

LEAD: Exploring Logit Space Evolution for Model Selection

Zixuan Hu¹ Xiaotong Li¹ Shixiang Tang³ Jun Liu⁴ Yichun Hu¹ Ling-Yu Duan^{1,2*}

¹ School of Computer Science, Peking University, Beijing, China, ² Peng Cheng Laboratory, Shenzhen, China,

³ The Chinese University of Hong Kong, Hongkong, China, ⁴ Singapore University of Technology and Design, Singapore

{hzxuan, hycc, lingyu}@pku.edu.cn, lixiaotong@stu.pku.edu.cn,

shixiangtang@cuhk.edu.hk, jun.liu@sutd.edu.sg

Abstract

The remarkable success of “pretrain-then-finetune” paradigm has led to a proliferation of available pre-trained models for vision tasks. This surge presents a significant challenge in efficiently choosing the most suitable pre-trained models for downstream tasks. The critical aspect of this challenge lies in effectively predicting the model transferability by considering the underlying fine-tuning dynamics. Existing methods often model fine-tuning dynamics in feature space with linear transformations, which do not precisely align with the fine-tuning objective and fail to grasp the essential nonlinearity from optimization. To this end, we present LEAD, a finetuning-aligned approach based on the network output of logits. LEAD proposes a theoretical framework to model the optimization process and derives an ordinary differential equation (ODE) to depict the nonlinear evolution toward the final logit state. Additionally, we design a class-aware decomposition method to consider the varying evolution dynamics across classes and further ensure practical applicability. Integrating the closely aligned optimization objective and nonlinear modeling capabilities derived from the differential equation, our method offers a concise solution to effectively bridge the optimization gap in a single step, bypassing the lengthy fine-tuning process. The comprehensive experiments on 24 supervised and self-supervised pre-trained models across 10 downstream datasets demonstrate impressive performances and showcase its broad adaptability even in low-data scenarios.

1. Introduction

The “pretrain-then-finetune” learning paradigm has exhibited remarkable success in a wide range of computer vision tasks [6, 28], leading to an increasing number of diverse pre-trained models open-sourced to the computer vision community. This surge poses a significant challenge in selecting the optimal pre-trained model for a downstream task

*Corresponding Author.

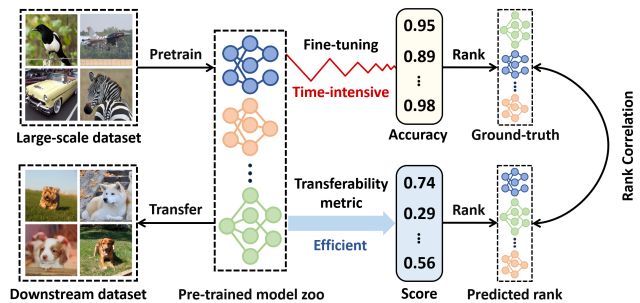


Figure 1. This diagram illustrates the task of model selection, which aims to rank pre-trained models for choosing the optimal one. The ground-truth ranking sequence is derived from the models’ accuracy after fine-tuning. However, due to the time-intensive nature of this process, it requires to design efficient transferability metrics for accurate ranking prediction. The performance of this metric is evaluated using rank correlation, *e.g.*, weighted Kendall’s tau [42], to compare the predicted and ground-truth rankings.

from a vast model zoo. Considering that the brute-force solution of fine-tuning all models on downstream tasks may be infeasible in many scenarios due to the substantial computational costs, the model selection task emerges [13, 47], which ranks the transferability of pre-trained models to the given task without incurring much cost, as shown in Fig. 1.

Predicting model transferability aims to efficiently estimate the final state after fine-tuning on the downstream task, without network optimization by backward propagation. Prior studies only depend on specific properties of the initial state (*e.g.*, representation similarity [13, 14], conditional distribution [33, 47]), to serve as transferability indicators. Recent studies reveal that modeling the dynamic process of fine-tuning has become the bottleneck to design more high-quality transferability metrics. Following this philosophy, recent works propose to simulate fine-tuning dynamics on features through multiple stages of linear transformations [32, 41], achieving some promising results.

Despite recent progress in this field, there are still two

considerable gaps between existing modeled fine-tuning dynamics and the real ones: (1) Insufficient objective alignment. The designed objectives in prior works do not seamlessly align with fine-tuning, as they often focus more on enhancing class separability via feature manipulation, rather than the classification optimization for both the feature and output spaces. (2) Disregarding nonlinear modeling. Existing methods relying solely on linear transformations cannot well capture the intricate nonlinear changes in the feature space during optimization. Such constraints contribute to growing deviations in dynamic modeling and result in unreliable predictions of the final state.

Delving deeper into the optimization process, we focus on modeling the dynamic changes within the output space (*i.e.*, logits for classification), which not only more closely relates to the fine-tuning objective but also embeds the overall nonlinear dynamics from the network optimization during fine-tuning. Building upon this foundation, we introduce an insightful framework to depict the trajectory of logits toward their final state in a dynamical modeling system. To capture and understand this evolution process of the logit space, the framework provides a discrete iterative optimization equation via gradient analysis and captures nonlinear logit space changes during optimization.

In contrast to the extensive iterations required in fine-tuning optimization, we further extend the framework to propose a new transferability metric “Logits Evolution Analysis through Dynamical Equation (LEAD)”, to seize the final evolutionary state in a single step. We perform a mathematical derivation to convert the discrete process into a continuous differential equation and simplify it into an ordinary differential equation (ODE) by leveraging the classical analysis tool, Neural Tangent Kernel (NTK) [26]. This ODE concisely models the nonlinear evolution in logit space through its closed-form solution, allowing for real-time estimation of logits. Additionally, we introduce a class-aware decomposition method to enhance prediction accuracy and break the barrier of theoretical assumption of NTK. As a result, our method provides a direct estimation of the evolution by incorporating both the initial model state and the impact of fine-tuning evolution, serving as an effective indicator assessing model transferability.

The contributions of this work are three-fold. (1) To the best of our knowledge, we are the first to explore model transferability by modeling the evolution process in the logit space. (2) We propose LEAD, a novel transferability metric with two key components: a theoretical framework providing a closed-form solution to depict the logits evolution towards the final state, and a class-aware decomposition method to guarantee the practical applicability of theoretical findings and improve predictive precision. (3) We validate the effectiveness and generalizability of our method through experiments on 24 different pre-trained

models across 10 downstream datasets, achieving SOTA results with an average of 17% gain in the rank correlation. In addition, our method can be naturally extended to low-data scenarios while maintaining competitive performance.

2. Related Work

2.1. Transferability Metric

The challenge of efficiently ranking pre-trained models has captured much interest of researchers, leading to the development of various metrics to evaluate the model transferability [2, 3, 12, 18, 36, 38, 47, 52]. For example, RSA [13], DDS [14] and GBC [38] are methods based on the distance measurements of the feature space. RSA and DDS assess transferability by evaluating the representation similarity between pre-trained and downstream datasets. GBC employs the Bhattacharyya coefficient to measure class separability in the feature space. \mathcal{N} leep [33] and Logme [52] estimate transferability by analyzing the conditional distribution, with \mathcal{N} leep utilizing Gaussian mixture model and Logme employing Bayesian optimization. However, these methods treat network outputs as static and overlook the changes introduced by the training process, leading to less effectiveness in scenes with larger fine-tuning dynamics.

Recently, SFDA [41] and PED [32] propose to simulate changes in the feature space, respectively projecting the features into a Fisher space and utilizing a physically-inspired model to enhance the class separability, achieving impacts similar to fine-tuning. While these methods achieve some improvements, they solely concentrate on the feature space, disregarding the dynamic changes introduced by the classifier, which is jointly optimized with the backbone. Additionally, they utilize linear transformations to simulate the intricate variations in fine-tuning, resulting in a significant limitation in simulation precision. Instead, our LEAD employs the ordinary differential equation to analyze logit dynamics, maintaining a closely aligned objective and simulating the nonlinear changes in fine-tuning.

2.2. Neural Tangent Kernel

Studying the behavior of neural network optimization processes has been an enduring challenge due to their non-linearity and nonconvexity. Neural Tangent Kernel (NTK) [20, 26], serving as a theoretical tool to characterize the optimization process, has attracted much research attention. Some studies have explored its theoretical properties and applications. Notable works include the NTK’s property of remaining constant under infinite network width [26], the explanation of the instability of training [48], the analysis of its bias and variance under finite width [20], and its acceleration algorithm [35]. While NTK can analyze network trainability [26, 35], it is constrained by the requirement of infinite width and neglects the consideration of initial logits, resulting in the inability to predict the changes starting

from the initial state for a finite-width network. To tackle these issues, we innovatively propose an analysis framework leveraging the property of NTK, suitable for model selection problems to bridge the gap between the initial and final states of finite-width networks.

3. Methodology

In this section, we first present the problem setup for the task of model selection. Then we analyze training mechanisms to model the fine-tuning optimization in a discrete process. Finally, we propose a transferability metric that converts the discrete process into a continuous dynamical equation with a concise solution to directly seize the evolution destination of logits, and feed updated logits into Cross-Entropy loss to assess transferability, as shown in Fig. 2.

3.1. Problem Setup

Consider M pre-trained models $\{\Psi_m\}_{m=1}^M$ and a downstream dataset $\mathcal{T} = \{x_n, y_n\}_{n=1}^N$ with K classes, where model Ψ consists of its backbone f which outputs the encoded feature, N is the number of images in the downstream task. The purpose of model selection is to design an efficient algorithm to predict the ranking of the performance of M pre-trained models after fine-tuning.

Specifically, given the ground-truth performance rankings $\{G_m\}_{m=1}^M$ of pre-trained models after fine-tuning and the predicted ranking $\{P_m\}_{m=1}^M$ by model selection algorithms, we can leverage the commonly-used rank-based correlation, *i.e.*, the weighted Kendall's τ_w , to evaluate the effectiveness of the proposed model selection methods, *i.e.*,

$$\tau_w = \frac{1}{\sum_{i \neq j} w_{ij}} \cdot \sum_{i \neq j} w_{ij} \text{sign}(G_i - G_j) \cdot \text{sign}(P_i - P_j), \quad (1)$$

where $w_{ij} = \frac{1}{i+j}$ and $\text{sign}(\cdot)$ is the signum function. A larger τ_w indicates more consistencies between the predicted ranking and the ground truth, which means a better model selection algorithm.

3.2. Revisiting the Fine-tuning Optimization

When a pre-trained model Ψ transfers to a downstream task $\mathcal{T} := \{\mathcal{X}, \mathcal{Y}\}$, it undergoes extensive optimization for evolving from its initial state (before fine-tuning) to the final state (after fine-tuning), resulting in complex changes in both feature space and output space. Since the transferability metric is designed to evaluate the final state without fine-tuning, it is essential and challenging to model the optimization process without lengthy computations, bridging the gap between the initial and the final state. To gain insights into the underlying optimization mechanisms, we conduct an analysis to compare the prior arts with fine-tuning to revisit the evolution process toward the final state.

Recent works SFDA [41] and PED [32] (called simulation methods) aim to approximate fine-tuning procedures

by simulating feature dynamics. They design an objective function U to assess the class separability of the feature space. Then, they apply linear transformations to increase U and feed modified features into a classifier to update logits, as shown below:

$$A, b = \arg \max_{A, b} U(A \cdot f(\mathcal{X}) + b, \mathcal{Y}), \quad (2)$$

$$\Delta \mathcal{F}(\mathcal{X}) = \mathcal{C}(A \cdot f(\mathcal{X}) + b) - \mathcal{C}(f(\mathcal{X})),$$

where A, b denote linear transformation parameters, \mathcal{C} denotes a machine learning classifier and \mathcal{F} denotes the function that outputs logits. As a result of the optimization in Eq. (2), the objective of enhancing class separability is achieved. However, fine-tuning directly minimizes the gap between predicted logits and labels and performs gradient descent on parameters to update the output logits as follows:

$$\begin{aligned} \Delta \theta &= -\eta \cdot \frac{d\mathcal{L}(\mathcal{F}(\mathcal{X}, \theta), \mathcal{Y})}{d\theta}, \\ \Delta \mathcal{F}(\mathcal{X}) &= \mathcal{F}(\mathcal{X}, \theta + \Delta \theta) - \mathcal{F}(\mathcal{X}, \theta) \approx \Delta \theta \cdot \frac{d\mathcal{F}}{d\theta} \quad (3) \\ &= -\eta \cdot \frac{d\mathcal{L}}{d\theta} \cdot \frac{d\mathcal{F}}{d\theta} = -\eta \cdot \frac{d\mathcal{L}}{d\mathcal{F}} \left(\frac{d\mathcal{F}}{d\theta} \cdot \frac{d\mathcal{F}}{d\theta} \right), \end{aligned}$$

where η denotes the learning rate, \mathcal{L} denotes loss function, and θ denotes parameters of \mathcal{F} . Eq. (3) provides the variation of logits within one step, capturing the nonlinear dynamics based on gradients. When comparing the optimization outlined in Eq. (2) and Eq. (3), it becomes evident that simulation methods differ from fine-tuning in terms of objectives and optimization approaches.

Despite preliminary exploration, such simulation methods come with severe limitations due to these inconsistencies, leading to deviations in predicting the final state, as illustrated in Fig. 3. Specifically, simulation methods aim to enhance class separability within the feature space, whereas fine-tuning drives logits closer to labels. The insufficient objective alignment introduces distinct inductive bias compared to fine-tuning, resulting in notable variations in modeling dynamics [27, 34]. Moreover, due to the nonlinear nature of neural networks and complex parameter interactions, linear approximations in simulation methods often fail to accurately capture the dynamic changes [21]. Such limitations hinder the effective modeling of nonlinearity, further constraining modeling precision.

3.3. Modeling Logits Evolution Process through Dynamical Equation

To address the aforementioned problems, our study delves deeper into the optimization analysis from Eq. (3). In fact, unlike features, the logits derived from the optimization output naturally align with the fine-tuning objectives and reflect the inherent nonlinearity from gradient descent. Therefore,

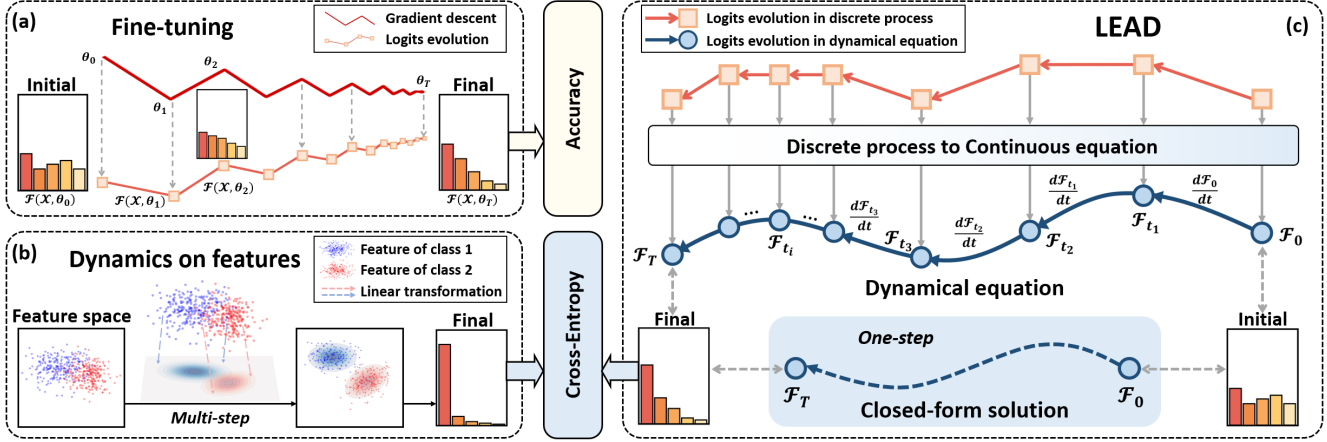


Figure 2. Comparison of different approaches for modeling the evolution process. (a) Fine-tuning iteratively updates parameters via gradient descent, revealing authentic logits evolution trajectory and classification accuracy for ground-truth ranking. (b) Recent works (e.g., SFDA [41], PED [32]) perform linear transformations in feature space to enhance class separability, updating logits with modified features. The ranking score is determined by Cross-Entropy loss on updated logits. While these approaches simulate fine-tuning impact on features, they often suffer huge deviations from the authentic results. (c) Our method, LEAD, transforms the discrete logit evolution during fine-tuning into a gradient-based dynamical equation to capture nonlinear changes and can efficiently predict the final state through the closed-form solution. It establishes an analysis framework linking the initial and final states, enabling the precise prediction in one step.

it provides a particularly relevant entry point to analyze dynamical changes during fine-tuning and assess the model transferability through its evolution process.

While the dynamical process of logit space analyzed by Eq. (3) remains consistent with fine-tuning, there still lacks an efficient approach for capturing logits evolution, due to the discrete process of iterating over thousands of steps by traditional optimization. To this end, we propose a new transferability metric called “Logits Evolution Analysis through Dynamical equation (LEAD)”, that bridges the evolution process from the initial logit space towards the final state. In contrast to extensive optimization iterations, our method proposes a dynamical analysis framework for analyzing the nonlinear effects within the logit space, providing an efficient prediction to capture the final state with a concise solution in a single step.

Theoretical Background. In our LEAD, we innovatively introduce the Neural Tangent Kernel (NTK) to simplify the dynamical equation. Here, we provide a brief introduction to the definition of the NTK kernel function and the constant-preserving property that we utilize.

$$\hat{\Phi}_{\theta_t}(\mathcal{X}, \mathcal{X}) \triangleq \nabla_{\theta_t} \mathcal{F}(\mathcal{X}; \theta_t) \nabla_{\theta_t} \mathcal{F}(\mathcal{X}; \theta_t)^\top, \quad (4)$$

$$\Phi(\mathcal{X}, \mathcal{X}) \triangleq \lim_{l \rightarrow \infty} \hat{\Phi}_{\theta_t}(\mathcal{X}, \mathcal{X}),$$

where ∇ denotes the Laplace Operator, and l denotes the root mean square of the widths of all layers in the network. The kernel function $\hat{\Phi}_{\theta_t}$ remains invariant throughout the training process as l approaches infinity with an approximation error of $O(\frac{1}{l^2})$ [20, 26]. For simplicity, in the remainder of this paper, we denote functions at time t as:

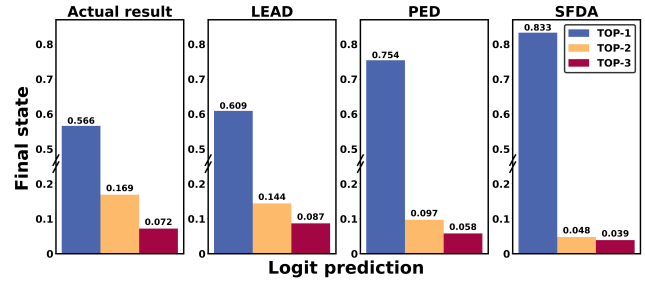


Figure 3. Comparison of the average prediction of final state logits across various methods on VOC2007 [16], where the actual result stems from fine-tuning. Closer to actual results indicates more accurate modeling of the final state. Simulation methods (SFDA, PED) capture the correct trend but exhibit a large deviation from the actual results, affecting the accuracy of model ranking.

$$\mathcal{F}_t(\cdot) \triangleq \mathcal{F}(\cdot, \theta_t), \hat{\Phi}_t(\cdot) \triangleq \hat{\Phi}_{\theta_t}(\mathcal{X}, \mathcal{X}), \Phi \triangleq \Phi(\mathcal{X}, \mathcal{X}).$$

Dynamical Equation and Closed-form Solution. To bypass the complexity of iterative computations, we extend the analysis in Eq. (3) and theoretically derive a dynamical equation that is efficient for analysis. Specifically, we seek for a mathematical derivation through the theory of limit approximation to convert the discrete process into a continuous differential equation, enabling us to track the trajectory of logits in the solution space of the equation. Furthermore, we leverage the property of the NTK in Eq. (4) to simplify the equation into an Ordinary Differential Equation (ODE) with an initial value condition. Due to space constraints, we directly provide the equation and the detailed proof can be found in the Appendix.

$$\frac{d\mathcal{F}_t}{dt} = -\eta \cdot \Phi \cdot \frac{d\mathcal{L}_t}{d\mathcal{F}_t}, \quad \mathcal{F}_0 = \log_{init}, \quad (5)$$

where \log_{init} is the initial state of logits obtained through feeding features and labels of downstream datasets to a classification algorithm. Compared to the discrete process, our proposed ODE depicts the continuous evolution of the fine-tuning dynamics, allowing for a more flexible modeling approach. It transforms discrete gradient effects into equivalent differential effects in continuous time, preserving the capability for modeling nonlinear changes and providing precise prediction of the evolution process. Furthermore, unlike employing numerical methods as the ODEsolver, we can directly obtain its closed-form solution through integration over time dimension, allowing us to bypass the lengthy evolution process and reach the optimization destination in a single step, as shown below:

$$\mathbb{E}(\mathcal{F}_t(\mathcal{X})) = (\mathbb{I} - e^{-\eta\Phi \cdot t}) \mathcal{Y} + e^{-\eta\Phi \cdot t} \cdot \log_{init}. \quad (6)$$

As depicted in Eq. (6), the solution is derived by interpolating between the one-hot label and the initial observation of logits, with the interpolation coefficients determined by the NTK matrix. It comprises two crucial elements: \log_{init} , representing the value of logits before optimization and determining the start of the evolution process; and Φ , indicating the convergence rate to the label and determining the trajectory of logit evolution. Our proposed framework efficiently models the evolution process, enabling a prediction of real-time changes.

Class-aware Decomposition. While Eq. (6) provides the theoretical result for the final state, computing the value of Φ under finite width remains challenging. Prior studies [20, 53] indicate that, even with finite width, the convergence rate can be assessed by using the eigenvalues of the initial state NTK matrix $\hat{\Phi}_0$. Inspired by this empirical finding, we further conduct a decomposition approach to bridge theory and practice, utilizing the eigenvalue decomposition on $\hat{\Phi}_0$ to replace the convergence rate determined by Φ in Eq. (6) with eigenvalues.

Due to the network exhibiting varying classification abilities across different classes, their convergence behavior and evolution dynamics also differ during the fine-tuning process. To enhance the modeling of behaviors in different classes, we separately compute NTK for each class, rather than mixing them together, to decouple the respective optimization effect. Specifically, we extract n samples of each class to calculate their respective NTK matrices and perform decomposition to obtain eigenvalues. The average eigenvalue of each class serves as the unified parameter to determine the interpolation coefficient for this class:

$$\mathbb{E}(\mathcal{F}_t(x)) = \left(\mathbb{I} - e^{-\eta\bar{\lambda} \cdot t}\right) y + e^{-\eta\bar{\lambda} \cdot t} \cdot \log_{init}, \quad (7)$$

where x, y is a given sample and its one-hot label, $\bar{\lambda}$ is the average eigenvalue of the NTK matrix associated with the class of x . Building upon the conclusion of Eq. (6), Eq.

(7) avoids the demand of infinite width, enabling practical computability of the prediction results. As shown in Fig. 2, we employ the results of Eq. (7) to efficiently obtain predictions for the final state logits. Subsequently, following the common practice [32, 33, 41], we feed predictions into the Cross-Entropy loss to obtain the transferability score for model ranking.

Summary. Our method explicitly models the fine-tuning optimization through the theoretical analysis framework and provides a concise dynamical solution to bypass the lengthy optimization process in a single step, which seamlessly integrates the initial state observations and the evolution trajectory towards the final state. Therefore, we can efficiently predict the final logit state through this dynamical solution. Through comparing logit prediction accuracy among different models, it can serve as an effective metric for predicting model transferability.

4. Experiments

To demonstrate the effectiveness and robustness of our approach, we evaluate LEAD on diverse pre-trained models, including both supervised and self-supervised convolutional neural networks. Our evaluation spans 10 widely used classification benchmarks in transfer learning, employing weighted Kendall's τ_w as rank correlation to evaluate the transferability metric. Additionally, we highlight the effectiveness of LEAD in the low-data regime, where only 2/5/10 samples are available per class.

4.1. Implementation Details

Given a pre-trained model Ψ , we append a random-initialized [22] MLP h as a classification head after its backbone f , and combine them as $\mathcal{F} := h \circ f$ to produce logits. To compute the logits in the final state, we only need to derive the NTK matrix of each class and \log_{init} through Eq. (7). We follow the approach in [35] to compute the NTK matrix and employ a robust and efficient machine learning classifier, the multi-class SVM [49], to generate \log_{init} . Due to space constraints, we provide a more detailed explanation and pseudocode in the Appendix.

4.2. Benchmarks

Target Datasets. We employ 10 classification benchmarks commonly utilized in transfer learning research including Caltech-101 [17], Stanford Cars [29], CIFAR10 [30], CIFAR100 [30], DTD [10], Oxford 102 Flowers [37], Food-101 [4], Oxford-IIIT Pets [39], SUN397 [51], and VOC2007 [16]. These datasets encompass abundant characteristics, including backgrounds, textures, and coarse/fine-grained scenes, spanning diverse fields.

Ground Truth Rank. To conduct the ground truth ranking for the model zoo, we follow the approach in [41] and [52]. Specifically, we employ a grid search strategy to determine

Table 1. Performance comparisons of different transferability metrics on the supervised model zoo. We measure the performance with weighted Kendall’s τ_w and a larger τ_w represents a better prediction rank. The best results are in bold and the second-best are in underline.

Method	Reference	Downstream Target Dataset									
		Food	Caltech	Flowers	Cars	CIFAR100	DTD	CIFAR10	Pets	SUN397	VOC2007
PARC [3]	NeurIPS’21	0.363	0.422	0.066	0.135	-0.092	0.536	0.310	0.114	-0.097	0.707
Logme [52]	ICML’21	0.336	0.349	0.425	0.538	0.603	<u>0.651</u>	0.783	0.372	0.440	0.673
\mathcal{N} leep [33]	CVPR’21	0.504	<u>0.678</u>	0.292	0.435	0.686	0.452	0.656	<u>0.686</u>	0.770	0.645
PACTran [12]	ECCV’22	0.709	0.372	0.480	0.151	0.742	0.372	0.757	0.451	0.422	0.536
SFDA [41]	ECCV’22	0.534	0.597	0.421	0.472	0.749	0.421	<u>0.817</u>	0.625	0.562	<u>0.728</u>
GBC [38]	CVPR’22	<u>0.844</u>	0.430	0.348	0.735	<u>0.830</u>	0.480	0.866	0.358	0.574	0.654
ETran [18]	ICCV’23	0.709	0.627	<u>0.534</u>	0.557	0.749	0.605	0.719	0.651	0.577	0.673
PED [32]	ICCV’23	0.573	0.597	0.451	0.472	0.749	0.421	0.770	0.625	0.562	<u>0.728</u>
LEAD	This paper	0.892	0.698	0.786	<u>0.586</u>	0.835	0.689	0.791	0.841	<u>0.609</u>	0.743

the actual performance of each model in the downstream task. This strategy involves trying various learning rates from the set $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ and different weight decay values from the set $\{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}\}$. To ensure reliability, we repeat each experiment 5 times with different random seeds and take the average as the result.

4.3. Evaluation on Supervised Models

Model Zoo. We construct a Model Zoo that includes 12 CNN models supervised pre-trained on ImageNet [11], covering widely-used architectures: ResNet-34 [23], ResNet-50 [23], ResNet-101 [23], ResNet-152 [23], DenseNet-121 [25], DenseNet-161 [25], DenseNet-169 [25], DenseNet-201 [25], MNet-A1 [45], MobileNetV2 [40], GoogleNet [43], and InceptionV3 [44]. The accuracy results of these models after fine-tuning can be found in the Appendix.

Result Analysis. To show the effectiveness of LEAD, we compare it with prior arts, such as Bayesian-based Logme, Separability-based GBC, and simulation methods SFDA and PED. As shown in Tab. 1, simulation methods consider underlying dynamic effects and exhibit promising performances on some datasets. However, as the aforementioned analysis in Sec. 3.2 shows, the bias in the simulation process causes these methods to have unstable improvements, particularly resulting in unsatisfactory performance on Flowers, Food, and Pets. Benefiting from a precise analysis of the evolution, our LEAD yields a performance gain of +0.335, +0.319, +0.216 compared to PED on these three datasets, respectively. Overall, LEAD achieves the best τ_w on 7 target datasets and obtains an average τ_w of 0.747 which is relatively 18% better than the latest SOTA ETran [18]. Our experiments highlight the importance of accurate modeling of dynamics and confirm the effectiveness of our method to model the evolution process through the dynamical equation.

4.4. Evaluation on Self-supervised Models

Model Zoo. Self-supervised learning models (SSL models) have shown their efficacy in transfer learning and often display superior generalization performance compared

to Supervised models. To illustrate the broad applicability of our approach, inspired by [32, 41], we assemble a self-supervised model zoo trained on ImageNet with BYOL [19], Infomin [46], PCLv1 [31], PCLv2 [31], Selav2 [1], InsDis [50], SimCLRv1 [7], SimCLRv2 [8], MoCov1 [24], MoCov2 [9], DeepClusterv2 [5], and SWAV [6].

Result Analysis. Due to differing pre-training and downstream classification objectives, SSL models undergo larger dynamic changes during the fine-tuning process, leading to distinct transferability properties compared to supervised models [15]. As shown in Tab. 1 and 2, methods that disregard dynamics (e.g. PARC, \mathcal{N} leep) perform competitively in the supervised model zoo, but fail on some datasets for SSL models. For example, \mathcal{N} leep and PARC have a negative small rank correlation on VOC2007 (-0.101) and CIFAR100 (-0.136), respectively. To maintain fairness, we refrain from comparing with GBC [38] in the SSL model zoo, which is designed for supervised scenarios. On the contrary, SFDA and PED exhibit better performances due to consideration of training dynamics. Nevertheless, the limitation of the simulating process still hinders their performance. Compared to them, LEAD achieves a consistent improvement on 7 datasets and obtains an average τ_w of 0.746 which is relatively 15% better than PED (0.648). The results validate the effectiveness of LEAD for SSL models.

4.5. Evaluation on Low-data Regime

Experimental Settings. In practical application, privacy and resource constraints may limit our access to the entire downstream dataset. To explore the utilization boundary, we evaluate whether LEAD can provide effective model ranking only using limited samples. We follow three data settings in PACTran [12] with increasing average number of samples per class $\{2, 5, 10\}$ and divide 10 datasets into two groups according to the number of classes: 100+ classes, including Food, Caltech, Flowers, Cars, CIFAR100, and SUN397; 10-99 classes, including DTD, CIFAR10, Pets, and VOC2007. For each setting, we repeatedly sample 5 times for evaluation and take the average as the result.

Result Analysis. As shown in Tab. 3, all methods, in-

Table 2. Performance comparisons on the self-supervised model zoo. The best results are in bold and the second-best are in underline.

Method	Reference	Downstream Target Dataset									
		Food	Caltech	Flowers	Cars	CIFAR100	DTD	CIFAR10	Pets	SUN397	VOC2007
PARC [3]	NeurIPS'21	0.359	0.196	0.622	0.424	-0.136	0.447	0.147	0.496	-0.006	0.618
Logme [52]	ICML'21	0.570	0.051	0.604	0.375	-0.008	0.627	0.295	<u>0.684</u>	0.217	0.158
Nleep [33]	CVPR'21	0.574	0.525	0.534	0.486	0.276	0.641	-0.044	0.792	<u>0.719</u>	-0.101
PACTran [12]	ECCV'22	<u>0.720</u>	<u>0.622</u>	0.601	0.474	0.529	0.614	0.477	0.641	0.638	<u>0.620</u>
SFDA [41]	ECCV'22	0.685	0.523	<u>0.749</u>	0.515	0.548	0.773	0.619	0.586	0.698	0.568
ETran [18]	ICCV'23	0.465	0.405	0.644	0.587	<u>0.650</u>	0.214	0.606	0.338	0.520	0.376
PED [32]	ICCV'23	0.581	0.614	0.777	<u>0.649</u>	0.568	0.907	<u>0.673</u>	0.462	0.661	0.583
LEAD	This paper	0.860	0.780	0.725	0.663	0.776	<u>0.825</u>	0.713	0.629	0.760	0.723

Table 3. Performance comparisons with different N/K , where K is the number of classes, N is the number of samples for computing the metric. We divide 10 benchmarks into two groups according to the number of classes: 100+ classes and 10-99 classes, and evaluate the average of τ_w on the two groups. Additionally, we also provide the average results on the 10 benchmarks, recorded as Avg. The best results are denoted in bold, the second best results are denoted in underline, and - denotes cannot give a valid ranking.

Method	Supervised / Self-supervised pre-trained Model								
	$N/K = 2$			$N/K = 5$			$N/K = 10$		
	100+ classes	10-99 classes	Avg	100+ classes	10-99 classes	Avg	100+ classes	10-99 classes	Avg
GBC	0.17 / 0.32	0.10 / 0.17	0.13 / 0.25	0.45 / 0.40	0.22 / 0.18	0.36 / 0.31	<u>0.59</u> / 0.42	0.42 / 0.22	0.52 / 0.34
PACTran	<u>0.43</u> / 0.49	<u>0.20</u> / 0.39	<u>0.33</u> / <u>0.45</u>	0.42 / <u>0.48</u>	0.31 / <u>0.44</u>	0.38 / <u>0.47</u>	0.56 / <u>0.52</u>	0.48 / 0.46	0.53 / <u>0.50</u>
ETran	0.42 / 0.43	0.12 / <u>0.41</u>	0.29 / 0.42	0.48 / 0.47	<u>0.31</u> / 0.38	<u>0.41</u> / 0.43	0.56 / 0.47	<u>0.56</u> / 0.48	<u>0.56</u> / 0.48
PED	- / 0.36	- / 0.24	- / 0.31	- / 0.43	- / 0.35	- / 0.40	0.30 / 0.50	0.22 / <u>0.56</u>	0.27 / 0.52
LEAD	0.45 / <u>0.48</u>	0.51 / 0.45	0.47 / 0.46	0.48 / 0.49	0.64 / 0.53	0.54 / 0.51	0.65 / 0.53	0.64 / 0.60	0.64 / 0.56

cluding PACTran designed for this scenario, exhibit significant performance declines under low-data settings. For instance, ETran’s performance in the supervised model zoo drops from 0.64 with the entire dataset to 0.29, 0.41, and 0.56 in three low-data settings. Moreover, some methods, such as SFDA and PED fail to provide prediction rankings in some settings. For example, when $N/K = 2, 5$, PED cannot correctly evaluate class separability due to insufficient data, imposing huge changes to completely separate different classes. This leads to all models perfectly classifying all samples, resulting in identical scores that cannot be ranked. In contrast, our LEAD achieves stable results, ensuring $\tau_w \geq 0.45$ even in extreme settings like $N/K = 2$, and consistently outperforms other methods. For example, under $N/K = 10$, its average τ_w in the supervised model zoo reaches 0.64, which is relatively 14% better than the latest SOTA ETran (0.56) and is equivalent to the latest SOTA on the complete dataset, with less than 5% of data size. The results demonstrate the robustness of our method in the face of data limitations.

5. Ablation Study and Efficiency Analysis

In this section, without loss of generality, we conduct ablation studies on Caltech, CIFAR10, and VOC2007, building upon the evaluation of SSL models in Sec. 4.4 for saving time. We conduct separate experiments to analyze the impact of interpolation coefficients, a key component of LEAD, aiming to delve into the influence of hyperparameters and implementation details, gaining insights into key

factors contributing to LEAD’s effectiveness. Additionally, we provide a runtime comparison to verify efficiency, and more ablation study results are provided in the appendix.

5.1. Interpolation Coefficient

The interpolation coefficients are determined by time coefficient t and the NTK matrix. We conduct ablation experiments on these two key factors:

Time Coefficient t . As we model the discrete process as a continuous ODE, the continuous variable t is proportional to the number of updates, determining the progress of the evolution process. In Fig. 4, we evaluate our method under varied time settings, keeping the unit as $1e6$ consistent across all experiments. It is evident that performance improves as the time increases within a specific range. However, if the dynamic effect persists for an extended period (e.g., 10.0), all model predictions converge too closely to the ground-truth label, reducing discrimination for model ranking. Hence, we set the time coefficient as 1 by default.

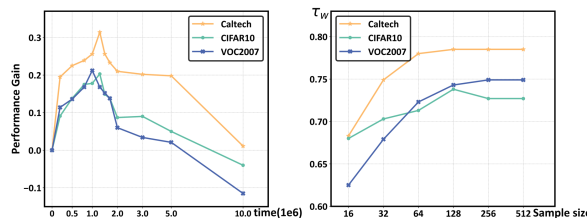


Figure 4. The left picture shows the performance gain in τ_w with respect to different time t and the right shows the performance in different sample sizes for each class NTK.

Sample Size for NTK. To enhance the perception of different dynamic intensities of classes, we propose the class-aware decomposition that employs S samples of each class to obtain NTK matrix and perform spectrum analysis to derive eigenvalues. In Fig. 4, we conduct ablation experiments to explore the effect of varying S . We observe that performance generally improves as S increases, and gradually reaches stability. This demonstrates that when the sample size reaches a certain size, eigenvalues reliably capture differences in dynamic intensity, offering a refined analysis result of the dynamics. Therefore, we set $S = 64$ to ensure the method’s efficiency and stability in all experiments.

5.2. Running Time Comparison

In experiment results, we have demonstrated the effectiveness of LEAD in various scenarios. In this section, we highlight the computational efficiency of LEAD. With the concise solution in Eq. (7), LEAD predicts the final state only requiring the NTK matrix and log_{init} . As shown in Tab. 4, the running time of LEAD is about half that of simulation methods (45s compared to 92s), ranking second after GBC which displays unstable performance. Notably, LEAD takes 3s to compute the NTK, while feature extraction takes 420s, resulting in only a marginal 0.7% increase in GPU time. The time to compute log_{init} is 42s CPU time, indicating significant potential for exploring faster classification algorithms to further reduce the overall time.

Table 4. The comparisons of average running time on CIFAR10.

Metrics	PARC	N/LEEP	PACTran	SFDA	GBC	PED	Ours
Running Time	111s	1430s	77s	92s	26s	97s	45s

6. Visualization and Analysis

In this section, we validate our approach through some visualizations and these results clearly illustrate how our model effectively addresses the model selection problem.

6.1. Logit Prediction

We have conducted a comparison of the predicted logits distribution on VOC2007 using different methods, as illustrated in Fig. 3. Notably, predictions obtained by LEAD closely match the authentic distribution after fine-tuning. To provide further quantitative analysis, we visualize the average bias in predicted logits in Fig. 5. Across three datasets, our method exhibits a substantial reduction compared to simulation methods, achieving nearly a 50% overall reduction. This visualization highlights the precision of our dynamical equation-based approach in capturing logit evolution for more accurate ranking sequences.

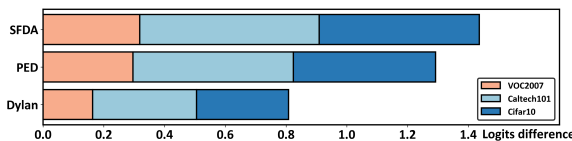


Figure 5. Comparison of the average difference between logits predicted by different methods and logits after fine-tuning. The difference is calculated utilizing the Euclidean distance.

6.2. Rank Variation

By employing our method, we can refine models that have poor initial observation but show better progress during training. To assess the effectiveness of our approach, we conduct a visualization comparing the model rankings between the initial observations and our predicted final state, as shown in Fig. 6. The results clearly illustrate a substantial enhancement in the calibration of model rankings when using our refined observations, with the optimal model now positioned closely to the reference line in the figure. Notably, models starting from an initially unfavorable point can be rapidly elevated to achieve a superior ranking, highlighting the effectiveness of our methodology.

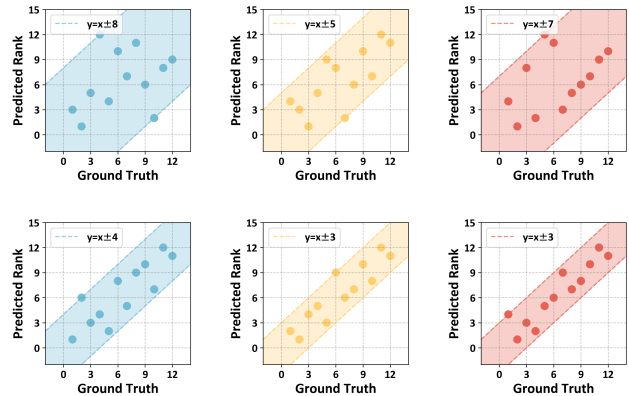


Figure 6. The variation in model ranking, utilizing initial states (first row) and predicted final states of LEAD (second row). Results span three datasets, from left to right: CIFAR10, Caltech, and VOC2007. Shaded areas reflect the range of the ranking error.

7. Conclusions and Future Work

In this work, we propose a dynamical equation designed to capture the evolution process within the logit space, offering an efficient solution for assessing model transferability. We construct a theoretical analysis framework and further devise a more practical and precise solution. And we demonstrate its consistent effectiveness across various scenarios including supervised and self-supervised, full-data and low-data settings. In future work, we intend to expand the current analysis framework beyond classification to cover a broader range of downstream tasks. We hope that our work can enhance the understanding of optimization mechanisms in deep neural networks and shed light on other fields, such as the design of optimization algorithms.

Acknowledgement

This work was supported by the National Natural Science Foundation of China under Grant 62088102, in part by the PKU-NTU Joint Research Institute (JRI) sponsored by a donation from the Ng Teng Fong Charitable Foundation and in part by AI Joint Lab of Future Urban Infrastructure sponsored by Fuzhou Chengtuo New Infrastructure Group and Boyun Vision Co. Ltd.

References

- [1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv*, 2019. [6](#)
- [2] Yajie Bao, Yang Li, Shao-Lun Huang, Lin Zhang, Lizhong Zheng, Amir Zamir, and Leonidas Guibas. An information-theoretic approach to transferability in task transfer learning. In *ICIP*, pages 2309–2313, 2019. [2](#)
- [3] Daniel Bolya, Rohit Mittapalli, and Judy Hoffman. Scalable diverse model selection for accessible transfer learning. In *NeurIPS*, pages 19301–19312, 2021. [2, 6, 7](#)
- [4] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101—mining discriminative components with random forests. In *ECCV*, 2014. [5](#)
- [5] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, pages 132–149, 2018. [6](#)
- [6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *NeurIPS*, pages 9912–9924, 2020. [1, 6](#)
- [7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. [6](#)
- [8] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, pages 22243–22255, 2020. [6](#)
- [9] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv*, 2020. [6](#)
- [10] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, pages 3606–3613, 2014. [5](#)
- [11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. [6](#)
- [12] Nan Ding, Xi Chen, Tomer Levinboim, Soravit Changpinyo, and Radu Soricut. Pactran: Pac-bayesian metrics for estimating the transferability of pretrained models to classification tasks. In *ECCV*, pages 252–268. Springer, 2022. [2, 6, 7](#)
- [13] Kshitij Dwivedi and Gemma Roig. Representation similarity analysis for efficient task taxonomy & transfer learning. In *CVPR*, pages 12387–12396, 2019. [1, 2](#)
- [14] Kshitij Dwivedi, Jiahui Huang, Radoslaw Martin Cichy, and Gemma Roig. Duality diagram similarity: a generic framework for initialization selection in task transfer learning. In *ECCV*, pages 497–513, 2020. [1, 2](#)
- [15] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *CVPR*, pages 5414–5423, 2021. [6](#)
- [16] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88:303–338, 2010. [4, 5](#)
- [17] Li Fei-Fei. Learning generative visual models from few training examples. In *CVPR workshop*, 2004. [5](#)
- [18] Mohsen Gholami, Mohammad Akbari, Xinglu Wang, Behnam Kamranian, and Yong Zhang. Etran: Energy-based transferability estimation. In *ICCV*, pages 18613–18622, 2023. [2, 6, 7](#)
- [19] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent—a new approach to self-supervised learning. In *NeurIPS*, pages 21271–21284, 2020. [6](#)
- [20] Boris Hanin and Mihai Nica. Finite depth and width corrections to the neural tangent kernel. In *ICLR*, 2020. [2, 4, 5](#)
- [21] Hangfeng He and Weijie Su. The local elasticity of neural networks. In *ICLR*, 2020. [3](#)
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015. [5](#)
- [23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. [6](#)
- [24] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020. [6](#)
- [25] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017. [6](#)
- [26] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018. [2, 4](#)
- [27] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, pages 7482–7491, 2018. [3](#)
- [28] Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *CVPR*, pages 2656–2666, 2018. [1](#)
- [29] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013. [5](#)
- [30] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. [5](#)
- [31] Junnan Li, Pan Zhou, Caiming Xiong, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv*, 2020. [6](#)
- [32] Xiaotong Li, Zixuan Hu, Yixiao Ge, Ying Shan, and Lingyu Duan. Exploring model transferability through the lens of potential energy. In *ICCV*, pages 5429–5438, 2023. [1, 2, 3, 4, 5, 6, 7](#)
- [33] Yandong Li, Xuhui Jia, Ruoxin Sang, Yukun Zhu, Bradley Green, Liqiang Wang, and Boqing Gong. Ranking neural checkpoints. In *CVPR*, pages 2663–2673, 2021. [1, 2, 5, 6, 7](#)
- [34] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017. [3](#)
- [35] Mohamad Amin Mohamadi and Danica J. Sutherland. A fast, well-founded approximation to the empirical neural tangent kernel. In *ICML*, 2022. [2, 5](#)

- [36] Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. Leep: A new measure to evaluate transferability of learned representations. In *ICML*, pages 7294–7305, 2020. [2](#)
- [37] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. [5](#)
- [38] Michal Pándy, Andrea Agostinelli, Jasper Uijlings, Vittorio Ferrari, and Thomas Mensink. Transferability estimation using bhattacharyya class separability. In *CVPR*, pages 9172–9182, 2022. [2](#), [6](#)
- [39] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, pages 3498–3505, 2012. [5](#)
- [40] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. [6](#)
- [41] Wenqi Shao, Xun Zhao, Yixiao Ge, Zhaoyang Zhang, Lei Yang, Xiaogang Wang, Ying Shan, and Ping Luo. Not all models are equal: Predicting model transferability in a self-challenging fisher space. In *ECCV*, pages 286–302, 2022. [1](#), [2](#), [3](#), [4](#), [5](#), [6](#), [7](#)
- [42] Grace S. Shieh. A weighted kendall’s tau statistic. *Statistics Probability Letters*, 39(1):17–24, 1998. [1](#)
- [43] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. [6](#)
- [44] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. [6](#)
- [45] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, pages 2820–2828, 2019. [6](#)
- [46] Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. What makes for good views for contrastive learning? In *NeurIPS*, pages 6827–6839, 2020. [6](#)
- [47] Anh T Tran, Cuong V Nguyen, and Tal Hassner. Transferability and hardness of supervised classification tasks. In *ICCV*, pages 1395–1405, 2019. [1](#), [2](#)
- [48] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. [2](#)
- [49] Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *Esann*, pages 219–224, 1999. [5](#)
- [50] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, pages 3733–3742, 2018. [6](#)
- [51] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492. IEEE, 2010. [5](#)
- [52] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. Logme: Practical assessment of pre-trained models for transfer learning. In *ICML*, pages 12133–12143, 2021. [2](#), [5](#), [6](#), [7](#)
- [53] Gizem Yüce, Guillermo Ortiz-Jiménez, Beril Besbinar, and Pascal Frossard. A structured dictionary perspective on implicit neural representations. In *CVPR*, pages 19228–19238, 2022. [5](#)