LEAD: Exploring Logit Space Evolution for Model Selection

Supplementary Material

In this supplementary material, we first present theoretical proof of our approach. Next, we introduce implementation details about LEAD and conduct more ablation studies for our approach to validate its effectiveness. Finally, we provide performance comparisons under different measurements and fine-tuning results of the ground-truth.

A. Theoretical Proof

Below, we will provide detailed proof of the theoretical results presented in the methodology section.

Notation. First, We recall the notation that we used in the main paper as well as this appendix:

$\{\mathcal{X},\mathcal{Y}\}$ denotes the downstream dataset. $\mathcal{F}$ denotes the logit function. η denotes the learning rate. l denotes root mean square of network widths. Φ denotes NTK matrix when l approaching infinite. $\mathbb{I}$ denotes the identity matrix. To simplify the notation, $\cdot$ denotes both dot product and scalar multiplication, $e^{\cdot}$, $\cdot^2$ denotes both scalar and matrix power operations, and $\frac{\mathrm{d}\times}{\mathrm{d}\cdot}$ denotes both differentiation of scalar functions and the Jacobian matrix of vector functions.

Dynamical Equation. Subsequently, we provide the proof for obtaining the continuous dynamical equation in Eq. (5) of the main paper through the limit approximation method.

Conclusion 1. *Consider an infinite-width neural network $\mathcal{F}$ and a downstream dataset $\mathcal{T}=\{\mathcal{X},\mathcal{Y}\}$. The evolution process of the output logits $\mathcal{F}_t(\mathcal{X})$ will follow the following differential equation:*

$$\frac{\mathrm{d}\mathcal{F}_t}{\mathrm{d}t}=-\eta\cdot\Phi\cdot\frac{\mathrm{d}\mathcal{L}_t}{\mathrm{d}\mathcal{F}_t},\ \mathcal{F}_0=log_{init}. \tag{1}$$

Proof. We need to consider the variation trend of $\mathcal{F}$. Therefore, we calculate its derivative at time t according to the definition:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{F}_t(\mathcal{X})}{\mathrm{d}t}&=\lim_{\Delta t\to 0}\frac{\mathcal{F}_{t+\Delta t}(\mathcal{X})-\mathcal{F}_t(\mathcal{X})}{\Delta t}\\&=\lim_{\Delta t\to 0}\frac{\mathcal{F}\left(\mathcal{X},\theta_{t+\Delta t}\right)-\mathcal{F}\left(\mathcal{X},\theta_t\right)}{\Delta t}.\end{aligned} \tag{2}$$

Through Taylor's theorem, we obtain the asymptotic series decomposition of $\mathcal{F}$ concerning the variation of θ:

$$\begin{aligned}\mathcal{F}\left(\mathcal{X},\theta_{t+\Delta t}\right)=\mathcal{F}\left(\mathcal{X},\theta_t\right)+\frac{\mathrm{d}\mathcal{F}\left(\mathcal{X},\theta_t\right)}{\mathrm{d}\theta}\left(\theta_{t+\Delta t}-\theta_t\right)\\+O\left(\left(\theta_{t+\Delta t}-\theta_t\right)^2\right),\end{aligned} \tag{3}$$

where $O(\cdot)$ denotes a infinitesimal quantity which has the same order with $\cdot$, describing the remainder of the asymptotic series. Meanwhile, according to the optimization of the gradient descent, we can determine the equivalent effects in corresponding continuous time:

$$\begin{aligned}\theta_{t+\Delta t}-\theta_t&=-\eta\cdot\frac{\mathrm{d}\mathcal{L}\left(\mathcal{F}\left(\mathcal{X},\theta_t\right),\mathcal{Y}\right)}{\mathrm{d}\theta}\cdot\Delta t,\\O\left(\left(\theta_{t+\Delta t}-\theta_t\right)^2\right)&=O\left(\eta^2\cdot\frac{\mathrm{d}\mathcal{L}_t}{\mathrm{d}\theta}\cdot\frac{\mathrm{d}\mathcal{L}_t}{\mathrm{d}\theta}\cdot\Delta t^2\right)\\&=O\left(\left(\Delta t\right)^2\right).\end{aligned} \tag{4}$$

Combining Eq. (3) and Eq. (4), we obtain the asymptotic series decomposition of $\mathcal{F}$ concerning the variation of t:

$$\begin{aligned}&\mathcal{F}\left(\mathcal{X},\theta_{t+\Delta t}\right)-\mathcal{F}\left(\mathcal{X},\theta_t\right)\\=&\frac{\mathrm{d}\mathcal{F}\left(\mathcal{X},\theta_t\right)}{\mathrm{d}\theta}\left(\theta_{t+\Delta t}-\theta_t\right)+O\left(\left(\theta_{t+\Delta t}-\theta_t\right)^2\right)\\=&-\eta\cdot\frac{\mathrm{d}\mathcal{F}\left(\mathcal{X},\theta_t\right)}{\mathrm{d}\theta}\cdot\frac{\mathrm{d}\mathcal{L}\left(\mathcal{F}\left(\mathcal{X},\theta_t\right),\mathcal{Y}\right)}{\mathrm{d}\theta}\Delta t+O\left(\left(\Delta t\right)^2\right).\end{aligned} \tag{5}$$

Combining Eq. (2) and Eq. (5) and employing the property of $O(\cdot)$, we can use limit approximation, discarding the remainder terms with a limit value of 0, to determine the value of the derivative:

$$\begin{aligned}\frac{\mathrm{d}\mathcal{F}_t(\mathcal{X})}{\mathrm{d}t}=&\lim_{\Delta t\to 0}-\eta\cdot\frac{\mathrm{d}\mathcal{F}\left(\mathcal{X},\theta_t\right)}{\mathrm{d}\theta}\cdot\frac{\mathrm{d}\mathcal{L}\left(\mathcal{F}\left(\mathcal{X},\theta_t\right),\mathcal{Y}\right)}{\mathrm{d}\theta}\\&+O\left(\left(\Delta_t\right)^2\right)/\Delta t\\=&-\eta\cdot\frac{\mathrm{d}\mathcal{F}\left(\mathcal{X},\theta_t\right)}{\mathrm{d}\theta}\cdot\frac{\mathrm{d}\mathcal{L}\left(\mathcal{F}\left(\mathcal{X},\theta_t\right),\mathcal{Y}\right)}{\mathrm{d}\theta}\\=&-\eta\cdot\frac{\mathrm{d}\mathcal{F}_t}{\mathrm{d}\theta}\cdot\frac{\mathrm{d}\mathcal{L}_t}{\mathrm{d}\theta}.\end{aligned} \tag{6}$$

where the last line simplifies the notation for convenience. Through the chain rule, we have:

$$\frac{\mathrm{d}\mathcal{F}_t}{\mathrm{d}t}=-\eta\cdot\frac{\mathrm{d}\mathcal{L}_t}{\mathrm{d}\mathcal{F}_t}\cdot\left(\frac{\mathrm{d}\mathcal{F}_t}{\mathrm{d}\theta}\cdot\frac{\mathrm{d}\mathcal{F}_t}{\mathrm{d}\theta}\right). \tag{7}$$

Finally, through the definition and the constant-preserving property of the NTK [15], we obtain the dynamical equation with its initial value condition:

$$\frac{\mathrm{d}\mathcal{F}_t}{\mathrm{d}t}=-\eta\cdot\Phi\cdot\frac{\mathrm{d}\mathcal{L}_t}{\mathrm{d}\mathcal{F}_t},\ \mathcal{F}_0=log_{init}. \tag{8}$$

Closed-form Solution. Finally, we solve the ordinary differential equation (ODE) in Eq. (8) through separation of variables and integration along the time dimension. As a commonly used scene in theoretical analysis, we consider the case where $\mathcal{L}$ is Mean Squared Error loss. And we can obtain concise closed-form solution as shown in Eq. (6) of the main paper:

Conclusion 2. *Consider the case where $\mathcal{L}$ is MSE loss. The solution of logits $\mathcal{F}_t(\mathcal{X})$ to the equation presented in Eq. (8) can be expressed as follows:*

$$\mathbb{E}\left(\mathcal{F}_t\left(\mathcal{X}\right)\right) = \left(\mathbb{I} - e^{-\eta\Phi\cdot t}\right)\mathcal{Y} + e^{-\eta\Phi\cdot t}\cdot log_{init}. \tag{9}$$

Proof. Due to $\frac{\mathrm{d}\mathcal{L}_t}{\mathrm{d}\mathcal{F}_t} = \mathcal{F}_t - \mathcal{Y}$, we have:

$$\frac{\mathrm{d}\mathcal{F}_t}{\mathrm{d}t} = -\eta\cdot\Phi\cdot\left(\mathcal{F}_t - \mathcal{Y}\right),\ \mathcal{F}_0 = log_{init}. \tag{10}$$

In order to transform both sides of Eq. (10) into independent differentials, we first introduce an integrating factor $e^{\eta\Phi t}$:

$$\begin{aligned} & e^{\eta\Phi t}\ \mathrm{d}\mathcal{F}_t = -\eta\Phi\cdot e^{\eta\Phi t}\left(\mathcal{F}_t - \mathcal{Y}\right)\mathrm{d}t \\ \Leftrightarrow\ & e^{\eta\Phi t}\ \mathrm{d}\mathcal{F}_t + \eta\Phi\cdot e^{\eta\Phi t}\mathcal{F}_t\ \mathrm{d}t = \eta\Phi\cdot e^{\eta\Phi t}\cdot\mathcal{Y}\ \mathrm{d}t. \end{aligned} \tag{11}$$

Through the differential properties of composite functions, the left-hand side of Eq. (11) is equivalent to the differential of a composite function:

$$\begin{aligned} & e^{\eta\Phi t}\ \mathrm{d}\mathcal{F}_t + \eta\Phi\cdot e^{\eta\Phi t}\mathcal{F}_t\ \mathrm{d}t \\ = & e^{\eta\Phi t}\ \mathrm{d}\mathcal{F}_t + \mathcal{F}_t\ \mathrm{d}e^{\eta\Phi t} = \mathrm{d}\left(\mathcal{F}_t\cdot e^{n\Phi t}\right). \end{aligned} \tag{12}$$

Combining Eq. (11) and Eq. (12), we get the following equation whose left and right sides are determined by the differential of independent composite functions:

$$\mathrm{d}\left(\mathcal{F}_t\cdot e^{n\Phi t}\right) = \mathcal{Y}\ \mathrm{d}\left(e^{\eta\Phi t}\right). \tag{13}$$

By integrating over the time dimension t, we have:

$$\begin{aligned} & \mathcal{F}_t\cdot e^{n\Phi t}|_0^t = \mathcal{Y}\cdot e^{\eta\Phi t}|_0^t \\ \Leftrightarrow\ & \mathcal{F}_t\cdot e^{n\Phi t} - \mathcal{F}_0 = \mathcal{Y}\cdot\left(e^{\eta\Phi t} - \mathbb{I}\right). \end{aligned} \tag{14}$$

Substituting the initial condition in Eq. (10), we obtain the solution to the dynamical equation:

$$\mathcal{F}_t\left(\mathcal{X}\right) = \left(\mathbb{I} - e^{-\eta\Phi\cdot t}\right)\mathcal{Y} + e^{-\eta\Phi\cdot t}\cdot log_{init}. \tag{15}$$

Since the constant-preserving property of Φ requires taking the expectation over the randomly initialized part of $\mathcal{F}$, our results need to take the expectation as well:

$$\mathbb{E}\left(\mathcal{F}_t\left(\mathcal{X}\right)\right) = \left(\mathbb{I} - e^{-\eta\Phi\cdot t}\right)\mathcal{Y} + e^{-\eta\Phi\cdot t}\cdot log_{init}. \tag{16}$$

B. Implementation Details

The implementation of our method are presented in the section of methodology and experiments. Additionally, we present more detailed information and the pseudocode of the overall algorithm in this section.

Classification Algorithm. In the experiments and ablation study section of the main paper, we have explained the reasons for choosing the SVM algorithm as classification algorithm to obtain log_{init}. Here, we will provide a detailed introduction to the Multi-class SVM. The original SVM is a binary classifier, that can only handle two classes problems. However, for multiclass problems, we can utilize the One-vs-Rest (OvR) strategy [29]. For a problem with K classes, we train K different binary classifiers, with each classifier specifically addressing one class, treating it as the positive class, and considering all other classes as negative. For a given sample, predictions are made using these classifiers and the prediction values are normalized to obtain the predicted probabilities (*i.e.,* logits) for K classes.

Calculation of NTK. For the computation of the NTK, we employ the approximation algorithm proposed in [20]. Specifically, we concatenate the model backbone with the classification head, a randomly initialized MLP (with two hidden layers of width 1024 and 2048). The combination of these two parts constitutes $\mathcal{F}$ to output logits. Subsequently, we calculate the NTK using the following equation:

$$\Phi_{i,j}(\mathcal{X},\mathcal{X}) = \left[\nabla_\theta\sum_{k=1}^{K}\mathcal{F}^{(k)}\left(x_i,\theta\right)\right]\left[\nabla_\theta\sum_{k=1}^{K}\mathcal{F}^{(k)}\left(x_j,\theta\right)\right] \tag{17}$$

where $\Phi_{i,j}$ denotes the element of i-th row and j-th column in Φ. x_i denotes i-th sample of $\mathcal{X}$. K denotes the output dimension of $\mathcal{F}$, $\mathcal{F}^{(k)}$ denotes the k-th output dimension and ∇ is only utilized on the MLP for efficiency. The computation method in Eq. (17) reduces the computational cost of the NTK. Meanwhile, [20] provides both theoretical proof and experimental validation that the eigenvalue range obtained through this approximation is close to that of the original computation method in [15].

Pseudocode. We provide the pseudocode for our LEAD to demonstrate the complete computation procedure.

C. More Ablation Study and Visualization

C.1. Initial State Observation

To capture the observation of log_{init}, we feed features and labels of downstream datasets into a classifier that can output the probability of each class. We evaluate several commonly used machine learning algorithms SVM [29], LDA [19], and GMM [22], comparing their impact on performance. As shown in Fig. 1, it's noteworthy that, whatever the classifier employed, direct utilization of log_{init} fails to yield satisfactory results due to overlooking the training dynamics. Conversely, the application of LEAD consistently improves all results to exceed the prior SOTA. This demonstrates that LEAD does not rely on a specific classifier and has uniform applicability to different classifier selections.

C.2. Different Interpolation Strategies

In our method, we use eigenvalues of the class-aware NTK matrix to determine the interpolation coefficient. In this section, we implement other strategies for comparison in

Algorithm 1: The algorithm of our method LEAD

```
Input: Downstream dataset $\{\mathcal{X},\mathcal{Y}\}$, with $K$ classes; Model zoo
       $\{\Psi_i\}_{i=1}^m$ with their backbone $\{f_i\}_{i=1}^m$; The sample size
       hyper-meter $S$ for class-aware decomposition; The time
       hyper-parameter $t$.
Output: The transferability score $\{P_i\}_{i=1}^m$ for each model in the
        model zoo and the predicted rank.
1  for $\Psi_i$ in Model zoo do
2      Encode images $\mathcal{X}$ to feature embeddings and feed features
       and corresponding labels into the classification algorithm $\mathcal{C}$
       to obtain initial state logits:
3          $\mathcal{Z} = f_i(\mathcal{X}),\ log_{init} = \mathcal{C}(\mathcal{Z},\mathcal{Y})$;
4      Concatenate a randomly initialized MLP $h$ as a classification
       head after its backbone $f$ and combine them as logit
       function $\mathcal{F} = h \circ f$.
5      for $k$ in range(K) do
6          Select $S$ samples $\mathcal{X}_k := \{x_{k_i}\}_{i=1}^S$ from k-th class.
7          Compute the NTK matrix for this class:
8              $\Phi_{u,v}(\mathcal{X}_k,\mathcal{X}_k) = \left[\nabla_\theta \sum_{j=1}^K \mathcal{F}^{(j)}\left(x_{k_u},\theta\right)\right]\cdot$
9                  $\left[\nabla_\theta \sum_{j=1}^K \mathcal{F}^{(j)}\left(x_{k_v},\theta\right)\right]$;
10         for $1 \le u,v \le K$.
11         Employ eigenvalue decomposition to $\Phi(\mathcal{X}_k,\mathcal{X}_k)$ and
           obtain the mean of eigenvalues $\overline{\lambda}$.
12         We can get the prediction value of the final state logits
           of samples belonging to this class:
           $\mathbb{E}\left(\mathcal{F}_t(x)\right) = \left(\mathbb{I} - e^{-\overline{\lambda}\cdot t}\right) y + e^{-\overline{\lambda}\cdot t}\cdot log_{init}(x)$;
13         for $x$ belonging to the k-th class.
14     end
15     Feed the predicted final state logits $\mathcal{F}_t(\mathcal{X})$ and ground-
       truth label $\mathcal{Y}$ into the Cross-Entropy loss to obtain the
       score $P_i$ for the model $\Psi_i$.
16 end
17 Rank models $\{\Psi_i\}_{i=1}^m$ according to their scores $\{P_i\}_{i=1}^m$.
```

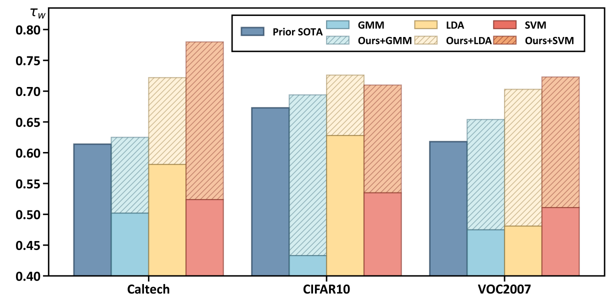

Figure 1. Performance comparison of our method under different classifier selection to compute log_{init}.

Fig. 2. Specifically, 'Constant' denotes directly choosing a constant c as the coefficient for all samples and its results are optimal ones selected through grid search within $c \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. 'Original NTK' denotes the computation of the NTK by mixing samples randomly chosen from all classes to determine interpolation coefficients. 'Class-aware NTK' denotes the approach we propose, which involves computing NTK separately for each class to determine interpolation coefficients.

As we can see, directly employing a constant will harm the performance because it incorrectly assumes that evolution processes of all samples are the same. 'Original NTK', which computes NTK by mixing samples, shows some improvement, especially for datasets with fewer classes like CIFAR10. However, for datasets with a larger number of classes like Caltech, it exhibits only marginal improvement due to a lack of ability to model the evolution of different classes. The results highlight the advantage of our proposed class-aware method in modeling the evolution process.

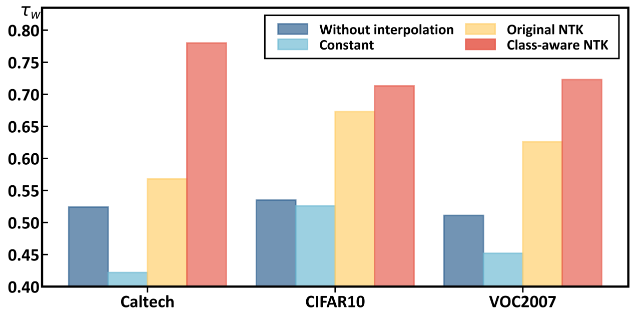

Figure 2. Performance comparison under different interpolation strategies.

C.3. Network Width

The width of the model backbone is fixed, but the hidden layer width of the classification head MLP can be adjusted. We evaluate the impact of different hidden layer widths on performance, and for the convenience of comparison, we set both two hidden layers to the same width, as shown in Fig. 3. As we can see, with the increase in the width of the hidden layers, the accuracy of the rank also improves. It indicates that, as the network width increases, the calculation condition of the NTK matrix gradually approach the required theoretical assumption, leading to the improvement of prediction precision. To balance computation cost and performance, we set the widths of two hidden layers to be 1024 and 2048.

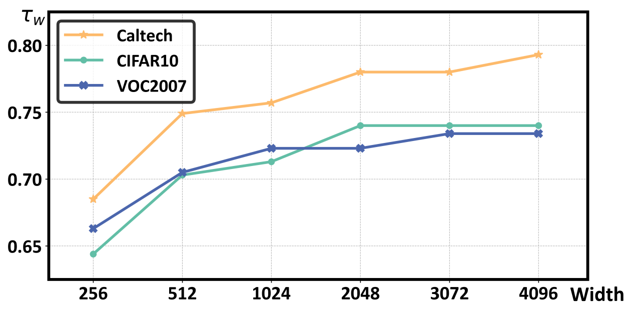

Figure 3. Performance comparison under different MLP hidden layer width.

C.4. Logit Predictions for Different Classes

In Fig. 3 and Fig. 6 of the main paper, we have demonstrated the accuracy of LEAD in predicting logits for the dataset through comparison with prior arts. In this section, we provide visualization results for more detailed comparison. As shown in Fig. 4, we present the differences between the logits predicted by LEAD and the actual values obtained after fine-tuning, focusing on six classes of the VOC2007 dataset. We can observe that the distribution of predicted logits closely aligns with the actual results across six categories, further highlighting the precision of our dynamical equation-based approach in capturing logit evolution for accurate ranking sequences.

D. Different Measurements of Transferability

Apart from utilizing weighted Kendall's tau (τ_w), here we employ various measurement metrics of the rank correlation to assess the performance of our method LEAD. These evaluation metrices include Kendall's tau (τ), weighted Pearson's correlation (r_w), and top-k relative accuracy (Rel-k).

Table 1. Performance comparison under different measurement metrics of rank correlation assessment on Caltech [9], CIFAR10 [16], and VOC2007 [8] datasets using self-supervised model zoo.

Dataset	Method	τ_w	τ	r_w	Rel-1	Rel-3
Caltech	PACTran [7]	0.622	0.455	0.457	**1.000**	**1.000**
	SFDA [24]	0.523	0.424	0.627	0.995	**1.000**
	ETran [10]	0.405	0.455	0.410	0.990	0.992
	PED [18]	0.614	0.530	0.675	0.995	**1.000**
	LEAD (Ours)	**0.780**	**0.758**	**0.851**	**1.000**	**1.000**
CIFAR10	PACTran [7]	0.477	0.485	0.497	**1.000**	**1.000**
	SFDA [24]	0.619	0.545	0.558	**1.000**	**1.000**
	ETran [10]	0.606	0.515	0.422	0.999	**1.000**
	PED [18]	0.673	0.606	0.645	**1.000**	**1.000**
	LEAD (Ours)	**0.713**	**0.667**	**0.792**	**1.000**	**1.000**
VOC2007	PACTran [7]	0.620	0.485	0.413	0.995	**1.000**
	SFDA [24]	0.568	0.424	0.636	0.998	0.998
	ETran [10]	0.376	0.455	0.468	0.974	0.998
	PED [18]	0.583	0.484	0.708	0.998	**1.000**
	LEAD (Ours)	**0.723**	**0.788**	**0.847**	**0.998**	**1.000**

Rel-k represents the ratio between the optimal fine-tuning accuracy within the models ranked in the top-k and the best fine-tuning accuracy across all models. To evaluate the robustness of transferability metrics across different measurements, we conduct experiments using self-supervised CNN models on Caltech, CIFAR10, and VOC2007 datasets, as shown in Tab. 1. The results show that τ_w is highly correlated with other metrics, and it can assign larger weights to models in higher ranking, which we are concerned with. Therefore, following [18, 21, 24], we employ τ_w in the main paper to evaluate performance. Meanwhile, our LEAD consistently outperforms prior arts such as PACTran, SFDA, ETran, and PED across the aforementioned measurements, highlighting the superior performance of LEAD.

E. Ground-truth Results

We obtained the ground-truth results after model fine-tuning which employs a grid-search strategy, following the implementation of [18, 24]. More details on this process are available in Sec. 4 of the main paper. In Tab. 2 and 3, we present the ground-truth results of 12 supervised pre-trained models and 12 self-supervised pre-trained models across 10 downstream tasks.

References

[1] Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning. *arXiv*, 2019. 6

[2] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *ECCV*, pages 132–149, 2018. 6

[3] 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. 6

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020. 6

[5] 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

[6] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv*, 2020. 6

[7] 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. 4

[8] 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

[9] Li Fei-Fei. Learning generative visual models from few training examples. In *CVPR workshop*, 2004. 4

[10] Mohsen Gholami, Mohammad Akbari, Xinglu Wang, Behnam Kamranian, and Yong Zhang. Etran: Energy-based transferability estimation. In *ICCV*, pages 18613–18622, 2023. 4

[11] 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

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 5

[13] 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

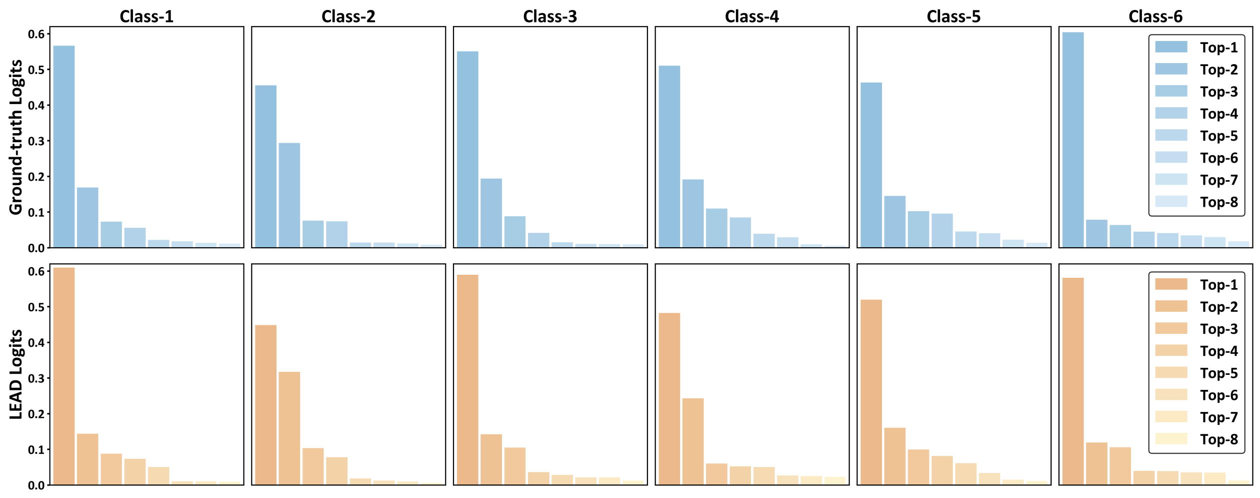

Figure 4. Comparison of the average prediction for final state logits between LEAD and Ground-truth results (obtain after fully fine-tuning) on six classes of VOC2007 dataset.

Table 2. The ground-truth results of the 12 supervised pre-trained models on 10 downstream tasks.

Supervised	Food	Caltech	Flowers	Cars	CIFAR100	DTD	CIFAR10	Pets	SUN397	VOC2007
ResNet-34 [12]	81.99	91.15	95.20	88.63	81.94	72.96	96.12	93.50	61.02	84.60
ResNet-50 [12]	84.45	91.98	96.26	89.09	82.80	74.72	96.28	93.88	63.54	85.80
ResNet-101 [12]	85.58	92.38	96.53	89.47	84.88	74.80	97.39	93.92	63.76	85.68
ResNet-152 [12]	86.28	93.10	96.86	89.88	85.66	76.44	97.53	94.42	64.82	86.32
DenseNet-121 [14]	84.99	91.50	97.02	89.34	82.75	74.18	96.45	93.07	63.26	85.28
DenseNet-161 [14]	87.13	93.13	97.59	89.62	84.98	76.21	97.48	94.35	64.25	85.69
DenseNet-169 [14]	85.84	92.51	97.32	89.02	84.26	74.72	96.77	93.62	64.10	85.77
DenseNet-201 [14]	86.71	93.14	97.10	89.44	84.88	76.04	97.02	94.03	64.57	85.67
MNet-A1 [27]	71.35	89.34	95.39	72.58	72.04	70.12	92.59	91.08	56.56	81.06
MobileNetV2 [23]	81.12	88.64	96.20	86.44	78.11	71.72	94.74	91.28	60.29	82.80
Googlenet [25]	79.30	90.85	95.76	87.76	79.84	72.53	95.54	91.38	59.89	82.58
InceptionV3 [26]	81.76	92.75	95.73	87.74	81.49	72.85	96.18	92.14	59.98	83.84

[14] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *CVPR*, pages 4700–4708, 2017. 5

[15] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018. 1, 2

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 4

[17] Junnan Li, Pan Zhou, Caiming Xiong, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv*, 2020. 6

[18] Xiaotong Li, Zixuan Hu, Yixiao Ge, Ying Shan, and Ling-Yu Duan. Exploring model transferability through the lens of potential energy. In *ICCV*, pages 5429–5438, 2023. 4

[19] Sebastian Mika, Gunnar Ratsch, Jason Weston, Bernhard Scholkopf, and Klaus-Robert Mullers. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX: Proceedings of the 1999 IEEE signal processing society workshop (cat. no. 98th8468)*, pages 41–48, 1999. 2

[20] Mohamad Amin Mohamadi and Danica J. Sutherland. A fast, well-founded approximation to the empirical neural tangent kernel. In *ICML*, 2022. 2

[21] Michal Pándy, Andrea Agostinelli, Jasper Uijlings, Vittorio Ferrari, and Thomas Mensink. Transferability estimation using bhattacharyya class separability. In *CVPR*, pages 9172–9182, 2022. 4

[22] Douglas A Reynolds et al. Gaussian mixture models. *Encyclopedia of biometrics*, 741(659-663), 2009. 2

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. 5

[24] 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. 4

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet,

Table 3. The ground-truth results of the 12 self-supervised pre-trained models on 10 downstream tasks.

Self-Supervised	Food	Caltech	Flowers	Cars	CIFAR100	DTD	CIFAR10	Pets	SUN397	VOC2007
BYOL [11]	85.44	91.90	96.80	89.83	83.86	76.37	96.98	91.48	63.69	85.13
Deepclusterv2 [2]	87.24	91.16	97.05	90.16	84.84	77.31	97.17	90.89	66.54	85.38
Infomin [28]	78.82	80.86	95.81	86.90	70.89	73.74	96.72	90.92	57.67	81.41
InsDis [30]	76.47	77.21	93.63	80.21	69.08	66.40	93.08	84.58	51.62	76.33
MoCov1 [13]	77.21	79.68	94.32	82.19	71.23	67.36	84.15	85.26	53.83	77.94
MoCov2 [6]	77.15	82.76	95.12	85.55	71.27	72.56	96.48	89.06	56.28	78.32
PCLv1 [17]	77.70	88.60	95.62	87.15	79.44	73.28	86.42	88.93	58.36	91.91
PCLv2 [17]	80.29	87.52	95.87	85.56	79.84	69.30	96.55	88.72	58.82	81.85
Selav2 [1]	86.37	90.53	96.22	89.85	84.36	76.03	96.85	89.61	65.74	85.52
SimCLRv1 [4]	82.20	90.94	95.33	89.98	84.49	73.97	97.09	88.53	63.46	83.29
SimCLRv2 [5]	82.23	88.58	95.39	88.82	78.91	94.71	96.22	89.18	60.93	83.08
SWAV [3]	87.22	89.49	97.11	89.81	83.78	76.68	96.81	90.59	66.10	85.06

Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 5

[26] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. 5

[27] 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. 5

[28] 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

[29] Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *Esann*, pages 219–224, 1999. 2

[30] 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