Ranking Neural Checkpoints

Yandong Li [1,2*] Xuhui Jia[2] Ruoxin Sang[2]
Yukun Zhu[2] Bradley Green[2] Liqiang Wang[1] Boqing Gong[2]
[1]University of Central Florida [2]Google
lyndon.leeseu@outlook.com lwang@cs.ucf.edu {xhjia,rxsang,yukun,brg,bgong}@google.com

Abstract

This paper is concerned with ranking many pre-trained deep neural networks (DNNs), called checkpoints, for the transfer learning to a downstream task. Thanks to the broad use of DNNs, we may easily collect hundreds of checkpoints from various sources. Which of them transfers the best to our downstream task of interest? Striving to answer this question thoroughly, we establish a neural checkpoint ranking benchmark (NeuCRaB) and study some intuitive ranking measures. These measures are generic, applying to the checkpoints of different output types without knowing how the checkpoints are pre-trained on which datasets. They also incur low computation cost, being practically meaningful. Our results suggest that the linear separability of the features extracted by the checkpoints is a strong indicator of transferability. We also arrive at a new ranking measure, $\mathcal{N}$LEEP, which gives rise to the best performance in the experiments. Code will be made publicly available.

1. Introduction

There is an increasing number of pre-trained deep neural networks (DNNs), which we call checkpoints. We may produce hundreds of intermediate checkpoints when we sweep through various learning rates, optimizers, and losses to train a DNN. Furthermore, semi-supervised [10, 4, 49, 36, 58, 39, 37, 8] and self-supervised [15, 26, 11, 62, 43] learning make it feasible to harvest DNN checkpoints with scarce or no labels. Fine-tuning [65, 44] has become a de facto standard to adapt the pre-trained checkpoints to target tasks. It leads to faster convergence [16, 27, 51] and better performance [35] on the downstream tasks.

However, not all checkpoints are equally useful for a target task, and some could even under-perform a randomly initialized checkpoint (cf. Section 2.2). This paper is concerned with **ranking neural checkpoints**, which aims to measure how effectively fine-tuning can transfer knowledge from the pre-trained checkpoints to the target task. The measurement should be *generic* enough for all the neural checkpoints, meaning that it works without knowing any pre-training details (e.g., pre-training examples, hyper-parameters, losses, early stopping stages, etc.) of the checkpoints. It also should be *lightweight*, ideally without training on the downstream task, to make it practically useful. We may use the measurement to choose the top few checkpoints before running fine-tuning, which is computationally more expensive than calculating the measurements.

Ranking neural checkpoints is crucial. Some domains or applications lack large-scale human-curated data, like medical images [48], raising a pressing need for high-quality pre-trained checkpoints as a warm start for fine-tuning. Fortunately, there exist hundreds of thousands of checkpoints of popular neural network architectures. For instance, many computer vision models are built upon ResNet [28], Inception-ResNet [56], and VGG [52]. As a result, we can construct a candidate pool by collecting the checkpoints released by different groups, for various tasks, and over distinct datasets.

It is nontrivial to rank the checkpoints for a downstream task. We explain this point by drawing insights from the related, yet arguably easier, task transferability problem [1, 20, 66, 41], which aims to provide high-level guidance about how well a neural network pre-trained in one task might transfer to another. However, not all checkpoints pre-trained in the same source task transfer equally well to the target task [70, 35]. The pre-training strategy also matters. Zhai *et al.* [68] find that combining supervision with self-supervision improves a network's transfer results on downstream tasks. He *et al.* [26] also show that self-supervised pre-training benefits object detection more than its supervised counterpart under the same fine-tuning setup.

We may also appreciate the challenge in ranking neural checkpoints by comparing it with another related line of work: predicting DNNs' generalization gaps [40, 31, 5]. Jiang *et al.* [30] use a linear regressor to predict a DNN's generalization gap, i.e., the discrepancy between its training and test accuracies, by exploring the training data's margin

*This work was done while the first author was an intern at Google.

distributions. Other signals studied in the literature include network complexity and noise stability. Ranking neural checkpoints is more challenging than predicting a DNN's generalization gap. Unlike the training and test sets that share the same underlying distribution, the downstream task may be arbitrarily distant from the source task over which a checkpoint is pre-trained. Moreover, we do not have access to the pre-training data at all. Finally, instead of keeping the networks static, fine-tuning dramatically changes all weights of the checkpoints.

We establish a neural checkpoint ranking benchmark (NeuCRaB) to study the problem systematically. NeuCRaB covers various checkpoints pre-trained on widely used, large-scale datasets by different training strategies and architectures at a range of early stopping stages. It also contains diverse downstream tasks, whose training sets are medium-sized, making it practically meaningful to rank and fine-tune existing checkpoints. Pairing up all the checkpoints and downstream tasks, we conduct careful fine-tuning with thorough hyper-parameter sweeping to obtain the best transfer accuracy for each checkpoint-downstream-task pair. Hence, we know the groundtruth ranking of the checkpoints for each downstream task according to the final accuracies (over the test/validation sets).

A functional checkpoint ranking measurement should be highly correlated with the groundtruth ranking and, equally importantly, incurs as low computation cost as possible. We study several intuitive methods for ranking the neural checkpoints. One is to freeze the checkpoints as feature extractors and use a linear classifier to evaluate the features' separability on the target task. Another is to run fine-tuning for only a few epochs (to avoid heavy computation) and then evaluate the resulting networks on the target task's validation set. We also estimate the mutual information between labels and the features extracted from a checkpoint.

Finally, we propose a lightweight measure, named Gaussian LEEP ($\mathcal{N}$LEEP), to rank checkpoints based on the recently proposed log expected empirical prediction (LEEP) [41]. LEEP was originally designed to measure between-task transferabilities. It cannot handle the checkpoints pre-trained by unsupervised or self-supervised learning since it requires all checkpoints to have a classification head. Its computation cost could blow up when the classification head corresponds to a large output space. Moreover, it depends on the classification head's probabilistic output, which, unfortunately, is often overly confident [25].

To tackle the above problems, we replace the checkpoints' output layer with a Gaussian mixture model (GMM). This simple change kills two birds with one stone. On the one hand, GMM's soft assignment of input to clusters seamlessly applies to LEEP, resulting in the lightweight, effective $\mathcal{N}$LEEP measure that works regardless of the checkpoints' output types. On the other hand, since we fit GMM to the target task's data, instead of the pre-training data of a different source task, the cluster assignment probabilities are likely more calibrated than the classification probabilities for the target task, if there exist classification heads.

2. The Neural Checkpoint Ranking Benchmark (NeuCRaB)

We formalize ranking neural checkpoints as follows. Suppose we have m pre-trained neural networks, called checkpoints, $\mathcal{C} := \{\theta_i\}_{i=1}^m$. Denote by $\mathcal{T}$ a distribution of tasks. Without loss of generality, we mainly study classification downstream tasks, each of which, $t \sim \mathcal{T}$, contains a training set and a test set. An evaluation procedure, $\mathbf{G} : \mathcal{C} \times \mathcal{T} \mapsto \mathbb{R}$, replaces the output layer of a checkpoint θ_i with a linear classifier for a downstream task t, followed by fine-tuning using the task's training set. It employs hyper-parameter sweeping and returns the best accuracy on the test set. We apply this evaluation procedure to all the checkpoints for task t and obtain their test accuracies:

$$\mathbf{G}_t := \{\mathbf{G}(\theta_i, t)\}_{i=1}^m \in \mathbb{R}^m, \tag{1}$$

which defines the groundtruth ranking list for task t.

Denote by $\mathcal{R}$ all measures that return a ranking score for any checkpoint-task pair under a computation budget $\mathbf{b}$. A measure $\mathbf{R} \in \mathcal{R}$ gives rise to the following ranking scores for a task t,

$$\mathbf{R}_t := \{\mathbf{R}(\theta_i, t; \mathbf{b})\}_{i=1}^m \in \mathbb{R}^m, \tag{2}$$

where we underscore the computation budget $\mathbf{b}$ in the measure $\mathbf{R}(\cdot, \cdot; \mathbf{b})$.

Our objective in ranking neural checkpoints is to find the best ranking measure in expectation,

$$\mathbf{R}^* \leftarrow \arg\max_{\mathbf{R} \in \mathcal{R}} \ \mathbb{E}_{t \sim \mathcal{T}} \ \mathcal{M}(\mathbf{R}_t, \mathbf{G}_t) \tag{3}$$

where $\mathcal{M}$ is a metric evaluating the ranking scores $\mathbf{R}_t$ against the test accuracies $\mathbf{G}_t$. Section 2.3 details the evaluation methods used in this work. Equipped with such a ranking measure $\mathbf{R}^*$, we can identify the checkpoints that potentially transfer to a downstream task better than the others without resorting to heavy computation.

2.1. Downstream Tasks $\mathcal{T}$

Following the design principle of [68], we study diverse downstream tasks including Caltech101 [22], Flowers102 [42], Sun397 [63], and Patch Camelyon [62]. These tasks are representative of general object recognition, fine-grained object recognition, scenery image classification, and medical image classification, respectively. Table 1 in Appendix A.1 provides more details of these tasks. A common theme is that their training sets are all medium-sized, making it especially beneficial to leverage pre-trained checkpoints to avoid overfitting.

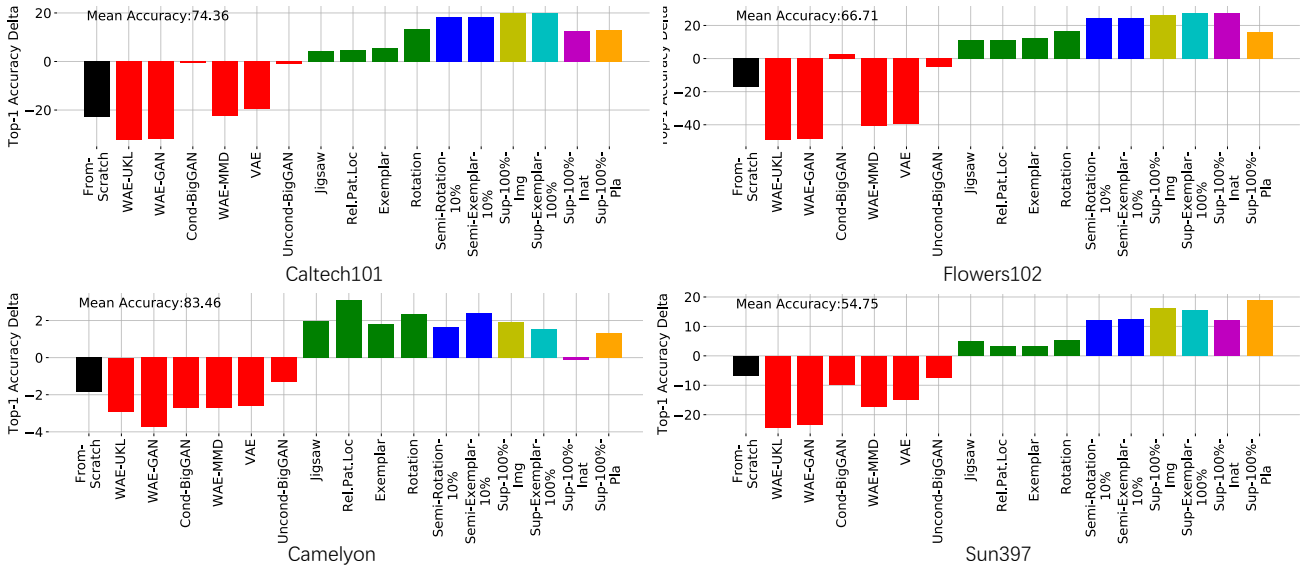

Figure 1. Fine-tuning the checkpoints in Group I on four downstream tasks. We keep the best fine-tuning accuracy for each checkpoint-task pair after hyper-parameter sweeping. For better visualization, the values are offset by their mean (cf. Table 4 in Appendix for the absolute values). (Best viewed in color. Red: generative models. Black: From-Scratch. Green: self-supervised models. Blue: semi-supervised models. Yellow, Pink, and Orange: supervised models trained on ImageNet, Inatualist, and Places365, respectively. Cyan: a hybridly-supervised model.)

2.2. Neural Checkpoints $\mathcal{C}$

Thanks to the broad use of DNNs, one may collect neural checkpoints of various types from multiple sources. To simulate this situation, we construct a rich set of checkpoints and separate them into three groups according to the pre-training strategies and network architectures.

Group I: Checkpoints of mixed supervision. The first group of checkpoints are pre-trained with mixed supervision till convergence, including supervised learning, self-supervised learning, semi-supervised learning, and the discriminators or encoders in deep generative models. It consists of 16 ResNet-50s [28]. We borrow 14 models pre-trained on ImageNet [14] from [68]. Among them, four are pre-trained by self-supervised learning (Jigsaw [43], Relative Patch Location [15], Exemplar [17], and Rotation [23]), six are the discriminators of generative models (WAE-UKL [50], WAE-GAN, WAE-MMD [59], Cond-BigGAN, Uncond-BigGAN [9], and VAE [34]), two are based on semi-supervised learning (Semi-Rotation-10% and Semi-Exemplar-10% [67]), one is by fully supervised learning (Sup-100%-Img [28]), and one is trained with a hybrid supervised loss (Sup-Exemplar-100% [67]). We also add two supervised checkpoints pre-trained on iNaturalist (Sup-100%-Inat) [61] and Places365 (Sup-100%-Pla) [69], respectively. Using the evaluation procedure $\mathbf{G}(\theta_i, t)$ (cf. equation (1)), we obtain their final accuracies on the downstream tasks described in Section 2.1.

Figure 1 shows the best fine-tuning accuracies offset by their mean for better visualization, and Table 4 (in Appendix) contains the absolute accuracy values. We include the training from scratch (From-Scratch) for comparison. Most of the checkpoints yield significantly better fine-tuning results than From-Scratch. Some of the discriminators in generative models, however, under-perform From-Scratch. The highest-performance checkpoints change from one downstream task to another.

Group II: Checkpoints at different pre-training stages. This group comprises 12 ResNet-50s pre-trained by fully supervised learning on ImageNet, iNaturalist, and Places-365. We save a checkpoint right after each learning rate decay, resulting in four checkpoints per dataset. Figure 2 and Table 5 in Appendix show the best fine-tuning accuracies over the four downstream tasks, where Img-90k refers to the checkpoint trained on ImageNet for 90k iterations. Interestingly, the downstream tasks favor different pre-training sources, indicating the necessity of studying between-task transferabilities [68, 66]. However, the source task information may be not known for all checkpoints. Moreover, the converged model over a source task does not always transfer the best to a downstream task (cf. Img-270k vs. Img-300k on Camelyon, Inat-270k vs. Inet-300k on Flowers102, etc.). We hence construct this NeuCRaB for studying the ranking of neural checkpoints without accessing how one pre-trained the checkpoints over which dataset.

Group III: Checkpoints of heterogeneous architec-

tures. Kornblith *et al.* [35] show that better network architectures can learn better features that can be transferred across vision-based tasks. Therefore, we construct the third group of checkpoints by using different neural architectures. Four of them belong to the Inception family [57], one is Inception-ResNet-v2 [56], six come from the MobileNet family [29], and two are from the ResNet-v1 family [28]. We train them on ImageNet till convergence. Figure 3 and Table 6 in Appendix visualize their fine-tuning accuracies on the four downstream tasks.

2.3. Evaluation Metrics $\mathcal{M}$

We use multiple metrics (cf. $\mathcal{M}$ in eq. (3)) to evaluate the checkpoint ranking measures.

Recall@k: A practitioner may have resources to test up to k checkpoints for their task of interest. We consider it a success if a measure ranks the highest-performance checkpoint into the top k. A measure's Recall@k is the ratio between the number of downstream tasks on which it succeeds and the total number of tasks. We employ $k = 1$ and $k = 3$ in the experiments.

Top-k relative accuracy (Rel@k): Given a task, a ranking measure returns an ordered list of the checkpoints. If the measure selects a high-performing checkpoint to the top k despite that it misses the highest-performance one, we do not want to overly penalize it. This Rel@k is the ratio between the best fine-tuning accuracy on the downstream task with the top k checkpoints and the the best fine-tuning accuracy with all the checkpoints.

Pearson correlation: We incorporate Pearson's r [45] to compute the linear correlation between a measure' ranking scores $\mathbf{R}_t$ and the evaluation procedure's final accuracies $\mathbf{G}_t$.

Kendall ranking correlation: We also include Kendall's τ [32] to measure the ordinal association between a ranking measure $\mathbf{R}$ and the evaluation procedure $\mathbf{G}$ for each task. After all, what matter is the order of the checkpoints rather than the precise ranking scores.

3. Checkpoint Ranking Methods

In this section, we describe some intuitive neural checkpoint ranking methods. These methods strive to achieve high correlation with the checkpoint evaluation procedure $\mathbf{G}$ at low computation cost.

3.1. Fine-tuning with Early Stopping

If there is no constraint over computing, the evaluation procedure $\mathbf{G}$ itself becomes the gold ranking measure. Hence, a natural ranking method is the fine-tuning with early stopping, by which the model is far from convergence. The premature models' test accuracies are the ranking scores. Experiments reveal that it is hard to forecast from the premature models.

3.2. Linear Classifiers

We derive the second ranking method also from the evaluation procedure $\mathbf{G}$, which replaces a checkpoint's output layer by a linear classifier tailored for the downstream task. We train the linear classifier while freezing the other layers. The ranking score equals the classifier's test accuracy. It is worth mentioning that self-supervised learning [11, 26, 24] often adopts this practice as well to evaluate the learned feature representations. We shall see that the linear separability of the features extracted from a checkpoint is a strong indicator of the performance of fine-tuning the full checkpoint.

3.3. Mutual Information

Suppose the extracted features' quality well correlates with a checkpoint's final accuracy on a downstream task. Besides the linear separability above, we can rank the checkpoints by their mutual information between the high-dimensional features and discrete labels of the downstream task. We employ the state-of-the-art I_α mutual information estimator [46], where α controls the trade-off between variance and bias. It is a variational lower bound parameterized by a neural network. Belghazi *et al.* [6] report that the neural estimators generally outperform prior mutual information estimations, especially when the variables are high-dimensional. We use the code released by the authors to calculate I_α [46].

3.4. LEEP for the Checkpoints with Classification Heads

To rank the checkpoints pre-trained over classification source tasks, the recently proposed LEEP [41] measure is directly applicable despite that it was originally designed for between-task transfer. Denote by $\mathcal{Z}$ the classification space of a checkpoint θ. We can interpret $\theta(x)_z$, the z-th (softmax) output element, as the probability of classifying the input x into the class $z \in \mathcal{Z}$. Given a downstream task $t \sim \mathcal{T}$ and its test set $\{(x_j, y_j)\}_{j=1}^n$, the LEEP ranking score for the checkpoint θ is calculated by

$$\begin{aligned} \mathbf{R}_{\text{LEEP}}(\theta, t) &:= \frac{1}{n} \sum_{j=1}^{n} \log P(y_j | x_j, \theta, t) \\ P(y|x, \theta, t) &:= \sum_{z \in \mathcal{Z}} \hat{P}(y|z) \theta(x)_z \end{aligned} \tag{4}$$

where $\hat{P}(y|z)$ is the empirical conditional distribution of the downstream task's label y given the source label $z \in \mathcal{Z}$, and $P(y|x, \theta, t)$ is a "dummy" classifier, which firstly draws a label z from the checkpoint $\theta(x)$ and then draws a class y from the conditional distribution $\hat{P}(y|z)$.

Denote by $\{x_j, y_j\}_{j=1}^{\tilde{n}}$, $y \in \mathcal{Y}$, the downstream task's training set. LEEP computes the conditional distribution $\hat{P}(y|z)$ by "counting". The joint distribution $\hat{P}(y, z)$ due to the checkpoint θ is

$$\hat{P}(y, z) = \frac{1}{\tilde{n}} \sum_{j:y_j=y} \theta(x_j)_z, \tag{5}$$

which gives rise to the conditional distribution $\hat{P}(y|z) = \hat{P}(y, z)/\hat{P}(z) = \hat{P}(y, z)/\sum_{y\in\mathcal{Y}} \hat{P}(y, z)$.

In the experiments, LEEP and the linear classifier are the second best ranking methods for the checkpoints pre-trained for classification. However, LEEP's computation cost is high when a checkpoint's classification output is high-dimensional (e.g., iNaturalist contains more than 8000 classes). Besides, its softmax estimation of the classification probability $\theta(x)_z$ is often poorly calibrated [25]. Finally, it does not apply to the checkpoints with no classification heads.

3.5. $\mathcal{N}$LEEP

We propose a variation to LEEP that applies to all types of checkpoints including those obtained from unsupervised learning and self-supervised learning. It can also avoids the overly confident softmax.

Feeding the training data of a downstream task into a checkpoint, we obtain their feature representations. The representations are thousands of dimensions, depending on the checkpoint's neural architecture. We reduce their dimension by using the principal component analysis (PCA). Denote by s the resultant low-dimensional representation of the input x.

We then fit a Gaussian mixture model (GMM), $P(s) = \sum_{v\in\mathcal{V}} \pi_v \mathcal{N}(s|\mu_v, \Sigma_v)$, to the training set $\{s_j\}_{j=1}^{\tilde{n}}$, where $\mathcal{V}$ is a collection of all the Gaussian components, and π_v, $v \in \mathcal{V}$, are the mixture weights. It is convenient to compute the posterior distribution:

$$P(v|x) = P(v|s) \propto \pi_v \mathcal{N}(s|\mu_v, \Sigma_v), \tag{6}$$

which is arguably more reliable than the class assignment probability $\theta(x)_z$ output by the softmax classifier because we fit GMM to the downstream task's training data, whereas the softmax classifier is learned from a different source task.

Hence, we arrive at an improved ranking measure, named $\mathcal{N}$LEEP, by replacing $\theta(x)_z$, the probability of classifying an input x to the class z, in equations (4–5) by the posterior distribution $P(v|x)$.

4. Experiments on NeuCRaB

There are free parameters in each of the ranking methods. Before presenting the main results, we study how the free parameters in $\mathcal{N}$LEEP affect its checkpoint ranking performance. Figure 2 illustrates $\mathcal{N}$LEEP's Kendall's τ values over Groups I and II with different PCA feature dimensions and the numbers of Gaussian components. Each Kendall's τ is averaged across all the downstream tasks; the higher, the better. Along the vertical axes, we change the feature dimensions by keeping different percentages of the PCA energies; PCA50 means the percentage is 50%. Along the horizontal axes, we adopt different numbers of Gaussian components in GMM; $2\times$ means the number is twice the class number of the downstream task. Notably, the Kendall's τ values remain relatively stable. In the remaining experiments with $\mathcal{N}$LEEP, we fix the PCA energy to 80% and the number of Gaussian components five times the class number of a downstream task.

4.1. Comparison Results

Tables 1, 2, and 3 show the checkpoint ranking methods' performance on Groups I (checkpoints of mixed supervision), II (different pre-training stages), and III (heterogeneous architectures), respectively. We also union the three groups and present the corresponding ranking performance in Table 2 in Appendix. The numbers in the tables are the average over all downstream tasks. In addition to the evaluation metrics detailed in Section 2.3, the GFLOPS column measures the ranking methods' computing performance; the lower, the better.

We report multiple variations of the ranking methods in the tables. Fine-tuning is computationally expensive, so we stop it after one or five epochs. The linear classifiers are less so as we save the feature representations of downstream tasks' after one forward pass to the checkpoints. We report the linear classifiers' ranking results after training them for one epoch, five epochs, and convergence. We test $\alpha = 0.01$ and $\alpha = 0.50$ in the I_α mutual information estimator. Additionally, we experiment with I_α after reducing the feature dimensions by using PCA.

4.2. Main Findings

In each column of Tables 1, 2, 3, and Table 2 in Appendix, we highlight the best and second best by the bold font and underscore, respectively.

The mutual information fails to rank high-performing checkpoints to the top and even produces negative Pearson and Kendall correlations, probably because of the features' high dimensions. Reducing the feature dimensions by PCA significantly improves the mutual information's ranking performance; MI w/ PCA (α=0.01) leads to the second best Rel@1, Recall@3 and Rel@3 among the ranking methods in Group III, the checkpoints of heterogeneous neural architectures. Varying α in the I_α mutual information estimator [46] can control the trade-off between variance and bias. MI w/ and w/o PCA (α=0.01) perform better than MI w/ and w/o PCA (α=0.50), respectively. It indicates that neural checkpoint ranking requires low-bias MI

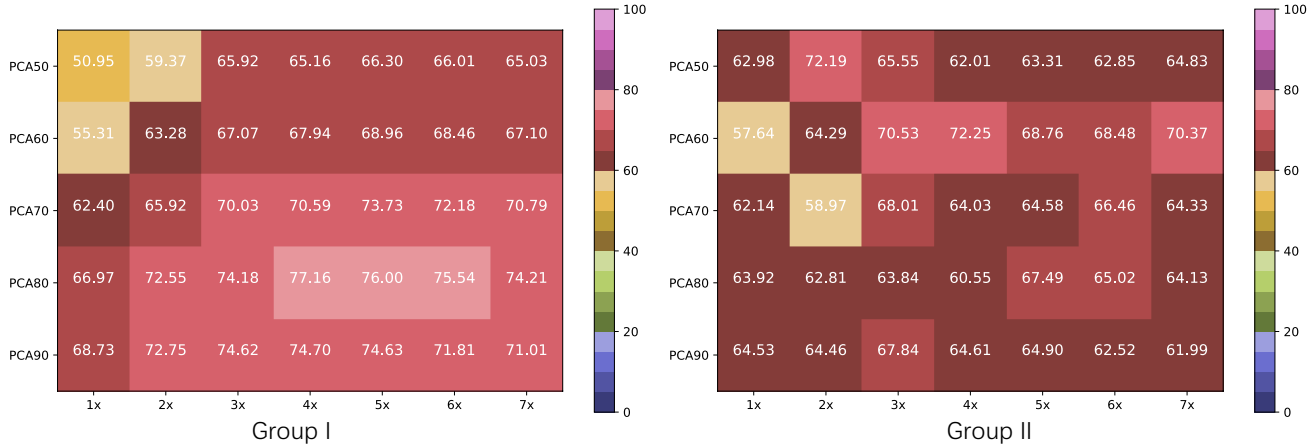

Figure 2. $\mathcal{N}$LEEP' checkpoint ranking performance, evaluated by Kendall's τ, on Groups I and II in NeuCRaB. We vary the PCA feature dimension and the number of Gaussian components in GMM.

estimator since smaller α means low-bias but high-variance estimation.

Fine-tuning up to some epochs turns out the worst ranking methods because it leads to low correlation with the groundtruth ranking and yet incurs heavy computation. Similarly, training the linear classifier up to one or five epochs does not perform well except in Group II. These results indicate that it is difficult to forecast the checkpoints' final performance from premature models. Fine-tuning (5 epochs) and Linear (5 epochs) perform better than Fine-tuning (1 epoch) and Linear (1 epoch) in terms of Person and Kendall correlation, respectively. However, they all fail to select the top checkpoint in Group I and Group III since they produce lower Recall@1 and Recall@3 than others. One possible reason is that the evaluation accuracies of checkpoints in the early stage tend to have large variance.

Feature qualities before fine-tuning the checkpoints. If we train the linear classifiers till convergence, they become the best in Group II, and the second best checkpoint ranking method in Groups I and III in terms of Pearson and Kendall correlations. It can also produce better Recall@1 and Recall@3 than Linear (1 epoch) and Linear (5 epoch) in Groups I, II and III since the evaluation accuracies of converged models are more stable than models in the early training stage. Note that the linear classifiers' accuracies, i.e., the ranking scores, imply the linear separability of the features extracted by the checkpoints. Recall that the mutual information with PCA feature dimension reduction is among the second best (Rel@1, Recall@3 and Rel@3) in Group III. Since both methods measure the feature representations' quality by the downstream tasks' labels, we conjecture that the quality of the features is a strong indicator of the checkpoints' final fine-tuning performance on the downstream tasks. It would be interesting to study other feature quality measures beyond the linear separability and mutual information in future work.

$\mathcal{N}$LEEP performs consistently well in all the groups of checkpoints over all the evaluation metrics with the lowest computation cost . In contrast, the original LEEP measure is not applicable to Group I, the checkpoints of mixed supervision, because it requires that the checkpoints have a classification output layer. Overall, LEEP is the second best over all evaluation metrics among the ranking methods in Groups II and III, whose checkpoints all have a classification output layer. Specifically, LEEP can produce the second best Recall@1, Recall@3 and Rel@3 in Group II, and the best Recall@3, the best Rel@3 and the second best Kendall correlation in Group III. It is a more consistent indicator than fine-tuning, linear classifier, or MI based ranking methods. However, LEEP can not produce better results than $\mathcal{N}$LEEP, and it requires slightly larger GFLOPS due to the extra computation cost from the classification head.

We conjecture that $\mathcal{N}$LEEP outperforms LEEP mainly because GMMs calibrate the posterior probabilities better than the checkpoints' softmax classifiers. The checkpoint ranking quality of LEEP score hinges on the performance of the 'dummy classifier' – $P(y|x, \theta, t)$, and $\theta(x)_z$ is the key element to calculate it. However, $\theta(x)$ can be poorly calibrated [41] and it can not represent a true probability. In contrast, $P(v|x)$ used in $\mathcal{N}$LEEP is indeed the probability that the sample belongs to one cluster from a mixture of Gaussian distributions and it can remedy the poor-calibrated problem in LEEP.

Computational costs. Moreover, we highlight the GFLOPS column in the tables. $\mathcal{N}$LEEP and LEEP exhibit a clear advantage over the other checkpoint ranking methods in terms of computing. The main reason is that $\mathcal{N}$LEEP and LEEP can avoid intensive computation from neural network

Method	Recall@1	Rel@1	Recall@3	Rel@3	Pearson	Kendall	GFLOPS
Linear (1 epoch)	0.00	96.97	25.00	98.79	23.56	18.44	4.95E4
Linear (5 epoch)	25.00	98.79	50.00	98.94	49.77	32.33	4.97E4
Linear (converged)	50.00	99.63	**75.00**	**99.65**	68.97	53.43	5.33E4
Fine-tune (1 epoch)	25.00	97.45	25.00	97.66	30.25	22.15	6.51E5
Fine-tune (5 epoch)	0.00	91.09	25.00	98.61	48.19	36.78	4.28E6
MI (α=0.01) [46]	0.00	64.67	0.00	87.96	2.39	-0.31	1.62E5
MI (α=0.50)	0.00	66.71	25.00	90.31	-4.91	-13.05	1.62E5
MI w/ PCA (α=0.01)	0.00	89.45	50.00	99.27	16.16	20.67	5.58E4
MI w/ PCA (α=0.50)	0.00	86.49	25.00	94.28	-24.72	-16.06	5.58E4
LEEP [41]	–	–	–	–	–	–	–
$\mathcal{N}$LEEP	**75.00**	**99.65**	**75.00**	**99.65**	**84.30**	**76.00**	**12.85**

Table 1. Checkpoint ranking results on Group I, the checkpoints of mixed supervision (GFLOPS excludes a forward pass on training data, which takes 3.04E5 GFLOPS shared by all methods)

Method	Recall@1	Rel@1	Recall@3	Rel@3	Pearson	Kendall	GFLOPS
Linear (1 epoch)	0.00	96.46	25.00	98.79	27.01	24.24	4.95E4
Linear (5 epochs)	50.00	99.57	**100.00**	**100.00**	55.07	51.28	4.97E4
Linear (converged)	75.00	99.95	**100.00**	**100.00**	**79.30**	**68.60**	5.33E4
Fine-tune (1 epoch)	25.00	99.05	25.00	99.47	19.61	15.52	6.51E5
Fine-tune (5 epochs)	25.00	99.55	**100.00**	**100.00**	68.47	58.33	4.28E6
MI (α=0.01) [46]	0.00	94.84	25.00	97.43	-29.41	-17.81	1.62E5
MI (α=0.50)	0.00	96.66	0.00	97.03	-11.36	-10.21	1.62E5
MI w/ PCA (α=0.01)	50.00	99.60	75.00	99.85	52.14	51.34	5.58E4
MI w/ PCA (α=0.50)	0.00	96.68	50.00	99.52	23.73	17.09	5.58E4
LEEP [41]	75.00	99.44	75.00	99.90	50.36	55.49	378.31
$\mathcal{N}$LEEP	**100.00**	**100.00**	**100.00**	**100.00**	72.84	67.49	**12.95**

Table 2. Checkpoint ranking results on Group II, the checkpoints at different pre-training stages (GFLOPS excludes a forward pass on training data, which takes 3.04E5 GFLOPS shared by all)

Method	Recall@1	Rel@1	Recall@3	Rel@3	Pearson	Kendall	GFLOPS
Linear (1 epoch)	25.00	98.17	25.00	99.35	30.14	13.80	3.37E4
Linear (5 epoch)	25.00	98.98	25.00	99.63	33.45	18.95	3.38E4
Linear (converged)	25.00	**99.66**	25.00	99.72	63.55	36.91	3.62E4
Fine-tune (1 epoch)	0.00	98.28	25.00	99.80	17.61	11.59	4.43E5
Fine-tune (5 epoch)	25.00	98.62	25.00	99.68	25.72	15.72	2.91E6
MI (α=0.01) [46]	25.00	98.29	25.00	99.34	4.42	2.94	1.30E5
MI (α=0.50)	25.00	98.36	25.00	99.37	-9.79	-6.81	1.30E5
MI w/ PCA (α=0.01)	0.00	99.18	50.00	99.82	61.94	38.83	5.56E4
MI w/ PCA (α=0.50)	0.00	96.34	0.00	98.47	33.17	21.26	5.56E4
LEEP [41]	25.00	97.36	**75.00**	**99.90**	42.99	45.06	247.56
$\mathcal{N}$LEEP	25.00	**99.66**	25.00	99.70	**66.94**	**51.14**	**12.68**

Table 3. Checkpoint ranking results on Group III, the checkpoints of heterogeneous architectures (GFLOPS excludes a forward pass on training data, which takes 2.73E5 GFLOPS shared by all)

training, and they only require one forward pass through the training data.

Comparing different groups of the checkpoints. Checkpoint ranking on different groups of checkpoints varies in degrees of difficulty. The most challenging group is Group III, the checkpoints of heterogeneous neural architectures. All the ranking methods produce lower correlations with the groundtruth ranking, and they can barely select the top checkpoints in this group. The main reason is that the neural architectures matter for transfer learning [35]. Besides, heterogeneous neural architectures can demonstrate various performance even if we train them from scratch on downstream tasks. Ranking neural checkpoints by the feature

representations of the last layer is not sufficient for those checkpoints. We may explore more advanced ranking methods considering the structures of the deep neural networks in the future.

Checkpoint ranking on Group II is easier than on Group I since all the ranking methods can achieve relatively better results over all evaluation metrics in Group II. The results indicate that checkpoints with various training strategies (Group I) can bring more complex knowledge from source domains, comparing with checkpoints with different early stopping stages (Group II). In addition, fine-tuning the entire models and training linear classifiers up to one or five epochs perform significantly better on Group II since those ranking methods are based on early stopping as well.

Additional experiments in the supplementary materials. To simulate a sufficiently large pool of checkpoints in the real applications, we finally combine the checkpoints in Group I, II, and III into one large group and conduct checkpoint ranking experiments on it. We also add one more group of checkpoints with ResNet-101s [28] to evaluate the checkpoint ranking on deeper models. Please see more details in Appendix A.3 and A.4. We also take object detection and instance segmentation as downstream tasks and conduct preliminary experiments on VOC [12] and Cityscapes [13]. Please refer to Appendix A.6 to see detailed discussions.

Although the benchmark can be easily extended to many downstream tasks in other modalities, e.g., voice, text, and cross-modal modalities, we steer our attention into comparing several intuitive ranking measures on the variants of checkpoints, covering different training strategies, source domains, and architectures at a range of early stopping stages. We formalize the checkpoint ranking idea, demonstrate the existence of an effective yet lightweight measure, $\mathcal{N}$LEEP, and hope it can shed light on more efficient ranking methods and practical applications.

5. Related Work

Our work is broadly related to task transferability and neural networks' generalization gap.

Task transferability. A task usually refers to a joint distribution over input and label. Task transferability aims to predict how well a deep neural network pre-trained on a source task transfers to the target task. One may estimate the task transferability by data similarities regardless of models being used. Some work in this line includes conditional entropy [60], data set distance as optimal transport [2], F-relatedness [7], A-distance [33], and discrepancy distance [38]. Besides, Poole *et al.* [46] derived information theoretic bounds. These methods are generally hard to compute in practice and rely on the availability of the source data. Some recent task transferability estimators involve both data and the models. Taskonomy [66] is a fully computation method, where task similarity scores are obtained by transfer learning experiments. Dwivedi *et al.* [19] analyzed the representation similarities to construct a task taxonomy. Besides the models trained on source tasks, all these methods also require a fine-tuned or independently trained model from the target task. In contrast, our work aims to find checkpoint ranking measures that are lightweight in computing and requires no access to the source tasks.

Recent works demonstrated that using pre-trained checkpoints that have similar feature representations as the target task's representations can improve transfer learning [19, 54, 55]. Song *et al.* [54, 55] employed attribution maps to compare two models and then quantified transferabilities by the similarity of two models. Those approaches all require a converged model on target datasets, incurring intensive computation. However, we want to design a lightweight method for ranking checkpoints, ideally without any training procedures.

Predicting neural networks' generation gap. The difference between a model's performance on the training data versus its performance on test data is known as the generalization gap. It is practically useful and theoretically impactful to predict a neural network's generalization gap. Most recent work does so by finding a set of features that is predictive of the generalization, e.g., by estimating data margins [5, 21, 53]. Jiang *et al.* [30] and Yak *et al.* [64] demonstrate how the margin signatures of a neural network can predict the generalization gap with small errors. Besides, the network complexity and noise stability are also useful cues [40, 31, 5, 3]. Our problem substantially differs from predicting the neural networks' generalization gap, which is concerned with the training and test data sets that share the same underlying distribution. We instead care about the results after fine-tuning a network's checkpoint.

6. Conclusion

Deep learning has triumphed over many fields in both research and real-world applications. There must exist hundreds of thousands of DNNs trained and released by various groups. To this end, it is natural to select an existing, promising DNN checkpoint as a warm start to a training procedure when solving a new task. How to identify useful checkpoints from a large pool for the target task? Towards answering this question, we present NeuCRaB, a thorough benchmark covering diverse downstream tasks and pre-trained DNN checkpoints, along with $\mathcal{N}$LEEP, a lightweight, effective checkpoint ranking measure.

The experiments with linear classifiers and mutual information (after PCA) reveal that the features extracted from the checkpoints are good indicators of the checkpoints' potential in transfer learning. It is worth exploring other ways of evaluating the features' quality in future work. It

is also interesting to investigate the checkpoints' inherent signatures, such as topology and stability to noise, which might be informative of their transferabilities. Finally, some learning-based methods in predicting networks' generalization gaps are also promising for the checkpoint ranking problem.

References

[1] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6430–6439, 2019. 1

[2] David Alvarez-Melis and Nicolò Fusi. Geometric dataset distances via optimal transport. *arXiv preprint arXiv:2002.02923*, 2020. 8

[3] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. *arXiv preprint arXiv:1802.05296*, 2018. 8

[4] Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. In *Advances in neural information processing systems*, pages 3365–3373, 2014. 1

[5] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017. 1, 8

[6] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: mutual information neural estimation. *arXiv preprint arXiv:1801.04062*, 2018. 4

[7] Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines*, pages 567–580. Springer, 2003. 8

[8] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5050–5060, 2019. 1

[9] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 3

[10] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009. 1

[11] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 1, 4

[12] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010. 8

[13] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 8

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 3, 12

[15] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015. 1, 3

[16] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655, 2014. 1

[17] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. In *Advances in neural information processing systems*, pages 766–774, 2014. 3

[18] Kshitij Dwivedi, Jiahui Huang, Radoslaw Martin Cichy, and Gemma Roig. Duality diagram similarity: a generic framework for initialization selection in task transfer learning. *arXiv preprint arXiv:2008.02107*, 2020.

[19] Kshitij Dwivedi and Gemma Roig. Representation similarity analysis for efficient task taxonomy & transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12387–12396, 2019. 8

[20] Eric Eaton, Terran Lane, et al. Modeling transfer relationships between learning tasks for improved inductive transfer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 317–332. Springer, 2008. 1

[21] Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large margin deep networks for classification. In *Advances in neural information processing systems*, pages 842–852, 2018. 8

[22] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006. 2, 12, 14

[23] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018. 3

[24] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 4

[25] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017. 2, 5

[26] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020. 1, 4

[27] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *Proceedings of the IEEE international conference on computer vision*, pages 4918–4927, 2019. 1

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 3, 4, 8, 12

[29] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 4

[30] Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the generalization gap in deep networks with margin distributions. *arXiv preprint arXiv:1810.00113*, 2018. 1, 8

[31] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. Generalization in deep learning. *arXiv preprint arXiv:1710.05468*, 2017. 1, 8

[32] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. 4

[33] Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *VLDB*, volume 4, pages 180–191. Toronto, Canada, 2004. 8

[34] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3

[35] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2661–2671, 2019. 1, 4, 7

[36] Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016. 1

[37] Yucen Luo, Jun Zhu, Mengxi Li, Yong Ren, and Bo Zhang. Smooth neighbors on teacher graphs for semi-supervised learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8896–8905, 2018. 1

[38] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. *arXiv preprint arXiv:0902.3430*, 2009. 8

[39] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018. 1

[40] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in neural information processing systems*, pages 5947–5956, 2017. 1, 8

[41] Cuong V Nguyen, Tal Hassner, Cedric Archambeau, and Matthias Seeger. Leep: A new measure to evaluate transferability of learned representations. *arXiv preprint arXiv:2002.12462*, 2020. 1, 2, 4, 6, 7, 14

[42] 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. 2, 12, 14

[43] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 1, 3

[44] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009. 1

[45] Karl Pearson. Vii. note on regression and inheritance in the case of two parents. *proceedings of the royal society of London*, 58(347-352):240–242, 1895. 4

[46] Ben Poole, Sherjil Ozair, Aaron van den Oord, Alexander A Alemi, and George Tucker. On variational bounds of mutual information. *arXiv preprint arXiv:1905.06922*, 2019. 4, 5, 7, 8, 14

[47] Joan Puigcerver, Carlos Riquelme, Basil Mustafa, Cedric Renggli, André Susano Pinto, Sylvain Gelly, Daniel Keysers, and Neil Houlsby. Scalable transfer learning with expert models. *arXiv preprint arXiv:2009.13239*, 2020.

[48] Maithra Raghu, Chiyuan Zhang, Jon Kleinberg, and Samy Bengio. Transfusion: Understanding transfer learning for medical imaging. In *Advances in neural information processing systems*, pages 3347–3357, 2019. 1

[49] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in neural information processing systems*, pages 3546–3554, 2015. 1

[50] Paul Rubenstein, Olivier Bousquet, Josip Djolonga, Carlos Riquelme, and Ilya O Tolstikhin. Practical and consistent estimation of f-divergences. In *Advances in Neural Information Processing Systems*, pages 4070–4080, 2019. 3

[51] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813, 2014. 1

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[53] Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel RD Rodrigues. Robust large margin deep neural networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017. 8

[54] Jie Song, Yixin Chen, Xinchao Wang, Chengchao Shen, and Mingli Song. Deep model transferability from attribution maps. In *Advances in Neural Information Processing Systems*, pages 6182–6192, 2019. 8

[55] Jie Song, Yixin Chen, Jingwen Ye, Xinchao Wang, Chengchao Shen, Feng Mao, and Mingli Song. Depara: Deep attribution graph for deep knowledge transferability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3922–3930, 2020. 8

[56] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016. 1, 4, 12

[57] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 4

[58] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*, pages 1195–1204, 2017. 1

[59] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017. 3

[60] Anh T Tran, Cuong V Nguyen, and Tal Hassner. Transferability and hardness of supervised classification tasks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1395–1405, 2019. 8

[61] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018. 3, 12

[62] Bastiaan S Veeling, Jasper Linmans, Jim Winkens, Taco Cohen, and Max Welling. Rotation equivariant cnns for digital pathology. In *International Conference on Medical image computing and computer-assisted intervention*, pages 210–218. Springer, 2018. 1, 2, 12, 14

[63] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010. 2, 12, 14

[64] Scott Yak, Javier Gonzalvo, and Hanna Mazzawi. Towards task and architecture-independent generalization gap predictors. *arXiv preprint arXiv:1906.01550*, 2019. 8

[65] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014. 1

[66] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 1, 3, 8

[67] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE international conference on computer vision*, pages 1476–1485, 2019. 3

[68] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. The visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019. 1, 2, 3, 12

[69] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 3, 12

[70] Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin D Cubuk, and Quoc V Le. Rethinking pre-training and self-training. *arXiv preprint arXiv:2006.06882*, 2020. 1

[71] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 12

A. Appendix

In this appendix, we provide the following details to support the main text:

Section A.1: Descriptions of the 4 downstream tasks.

Section A.2: Training details of pre-training and fine-tuning.

Section A.3: Comparison results on the combined group of checkpoints in Groups I, II and III.

Section A.4: Another group of checkpoints with ResNet101s at different pre-training stages.

Section A.5: More experiment results on Groups I-IV.

Section A.6: Neural checkpoints ranking on object detection and instance segmentation.

A.1. Downstream tasks

In this section, we describe the datasets used for the downstream tasks as shown in Table 4. More specifically, **Caltech101** [22] contains 101 classes, including animals, airplanes, chairs and etc, the image size varies from 200 to 300 pixels per edge. **Flowers102** [42] have 102 classes, with 40 to 248 training images per class, each image has at least 500 pixels. **Patch Camelyon** [62] contains 327,680 images of histopathologic scans of lymph node sections with image size of 96x96, which is collected to predict the presence of metastatic tissue. **Sun397** [63] is a scenery benchmark with 397 classes, including cathedral, staircase, shelter, river, or archipelago. There are at least 100 images per class. The images are in 200x200 or higher resolutions. We believe the dataset portfolio well represents a broad set of vision tasks.

A.2. Hyper-parameter Sweep

We adopt the similar experiment setting as in [68] to fine-tune the neural networks on the downstream tasks. Specifically, we set the batch size to 512 and use SGD with momentum of 0.9. We do not use weight decay for fine-tuning, and we set it to be 0.01 times the learning rate [71] when training from scratch. We perform per-task hyper-parameter search. For each task, we sweep the learning rate in $\{0.0001, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.5\}$ and the training step in $\{2500, 5000, 10000, 15000, 20000, 400000\}$. We incorporate inception data augmentation [56] for pre-training checkpoints and we do not use data-augmentation when we fine-tune the neural networks on the downstream tasks to emphasize the effect of transfer learning.

A.3. Comparison results on all checkpoints in Groups I, II, III

To obtain a comprehensive analysis, we also consolidate the checkpoints from Group I, II and III into one group (including 41 checkpoints in total) and then apply the ranking methods on it. Table 5 shows the comparison results. The results further evaluate our observations in Section 4 of the main text. $\mathcal{N}$LEEP performs consistently well on the big group of checkpoints with lowest computation cost. Linear separability of the feature representation is also a good indicator for ranking a large group of neural checkpoints. Fine-tuning with early stopping and mutual information estimator produce poor correlations. The ranking qualities of different ranking methods on the large group of checkpoints are in sharper contrast than on small groups. For instance, the Pearson's r of $\mathcal{N}$LEEP *vs.* Finetune (5 epochs) on the large group is 83.71 *vs.* 27.84 but they perform 72.84 *vs.* 68.47 on Group II (Table 2 in the main text). It indicates that $\mathcal{N}$LEEP is a low-variance and low-bias checkpoint ranking estimator, while early stopping may produce high-variance ranking results.

A.4. Group IV: Supervised ResNet101s

We incorporate another group of checkpoints, including 12 ResNet101 [28] models pre-trained by fully supervised learning on ImageNet [14], iNaturalist [61], and Places-365 [69]. We obtain the checkpoints in the same way as we have done for Group II, but with ResNet101 architecture. We want to study how different model architecture and model size affect the ranking quality.

Figure 3 and Table 10 show the fine-tuning accuracy on 4 downstream tasks. The relative fine-tuning accuracies are similar to the accuracies on Group II. We also observe that a converged checkpoint does not necessarily demonstrates the best performance on the downstream tasks (cf. Img-270k is better than Img-300k on Flowers102 [42]). Table 6 shows the comparison results of ranking methods on those checkpoints. The relative performance among the ranking methods is similar to what they do in Group II (Table 2 in the main text). Except that they perform better on ResNet101s, e.g., Linear (converged) can achieve 68.60 in terms of Kendall's τ on ResNet50s versus 73.48 on ResNet101s, $\mathcal{N}$LEEP can get 72.84 in terms of Pearson's r on ResNet50s versus 83.22 on ResNet101s. The observation reveals that the ranking of deeper checkpoints may be more predictable than shallow ones.

A.5. More experimental results on Groups I-IV

We show more comparison results on NeuCRaB in this section. Figures 4 and 5 show the best fine-tuning accuracies offset by their mean (for better visualization) on Groups II and III, respectively. Table 7, 8, 9, 10 demonstrate the

absolute best fine-tuning accuracies on Groups I-IV, respectively.

A.6. Neural checkpoint ranking for object detection and instance segmentation

We also evaluate on object detection and segmentation tasks, and show the results in Tables 11. Specifically, we incorporate the recent self-supervised MoCo models (MoCov1, MoCov2, MoCov2-800epoch) and a ResNet50 model (supervised pretrained on ImageNet) into a new group of checkpoints. We evaluate checkpoint ranking on Pascal VOC (object detection) and Cityscapes (instance segmentation). In order to adapt $\mathcal{N}$LEEP to detection and segmentation tasks, we assign multiple ground truth labels for one image if it includes multiple object categories and extract the image-level features to perform GMM. We adapt $\mathcal{N}$LEEP to detection and segmentation tasks by assigning multi-labels to images with multiple object categories. The experiment results demonstrate that $\mathcal{N}$LEEP consistently outperforms the fine-tune and linear evaluation based approachs. We plan to include more diverse downstream tasks in NeuCRaB to facilitate future research.

Dataset	Training	Evaluation	Number of Classes
Caltech101 [22]	3060	6084	101
Flower102 [42]	2040	6149	102
Patch-Camelyon [62]	262144	32768	2
Sun397 [63]	76128	10875	397

Table 4. Statistics of the datasets associated with the downstream tasks

Method	Recall@1	Rel@1	Recall@3	Rel@3	Pearson	Kendall	GFLOPS
Linear (1 epoch)	0.00	99.13	25.00	99.46	22.30	13.42	4.45E4
Linear (5 epochs)	0.00	99.13	25.00	99.21	42.99	31.64	4.47E4
Linear (converged)	25.00	99.42	**50.00**	99.73	76.22	61.22	4.79E4
Fine-tune (1 epoch)	0.00	96.69	0.00	98.16	3.84	6.50	5.85E5
Fine-tune (5 epochs)	0.00	**99.49**	0.00	99.49	27.20	27.16	3.84E6
MI (α=0.01) [46]	0.00	77.50	0.00	81.44	1.12	7.16	1.52E5
MI (α=0.50)	0.00	66.51	0.00	90.07	-4.05	-14.22	1.52E5
MI w/ PCA (α=0.01)	0.00	89.18	**50.00**	**99.84**	12.14	20.99	5.57E4
MI w/ PCA (α=0.50)	0.00	97.07	0.00	98.70	-14.03	-2.39	5.57E4
LEEP [41]	–	–	–	–	–	–	–
$\mathcal{N}$LEEP	**50.00**	99.47	**50.00**	99.78	**83.71**	**68.18**	**12.86**

Table 5. Comparison results on all checkpoints in Group I, II, III (GFLOPS excludes a forward pass on training data, which takes 2.73E5 GFLOPS shared by all).

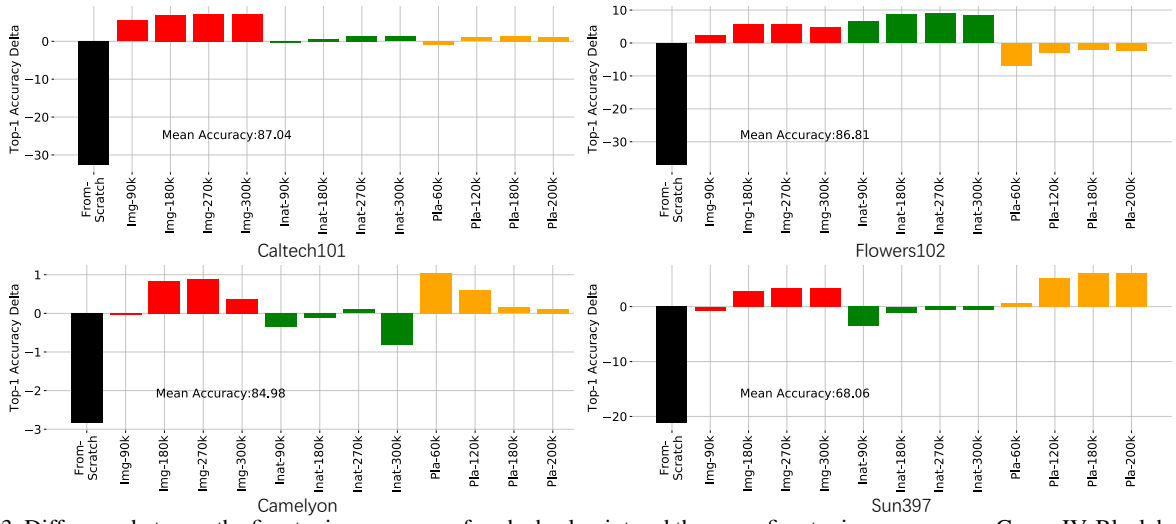

Figure 3. Difference between the fine-tuning accuracy of each checkpoint and the mean fine-tuning accuracy on Group IV. Black bar means From-Scratch. Red, green and orange bars represent ImageNet models, iNaturalist models and Places365 models, respectively. Img-90k means the checkpoint obtained by early stopping at the 90k-th iteration on ImageNet, and so on.

Method	Recall@1	Rel@1	Recall@3	Rel@3	Pearson	Kendall	GFLOPS
Linear (1 epoch)	0.00	98.58	25.00	98.99	46.75	27.27	1.021E5
Linear (5 epochs)	0.00	98.72	75.00	99.95	59.27	41.32	1.023E5
Linear (converged)	25.00	99.81	75.00	99.95	82.17	73.48	1.06E5
Fine-tune (1 epoch)	0.00	96.19	25.00	99.34	29.64	21.21	1.34E6
Fine-tune (5 epochs)	**75.00**	**99.98**	75.00	99.94	69.19	50.00	8.81E6
MI (α=0.01) [46]	0.00	97.25	75.00	98.46	12.96	13.21	1.62E5
MI (α=0.50)	25.00	98.60	50.00	99.54	30.16	18.21	1.62E5
MI w/ PCA (α=0.01)	0.00	99.85	75.00	99.95	51.85	48.91	5.58E4
MI w/ PCA (α=0.50)	0.00	95.99	50.00	98.41	48.64	44.31	5.58E4
LEEP [41]	25.00	99.52	75.00	99.72	54.54	46.43	378.31
$\mathcal{N}$LEEP	**75.00**	**99.98**	**100.00**	**100.00**	**83.22**	**73.80**	**12.95**

Table 6. Comparison results on Group IV (GFLOPS excludes a forward pass on training data, which takes 6.27E5 GFLOPS shared by all).

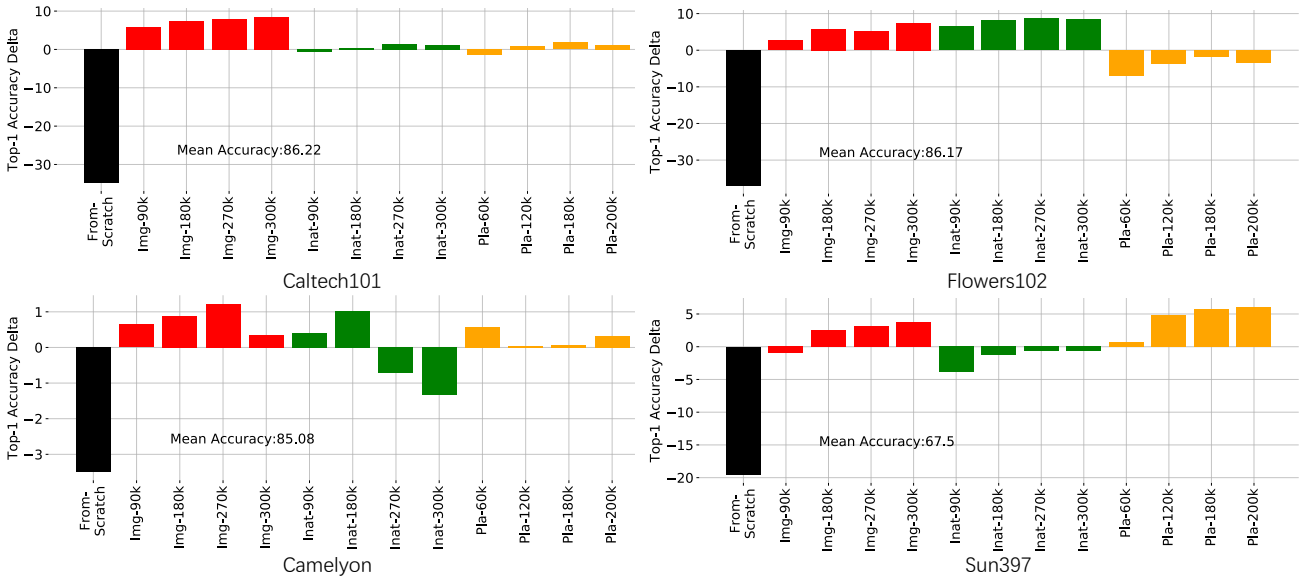

Figure 4. Difference between the fine-tuning accuracy of each checkpoint and the mean fine-tuning accuracy on Group II. Black bar means From-Scratch. Red, green and orange bars represent ImageNet models, iNaturalist models and Places365 models, respectively. Img-90k means the checkpoint obtained by early stopping at the 90k-th iteration on ImageNet, and so on.

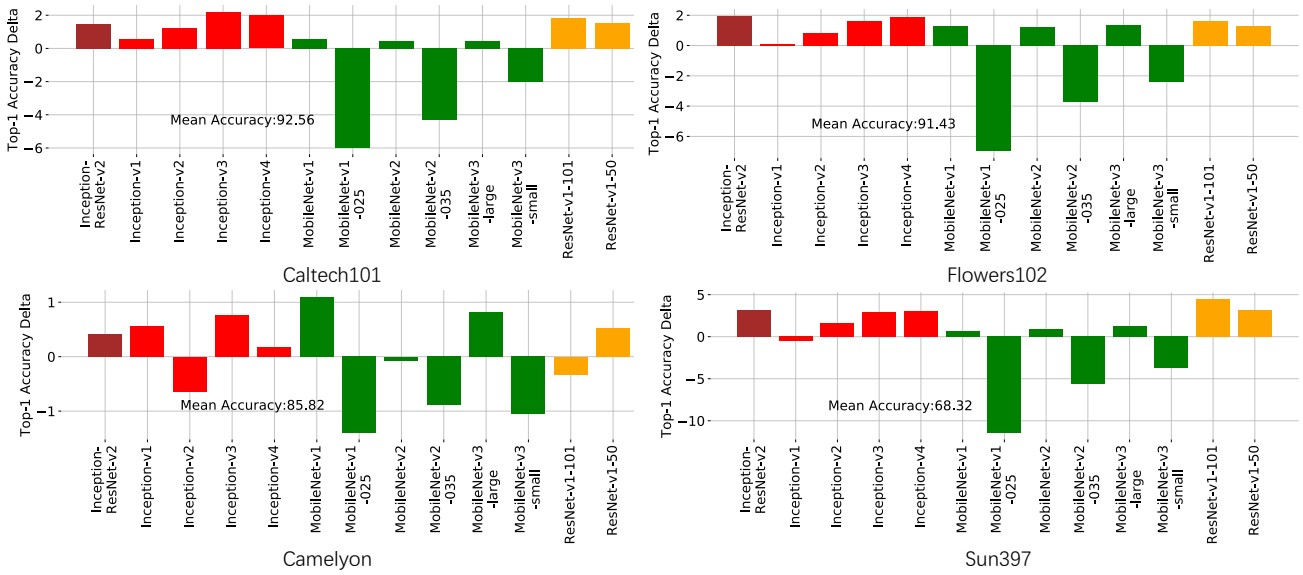

Figure 5. Difference between the fine-tuning accuracy of each checkpoint and the mean fine-tuning accuracy on Group III. The colors of bars represent the models trained with different architectures. Brown: Inception-ResNet-V2. Red: Inception family. Green: MobileNet family and their variants. Orange: ResNet-v1 family.

Dataset	From-Scratch	WAE-UKL	WAE-GAN	Cond-BigGAN	WAE-MMD	VAE	Uncond-BigGAN	Jigsaw	Rel.Pat.Loc	Exemplar	Rotation	Semi-Rotation-10%	Semi-Exemplar-10%	Sup-100%-Img	Sup-Exemplar-100%	Sup-100%-Inat	Sup-100%-Pla
Caltech101	51.44	41.99	42.37	73.88	51.83	54.91	73.15	78.85	79.09	80.02	87.91	92.73	92.77	94.42	94.5	87.00	87.27
Flowers102	49.28	17.46	17.96	69.67	26.05	26.98	61.72	77.69	78.08	78.87	83.29	91.46	91.28	93.05	93.91	94.47	82.79
Camelyon	81.59	80.55	79.71	80.73	80.73	80.87	82.14	85.43	86.58	85.27	85.81	85.09	85.83	85.37	84.98	83.33	84.77
Sun397	47.97	30.5	31.52	44.99	37.56	39.85	47.36	59.71	58.06	58.05	60.00	66.74	67.27	70.98	70.06	66.65	73.45

Table 7. Absolute fine-tuning accuracy on Group I.

Dataset	From-Scratch	Img-90k	Img-180k	Img-270k	Img-300k	Inat-90k	Inat-180k	Inat-270k	Inat-300k	Pla-60k	Pla-120k	Pla-180k	Pla-200k
Caltech101	51.44	92.08	93.55	94.18	94.73	85.69	86.54	87.66	87.43	84.91	87.11	88.01	87.48
Flowers102	49.28	88.8	91.96	91.36	93.6	92.71	94.32	94.78	94.6	79.15	82.47	84.36	82.88
Camelyon	81.59	85.73	85.97	86.3	85.43	85.48	86.12	84.37	83.75	85.67	85.13	85.16	85.4
Sun397	47.97	66.56	70.07	70.69	71.24	63.63	66.24	66.84	66.87	68.14	72.34	73.25	73.6

Table 8. Absolute fine-tuning accuracy on Group II.

Dataset	Inception-ResNet-v2	Inception-v1	Inception-v2	Inception-v3	Inception-v4	MobileNet-v1	MobileNet-v1-025	MobileNet-v2	MobileNet-v2-035	MobileNet-v3-large	MobileNet-v1-small	ResNet-v1-101	ResNet-v1-50
Caltech101	94.02	93.13	93.77	94.76	94.55	93.15	86.59	93.02	88.28	93.02	90.52	94.42	94.09
Flowers102	93.39	91.5	92.25	93.07	93.31	92.69	84.44	92.68	87.71	92.81	89.01	93.03	92.72
Camelyon	86.23	86.38	85.18	86.58	86.0	86.91	84.42	85.75	84.93	86.64	84.77	85.49	86.35
Sun397	71.52	67.82	69.95	71.23	71.41	69.03	56.88	69.22	62.7	69.61	64.63	72.74	71.44

Table 9. Absolute fine-tuning accuracy on Group III.

Dataset	From-Scratch	Img-90k	Img-180k	Img-270k	Img-300k	Inat-90k	Inat-180k	Inat-270k	Inat-300k	Pla-60k	Pla-120k	Pla-180k	Pla-200k
Caltech101	54.47	92.56	93.91	94.23	94.32	86.68	87.64	88.51	88.28	86.22	88.09	88.44	88.14
Flowers102	49.85	89.13	92.45	92.37	91.47	93.46	95.44	95.7	95.26	79.96	84.05	84.72	84.62
Camelyon	82.14	84.95	85.81	85.87	85.35	84.64	84.87	85.08	84.16	86.03	85.59	85.13	85.09
Sun397	46.87	67.36	70.83	71.41	71.44	64.6	66.97	67.44	67.42	68.78	73.21	74.22	74.24

Table 10. Absolute fine-tuning accuracy on Group IV.

Method	Recall@1	Pearson	Kendall
Linear (1 epoch)	✗	18.45	-33.33
Linear (5 epoch)	✗	40.77	0.00
Linear (converged)	✓	61.57	54.77
Fine-tune (1 epoch)	✗	20.55	0.00
Fine-tune (5 epoch)	✓	50.46	33.33
$\mathcal{N}$LEEP	✓	**66.59**	**66.67**

Method	Recall@1	Pearson	Kendall
Linear (1 epoch)	✗	23.55	-33.33
Linear (5 epoch)	✗	55.43	0.00
Linear (converged)	✓	68.32	33.33
Fine-tune (1 epoch)	✗	38.85	-33.33
Fine-tune (5 epoch)	✗	51.22	33.33
$\mathcal{N}$LEEP	✓	**82.66**	**66.67**

Table 11. Left: Checkpoint ranking results on the Pascal VOC Object Detection Benchmark (trained on VOC 2007 train+val + VOC 2012 train+val, tested on VOC 2007 using AP). Right: Checkpoint ranking for Cityscapes instance segmentation.