arXiv:2102.11005v3 [cs.LG] 23 Jun 2021

LogME: Practical Assessment of Pre-trained Models for Transfer Learning

Kaichao You[*1] **Yong Liu**[*1] **Jianmin Wang**[1] **Mingsheng Long**[1]

Abstract

This paper studies task adaptive pre-trained model selection, an underexplored problem of assessing pre-trained models for the target task and select best ones from the model zoo *without fine-tuning*. A few pilot works addressed the problem in transferring supervised pre-trained models to classification tasks, but they cannot handle emerging unsupervised pre-trained models or regression tasks. In pursuit of a practical assessment method, we propose to estimate the maximum value of label evidence given features extracted by pre-trained models. Unlike the maximum likelihood, the maximum evidence is *immune to over-fitting*, while its expensive computation can be dramatically reduced by our carefully designed algorithm. The Logarithm of Maximum Evidence (LogME) can be used to assess pre-trained models for transfer learning: a pre-trained model with a high LogME value is likely to have good transfer performance. LogME is *fast, accurate, and general*, characterizing itself as the first practical method for assessing pre-trained models. Compared with brute-force fine-tuning, LogME brings at most $3000\times$ speedup in wall-clock time and requires only 1% memory footprint. It outperforms prior methods by a large margin in their setting and is applicable to new settings. It is general enough for diverse pre-trained models (supervised pre-trained and unsupervised pre-trained), downstream tasks (classification and regression), and modalities (vision and language). Code is available at this repository: https://github.com/thuml/LogME.

1. Introduction

Human performance on many recognition tasks has been surpassed by deep neural networks (He et al., 2015; 2016) trained with large-scale supervised data (Deng et al., 2009; Russakovsky et al., 2015) and specialized computational devices (Jouppi et al., 2017). These trained neural networks, also known as pre-trained models, not only work well on tasks they are intended for but also produce generic representations (Donahue et al., 2014) that benefit downstream tasks such as object detection (Girshick et al., 2014).

Apart from serving as fixed feature extractors, pre-trained models can be fine-tuned (Yosinski et al., 2014; He et al., 2019) to serve downstream tasks better. The transfer learning paradigm "pre-training $\rightarrow$ fine-tuning" enjoys tremendous success in both vision (Kornblith et al., 2019) and language (Devlin et al., 2019) communities, and continues to expand to communities like geometric learning (Hu et al., 2020). *Transfer of pre-trained models has become one of the cornerstones of deep learning.*

Nowadays, there are numerous public pre-trained models offered by PyTorch (Benoit et al., 2019), TensorFlow (Abadi et al., 2016) and third-party libraries like HuggingFace Transformers (Wolf et al., 2020). When a practitioner wants to employ transfer learning to solve a specific task, the first problem is to select a good pre-trained model to start from. The problem is non-trivial and task adaptive, considering that *different tasks favor different pre-trained models*. Figure 1 intuitively illustrates the problem.

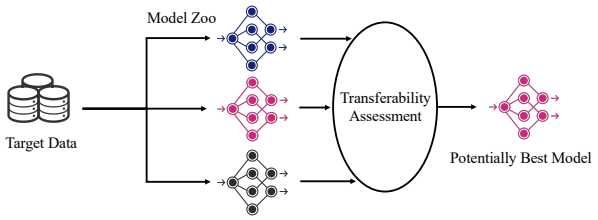

Figure 1. Illustration of task adaptive pre-trained model selection.

The problem challenges researchers to develop a practical assessment method that is fast, accurate and general. It should be fast enough compared to brute-force fine-tuning all available pre-trained models (Zamir et al., 2018), should be accurate enough so that potentially best models can be identified, and should be general enough to tackle a wide variety of common learning scenarios.

Despite its practical significance, there is limited guidance

[*]Equal contribution [1]School of Software, BNRist, Tsinghua University, Beijing 100084, China.
Kaichao You <youkaichao@gmail.com>. Correspondence to: Mingsheng Long <mingsheng@tsinghua.edu.cn>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

Table 1. Applicability of prior methods and the proposed LogME. "LM" means language modeling.

Modality	Pre-train	Target	LEEP	NCE	LogME
vision	classification	classification	✓	✓	✓
	classification	regression	✗	✗	✓
	contrastive	classification	✗	✗	✓
	contrastive	regression	✗	✗	✓
language	LM	classification	✗	✗	✓

on task adaptive pre-trained model selection. Based on NCE (Tran et al., 2019), Nguyen et al. (2020) recently studied the problem when both the pre-train task and the downstream task are classification. They construct an empirical predictor by estimating the joint distribution over the pre-trained and target label spaces and take the performance of the empirical predictor (LEEP) to assess pre-trained models. Though being fast, prior methods are not accurate and are specialized for transferring supervised pre-trained models to classification. They cannot apply to either contrastive pre-trained models (He et al., 2020; Chen et al., 2020a), unsupervised pre-trained language models (Devlin et al., 2019; Liu et al., 2019), or regression tasks.

Table 1 shows the applicability of pre-trained model selection methods. Prior to this paper, for most (4 out of 5) transfer learning settings, task adaptive pre-trained model selection does not have a decent solution.

To provide a general method for pre-trained model selection in various settings, we consider the features extracted by pre-trained models, thus being agnostic to how models are pre-trained. The maximum value of label evidence (marginalized likelihood) given extracted features is calculated, providing a general probabilistic approach that is applicable to both classification and regression tasks. Finally, the logarithm of maximum evidence (LogME) is used to assess pre-trained models for transfer learning. The maximum evidence is less prone to over-fitting (Bishop, 2006), and its humongous computational cost is dramatically reduced by our carefully designed algorithm.

The contributions of this paper are two-fold:

- We propose LogME for task adaptive pre-trained model selection, and develop a fast algorithm to accelerate the computation. LogME is easy to interpret and is extremely efficient. It brings at most $3000\times$ speedup in wall-clock time and requires just 1% memory footprint, characterizing itself as the first practical method for assessing pre-trained models in various transfer learning settings.
- We extensively validate the generality and superior performance of LogME on 22 *pre-trained models* and 17 *downstream tasks*, covering various pre-trained models (supervised pre-trained and unsupervised pre-trained), downstream tasks (classification and regression), and modalities (vision and language).

2. Related Works

2.1. Transfer learning

Transfer learning (Thrun & Pratt, 1998) is a broad research area containing transductive transfer, inductive transfer, task transfer learning, and so on. Transductive transfer is commonly known as domain adaptation (Quionero-Candela et al., 2009; Ganin & Lempitsky, 2015; Long et al., 2015), with the focus on eliminating domain shifts between two domains. Inductive transfer, or fine-tuning (Erhan et al., 2010; Yosinski et al., 2014), leverages an inductive bias (a pre-trained model) to improve the performance on a target task and is extremely popular in deep learning. In task transfer learning (Zamir et al., 2018), researchers investigate how to transfer between tasks rather than pre-trained models. They aim to discover the relationship among tasks (Ben-David & Schuller, 2003) and to exploit the relationship for further development. In the context of deep learning, transfer learning usually refers to inductive transfer, the topic we are concerned about in this paper.

Besides the naïve fine-tuning where pre-trained models only serve as good initializations, there are sophisticated fine-tuning techniques like regularization (Li et al., 2018), additional supervision (You et al., 2020), specially designed architecture (Kou et al., 2020), and intermediate-task training which continues to pre-train on an intermediate task (Gururangan et al., 2020; Pruksachatkun et al., 2020; Garg et al., 2020). They can improve transfer learning performance especially when the amount of target data is small, but in general, *they do not change the ranking of pre-trained models in downstream tasks*. If pre-trained model A is better than pre-trained model B in a task with vanilla fine-tuning, typically A is still better than B when those sophisticated techniques are turned on. For example, on three datasets and four sampling rates from Table 2 in You et al. (2020), better fine-tuning performance mostly indicates better Co-Tuning (their proposed method) performance. Therefore we focus on vanilla fine-tuning rather than these techniques in the rest of the paper, but practitioners are encouraged to adopt them for further improvement after selecting a pre-trained model.

2.2. Pre-trained models

Pre-trained models are neural networks trained on large-scale datasets and can be transferred to downstream tasks. Popular pre-trained models are reviewed in the following.

Supervised pre-trained models. ImageNet is the most famous dataset for supervised pre-training. In the ImageNet classification challenge, He et al. (2015) developed

the first deep neural network that surpassed human performance. InceptionNet (Szegedy et al., 2015) is another family of deep neural networks with parallel convolution filters. ResNet (He et al., 2016) introduces skip connections to ease the training and becomes much deeper with better performance. DenseNet (Huang et al., 2017) has carefully designed densely-connected blocks. MobileNet (Sandler et al., 2018) pays attention to mobile-friendly network structures, and the structure can be further optimized by network architecture search (Tan et al., 2019).

Contrastive pre-trained models. Although ImageNet pre-training is popular, the labeling cost of ImageNet is very high. Given the large amount of unlabeled data on the Internet, unsupervised pre-training has gained much attention in the past year. By exploiting self-supervised learning (Jing & Tian, 2020) on unlabeled data (Mahajan et al., 2018) with contrastive loss (Gutmann & Hyvärinen, 2010), unsupervised contrastive pre-training produces a family of pre-trained models besides supervised pre-trained models. He et al. (2020) proposed Momentum Contrast with a queue structure to fully exploit unlabeled data and obtained representations on par with supervised pre-training in terms of quality. Chen et al. (2020a) greatly improved the performance by exploring data augmentation, multi-layer projection head and many empirical design choices. How to design better contrastive pre-training strategies is still under active research (Tian et al., 2020).

Pre-trained language models. In the language community, unsupervised pre-training has been well established by training masked language models (Devlin et al., 2019) or autoregressive language models (Yang et al., 2019) on a large unlabeled corpus. Liu et al. (2019) explored many practical details on how to improve the training of these models. Because pre-trained language models are very large, Sanh et al. (2019) proposed distillation to get smaller and faster models. These pre-trained language models become an indispensable component in winning submissions on common benchmarks like GLUE (Wang et al., 2018) and SQuAD (Rajpurkar et al., 2016), and have profound industrial influence.

Pre-trained models are hosted in model zoos like TorchVision and HuggingFace. There are so many pre-trained models, but no one can overwhelmingly outperform the rest in all downstream tasks. The best model for a downstream task depends on the characteristic of both the task and the pre-trained model, thus being task adaptive. Practitioners can have a hard time choosing which pre-trained model to use for transfer learning, calling for a practical method to assess pre-trained models without brute-force fine-tuning.

2.3. Assessing transferablitiy of pre-trained models

Assessing transferability of pre-trained models has a great significance to guide common practice. Yosinski et al. (2014) studied which layer of a pre-trained model can be transferred while Kornblith et al. (2019) studied a wide variety of modern pre-trained models in computer vision. These papers aim for a deeper understanding of transfer learning (Neyshabur et al., 2020). Nonetheless, they draw conclusions by expensive and exhaustive fine-tuning with humongous computation cost (Section 5.5) which is hard for practitioners to afford.

To *efficiently* assess the transferability of pre-trained models, Nguyen et al. (2020) pioneered to develop LEEP with a focus on supervised pre-trained models transferred to classification tasks. The joint distribution over pre-trained labels and the target labels is estimated to construct an empirical predictor. The log expectation of the empirical predictor (LEEP) is used as a transferability measure. The LEEP method is closely related to Negative Conditional Entropy (NCE) proposed by Tran et al. (2019), an information-theoretic quantity (Cover, 1999) to study the transferability and hardness between classification tasks.

LEEP (Nguyen et al., 2020) and NCE (Tran et al., 2019), the only two prior methods for pre-trained model selection, shed light on this problem but leave plenty of room for further performance improvement. In addition, they can only handle classification tasks with supervised pre-trained models. Since contrastive pre-training and language modeling tasks do not have categorical labels, *prior methods cannot deal with these increasingly popular models*. To promote pre-trained model selection, we propose LogME which is broadly applicable to various pre-trained models, downstream tasks, and even data modalities.

3. Problem setup

In task adaptive pre-trained model selection, we are given M pre-trained models $\{\phi_m\}_{m=1}^M$ and a target dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ with n labeled data points. The dataset has an evaluation metric (accuracy, MAP, MSE *etc.*) to measure the ground-truth transfer performance T_m of fine-tuning ϕ_m with proper hyper-parameter tuning. A practical assessment method should produce a score S_m for each pre-trained model ϕ_m (ideally without fine-tuning ϕ_m on $\mathcal{D}$), and the scores $\{S_m\}_{m=1}^M$ should well correlate with $\{T_m\}_{m=1}^M$ so that top performing pre-trained models can be selected by simply evaluating the scores.

How to measure the performance of pre-trained model assessing methods. A perfect pre-trained model assessing method would output $\{S_m\}_{m=1}^M$ with exactly the same order as $\{T_m\}_{m=1}^M$. To measure the deviation from the perfect method, we can use simple metrics like top-1 accuracy or top-k accuracy (whether top-k in $\{S_m\}_{m=1}^M$ are also top-k in $\{T_m\}_{m=1}^M$). But top-1 accuracy is too conservative and top-k accuracy is not comparable across different val-

ues of M. Therefore we turn to rank correlation (Fagin et al., 2003) to directly measure the correlation between $\{S_m\}_{m=1}^M$ and $\{T_m\}_{m=1}^M$. The prior work (Nguyen et al., 2020) adopted Pearson's linear correlation coefficient, but neither Pearson's linear correlation nor its variant (Spearman's rank correlation) has a simple interpretation (see the interpretation of τ below).

Since the purpose of assessment is to choose a good pre-trained model, we hope *T_i is better than T_j if S_i is better than S_j*, which can be well captured by Kendall's τ coefficient (Kendall, 1938) as described in the following.

To simplify the discussion, assume larger value of transfer performance T and score S are preferred (*e.g.* accuracy). If this is not the case (*e.g.* transfer performance is measured by mean square error), the negation can be considered. For a pair of measures (T_i, S_i) and (T_j, S_j), the pair is concordant if $T_i < T_j \wedge S_i < S_j$ or $T_i > T_j \wedge S_i > S_j$ (concisely speaking, $\mathrm{sgn}(T_i - T_j)\mathrm{sgn}(S_i - S_j) = 1$). The Kendall's τ coefficient is defined by the following equation, which enumerates all $\binom{M}{2}$ pairs and counts the number of concordant pairs minus the number of discordant pairs.

$$\tau = \frac{2}{M(M-1)} \sum_{1 \le i < j \le M} \mathrm{sgn}(T_i - T_j)\mathrm{sgn}(S_i - S_j)$$

How to interpret τ (Fagin et al., 2003). The range of τ is $[-1, 1]$. $\tau = 1$ means T and S are perfectly correlated ($S_i > S_j \iff T_i > T_j$), and $\tau = -1$ means T and S are reversely correlated ($S_i > S_j \iff T_i < T_j$). *If T and S have correlation of τ, the probability of $T_i > T_j$ is $\frac{\tau+1}{2}$ when $S_i > S_j$.*

Pay attention to top performing models. Since a major application of assessing pre-trained models is to select top performing pre-trained models, discordant / concordant pairs should be weighted more if T_i, T_j, S_i, S_j are larger. This can be taken care of by τ_w (Vigna, 2015). The details of calculating τ_w can be found in implementation from SciPy (Virtanen et al., 2020).

In short, we measure the correlation between $\{S_m\}_{m=1}^M$ and $\{T_m\}_{m=1}^M$ by the weighted variant τ_w (Vigna, 2015). *Larger τ_w indicates better correlation and better assessment.*

Note that how to measure the performance of pre-trained model assessing methods is neither the focus nor the claimed novelty of this paper. We use weighted Kendall's τ because it is easy to interpret, but any proper rank correlation metric (such as Pearson's linear correlation and Spearman's rank correlation) can be adopted and should yield similar conclusions on superiority of our proposed method.

4. The LogME approach

For each pre-trained model ϕ_m, the algorithm should produce a score S_m independent from the rest of pre-trained models. We thus drop the subscript m in this section.

To be fast, we try to *avoid gradient optimization*. The pre-trained model ϕ serves as a fixed feature extractor. Features $\{f_i = \phi(x_i)\}_{i=1}^n$ and labels $\{y_i\}_{i=1}^n$ are used to assess pre-trained models. Note that Nguyen et al. (2020) used a pre-trained classification head h besides the pre-trained representation model ϕ, limiting their method to supervised pre-trained models. In contrast, *we only use the pre-trained representation model ϕ so that the proposed method can be applied to any pre-trained model* (whether supervised pre-trained or unsupervised pre-trained).

Without gradient optimization, the problem is cast into estimating the compatibility of features $\{f_i = \phi(x_i)\}_{i=1}^n$ and labels $\{y_i\}_{i=1}^n$, which is discussed in the rest of this section.

4.1. Evidence calculation

We first consider a simple case, with features $f_i \in \mathbb{R}^D$ and scalar labels $y_i \in \mathbb{R}$. The feature matrix $F \in \mathbb{R}^{n \times D}$ contains all the features and $y \in \mathbb{R}^n$ denotes all the labels.

A direct measurement of the compatibility between features F and labels y is the probability density $p(y|F)$, which is intractable without a parametrized model. Since *the rule-of-thumb transfer learning practice is to add a fully-connected layer on top of the pre-trained model*, we use a linear model upon features parametrized by w.

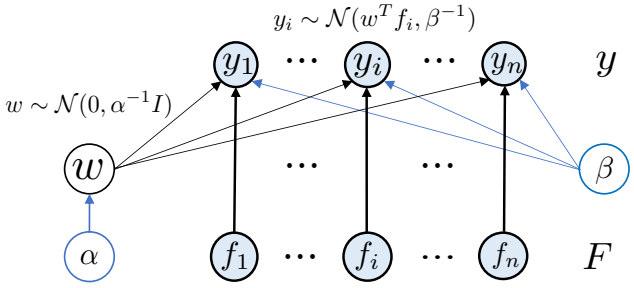

Figure 2. The directed graphical model for calculating evidence.

A naïve approach to deal with the linear model is to find the best w^* by logistic / linear regression and to assess pre-trained models by likelihood $p(y|F, w^*)$. However, it is well-known that *likelihood is prone to over-fitting* (Bishop, 2006), which is experimentally observed in Supplementary B. A better approach is to use the evidence (marginalized likelihood) $p(y|F) = \int p(w)p(y|F, w)\mathrm{d}w$, which integrates over all possible values of w and is better than simply using one optimal value w^*. This evidence-based approach is an elegant model selection approach and has a rigorous theoretical foundation (Knuth et al., 2015). For $p(w)$ and $p(y|F, w)$, we use the commonly adopted graphical model (Figure 2) specified by two positive parameters α and β: the prior distribution of the weight is an isotropic multivariate Gaussian $w \sim \mathcal{N}(0, \alpha^{-1} I)$, and the distribution of

each observation is a one-dimensional normal distribution $p(y_i|f_i,w,\beta)=\mathcal{N}(y_i|w^Tf_i,\beta^{-1})$.

According to the causal structure in Figure 2 and the basic principles in graphical models (Koller & Friedman, 2009), the evidence can be calculated analytically as Eq. 1.

$$\begin{aligned} p(y|F,\alpha,\beta) &= \int p(w|\alpha)p(y|F,w,\beta)\mathrm{d}w \\ &= \int p(w|\alpha)\prod_{i=1}^{n} p(y_i|f_i,w,\beta)\mathrm{d}w \\ &= (\frac{\beta}{2\pi})^{\frac{n}{2}}(\frac{\alpha}{2\pi})^{\frac{D}{2}}\int e^{-\frac{\alpha}{2}w^Tw-\frac{\beta}{2}||Fw-y||^2}\mathrm{d}w \end{aligned} \tag{1}$$

As $\int e^{-\frac{1}{2}(w^TAw+b^Tw+c)}\,\mathrm{d}w = \sqrt{\frac{(2\pi)^D}{|A|}}e^{-\frac{1}{2}c+\frac{1}{8}b^TA^{-1}b}$ when A is positive definite, Eq. 1 can be simplified. By taking the logarithm to make the equation simple, Eq. 2 shows the logarithm of the evidence as a function of α,β, where $A=\alpha I+\beta F^TF, m=\beta A^{-1}F^Ty$.

$$\begin{aligned} \mathcal{L}(\alpha,\beta) &= \log p(y|F,\alpha,\beta) \\ &= \frac{n}{2}\log\beta+\frac{D}{2}\log\alpha-\frac{n}{2}\log 2\pi \\ &- \frac{\beta}{2}||Fm-y||_2^2-\frac{\alpha}{2}m^Tm-\frac{1}{2}\log|A| \end{aligned} \tag{2}$$

4.2. Evidence maximization and LogME

A remaining issue of Eq. 2 is how to determine α,β. Gull (1989) suggested that we should choose α,β to maximize the evidence, *i.e.* use $(\alpha^*,\beta^*)=\arg\max_{\alpha,\beta}\mathcal{L}(\alpha,\beta)$. Because m and A are coupled, maximizing $\mathcal{L}(\alpha,\beta)$ is generally a difficult problem. However, this form of maximization can be achieved by alternating between evaluating m,γ and maximizing α,β with m,γ fixed (Gull, 1989), resulting the following formula, where σ_i's are singular values of F^TF.

$$A=\alpha I+\beta F^TF, m=\beta A^{-1}F^Ty, \gamma=\sum_{i=1}^{D}\frac{\beta\sigma_i}{\alpha+\beta\sigma_i}$$

$$\alpha\leftarrow\frac{\gamma}{m^Tm}, \beta\leftarrow\frac{n-\gamma}{||Fm-y||_2^2}$$

When the fixed-point iteration converges (empirically it converges with no more than three iterations), the logarithm maximum evidence $\mathcal{L}(\alpha^*,\beta^*)$ is used to evaluate the compatibility between features and labels. Because $\mathcal{L}(\alpha^*,\beta^*)$ scales linearly with n, we normalize it by $\frac{\mathcal{L}(\alpha^*,\beta^*)}{n}$ and term it LogME (logarithm of of maximum evidence). It can be intuitively interpreted as the average maximum log evidence of labels given the pre-trained features.

Extending LogME to complex cases. The LogME approach described above starts from a single-target regression. If the target problem is a multivariate-regression task, *i.e.* $y\in\mathbb{R}^{n\times K}$, we can calculate LogME for each dimension k $(1\le k\le K)$ and average them over the K dimension. If the target problem is a classification task with K classes, Eq. 1 cannot be calculated analytically (Daunizeau, 2017) with a categorical prior distribution, but we can convert the labels to one-hot labels and treat the problem as multivariate regression. Therefore, LogME can be used in both classification and regression tasks. The overall algorithm of LogME is described in Algorithm 1.

Algorithm 1 LogME

1: **Input:** Pre-trained model ϕ
   Target dataset $\mathcal{D}=\{(x_i,y_i)\}_{i=1}^n$
2: **Output:** logarithm of maximum evidence (LogME)
3: Extract features using pre-trained model ϕ:
   $F\in\mathbb{R}^{n\times D}, f_i=\phi(x_i), Y\in\mathbb{R}^{n\times K}$
4: Compute SVD $F^TF=V\mathrm{diag}\{\sigma\}V^T$
5: **for** $k=1$ to K **do**
6: Let $y=Y^{(k)}\in\mathbb{R}^n$, initialize $\alpha=1,\beta=1$
7: **while** α,β not converge **do**
8: Compute $\gamma=\sum_{i=1}^{D}\frac{\beta\sigma_i}{\alpha+\beta\sigma_i}, \Lambda=\mathrm{diag}\{(\alpha+\beta\sigma)\}$
9: **Naïve**: $A=\alpha I+\beta F^TF, m=\beta A^{-1}F^Ty$
10: **Optimized**: $m=\beta(V(\Lambda^{-1}(V^T(F^Ty))))$
11: Update $\alpha\leftarrow\frac{\gamma}{m^Tm}, \beta\leftarrow\frac{n-\gamma}{||Fm-y||_2^2}$
12: **end while**
13: Compute $\mathcal{L}_k=\frac{1}{n}\mathcal{L}(\alpha,\beta)$ using Eq. 2
14: **end for**
15: Return LogME $\frac{1}{K}\sum_{k=1}^{K}\mathcal{L}_k$

4.3. Computational speedup

Although the Bayesian approach of maximum evidence has many nice properties (Knuth et al., 2015), it inherits the common drawback of Bayesian methods with high computational complexity. The naïve implementation of Algorithm 1 has a complexity of $\mathcal{O}(KD^3+nKD^2)$. For typical usage with $D\approx 10^3, n\approx 10^4, K\approx 10^3$, the computational cost is 10^{13}, making the wall-clock time comparable to fine-tuning the pre-trained model ϕ.

Notice that the most expensive operations are Line 9 with matrix inversion A^{-1} and matrix multiplication $A^{-1}F^T$. These expensive operations, however, can be avoided by exploiting the decomposition of F^TF, which is readily accessible from Line 4.

To avoid matrix inversion A^{-1}, we exploit the decomposition $F^TF=V\mathrm{diag}\{\sigma\}V^T$ (V is an orthogonal matrix). Let $\Lambda=\mathrm{diag}\{(\alpha+\beta\sigma)\}$, then $A=\alpha I+\beta F^TF=V\Lambda V^T$, and $A^{-1}=V\Lambda^{-1}V^T$. To avoid the matrix-matrix multiplication $A^{-1}F^T$, we notice that y is a column vector and the associative law admits a fast computation $A^{-1}F^Ty=(V(\Lambda^{-1}(V^T(F^Ty))))$. In each for-loop, we only need to

update Λ rather than the expensive A^{-1}. In this way, all matrix-matrix multiplications are reduced to matrix-vector product, and the matrix inversion is avoided, as described in Line 10. Table 2 analyzes the complexity in detail. The optimized algorithm makes a time-consuming Bayesian approach fast enough, reducing the wall-clock time by the order of 10^2 (see Section 5.5).

Table 2. Computational complexity of Algorithm 1.

	Complexity per for-loop	Overall complexity
naïve	$\mathcal{O}(D^3 + nD^2)$	$\mathcal{O}(KD^3 + nKD^2)$
optimized	$\mathcal{O}(D^2 + nD)$	$\mathcal{O}(KD^2 + nKD + D^3 + nD^2)$

The proposed LogME is easy to interpret, has a solid theoretical foundation, and is applicable to various settings. Its computational cost is dramatically reduced by our optimized implementation.

5. Experiments

We first present the illustration of LogME on toy problems, and then focus on task adaptive pre-trained model selection. Original data are available in Supplementary C.

Illustration with toy data. To give readers an intuitive sense of how LogME works, we generate features with increasing noise to mimic the features extracted by pre-trained models with decreasing transferability and to check if LogME can measure the quality of features.

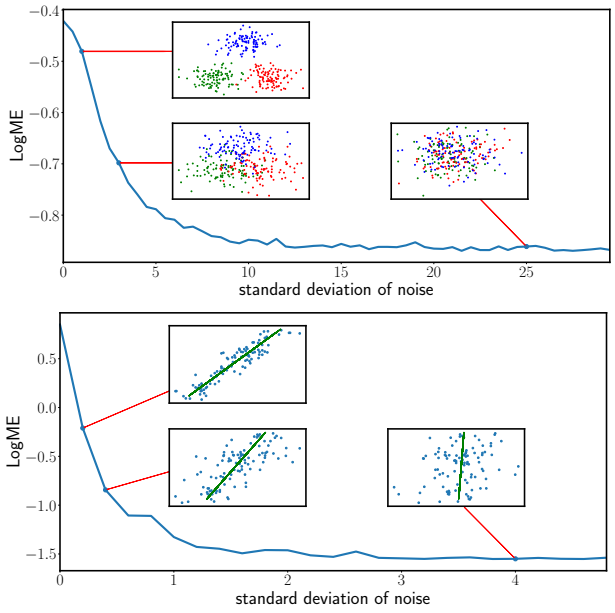

Figure 3. Illustration of LogME with toy data. It is clear that LogME decreases with decreasing feature quality.

For classification (Figure 3 top), three clusters in 2-D plane are generated, with colors indicating the categories. Initially, the features are separable so LogME has a large value. Then we add Gaussian noise with increasing variance and LogME becomes smaller as expected.

For regression (Figure 3 bottom), x is uniformly distributed and the output $y = 2x + \epsilon$ with observation error $\epsilon \sim \mathcal{N}(0, 0.1^2)$. By adding noise to the feature $x' = x + \mathcal{N}(0, t^2)$, the quality of feature x' becomes worse and it is harder to predict y from x'. With larger t (the standard deviation of noise), LogME becomes smaller as expected.

These toy experiments on synthesized data shows that LogME is a good measure of the feature quality, and therefore can provide a general assessment of pre-trained models for transfer learning.

5.1. Transferring supervised pre-trained models to classification tasks

We use 10 ImageNet pre-trained models available from PyTorch: Inception V1 (Szegedy et al., 2015), Inception V3 (Szegedy et al., 2016), ResNet 50 (He et al., 2016), ResNet 101 (He et al., 2016), ResNet 152 (He et al., 2016), DenseNet 121 (Huang et al., 2017), DenseNet 169 (Huang et al., 2017), DenseNet 201 (Huang et al., 2017), MobileNet V2 (Sandler et al., 2018), and NASNet-A Mobile (Tan et al., 2019). These pre-trained models cover most of the supervised pre-trained models in transfer learning that practitioners frequently use.

For downstream classification tasks, we take 9 commonly used datasets: Aircraft (Maji et al., 2013), Birdsnap (Berg et al., 2014), Caltech (Fei-Fei et al., 2004), Cars (Krause et al., 2013), CIFAR10 (Krizhevsky & Hinton, 2009), CIFAR100 (Krizhevsky & Hinton, 2009), DTD (Cimpoi et al., 2014), Pets (Parkhi et al., 2012), and SUN (Xiao et al., 2010). Due to space limit, we leave the description of each dataset and data statistics in Supplementary A.

To compute the value of transfer performance $\{T_m\}_{m=1}^M$ ($M = 10$), we carefully fine-tune pre-trained models with grid-search of hyper-parameters. As pointed out by Li et al. (2020), learning rates and weight decays are the two most important hyper-parameters. Hence we grid search learning rates and weight decays (7 learning rates from 10^{-1} to 10^{-4}, 7 weight decays from 10^{-6} to 10^{-3}, all logarithmically spaced) to select the best hyper-parameter on the validation set and compute the accuracy on the test set. *It is noteworthy that LogME requires neither fine-tuning nor grid search.* Here we fine-tune pre-trained models to evaluate LogME itself, but practitioners can straightforwardly use LogME to evaluate pre-trained models without fine-tuning.

We compare LogME against LEEP (Nguyen et al., 2020) and NCE (Tran et al., 2019). Prior to this paper, LEEP

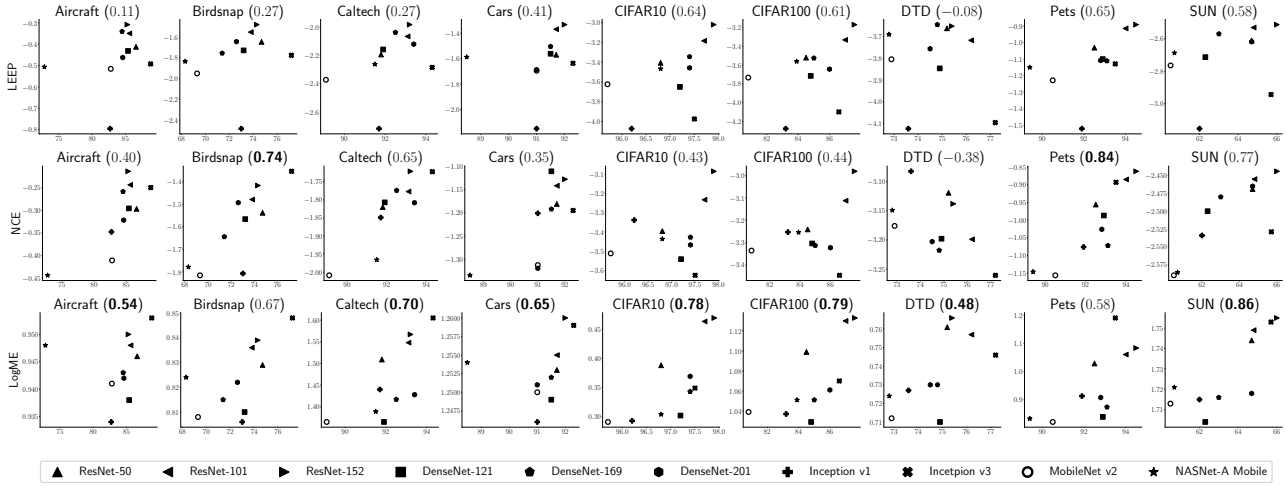

Figure 4. Correlation (τ_w) between fine-tuned accuracy (X-axis) and three methods for pre-trained model selection on 9 datasets with 10 pre-trained models. One row for each method, one column for each dataset (with τ_w in the bracket near the dataset name), and one marker for each pre-trained model. The best τ_w in each dataset is marked in bold.

and NCE are the only two methods for pre-trained model selection without fine-tuning, and they are dedicated to transferring supervised pre-trained models to classification tasks. We use LEEP, NCE and LogME to compute scores $\{S_m\}_{m=1}^M$ by applying 10 pre-trained models to the datasets. The correlation τ_w between scores and fine-tuned accuracies are presented in Figure 4.

We can find that LogME has consistently better correlation than LEEP, and outperforms NCE on most datasets (7 datasets out of 9 datasets). Note that LEEP and NCE even show a negative correlation in DTD (Cimpoi et al., 2014), because they rely on the relationship between classes of the pre-trained task and the target task while DTD classes are very different from ImageNet categories. In contrast, LogME still performs reasonably well for DTD.

The smallest τ_w of LogME in Figure 4 is around 0.5, so the probability of a pre-trained model ϕ_1 transferring better than ϕ_2 is at least 75% if ϕ_1 has a larger LogME. For most tasks τ_w of LogME is 0.7 or 0.8, so the probability of correct selection is 85% or 90%, sufficient for practical usage.

5.2. Transferring supervised pre-trained models to a regression task

Besides extensive classification tasks considered above, this section shows how LogME can be used to assess pre-trained models for a regression task, while prior methods (LEEP and NCE) cannot.

The regression task we use is dSprites (Matthey et al., 2017) from VTAB (Zhai et al., 2020) which is commonly used for evaluating the quality of learned representations. The input is an image containing a sprite (heart, square, and ellipse) with varying scale, orientation, and position. Pre-trained models are transferred to predict four scalars (scale, orientation, and (x, y) positions) together, and mean square error (MSE) on the test data is reported. The supervised pre-trained models are the same as Section 5.1 and hyper-parameter tuning scheme follows.

Results are plotted in Figure 5. It is clear that LogME and MSE are well correlated and the correlation coefficient $\tau_w = 0.84$ is very large: if a pre-trained model ϕ_1 has larger LogME than ϕ_2, with 92% probability ϕ_1 is better (has smaller MSE) than ϕ_2 after actually fine-tuning.

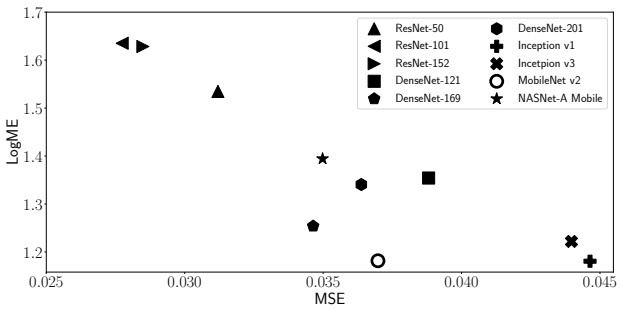

Figure 5. Supervised pre-trained models transferred to dSprites.

5.3. Transferring contrastive pre-trained models to downstream tasks

The recently emerging unsupervised pre-trained models (He et al., 2020) have a projection head with continuous output. However, LEEP and NCE cannot be extended to deal with the projection head of contrastive-based unsupervised pre-trained models because they rely on the relationship between pre-training categories and target categories.

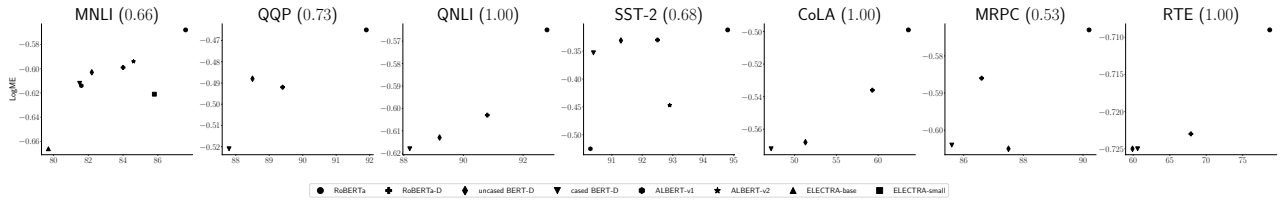

Figure 6. Correlation (τ_w) between fine-tuned accuracy (X-axis) and LogME in 7 GLUE tasks with 8 popular pre-trained language models. One column for each task (with τ_w in the bracket near the task name), and one marker for each pre-trained model.

Since LogME only requires features extracted from pre-trained models, it can be applied to contrastive pre-trained models. To demonstrate this, we use three popular models pre-trained with various training scheme: MoCo V1 (He et al., 2020) with momentum contrast, MoCo V2 (Chen et al., 2020b) with an MLP projection head and strong data augmentation, MoCo 800 trained with 800 epochs as suggested by Chen et al. (2020a), and SimCLR (Chen et al., 2020a) with carefully designed implementation.

Aircraft (Maji et al., 2013), the first dataset (alphabetically) in Section 5.1 is used as the classification task, and dSprites (Matthey et al., 2017) is used as the regression task. Results are shown in Table 3. SimCLR on dSprites is not reported because it does not converge after several trials. LogME gives the *perfect order* of both transferred accuracy and MSE. Note that the order in Aircraft (MoCo V1 $<$ MoCo V2 $<$ MoCo 800) is different from the order in dSprites (MoCo V1 $<$ MoCo 800 $<$ MoCo V2), so the transfer learning performance depends on both the pre-trained model and the target data, emphasizing the importance of *task adaptive* pre-trained model selection. We also observe that LogME values of unsupervised pre-trained models are similar, mainly because unsupervised features are not very discriminative.

Table 3. Use LogME to assess unsupervised pre-trained models.

Pre-trained Network	Aircraft		dSprites	
	Accuracy (%)	LogME	MSE	LogME
MoCo V1	81.68	0.934	0.069	1.52
MoCo V2	84.16	0.941	0.047	1.64
MoCo 800	86.99	0.946	0.050	1.58
SimCLR	88.10	0.950	-	-
	τ_w: 1.0		τ_w: 1.0	

5.4. Transferring pre-trained language models to the GLUE benchmark

To further demonstrate the generality of LogME, we show how LogME can work for pre-trained language models. Again prior works (LEEP and NCE) cannot deal with these pre-trained language models.

Here we take an alternative approach of evaluating the transfer performance $\{T_m\}_{m=1}^{M}$. We do not fine-tune pre-trained models ourselves, but directly use accuracies tuned by others, and check if LogME can correlate well with the results. The HuggingFace Model Hub generously provides lots of pre-trained language models and even provides carefully tuned transfer learning results in some GLUE (Wang et al., 2018) tasks for some models. We take out pre-trained models that have GLUE performance tuned by the HuggingFace organization, and select the top 8 downloaded models: RoBERTa (Liu et al., 2019), RoBERTa-D, uncased BERT-D, cased BERT-D, ALBERT-v1 (Lan et al., 2020), ALBERT-v2 (Lan et al., 2020), ELECTRA-base (Clark et al., 2020), and ELECTRA-small (Clark et al., 2020) ("D" means distilled version). The LogME on seven GLUE classification tasks together with fine-tuned accuracy are plotted in Figure 6. Some models only have results for certain tasks and we keep them as they are. Even though these accuracy numbers are tuned by the HuggingFace organization, *LogME perfectly estimates the ranking of transfer performance for* 3 *tasks* (with $\tau_w = 1$), showing the surprising effectiveness of LogME in pre-trained model selection.

5.5. Efficiency of LogME

LogME is a practical method to assess pre-trained models for transfer learning because it is general, accurate, and efficient. Section 4 shows the generality of LogME by considering features and labels in the general form. Results in this section validates the strong correlation between LogME and ground-truth transfer learning performance, demonstrating that LogME is accurate. Next we quantitatively measure the efficiency of LogME compared to brute-force fine-tuning. The algorithmic complexity is presented in Section 4.3, thus we focus on wall-clock time and memory footprint here.

Results are shown in Table 4. ResNet 50 on Aircraft is used for computer vision, and RoBERTa-D on MNLI task is used for NLP. Both wall-clock time and memory footprint is reported. The cost of computing ground-truth transferability T_m (fine-tuning with hyper-parameter search) serves as the upper bound of pre-trained model assessment. We also list the cost of extracting features by pre-trained models as a reference, which is the lower bound of pre-trained model as-

Table 4. Efficiency of LogME.

	wall-clock time		memory footprint	
Computer Vision	fine-tune (upper bound)	161000s	fine-tune (upper bound)	6.3 GB
	extract feature (lower bound)	37s	extract feature (lower bound)	43 MB
	LogME	50s	LogME	53 MB
	benefit	3200 ↑	benefit	120 ↑
Natural Language Processing	fine-tune (upper bound)	100200s	fine-tune (upper bound)	88 GB
	extract feature (lower bound)	1130s	extract feature (lower bound)	1.2 GB
	LogME	1157s	LogME	1.2 GB
	benefit	86 ↑	benefit	73 ↑

sessment. The cost for the rest models and datasets vary, but the proportion is similar. Note that, because carelessly tuned hyper-parameters cannot tell good models from bad models, it is necessary to attribute the cost of hyper-parameter search to brute-force fine-tuning while LogME does not need hyper-parameter tuning.

It is clear that brute-force fine-tuning is computationally expensive, requiring about a day for one dataset with one pre-trained model. Selecting the best pre-trained model out of 10 models would cost 10 days. Extracting features is very cheap and costs much less. In computer vision, the wall-clock time of LogME is reduced dramatically to $0.31‰$ of fine-tuning, bringing over $3000\times$ speedup while requiring $120\times$ less memory footprint. In the NLP domain, feature extraction is much slower and therefore the wall-clock time speedup is not as striking as computer vision, but still reaching $86\times$ speedup. *In all cases, LogME costs almost the same as the lower bound (feature extraction)*, meaning that LogME makes practical assessment possible with minimal additional cost.

6. Conclusion

A fast, accurate, and general assessment of pre-trained models for transfer learning has great practical significance. This paper takes a probabilistic approach and proposes logarithm of maximum evidence (LogME) to tackle the task adaptive pre-trained model selection problem. The expensive computation of maximizing the marginalized likelihood is optimized by careful implementation, leading to over $3000\times$ speedup compared to vanilla fine-tuning. LogME is applicable to vast transfer learning settings with supervised pre-trained models and unsupervised pre-trained models, downstream classification and regression tasks, vision and language modalities. The impressive generality of LogME and its substantially better performance over prior methods can be interesting to many practitioners.

This paper measures the quality of pre-trained models by their static representations (*i.e.* representations before fine-tuning). It is interesting to consider the dynamic representations (*i.e.* representations after fine-tuning) of pre-trained models to account for the change of pre-trained models during fine-tuning. We leave it as a future work.

Acknowledgements

We would like to thank Ximei Wang, Xinyang Chen, Yang Shu, and Yonglong Tian for helpful discussions. This work was supported by the National Key R&D Program of China (2020AAA0109201), NSFC grants (62022050, 62021002, 61772299), Beijing Nova Program (Z201100006820041), and MOE Innovation Plan of China.

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., and Isard, M. Tensorflow: a system for large-scale machine learning. In *OSDI*, 2016.

Ben-David, S. and Schuller, R. Exploiting task relatedness for multiple task learning. In *COLT*, 2003.

Benoit, S., Zachary, D., Soumith, C., Sam, G., Adam, P., Francisco, M., Adam, L., Gregory, C., Zeming, L., Edward, Y., Alban, D., Alykhan, T., Andreas, K., James, B., Luca, A., Martin, R., Natalia, G., Sasank, C., Trevor, K., Lu, F., and Junjie, B. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*, 2019.

Berg, T., Liu, J., Woo Lee, S., Alexander, M. L., Jacobs, D. W., and Belhumeur, P. N. Birdsnap: Large-scale fine-grained visual categorization of birds. In *CVPR*, 2014.

Bishop, C. M. *Pattern recognition and machine learning*. 2006.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A Simple Framework for Contrastive Learning of Visual Representations. In *ICML*, 2020a.

Chen, X., Fan, H., Girshick, R., and He, K. Improved Baselines with Momentum Contrastive Learning. *arXiv:2003.04297 [cs]*, 2020b.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. Describing textures in the wild. In *CVPR*, 2014.

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. In *ICLR*, 2020.

Cover, T. M. *Elements of information theory*. 1999.

Daunizeau, J. Semi-analytical approximations to statistical moments of sigmoid and softmax mappings of normal variables. *arXiv preprint arXiv:1703.00091*, 2017.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*, 2019.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

Erhan, D., Courville, A., Bengio, Y., and Vincent, P. Why does unsupervised pre-training help deep learning? In *AISTATS*, 2010.

Fagin, R., Kumar, R., and Sivakumar, D. Comparing top k lists. In *SODA*, 2003.

Fei-Fei, L., Fergus, R., and Perona, P. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *CVPR Workshops*, 2004.

Ganin, Y. and Lempitsky, V. Unsupervised Domain Adaptation by Backpropagation. In *ICML*, 2015.

Garg, S., Vu, T., and Moschitti, A. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. In *AAAI*, 2020.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Gull, S. F. Developments in maximum entropy data analysis. In *Maximum entropy and Bayesian methods*. 1989.

Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. In *ACL*, 2020.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

He, K., Girshick, R., and Dollár, P. Rethinking imagenet pre-training. In *ICCV*, 2019.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., and Leskovec, J. Strategies for Pre-training Graph Neural Networks. In *ICLR*, 2020.

Huang, G., Liu, Z., Weinberger, K. Q., and van der Maaten, L. Densely connected convolutional networks. In *CVPR*, 2017.

Jing, L. and Tian, Y. Self-supervised visual feature learning with deep neural networks: A survey. *TPAMI*, 2020.

Jouppi, N. P., Young, C., Patil, N., Patterson, D., Agrawal, G., Bajwa, R., Bates, S., Bhatia, S., Boden, N., and Borchers, A. In-datacenter performance analysis of a tensor processing unit. In *ISCA*, 2017.

Kendall, M. G. A new measure of rank correlation. *Biometrika*, 1938.

Knuth, K. H., Habeck, M., Malakar, N. K., Mubeen, A. M., and Placek, B. Bayesian Evidence and Model Selection. *Digital Signal Processing*, 2015.

Kolesnikov, A., Zhai, X., and Beyer, L. Revisiting Self-Supervised Visual Representation Learning. In *CVPR*, 2019.

Koller, D. and Friedman, N. *Probabilistic graphical models: principles and techniques*. 2009.

Kornblith, S., Shlens, J., and Le, Q. V. Do better imagenet models transfer better? In *CVPR*, 2019.

Kou, Z., You, K., Long, M., and Wang, J. Stochastic Normalization. In *NeurIPS*, 2020.

Krause, J., Deng, J., Stark, M., and Fei-Fei, L. Collecting a large-scale dataset of fine-grained cars. 2013.

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, 2009.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *ICLR*, 2020.

Li, H., Chaudhari, P., Yang, H., Lam, M., Ravichandran, A., Bhotika, R., and Soatto, S. Rethinking the Hyperparameters for Fine-tuning. In *ICLR*, 2020.

Li, X., Grandvalet, Y., and Davoine, F. Explicit Inductive Bias for Transfer Learning with Convolutional Networks. In *ICML*, 2018.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Long, M., Cao, Y., Wang, J., and Jordan, M. Learning Transferable Features with Deep Adaptation Networks. In *ICML*, 2015.

Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., and van der Maaten, L. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018.

Maji, S., Rahtu, E., Kannala, J., Blaschko, M., and Vedaldi, A. Fine-Grained Visual Classification of Aircraft. *arXiv:1306.5151 [cs]*, 2013.

Matthey, L., Higgins, I., Hassabis, D., and Lerchner, A. dsprites: Disentanglement testing sprites dataset, 2017.

Neyshabur, B., Sedghi, H., and Zhang, C. What is being transferred in transfer learning? In *NeurIPS*, 2020.

Nguyen, C., Hassner, T., Seeger, M., and Archambeau, C. LEEP: A New Measure to Evaluate Transferability of Learned Representations. In *ICML*, 2020.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. V. Cats and dogs. In *CVPR*, 2012.

Pruksachatkun, Y., Phang, J., Liu, H., Htut, P. M., Zhang, X., Pang, R. Y., Vania, C., Kann, K., and Bowman, S. R. Intermediate-Task Transfer Learning with Pretrained Language Models: When and Why Does It Work? In *ACL*, 2020.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. *Dataset shift in machine learning*. 2009.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., and Bernstein, M. Imagenet large scale visual recognition challenge. *IJCV*, 2015.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *CVPR*, 2018.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. 2015.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, 2016.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019.

Thrun, S. and Pratt, L. Learning to Learn: Introduction and Overview. In *Learning to Learn*. 1998.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What Makes for Good Views for Contrastive Learning? In *NeurIPS*, 2020.

Tran, A. T., Nguyen, C. V., and Hassner, T. Transferability and hardness of supervised classification tasks. In *ICCV*, 2019.

Vigna, S. A Weighted Correlation Index for Rankings with Ties. In *WWW*, 2015.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *EMNLP*, 2018.

Wolf, T., Chaumond, J., Debut, L., Sanh, V., Delangue, C., Moi, A., Cistac, P., Funtowicz, M., Davison, J., and Shleifer, S. Transformers: State-of-the-art natural language processing. In *EMNLP*, 2020.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., and Le, Q. V. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, 2019.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *NeurIPS*, 2014.

You, K., Kou, Z., Long, M., and Wang, J. Co-Tuning for Transfer Learning. In *NeurIPS*, 2020.

Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. Taskonomy: Disentangling Task Transfer Learning. In *CVPR*, 2018.

Zhai, X., Puigcerver, J., Kolesnikov, A., Ruyssen, P., Riquelme, C., Lucic, M., Djolonga, J., Pinto, A. S., Neumann, M., Dosovitskiy, A., Beyer, L., Bachem, O., Tschannen, M., Michalski, M., Bousquet, O., Gelly, S., and Houlsby, N. A Large-scale Study of Representation Learning with the Visual Task Adaptation Benchmark. *arXiv:1910.04867 [cs, stat]*, 2020.

A. Dataset description and statistics

Aircraft: The dataset contains fine-grained classification of 10,000 aircraft pictures which belongs to 100 classes, with 100 images per class.

Birdsnap: The dataset contains 49,829 images of 500 species of North American birds.

Caltech: The dataset contains 9,144 pictures of objects belonging to 101 categories. There are about 40 to 800 images per category. Most categories have about 50 images.

Cars: The dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images.

CIFAR 10: The dataset consists of 60,000 32x32 colorful images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images.

CIFAR 100: The dataset is just like the CIFAR 10, except it has 100 classes containing 600 images each.

DTD: The dataset contains a collection of 5,640 textural images in the wild, annotated with a series of human-centric attributes. It has 47 classes and 120 images per class.

Pets: The dataset contains 7,049 images of cat and dog species which belongs to 47 classes, with around 200 images per class.

SUN: The dataset contains 39,700 scenery pictures with 397 classes and 100 samples per class.

For all the datasets we use, we respect the official train / val / test splits if they exist, otherwise we use 60% data for training, 20% data for validation (hyper-parameter tuning) and 20% data for testing.

B. Comparing LogME to re-training head

A naïve way to measure the relationship between features and labels is to train a classification / regression head for the downstream task, and to use the head's performance as an assessment (sometimes it is called "linear probing" or "linear protocol evaluation"). Actually we have considered this idea but find that it works not as well as expected.

The issues of re-training head are studied by researchers in visual representation learning, too. Kolesnikov et al. (2019) found that (1) re-training head by second-order optimization is impractical; (2) first-order optimization with gradients is sensitive to the learning rate schedule and takes a long time to converge.

Apart from issues discussed by Kolesnikov et al. (2019), Kornblith et al. (2019) also note that hyper-parameter of logistic regression (strength of L2 regularization) should be tuned extensively, making head re-training inefficient.

Our empirical experiments agree with the above concerns with re-training head, and also find that re-training head does not work as well as expected. In the Caltech dataset, we extract features from 10 pre-trained models, train softmax regressors with tuned hyper-parameters (the L2 regularization strength), and plot the correlation between the best head accuracy and the transfer performance *w.r.t.* the number of hyper-parameter trials in Figure 7. The correlation of LogME is plotted as a reference. Computing LogME requires $3\times$ less time than re-training a head with one fixed hyper-parameter, and re-training head with exhaustive hyper-parameter search is still much inferior to LogME.

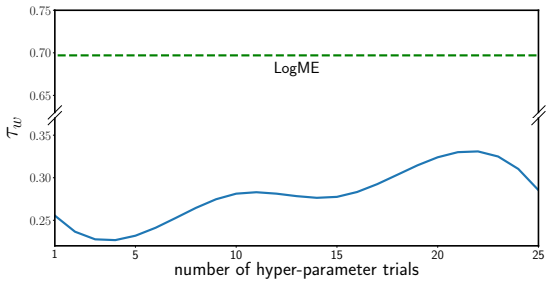

Figure 7. The correlation of re-training head *w.r.t.* the number of hyper-parameter trials. It is clear that re-training head is much worse than LogME.

As a side issue, even if we re-train a head for the downstream task, it is unclear what quantity of the head should be used to measure pre-trained models. Since the performance of downstream tasks are evaluated by accuracy and MSE in transfer learning, it may somewhat cause over-fitting if we use the accuracy and MSE of the re-trained head. Indeed, in Figure 7, when the number of hyper-parameter trials increases, the correlation can even go down, showing the effect of somewhat over-fitting.

Therefore, *re-training head is neither efficient nor effective as LogME*.

C. Original Results in Figures

Original results in figures are shown in the Table 5, Table 6, and Table 7.

Table 5. Original results in Figure 4.

task		ResNet-50	ResNet-101	ResNet-152	DenseNet-121	DenseNet-169	DenseNet-201	Inception v1	Inception v3	MobileNet v2	NASNet-A Mobile	τ_w
Aircraft	Accuracy	86.6	85.6	85.3	85.4	84.5	84.6	82.7	88.8	82.8	72.8	-
	LEEP	-0.412	-0.349	-0.308	-0.431	-0.340	-0.462	-0.795	-0.492	-0.515	-0.506	0.11
	NCE	-0.297	-0.244	-0.214	-0.296	-0.259	-0.322	-0.348	-0.250	-0.411	-0.444	0.40
	LogME	0.946	0.948	0.950	0.938	0.943	0.942	0.934	0.953	0.941	0.948	**0.54**
Birdsnap	Accuracy	74.7	73.8	74.3	73.2	71.4	72.6	73.0	77.2	69.3	68.3	-
	LEEP	-1.647	-1.553	-1.481	-1.729	-1.756	-1.645	-2.483	-1.776	-1.951	-1.835	0.27
	NCE	-1.538	-1.479	-1.417	-1.566	-1.644	-1.493	-1.807	-1.354	-1.815	-1.778	**0.74**
	LogME	0.829	0.836	0.839	0.810	0.815	0.822	0.806	0.848	0.808	0.824	0.67
Caltech	Accuracy	91.8	93.1	93.2	91.9	92.5	93.4	91.7	94.3	89.1	91.5	-
	LEEP	-2.195	-2.067	-1.984	-2.159	-2.039	-2.122	-2.718	-2.286	-2.373	-2.263	0.27
	NCE	-1.820	-1.777	-1.721	-1.807	-1.774	-1.808	-1.849	-1.722	-2.009	-1.966	0.65
	LogME	1.509	1.548	1.567	1.365	1.417	1.428	1.440	1.605	1.365	1.389	**0.70**
Cars	Accuracy	91.7	91.7	92.0	91.5	91.5	91.0	91.0	92.3	91.0	88.5	-
	LEEP	-1.570	-1.370	-1.334	-1.562	-1.505	-1.687	-2.149	-1.637	-1.695	-1.588	0.41
	NCE	-1.181	-1.142	-1.128	-1.111	-1.192	-1.319	-1.201	-1.195	-1.312	-1.334	0.35
	LogME	1.253	1.255	1.260	1.249	1.252	1.251	1.246	1.259	1.250	1.254	**0.65**
CIFAR10	Accuracy	96.8	97.7	97.9	97.2	97.4	97.4	96.2	97.5	95.7	96.8	-
	LEEP	-3.407	-3.184	-3.020	-3.651	-3.345	-3.458	-4.074	-3.976	-3.624	-3.467	0.64
	NCE	-3.395	-3.232	-3.084	-3.541	-3.427	-3.467	-3.338	-3.625	-3.511	-3.436	0.43
	LogME	0.388	0.463	0.469	0.302	0.343	0.369	0.293	0.349	0.291	0.304	**0.78**
CIFAR100	Accuracy	84.5	87.0	87.6	84.8	85.0	86.0	83.2	86.6	80.8	83.9	-
	LEEP	-3.520	-3.330	-3.167	-3.715	-3.525	-3.643	-4.279	-4.100	-3.733	-3.560	0.61
	NCE	-3.241	-3.112	-2.980	-3.304	-3.313	-3.323	-3.253	-3.447	-3.336	-3.254	0.44
	LogME	1.099	1.130	1.133	1.029	1.051	1.061	1.037	1.070	1.039	1.051	**0.79**
DTD	Accuracy	75.2	76.2	75.4	74.9	74.8	74.5	73.6	77.2	72.9	72.8	-
	LEEP	-3.663	-3.718	-3.653	-3.847	-3.646	-3.757	-4.124	-4.096	-3.805	-3.691	-0.08
	NCE	-3.119	-3.199	-3.138	-3.198	-3.218	-3.203	-3.082	-3.261	-3.176	-3.149	-0.38
	LogME	0.761	0.757	0.766	0.710	0.730	0.730	0.727	0.746	0.712	0.724	**0.48**
Pets	Accuracy	92.5	94.0	94.5	92.9	93.1	92.8	91.9	93.5	90.5	89.4	-
	LEEP	-1.031	-0.915	-0.892	-1.100	-1.111	-1.108	-1.520	-1.129	-1.228	-1.150	0.65
	NCE	-0.956	-0.885	-0.862	-0.987	-1.072	-1.026	-1.076	-0.893	-1.156	-1.146	**0.84**
	LogME	1.029	1.061	1.084	0.839	0.874	0.908	0.913	1.191	0.821	0.833	0.58
SUN	Accuracy	64.7	64.8	66.0	62.3	63.0	64.7	62.0	65.7	60.5	60.7	-
	LEEP	-2.611	-2.531	-2.513	-2.713	-2.570	-2.618	-3.153	-2.943	-2.764	-2.687	0.58
	NCE	-2.469	-2.455	-2.444	-2.500	-2.480	-2.465	-2.534	-2.529	-2.590	-2.586	0.77
	LogME	1.744	1.749	1.755	1.704	1.716	1.718	1.715	1.753	1.713	1.721	**0.86**

Table 6. Original results in Figure 5.

task		ResNet-50	ResNet-101	ResNet-152	DenseNet-121	DenseNet-169	DenseNet-201	Inception v1	Inception v3	MobileNet v2	NASNet-A Mobile	τ_w
dSprites	MSE	0.031	0.028	0.028	0.039	0.035	0.036	0.045	0.044	0.037	0.035	-
	LogME	1.53	1.64	1.63	1.35	1.25	1.34	1.18	1.22	1.18	1.39	0.84

Table 7. Original results in Figure 6.

task		RoBERTa	RoBERTa-D	uncased BERT-D	cased BERT-D	ALBERT-v1	ALBERT-v2	ELECTRA-base	ELECTRA-small	τ_w
MNLI	Accuracy	87.6	84.0	82.2	81.5	81.6	84.6	79.7	85.8	-
	LogME	-0.568	-0.599	-0.603	-0.612	-0.614	-0.594	-0.666	-0.621	0.66
QQP	Accuracy	91.9	89.4	88.5	87.8	-	-	-	-	-
	LogME	91.9	89.4	88.5	87.8	-	-	-	-	0.73
QNLI	Accuracy	92.8	90.8	89.2	88.2	-	-	-	-	-
	LogME	-0.565	-0.603	-0.613	-0.618	-	-	-	-	1.00
SST-2	Accuracy	94.8	92.5	91.3	90.4	90.3	92.9	-	-	-
	LogME	-0.312	-0.330	-0.331	-0.353	-0.525	-0.447	-	-	0.68
CoLA	Accuracy	63.6	59.3	51.3	47.2	-	-	-	-	-
	LogME	-0.499	-0.536	-0.568	-0.572	-	-	-	-	1.00
MRPC	Accuracy	90.2	86.6	87.5	85.6	-	-	-	-	-
	LogME	-0.573	-0.586	-0.605	-0.604	-	-	-	-	0.53
RTE	Accuracy	78.7	67.9	59.9	60.6	-	-	-	-	-
	LogME	-0.709	-0.723	-0.725	-0.725	-	-	-	-	1.00