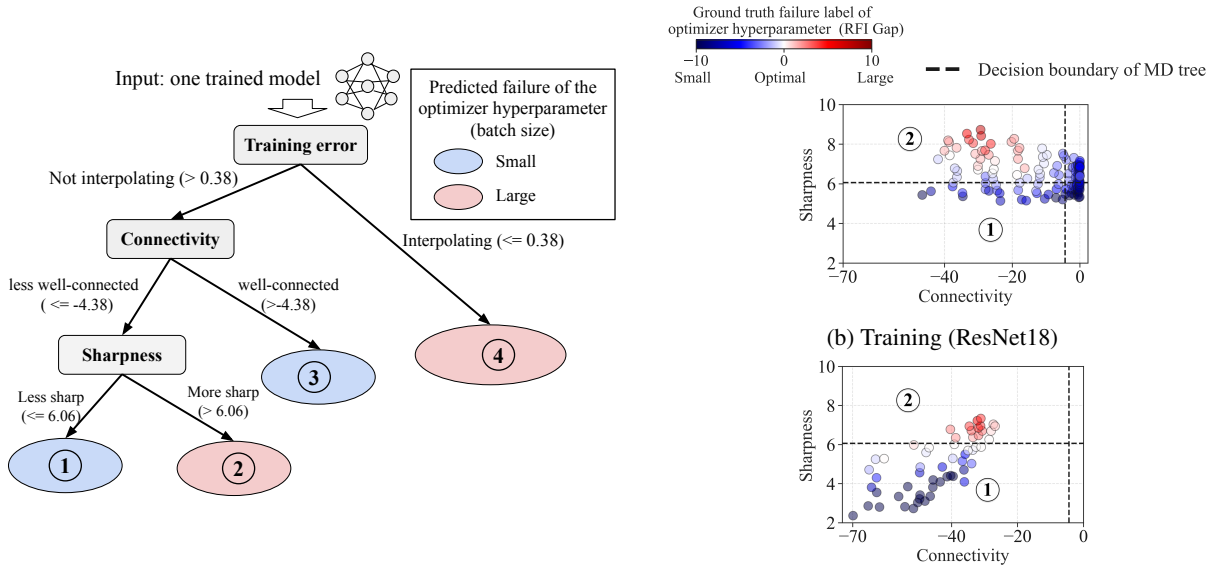
Transformer Architecture. Figure 14 shows a significant shift in the training and validation error ranges between ResNet18 models in the training set and Vision Transformer (ViT) models in the test set. This shift leads to poor performance when transferring diagnostic results from ResNet18 models to ViT models based on validation metrics. This discrepancy aligns with findings by Raghu et al. (2021) and d'Ascoli et al. (2021), which highlight the challenges of training ViT models from scratch on small-scale datasets, resulting in high training and validation errors. In contrast, convolutional architectures, such as ResNet18, train efficiently on small datasets and exhibit low errors.

Figure 15 reveals that ViT models fall into Regimes ① and ② within our MD tree constructed from ResNet18 models, with Regimes ③ and ④ absent for these ViT models. This finding shows the differing loss landscape properties between convolutional neural networks and transformer architectures. Despite this difference, the MD tree maintains strong diagnostic performance for ViT models. This result is attributed to the effective transfer of the sharpness metric's decision boundary, which separates Regimes ① and ②, from ResNet18 to ViT models.



(a) Training (ResNet18) (b) Test (ViT)

Figure 14. **(Applying validation metrics to diagnose the ViT-tiny models trained with CIFAR-10).** (a) training set comprises ResNet18 models trained on CIFAR-10, (b) test set comprises ViT-tiny models trained on CIFAR-10.



(a) MD tree

(b) Training (ResNet18)

(c) Test (ViT)

Figure 15. **(Visualizing MD tree and its diagnosis results on ViT-tiny models trained on CIFAR-10).** *Left side*: structure of the MD tree. The color of the leaf node indicates the predicted class by `MD tree`. The threshold values are learned from the training set. *Right side*: The first row represents training samples, and the second row represents test samples. Each colored circle represents one sample (which is one pre-trained model), and the color represents the ground-truth label: blue means the hyperparameter is too large, while red means small. The black dashed line indicates the decision boundary of `MD tree`. Each numbered regime on the right corresponds to the leaf node with the same number on the tree.

F. Corroborating Results

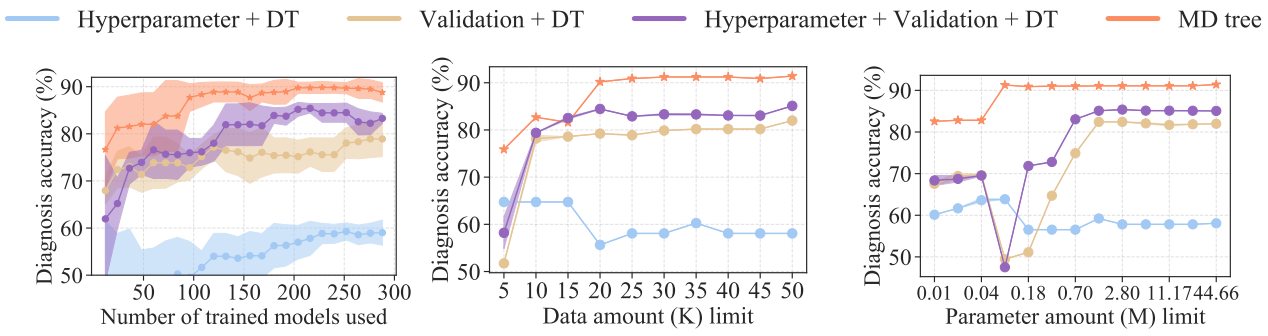
We provide additional results to corroborate the findings in the main paper. Section F.1 presents the comparison with more baseline methods. Section F.2 demonstrates that `MD tree`'s tree hierarchy helps improve its generalization ability. Section F.3 visualizes the validation metrics of the pre-trained models in Q1 and Q2 tasks, which is used to explain why validation metrics are not effective in failure source diagnosis. Section F.4 visualizes the loss landscape metrics of the pre-trained models under scale transfer scenario in Q1 and Q2 tasks, which is used to explain why `MD tree` can transfer from small-scale models to large-scale models.

F.1. More Baseline Comparison

In Figure 16, we present additional results by comparing our method to a stronger baseline that utilizes the validation metric and hyperparameter as the features of the decision tree. This method is titled "Hyperparameter + Validation + DT" and is represented as the purple curve. We can see that the additional baseline outperforms the previous two baseline methods. However, `MD tree` (orange curve) still outperforms the three baselines in both the dataset and the scale transfer scenarios. The experiments are repeated for five runs and the mean and standard deviation are reported.

F.2. Investigating Importance of `MD tree`'s Tree Hierarchy

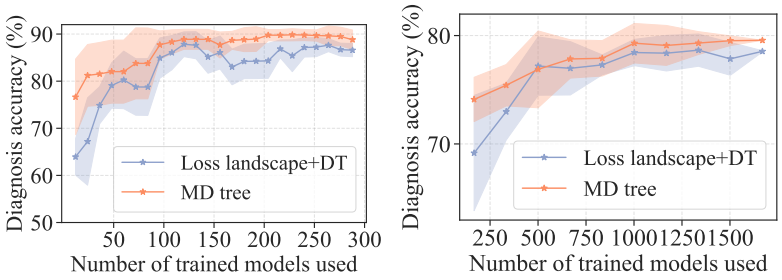
We summarize the two main reasons why `MD tree` can generalize to diagnosing unseen models: 1) the multi-phase pattern in the NN hyperparameter space, and 2) the fixed tree hierarchy of our method. The first reason has been discussed in Section 5.1, here we elaborate on the second reason. To achieve better generalization, we restricted the capacity of the decision tree by using a decision tree with only a few loss landscape features and fixing the tree hierarchy, prioritizing those features that are shown by Yang et al. (2021) to have sharp phase transitions. The goal is to evaluate whether the loss landscape metrics are useful and avoid overfitting. In Figure 17, we compare two methods on Q1 and Q2: 1) `MD tree` is our method, which uses a fixed hierarchy. 2) "Loss landscape + DT" is the baseline method, which uses the same set of loss landscape metrics as `MD tree`, but uses a decision tree with unfixed and learnable architecture. The results show that `MD tree` outperforms the baseline method in both tasks, particularly when using fewer models as the training set. The same advantage of `MD tree` holds for scale transfer in Figure 18. These findings demonstrate the importance of a fixed tree



(a) Dataset transfer (b) Scale transfer (c) Scale transfer

Figure 16. **(Comparing `MD tree` to additional baseline method (Hyperparameter + Validation + DT) on Q1 tasks of predicting optimizer hyperparameter is large or small).** The y-axis indicates the diagnosis accuracy. (a) The x-axis indicates the number of pre-trained models used for building the training set. (b) The x-axis indicates the maximum amount of training (image) data for training models in the training set. (c) The x-axis indicates the maximum number of parameters of the models in the training set. The additional baseline outperforms the other two baseline methods, but our method `MD tree` still outperforms all three in dataset transfer and scale transfer scenarios.

hierarchy in enhancing the generalizability of the diagnostic method.



(a) Q1: optimizer hyper. (b) Q2: optimizer vs. model

Figure 17. **(Comparing `MD tree` to standard DT on Q1 and Q2 task, with both utilizing the loss landscape metrics as features).** The y-axis indicates the diagnosis accuracy. The x-axis indicates the number of pre-trained models used for building the training set. `MD tree` outperforms normal DT with unfixed hierarchy trained with loss landscape metrics especially when using fewer models as the training set.

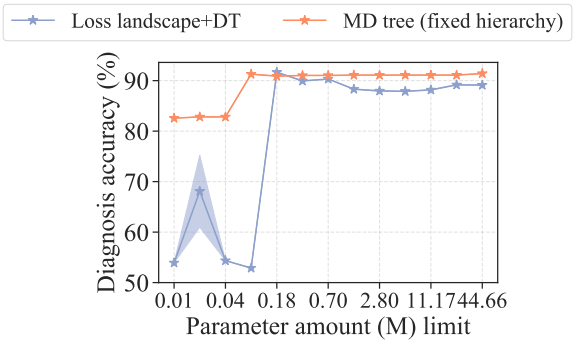
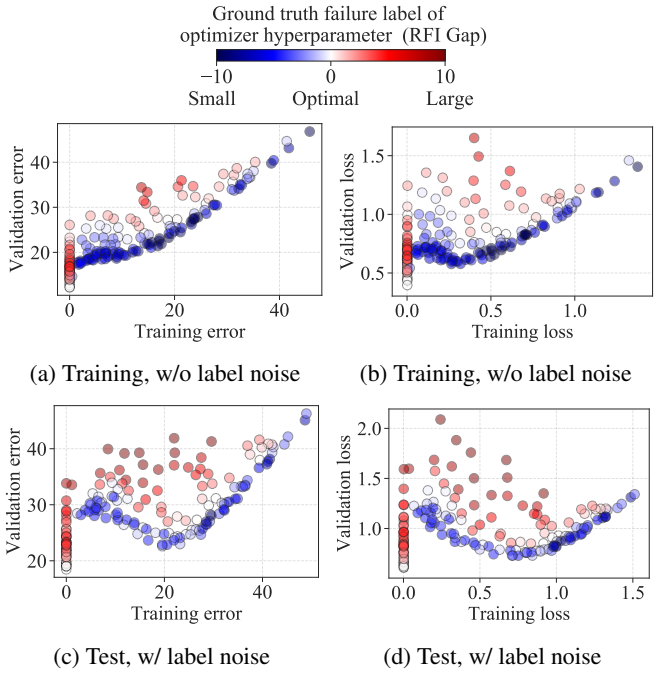


Figure 18. **(The fixed hierarchy of MD tree helps `MD tree` to generalize to unseen models with larger scales).** The y-axis indicates diagnosis accuracy in Q1 while the x-axis represents the maximum number of parameters of the models in the training set. `MD tree` outperforms normal DT with unfixed hierarchy trained with loss landscape metrics especially when using small-scale models as the training set.

F.3. Visualization for Validation Metrics in Q1 and Q2



(a) Training, w/o label noise (b) Training, w/o label noise

(c) Test, w/ label noise (d) Test, w/ label noise

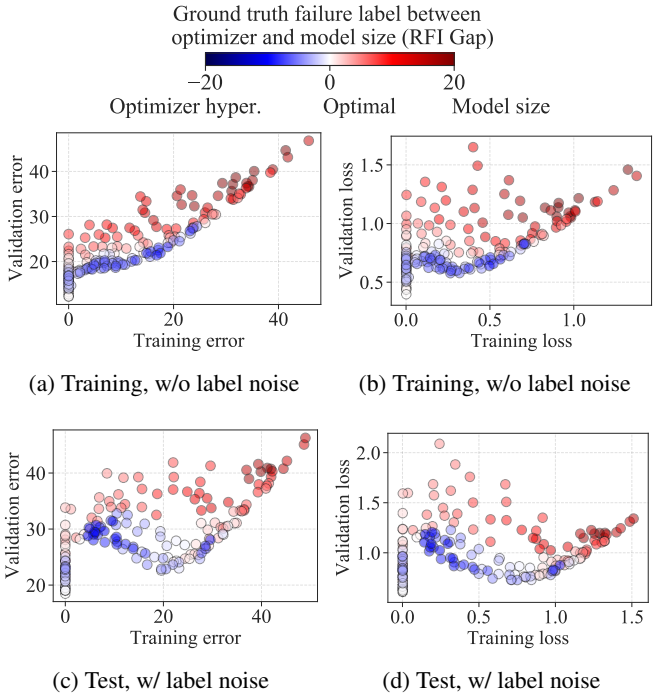
Figure 19. **(Applying validation metrics such as training/validation error and loss to diagnose the failure source of models).** (a)(b) Training set comprises models trained without label noise, (c)(d) test set comprises models trained with label noise.

Figure 19 visualizes the validation metrics and the ground-truth labels Q1 diagnosis task (optimizer hyperparameter is large or small) of pre-trained models from the training or test set. We can tell that the validation metrics have a nonlinear and complicated correlation with the failure source labels. Also, the red-versus-blue boundary in the training set (Figure 19(a) and 19(b)) are unable to make a good transfer to the test set (Figure 19(c) and 19(d)). This indicates the low transferability of validation metrics between different sets of models.

Figure 20 presents the visualization of validation metrics of pre-trained models in the Q2 diagnosis task (optimizer versus model size). Compared to those in Q1, we observe a more complicated correlation between failure source labels and validation metrics and worse transferability of decision boundary between training and test.

F.4. Visualization for Scale Transfer in Q1 and Q2

Figure 21 visualizes the loss landscape metrics of pre-trained models in Q1 and data amount transfer scenarios. We can find that the decision boundaries established by `MD tree` for small data sets (20K data points) closely resemble those for larger scales (up to 50K data points), demonstrating the transferability of `MD tree`. Figure 22 visualizes the loss landscape metrics of pre-trained models in Q2 and data amount transfer scenarios. For this task, we observe that the transferability of `MD tree` can be better even in the smallest data amount (less than 5K). Figure 23 visualizes the loss landscape metrics of pre-trained models in Q1 and parameter amount transfer scenarios. Again, it shows that boundaries determined for small-size models (less than 0.1M parameters) resemble those for larger scales (up to 44.66M parameters). From all figures, we observe notable transitions of decision boundaries varying from small-scale to large-scale. These transitions occur because very small-scale models often exhibit limited ranges in loss landscape metrics, such as consistently high sharpness or consistently negative connectivity. This demonstrates the limitations of transferring from very small-scale to large-scale models.



(a) Training, w/o label noise (b) Training, w/o label noise

(c) Test, w/ label noise (d) Test, w/ label noise

Figure 20. **(Applying validation metrics such as training/validation error and loss to diagnose the failure source of models.)** (a)(b) training set comprises models trained without label noise, (c)(d) test set comprises models trained with label noise.

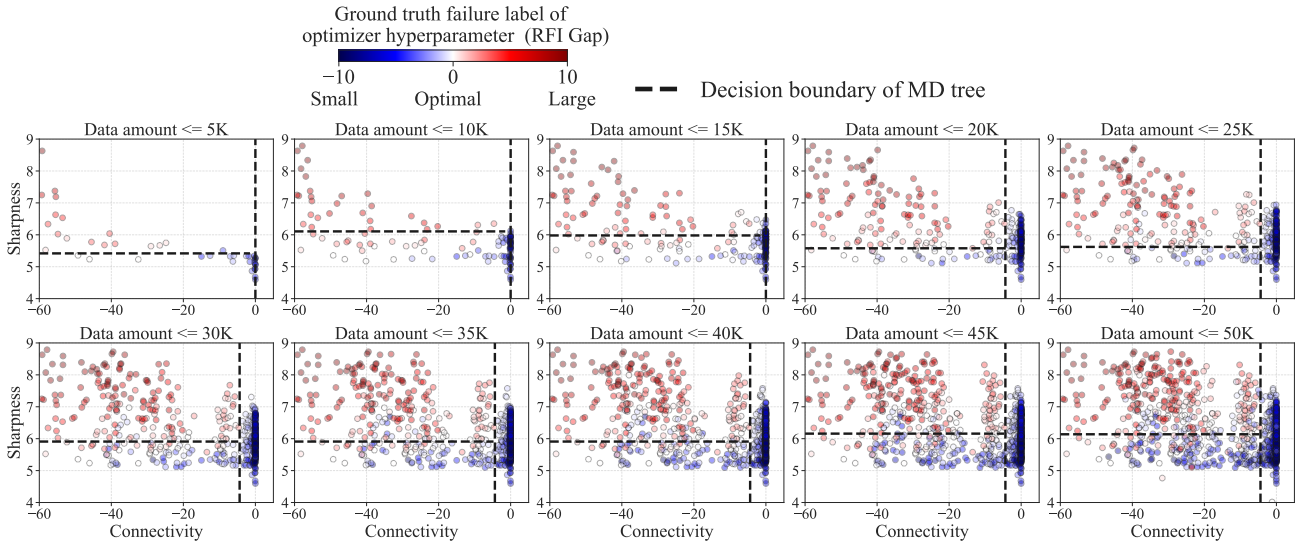


Figure 21. **(Visualization of models in training set in scale (data amount) transfer and decision boundary of `MD tree` in Q1).** Each subfigure presents the models under a specific data amount limit. The x-axis presents the connectivity and the y-axis presents the sharpness.

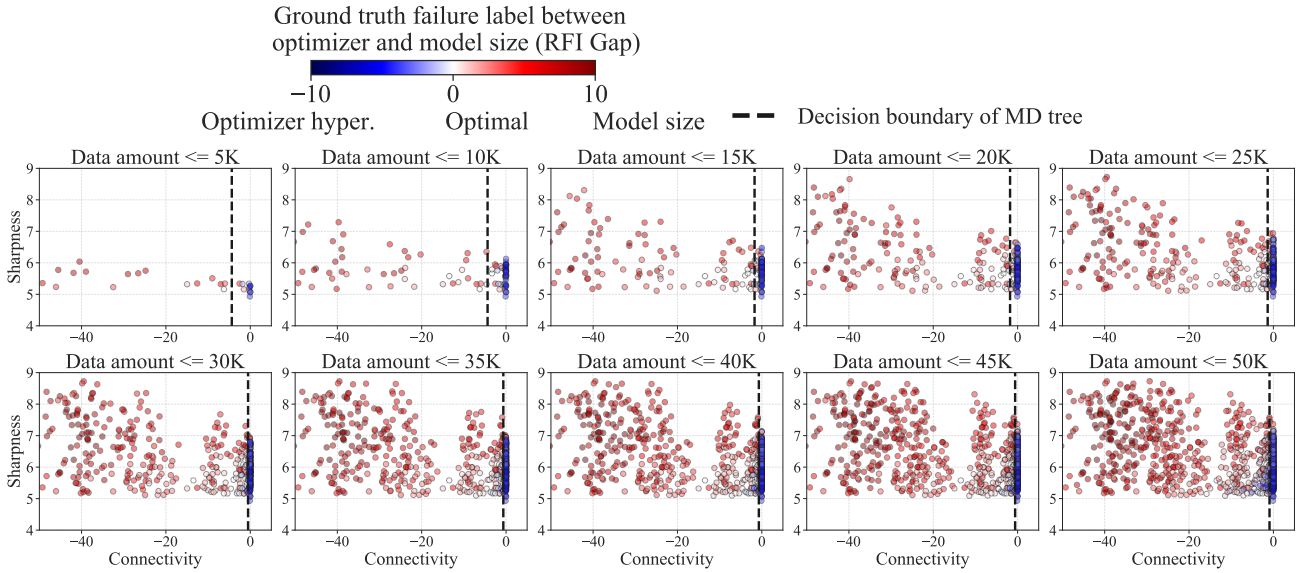


Figure 22. **(Visualization of models in training set in scale (data amount) transfer and decision boundary of `MD tree` in Q2 task).** Each subfigure presents the models under a specific data amount limit. The models are partitioned into non-interpolating (larger training error) regimes by `MD tree`. The x-axis presents the connectivity and the y-axis presents the sharpness.

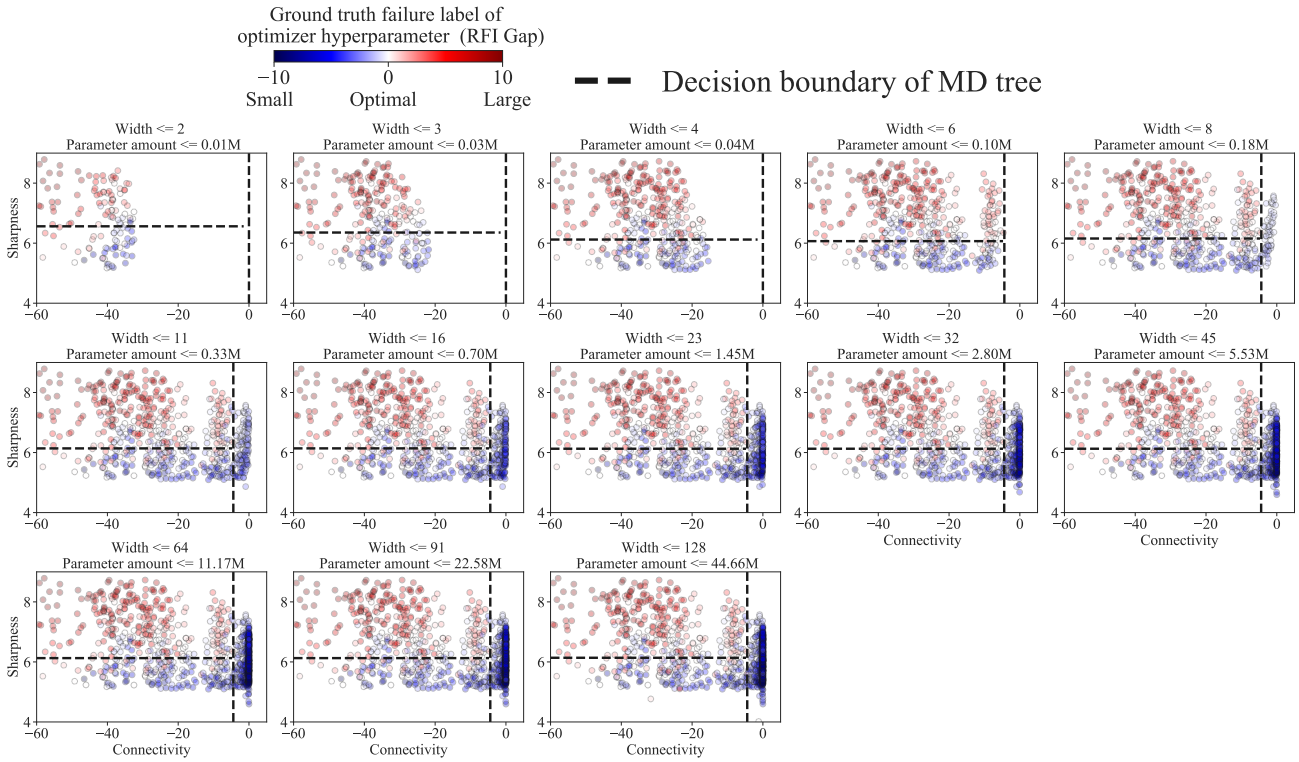
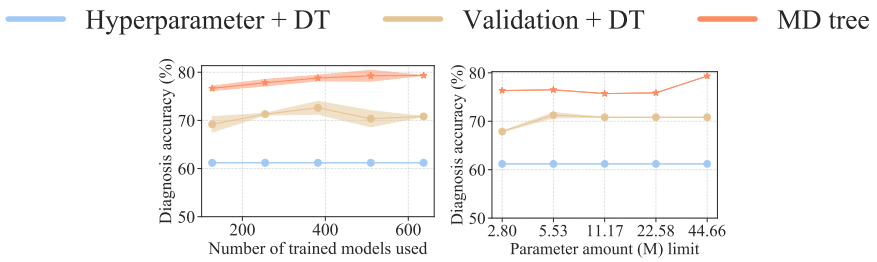


Figure 23. **(Visualization of models in training set in scale (number of model parameters) transfer and decision boundary of `MD tree` in Q1 task).** Each subfigure presents the models under a specific parameter amount limit. The x-axis presents the connectivity and the y-axis presents the sharpness.

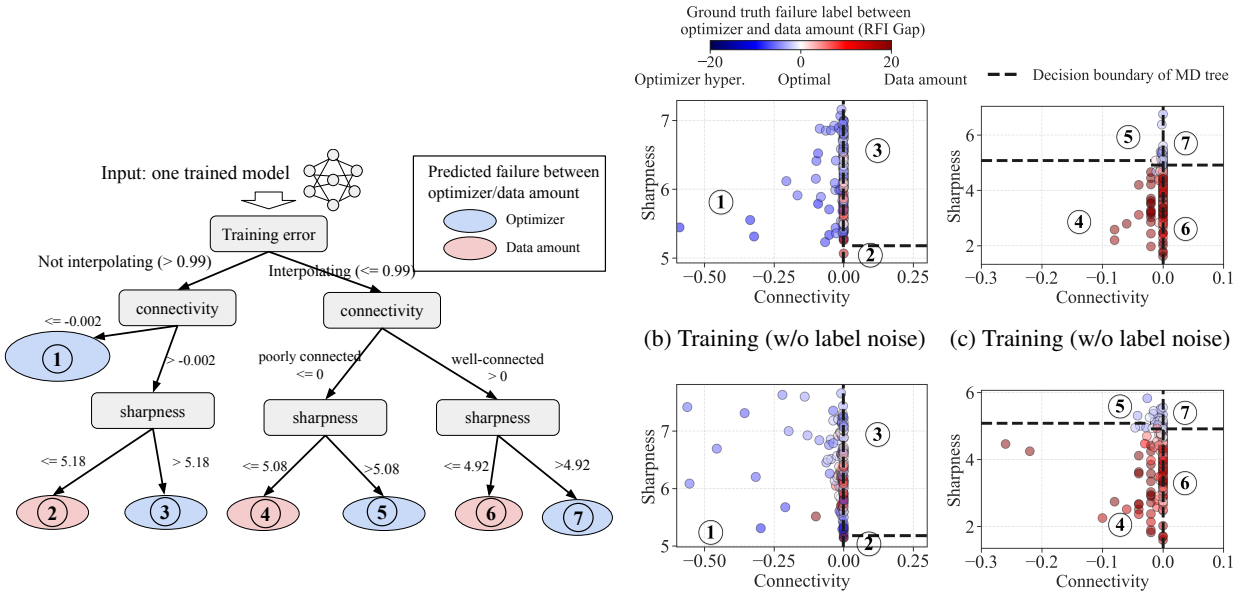
G. A Variant of Q2: Which Hyperparameter (Data amount or Optimizer) Leads to failure



(a) Dataset transfer (b) Scale transfer

Figure 24. **(Comparing `MD tree` to baseline methods in Q2 task in dataset and scale transfer).** y-axis indicates the diagnosis accuracy. (a) x-axis indicates the number of pre-trained models used for building the training set. (b) x-axis indicates the maximum number of parameters of trained models in the training set for fitting the classifier.

We investigate a variation of Q2, aiming to identify the failure source between data amount and optimizer settings. The specifics of the dataset and the transfer setup are outlined in Section C. Diagnosis accuracy, presented in Figure 24, indicates that `MD tree` exceeds validation-based methods by a 10% accuracy margin. Moreover, `MD tree` offers clear visual insights in Figure 25, showing that models with lower sharpness typically suffer from insufficient data, whereas those with higher sharpness face optimizer issues. A case study in Figure 26 contrasts `MD tree` with validation-based approaches, demonstrating `MD tree`'s ability to differentiate between two distinct failure sources that both result in overfitting.



(a) MD tree for Q2

(b) Training (w/o label noise) (c) Training (w/o label noise)

(d) Test (w/ label noise) (e) Test (w/ label noise)

Figure 25. **(Visualizing MD tree and its diagnosis results for a variant of Q2 (determining models' failure source on optimizer hyperparameter or data amount)).** *Left*: structure of the tree defined in Section 4.2. The color of the leaf node indicates the predicted class. The threshold values are learned from the training set. *Right*: each circle represents one model configuration, the color represents the ground truth label RFI gap G, the blue color means the failure source is the optimizer, while red means the data amount. The black dashed line indicates the decision boundary of `MD tree`. The samples in 25(b) and those in 25(c) are separated by training error. The same applies to 25(d) and 25(e).

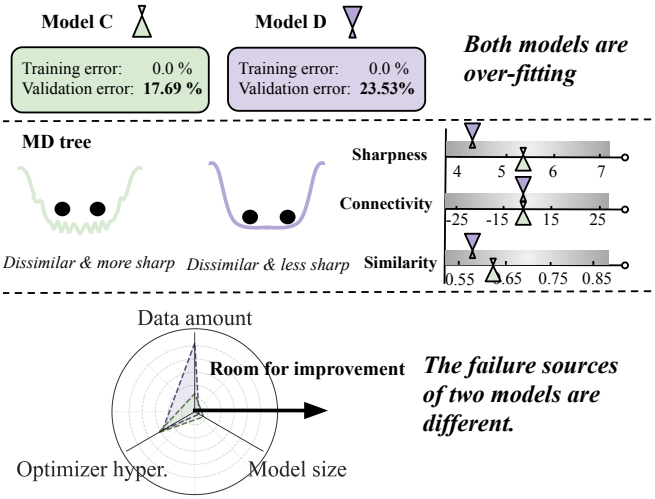
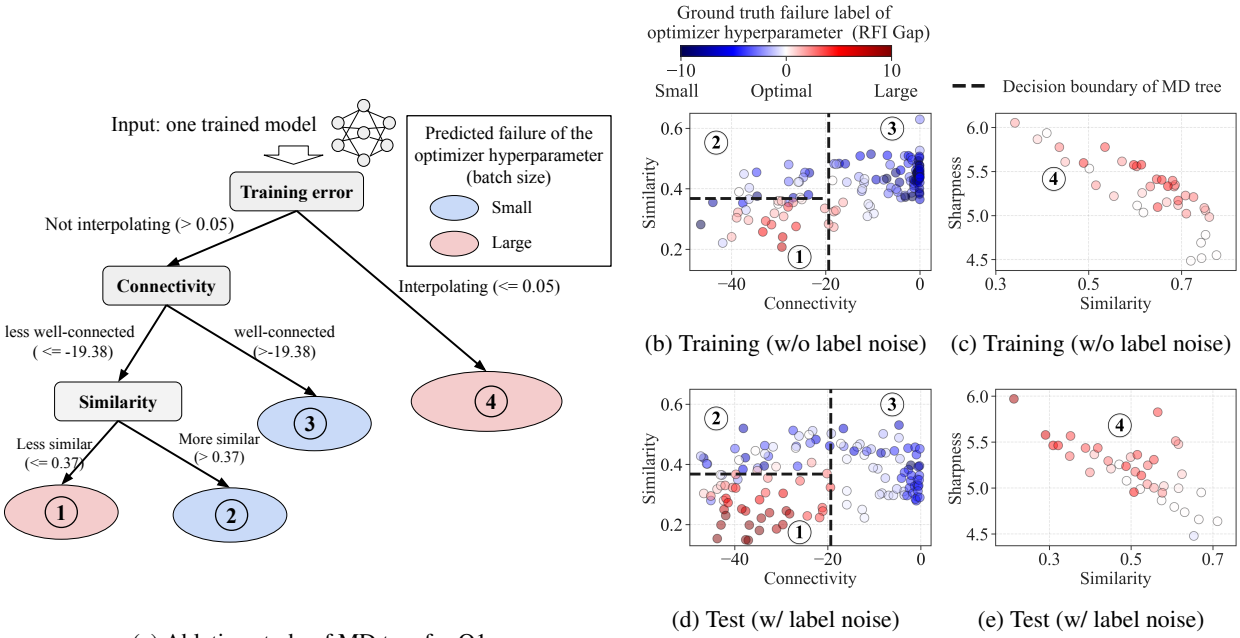


Figure 26. **(Case Study: `MD tree` vs. validation-based method for Q2).** *Top*: Validation metrics provide limited diagnosis: both models are overfitting. *Middle*: Loss landscape metrics further distinguish the two models: Model C has sharper local minima, while Model D has lower similarity. *Bottom*: Model C has an optimizer issue, while Model D's problem is due to insufficient data.



(a) Ablation study of MD tree for Q1

(b) Training (w/o label noise)

(c) Training (w/o label noise)

(d) Test (w/ label noise)

(e) Test (w/ label noise)

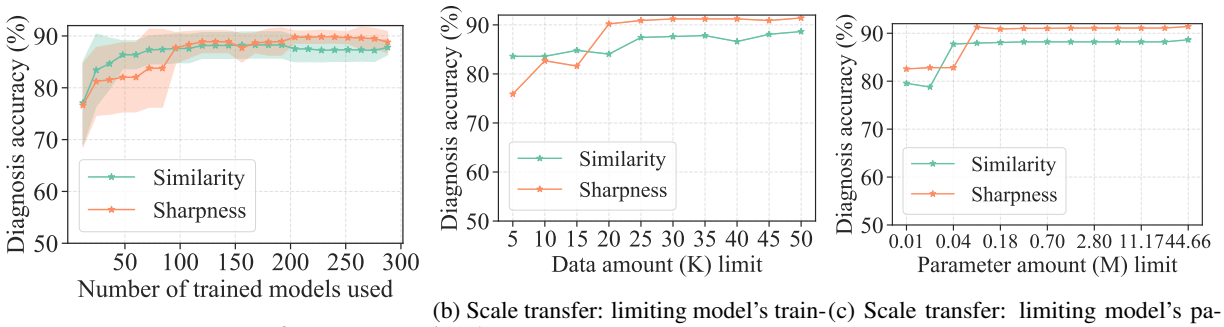
Figure 27. **(Ablation study of `MD tree` structure by using similarity instead of sharpness for Q1).** *Left*: structure of the tree. The color of the leaf node indicates the predicted class. The threshold values are learned from the training set. *Right*: The first row represents training samples, and the second row represents test samples. Each colored circle represents one sample (which is one pre-trained model), and the color represents the ground-truth label: blue means the hyperparameter is too large, while red means small. The black dashed line indicates the decision boundary of `MD tree`. Each numbered regime on the right corresponds to the leaf node with the same number on the tree. The samples in 27(b) and those in 27(c) are separated by training error. The same applies to 27(d) and 27(e).

H. Ablation Study on Structure of `MD tree`

In this section, we present an ablation study that uses a similarity metric for splitting nodes at deeper levels of the tree, as opposed to employing sharpness as outlined in the main paper, for diagnostic task Q1. We demonstrate that similarity, much like sharpness, is a useful metric for constructing the decision tree, offering an effective alternative for node splitting.

In the main paper, we mainly use training error $\mathcal{E}_{\text{tr}}$, connectivity $\mathcal{C}$, and sharpness $\mathcal{H}_t$ to construct the tree structure referred to as `MD tree`. The tree structure to solve Q1 is shown in Figure 4(a). In this ablation study, we replace the sharpness metric at the leftmost deep internal node with the similarity metric $\mathcal{S}$. The resulting tree structure is shown in Figure 27(a). We follow the same experimental setup in Section C. From Figure 27(b) and Figure 27(d), we can see that the similarity metric separates the left space into regime ① and regime ②. In each regime, models have the same failure source. This achieves a similar effect as sharpness in Figure 4(b) and 4(d). This means that the alternative structure of `MD tree` using similarity also provides precise and interpretable classification results.

In Figure 28, we compare two tree structures: one using similarity and the other using sharpness, across dataset transfer and scale transfer for Q1. Both structures achieve comparably high diagnosis accuracy in all three settings. The structure employing sharpness demonstrates marginally superior performance in scale transfer scenarios. Based on these findings, we conclude that both sharpness and similarity are valuable metrics for node splitting at deeper levels of the tree, offering practical utility in constructing diagnostic tree structures.



(a) Dataset transfer

(b) Scale transfer: limiting model's training data amount

(c) Scale transfer: limiting model's parameter amount

Figure 28. **(Varying `MD tree` structure by using similarity or sharpness in splitting the node in the deepest level for task Q1).** y-axis indicates the diagnosis accuracy. (a) x-axis indicates the number of pre-trained models used for building the training set. (b) x-axis indicates the maximum amount of training (image) data for training models in the training set for fitting the classifier. (c) x-axis indicates the maximum number of parameters of the models in the training set.