Ranking Neural Checkpoints

Yandong Li\textsuperscript{1,2}\textsuperscript{*} Xuhui Jia\textsuperscript{2} Ruoxin Sang\textsuperscript{2} Yukun Zhu\textsuperscript{2} Bradley Green\textsuperscript{2} Liqiang Wang\textsuperscript{1} Boqing Gong\textsuperscript{2} \\
\textsuperscript{1}University of Central Florida \textsuperscript{2}Google \\
lyndon.lee@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, \textit{NLEEP}, 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 \cite{pmlr-v102-chen20a, pmlr-v102-fang20a, pmlr-v102-kang20a, pmlr-v102-laskin20a, pmlr-v102-li20a, pmlr-v102-li20b, pmlr-v102-luo20a, pmlr-v102-miyato20a, pmlr-v102-park20a} and self-supervised \cite{pmlr-v102-chen20b, pmlr-v102-fang20b, pmlr-v102-glennberg20a, pmlr-v102-kang20b, pmlr-v102-laskin20b, pmlr-v102-miyato20b, pmlr-v102-park20b} learning make it feasible to harvest DNN checkpoints with scarce or no labels. Fine-tuning \cite{pmlr-v102-liu20a, pmlr-v102-xu20a} has become a de facto standard to adapt the pre-trained checkpoints to target tasks. It leads to faster convergence \cite{pmlr-v102-fei20a, pmlr-v102-lou20a, pmlr-v102-li20c} and better performance \cite{pmlr-v102-yang20a} 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 \ref{sec:related}). This paper is concerned with \textbf{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 \textit{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 \textit{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 \cite{pmlr-v69-lee17a}, 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 \cite{pmlr-v28-he16a, pmlr-v70-tan19b}, Inception-ResNet \cite{pmlr-v48-szegedy17b}, and VGG \cite{pmlr-v7-paszke17a}. 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 \cite{pmlr-v102-zhang19a, pmlr-v102-jin19a, pmlr-v102-li20d, pmlr-v102-chang20a}, 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 \cite{pmlr-v102-zhang19b, pmlr-v102-zhao20a}. The pre-training strategy also matters. Zhai \textit{et al.} \cite{pmlr-v102-zhai19a} find that combining supervision with self-supervision improves a network’s transfer results on downstream tasks. He \textit{et al.} \cite{pmlr-v102-he19a} 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 \cite{pmlr-v73-kim18a, pmlr-v73-kumar18a, pmlr-v97-luo19a, pmlr-v95-li19a}. Jiang \textit{et al.} \cite{pmlr-v95-li19a} 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.
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 (\(\text{LEEP}\)), to rank checkpoints based on the recently proposed log expected empirical prediction (LEEP) \cite{LEEP}. 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 \cite{LEEP}.

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 \(\text{LEEP}\) 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, \(C := \{\theta_i\}_{i=1}^m\). Denote by \(T\) a distribution of tasks. Without loss of generality, we mainly study classification downstream tasks, each of which, \(t \sim T\), contains a training set and a test set. An evaluation procedure, \(G : C \times T \rightarrow \mathbb{R}\), replaces the output layer of a checkpoint \(\theta_t\) 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:

\[
G_t := \{G(\theta_i, t)\}_{i=1}^m \in \mathbb{R}^m, \quad (1)
\]

which defines the groundtruth ranking list for task \(t\).

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

\[
R_t := \{R(\theta_i, t; b)\}_{i=1}^m \in \mathbb{R}^m, \quad (2)
\]

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

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

\[
R^* \leftarrow \arg \max_{R \in \mathbb{R}} \mathbb{E}_{t \sim T} M(R_t, G_t) \quad (3)
\]

where \(M\) is a metric evaluating the ranking scores \(R_t\) against the test accuracies \(G_t\). Section 2.3 details the evaluation methods used in this work. Equipped with such a ranking measure \(R^*\), 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 \(T\)

Following the design principle of \cite{leep}, we study diverse downstream tasks including Caltech101 \cite{Caltech}, Flowers102 \cite{Flowers}, Sun397 \cite{Sun}, and Patch Camelyon \cite{Patch}. 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.
2.2. Neural Checkpoints

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 [27]. We borrow 14 models pre-trained on ImageNet [13] from [66]. Among them, four are pre-trained by self-supervised learning (Jigsaw [42], Relative Patch Location [14], Exemplar [16], and Rotation [22]), six are the discriminators of generative models (WAE-UKL [48], WAE-GAN, WAE-MMD [57], Cond-BigGAN, Uncond-BigGAN [9], and VAE [33]), two are based on semi-supervised learning (Semi-Rotation-10% and Semi-Exemplar-10% [65]), one is by fully supervised learning (Sup-100%-Img [27]), and one is trained with a hybrid supervised loss (Sup-Exemplar-100% [65]). We also add two supervised checkpoints pre-trained on iNaturalist (Sup-100%-Inat) [59] and Places365 (Sup-100%-Pla) [67], respectively. Using the evaluation procedure $G(\theta_i, t)$ (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 [66, 64]. 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 architectures.** Kornblith et al. [34] show that better network ar-
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 [55], one is Inception-ResNet-v2 [54], six come from the MobileNet family [28], and two are from the ResNet-v1 family [27]. 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 $r$ [44] 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 $\tau$ [31] 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, 25, 23] 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 $I_\alpha$ mutual information estimator [45], where $\alpha$ 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 $I_\alpha$, [45].

3.4. LEEP for the Checkpoints with Classification Heads

To rank the checkpoints pre-trained over classification source tasks, the recently proposed LEEP [40] measure is directly applicable despite that it was originally designed for between-task transfer. Denote by $\mathcal{Z}$ the classification space of a checkpoint $\theta$. 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 $\theta$ is calculated by

$$R_{\text{LEEP}}(\theta, t) := \frac{1}{n} \sum_{j=1}^{n} \log P(y_j|x_j, \theta, t)$$

and

$$P(y|x, \theta, t) := \sum_{z \in \mathcal{Z}} \hat{P}(y|z)\theta(x)_z$$

(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 $\{(x_j, y_j)\}_{j=1}^n, y \in \mathcal{Y}$, 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
show the checkpoint ranking methods’
illustrates
sifying an input
is a collection of all the Gaussian components, and
π
N
named
we fit GMM to the downstream task’s training data, whereas
probability
V
posterior distribution:
Denote by
mension by using the principal component analysis (PCA).
the checkpoint’s neural architecture. W e reduce their di-
finally , it does not apply to the checkpoints with no classifi-
cation heads.
3.5. \( N/\text{LEEP} \)

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), \( P(s) = \sum_{v \in V} \pi_v \mathcal{N}(s | \mu_v, \Sigma_v) \), to the training set \( \{ s_j \}_{j=1}^n \), where \( V \) is a collection of all the Gaussian components, and \( \pi_v, v \in V \), are the mixture weights. It is convenient to compute the posterior distribution:
\[
P(v|x) = P(v|s) \propto \pi_v \mathcal{N}(s | \mu_v, \Sigma_v),
\]
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 \( N/\text{LEEP} \), 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 \( N/\text{LEEP} \) affect its checkpoint ranking performance. Figure 2 illustrates \( N/\text{LEEP} \)’s Kendall’s \( \tau \) values over Groups I and II with different PCA feature dimensions and the numbers of Gaussian components. Each Kendall’s \( \tau \) 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 \( \tau \) values remain relatively stable. In the remaining experiments with \( N/\text{LEEP} \), 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 \( I_\alpha \) mutual information estimator. Additionally, we experiment with \( I_\alpha \) 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 (\( \alpha=0.01 \)) 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 \( \alpha \) in the \( I_\alpha \) mutual information estimator [45] can control the trade-off between variance and bias. MI w/ and w/o PCA (\( \alpha=0.01 \)) perform better than MI w/ and w/o PCA (\( \alpha=0.50 \)), respectively. It indicates that neural checkpoint ranking requires low-bias MI estimator since smaller \( \alpha \) 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.

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 LLEEP, and it requires slightly larger GFLOPS due to the extra computation cost from the classification head.

We conjecture that LLEEP 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 \( \theta(x) \) is the key element to calculate it. However, \( \theta(x) \) can be poorly calibrated [40] and it can not represent a true probability. In contrast, \( P(v|x) \) used in LLEEP 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. LLEEP and LEEP exhibit a clear advantage over the other checkpoint ranking methods in terms of computing. The main reason is that LLEEP and LEEP can avoid intensive computation from neural network training, and they only require one forward pass through the training data.
### 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)

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall@1</th>
<th>Rel@1</th>
<th>Recall@3</th>
<th>Rel@3</th>
<th>Pearson</th>
<th>Kendall</th>
<th>GFLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear (1 epoch)</td>
<td>0.00</td>
<td>96.97</td>
<td>25.00</td>
<td>98.79</td>
<td>23.56</td>
<td>18.44</td>
<td>4.95E4</td>
</tr>
<tr>
<td>Linear (5 epoch)</td>
<td>25.00</td>
<td>98.79</td>
<td>50.00</td>
<td>98.94</td>
<td>49.77</td>
<td>32.33</td>
<td>4.97E4</td>
</tr>
<tr>
<td>Linear (converged)</td>
<td>50.00</td>
<td>99.63</td>
<td>75.00</td>
<td>99.65</td>
<td>68.97</td>
<td>53.43</td>
<td>5.33E4</td>
</tr>
<tr>
<td>Fine-tune (1 epoch)</td>
<td>25.00</td>
<td>97.45</td>
<td>25.00</td>
<td>97.66</td>
<td>30.25</td>
<td>22.15</td>
<td>6.51E5</td>
</tr>
<tr>
<td>Fine-tune (5 epoch)</td>
<td>0.00</td>
<td>91.09</td>
<td>25.00</td>
<td>98.61</td>
<td>48.19</td>
<td>36.78</td>
<td>4.28E6</td>
</tr>
<tr>
<td>MI (α=0.01) [45]</td>
<td>0.00</td>
<td>64.67</td>
<td>0.00</td>
<td>87.96</td>
<td>2.39</td>
<td>-0.31</td>
<td>1.62E5</td>
</tr>
<tr>
<td>MI (α=0.50)</td>
<td>0.00</td>
<td>66.71</td>
<td>25.00</td>
<td>90.31</td>
<td>-4.91</td>
<td>-13.05</td>
<td>1.62E5</td>
</tr>
<tr>
<td>MI w/ PCA (α=0.01)</td>
<td>0.00</td>
<td>89.45</td>
<td>50.00</td>
<td>99.27</td>
<td>16.16</td>
<td>20.67</td>
<td>5.58E4</td>
</tr>
<tr>
<td>MI w/ PCA (α=0.50)</td>
<td>0.00</td>
<td>86.49</td>
<td>25.00</td>
<td>94.28</td>
<td>-24.72</td>
<td>-16.06</td>
<td>5.58E4</td>
</tr>
<tr>
<td>LEEP [40]</td>
<td>75.00</td>
<td>99.65</td>
<td>75.00</td>
<td>99.65</td>
<td>84.30</td>
<td>76.00</td>
<td>12.85</td>
</tr>
</tbody>
</table>

### 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)

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall@1</th>
<th>Rel@1</th>
<th>Recall@3</th>
<th>Rel@3</th>
<th>Pearson</th>
<th>Kendall</th>
<th>GFLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear (1 epoch)</td>
<td>0.00</td>
<td>96.46</td>
<td>25.00</td>
<td>98.79</td>
<td>27.01</td>
<td>24.24</td>
<td>4.95E4</td>
</tr>
<tr>
<td>Linear (5 epochs)</td>
<td>50.00</td>
<td>99.57</td>
<td>100.00</td>
<td>100.00</td>
<td>55.07</td>
<td>51.28</td>
<td>4.97E4</td>
</tr>
<tr>
<td>Linear (converged)</td>
<td>75.00</td>
<td>99.95</td>
<td>100.00</td>
<td>100.00</td>
<td>79.30</td>
<td>68.60</td>
<td>5.33E4</td>
</tr>
<tr>
<td>Fine-tune (1 epoch)</td>
<td>25.00</td>
<td>99.05</td>
<td>25.00</td>
<td>99.47</td>
<td>19.61</td>
<td>15.52</td>
<td>6.51E5</td>
</tr>
<tr>
<td>Fine-tune (5 epochs)</td>
<td>25.00</td>
<td>99.55</td>
<td>100.00</td>
<td>100.00</td>
<td>68.47</td>
<td>58.33</td>
<td>4.28E6</td>
</tr>
<tr>
<td>MI (α=0.01) [45]</td>
<td>0.00</td>
<td>94.84</td>
<td>25.00</td>
<td>97.43</td>
<td>-29.41</td>
<td>-17.81</td>
<td>1.62E5</td>
</tr>
<tr>
<td>MI (α=0.50)</td>
<td>0.00</td>
<td>96.66</td>
<td>0.00</td>
<td>97.03</td>
<td>-11.36</td>
<td>-10.21</td>
<td>1.62E5</td>
</tr>
<tr>
<td>MI w/ PCA (α=0.01)</td>
<td>0.00</td>
<td>99.60</td>
<td>50.00</td>
<td>99.52</td>
<td>23.73</td>
<td>17.09</td>
<td>5.58E4</td>
</tr>
<tr>
<td>MI w/ PCA (α=0.50)</td>
<td>25.00</td>
<td>99.95</td>
<td>75.00</td>
<td>99.90</td>
<td>50.36</td>
<td>55.49</td>
<td>378.31</td>
</tr>
<tr>
<td>LEEP [40]</td>
<td>75.00</td>
<td>99.44</td>
<td>75.00</td>
<td>99.90</td>
<td>50.36</td>
<td>55.49</td>
<td>378.31</td>
</tr>
</tbody>
</table>

### 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)

<table>
<thead>
<tr>
<th>Method</th>
<th>Recall@1</th>
<th>Rel@1</th>
<th>Recall@3</th>
<th>Rel@3</th>
<th>Pearson</th>
<th>Kendall</th>
<th>GFLOPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear (1 epoch)</td>
<td>25.00</td>
<td>98.17</td>
<td>25.00</td>
<td>99.35</td>
<td>30.14</td>
<td>13.80</td>
<td>3.37E4</td>
</tr>
<tr>
<td>Linear (5 epochs)</td>
<td>25.00</td>
<td>98.98</td>
<td>25.00</td>
<td>99.63</td>
<td>33.45</td>
<td>18.95</td>
<td>3.38E4</td>
</tr>
<tr>
<td>Linear (converged)</td>
<td>25.00</td>
<td>99.66</td>
<td>25.00</td>
<td>99.72</td>
<td>63.55</td>
<td>36.91</td>
<td>3.62E4</td>
</tr>
<tr>
<td>Fine-tune (1 epoch)</td>
<td>25.00</td>
<td>98.62</td>
<td>25.00</td>
<td>99.68</td>
<td>25.72</td>
<td>17.09</td>
<td>5.58E4</td>
</tr>
<tr>
<td>Fine-tune (5 epochs)</td>
<td>25.00</td>
<td>98.28</td>
<td>25.00</td>
<td>99.80</td>
<td>25.72</td>
<td>17.09</td>
<td>5.58E4</td>
</tr>
<tr>
<td>MI (α=0.01) [45]</td>
<td>25.00</td>
<td>98.29</td>
<td>25.00</td>
<td>99.34</td>
<td>4.42</td>
<td>2.94</td>
<td>1.30E5</td>
</tr>
<tr>
<td>MI (α=0.50)</td>
<td>25.00</td>
<td>98.36</td>
<td>25.00</td>
<td>99.37</td>
<td>-9.79</td>
<td>-6.81</td>
<td>1.30E5</td>
</tr>
<tr>
<td>MI w/ PCA (α=0.01)</td>
<td>25.00</td>
<td>99.18</td>
<td>50.00</td>
<td>99.82</td>
<td>61.94</td>
<td>38.83</td>
<td>5.56E4</td>
</tr>
<tr>
<td>MI w/ PCA (α=0.50)</td>
<td>25.00</td>
<td>96.34</td>
<td>0.00</td>
<td>98.47</td>
<td>33.17</td>
<td>21.26</td>
<td>5.56E4</td>
</tr>
<tr>
<td>LEEP [40]</td>
<td>25.00</td>
<td>97.36</td>
<td>75.00</td>
<td>99.90</td>
<td>42.99</td>
<td>45.06</td>
<td>247.56</td>
</tr>
<tr>
<td>▼LEEP</td>
<td>25.00</td>
<td>99.66</td>
<td>25.00</td>
<td>99.70</td>
<td>66.94</td>
<td>51.14</td>
<td>12.68</td>
</tr>
</tbody>
</table>

**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 [34]. 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 re-
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 [27] 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 [20] and Cityscapes [12]. 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, NLEEP, 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 [58], data set distance as optimal transport [2], F-relatedness [7], A-distance [32], and discrepancy distance [37]. Besides, Poole et al. [45] 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 [64] is a fully computation method, where task similarity scores are obtained by transfer learning experiments. Dwivedi et al. [17] 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 [17, 52, 53]. Song et al. [52, 53] 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, 19, 51]. Jiang et al. [29] and Yak et al. [62] 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 [39, 30, 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 NLEEP, 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


[28] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco An


[58] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and


