Transferability Between Regression Tasks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Transfer learning has been a widely used technique to adapt a deep learning model trained for one task to another when there is a data distribution shift between these tasks. To improve the effectiveness of transfer learning and to understand relationships between tasks, we consider the problem of transferability estimation between regression tasks and propose two novel transferability estimators that are simple, computationally efficient, yet effective and theoretically grounded. We test our proposed methods extensively in various challenging, practical scenarios and show they significantly outperform existing state-of-the-art regression task transferability estimators in both accuracy and efficiency.

1 Introduction

Transfer learning [18, 27] has been a popular technique to adapt a deep learning model trained for one task to another when there is a data distribution shift between these tasks. To improve transfer learning and to understand task relationships, transferability estimation [4, 23, 30] has recently been studied. The aim of *transferability estimation* is to derive a computationally efficient metric (a score) that can predict the effectiveness of transferring a deep learning model from source to target tasks. This problem has recently gained attention as a means for model and task selection [4, 7, 23, 30, 35] that can potentially improve the performance and reduce the cost of transfer learning. Transferability estimators were also developed and used for checkpoint ranking [15, 17] and few-shot learning [29].

Nearly all existing methods consider only the transferability of classification tasks [4, 10, 16, 17, 22, 23, 28, 30] with very few designed for regression tasks [16, 35]. Furthermore, those few methods proposed for estimating regression model transferability have so far been either complex and inefficient [35] or ineffective [16], as we will show in our experiments. Compared to these previous work, we offer *efficient and theoretically grounded* transferability estimation for regression problems.

Specifically, we propose two families of simple transferability estimators that utilize the penalized mean squared error (MSE) of a linear regression model computed from the source and target training sets. The first family, *Linear MSE*, uses features extracted from a model trained on the source task (the *source model*), while the second family, *Label MSE*, uses the *dummy* labels obtained from the source model. In special cases where the source and target data share the inputs, the Label MSE estimators can be computed efficiently from true labels without a source model. Besides simplicity, our methods also have theoretical properties relating to the expected risk of the target model.

We conduct experiments on two real-world keypoint detection datasets, CUB-200-2011 [32] and OpenMonkey [34], to show the advantages of our methods. The results clearly demonstrate that the proposed methods outperform recently published, state-of-the-art (SotA) transferability estimators for regression problems, such as LogME [35] and TransRate [16], in both effectiveness and efficiency.

Other Related Work. Besides transferability estimation, our paper is also related to previous work that developed theoretical bounds for transfer learning [3, 5, 6, 20, 31, 33]. These bounds are

Submitted to the Workshop on Distribution Shifts, 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

usually hard to compute in practice and thus unsuitable for transferability estimation. Some previous transferability estimators have theoretical bounds on the empirical loss [23, 30], but they were shown for classification and did not bound the expected loss. Our work, on the other hand, focuses on regression and proves generalization bounds for the expected loss.

2 Transferability Between Regression Tasks

Transfer learning settings. Consider a source training set $\mathcal{D}_s = \{(x_i^s, y_i^s)\}_{i=1}^{n_s}$ and a target training set $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^{n_t}$ consisting of n_s and n_t examples respectively, where $x_i^s, x_i^t \in \mathbb{R}^d$ are d-dimensional input vectors, $y_i^s \in \mathbb{R}^{d_s}$ is a d_s-dimensional source label vector, and $y_i^t \in \mathbb{R}^{d_t}$ is a d_t-dimensional target label vector. Note that we allow multi-output regression tasks where $d_s, d_t > 1$.

From the source dataset $\mathcal{D}_s$, we train a deep learning model (w^*, h^*) consisting of a feature extractor w^* and a regression head h^* that minimizes the empirical MSE loss. That is, $w^*, h^* = \operatorname{argmin}_{w,h} \mathcal{L}(w, h; \mathcal{D}_s)$, where $w : \mathbb{R}^d \to \mathbb{R}^{d_r}$ is a feature extractor network, $h : \mathbb{R}^{d_r} \to \mathbb{R}^{d_s}$ is a source regression head network, and $\mathcal{L}(w, h; \mathcal{D}_s) = \frac{1}{n_s} \sum_{i=1}^{n_s} \|y_i^s - h(w(x_i^s))\|^2$ is the empirical MSE loss of the whole model (w, h) on $\mathcal{D}_s$, with $\|\cdot\|$ being the ℓ_2 norm. In practice, we usually consider the source model as a whole (e.g., a ResNet model [14]) and use its first l layers from the input (for some chosen number l) as the feature extractor w. The regression head h is the remaining part of the model from the l-th layer to the output layer, and the prediction for any input x is $h(w(x))$.

After training the optimal source model (w^*, h^*), we perform transfer learning to the target task by freezing the optimal feature extractor w^* and re-training a new regression head k^* using the target dataset $\mathcal{D}_t$, also by minimizing the empirical MSE loss: $k^* = \operatorname{argmin}_k \mathcal{L}(w^*, k; \mathcal{D}_t) = \operatorname{argmin}_k \left\{ \frac{1}{n_t} \sum_{i=1}^{n_t} \|y_i^t - k(w^*(x_i^t))\|^2 \right\}$, where $k : \mathbb{R}^{d_r} \to \mathbb{R}^{d_t}$ is a target regression head network that may have a different architecture than that of h. In general, the regression heads h and k may contain multiple layers and are not necessarily linear. This transfer learning setting is usually called *head re-training* [11, 24, 26] and will be used in our theoretical analysis. In our experiments, we also consider another transfer learning setting called *fine-tuning*, where we fine-tune the trained feature extractor w^* on the target set in addition to re-training k^* [1, 8, 12].

Transferability Estimators. A transferability estimator measures the effectiveness of transfer learning given a pair of source and target tasks. Intuitively, we can use the MSE of the transferred target model $\mathcal{L}(w^*, k^*; \mathcal{D}_t)$ as an estimator. However, that means we need to run the actual transfer learning algorithm, which could be expensive if the network architecture of the target regression heads (e.g., k and k^*) is complex. A simple way to reduce this computational cost is to approximate $\mathcal{L}(w^*, k^*; \mathcal{D}_t)$ using an ℓ_2 regularized linear regression (or Ridge regression) head; that is, we consider the family of Linear MSE estimators below. In our definitions, $\|\cdot\|_F$ is the Frobenius norm.

Definition 1. *A **Linear MSE** transferability estimator between a source dataset $\mathcal{D}_s$ and a target dataset $\mathcal{D}_t$, with a regularization parameter $\lambda \geq 0$, is defined as: $\mathcal{T}_\lambda^{\text{lin}}(\mathcal{D}_s, \mathcal{D}_t) := \min_{A,B} \left\{ \frac{1}{n_t} \sum_{i=1}^{n_t} \|y_i^t - Aw^*(x_i^t) - B\|^2 + \lambda \|A\|_F^2 \right\}$, where $A \in \mathbb{R}^{d_r \times d_t}$ and $B \in \mathbb{R}^{d_t}$.*

We use a regularizer in the estimator to avoid overfitting when $\mathcal{D}_t$ is small. Previous work such as LogME [35] prevents overfitting by taking a more complicated and expensive Bayesian approach. Here we argue that a simple regularization can tackle this issue effectively and more efficiently. Given a pre-trained feature extractor w^*, we can compute $\mathcal{T}_\lambda^{\text{lin}}(\mathcal{D}_s, \mathcal{D}_t)$ efficiently by solving a Ridge regression problem. If the target head is restricted to only linear regression model, the non-regularized Linear MSE $\mathcal{T}_0^{\text{lin}}(\mathcal{D}_s, \mathcal{D}_t)$ is equal to the MSE of the transferred target model (w^*, k^*) on $\mathcal{D}_t$. If the target head has more than one layer with a non-linear activation function, $\mathcal{T}_\lambda^{\text{lin}}(\mathcal{D}_s, \mathcal{D}_t)$ can be regarded as using a regularized linear model to approximate this non-linear head.

Although Linear MSE scores can be computed efficiently, they may still be expensive if the feature vectors $w^*(x_i^t)$ are high-dimensional. Furthermore, in many cases, we need to compute and compare the scores for several pairs of datasets, resulting in high computational costs. To further reduce the costs, we propose the Label MSE transferability estimators below, which replaces $w^*(x_i^t)$ by the "dummy" source label $z_i = h^*(w^*(x_i^t))$. Using dummy labels from the trained source model (w^*, h^*) is a technique previously used in the LEEP transferability scores for classification tasks [23].

Definition 2. *A Label MSE transferability estimator between a source dataset $\mathcal{D}_s$ and a target dataset $\mathcal{D}_t$, with a regularization parameter $\lambda \geq 0$, is defined as: $\mathcal{T}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t) := \min_{A,B} \left\{ \frac{1}{n_t} \sum_{i=1}^{n_t} \|y_i^t - Az_i - B\|^2 + \lambda \|A\|_F^2 \right\}$, where $A \in \mathbb{R}^{d_s \times d_t}$, $B \in \mathbb{R}^{d_t}$, and $z_i = h^*(w^*(x_i^t)), \forall i$.*

Table 1: Correlation coefficients when transferring from OpenMonkey to CUB-200-2011.

Transfer setting	Label-based method				Feature-based method			
	LabLogME	LabTransRate	LabMSE0	LabMSE1	LogME	TransRate	LinMSE0	LinMSE1
Head re-training	0.958	0.165	**0.992***	**0.992***	0.969	0.121	0.982	0.988*
Half fine-tuning	0.706	0.392	**0.882***	**0.882***	0.870	0.304	0.865	0.872*
Full fine-tuning	0.691	0.410	**0.870***	**0.870***	0.861*	0.311	0.854	0.861*

From our definitions, the lower the Linear MSE or Label MSE, the better the transferability. Since the size of z_i is usually much smaller than that of $w^*(x_i^t)$ (i.e., $d_s \ll d_r$), computing the Label MSE is usually cheaper than computing the Linear MSE. We prove novel properties of Label MSE below.

Theoretical Results. We prove some theoretical properties (in the form of generalization bounds) for the Label MSE with ReLU feed-forward neural networks. We first make the standard assumption that the source data and the target data are drawn iid from the unknown distributions $\mathbb{P}(X^s, Y^s)$ and $\mathbb{P}(X^t, Y^t)$ respectively. Given any model (w, k) for the target task, the expected risk of (w, k) is defined as $\mathcal{R}(w, k) = \mathbb{E}_{(x^t,y^t)\sim\mathbb{P}(X^t,Y^t)}\{\|y^t - k(w(x^t))\|^2\}$. Our main theoretical result is Theorem 1 below, with detailed proof in Appendix A.1. In this theorem, L is the number of layers of the ReLU feed-forward neural network (w, k), and we assume the number of hidden nodes and parameters in each layer are upper bounded by H and $M \geq 1$ respectively.

Theorem 1. *For any $\lambda \geq 0$ and $\delta > 0$, with probability at least $1 - \delta$, we have: $\mathcal{R}(w^*, k^*) \leq \mathcal{T}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t) + C(d, d_t, M, H, L, \delta)/\sqrt{n_t}$.*

This theorem shows that the expected risk is upper bounded by the Label MSE plus a complexity term $C(d, d_t, M, H, L, \delta)/\sqrt{n_t}$ that depends on the architecture of the target network and the target dataset (i.e., the input/output dimensions and the dataset size). When the complexity term is small (e.g., when n_t is large), the Label MSE will be a tighter bound for the expected risk. An experiment to investigate the usefulness of our bounds is in Appendix B.5.

Setting with Shared Inputs. When the source and target data have the same inputs, i.e., $x_i^s = x_i^t = x_i$ and $n_s = n_t = n$, we can compute the Label MSE directly from the labels without training the source model (w^*, h^*) or computing the dummy labels. Formally, we can use the following new definition for the Label MSE: $\widehat{\mathcal{T}}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t) := \min_{A,B} \left\{\frac{1}{n}\sum_{i=1}^n \|y_i^t - Ay_i^s - B\|^2 + \lambda\|A\|_F^2\right\}$, where we train a Ridge regression model directly from the corresponding label pairs (y_i^s, y_i^t). This new transferability estimator is *less expensive* since we do not need to train the source model. We can also prove similar generalization bounds with $\widehat{\mathcal{T}}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)$ (see Appendix A.2).

3 Experiments

We evaluate our transferability estimators on the keypoint regression tasks using two large-scale public datasets: (1) **CUB-200-2011** [32]: This dataset contains 11,788 bird images with 15 labeled keypoints indicating 15 different parts of a bird body, and (2) **OpenMonkey** [34]: This dataset contains over 100,000 monkey images in natural contexts, annotated with 17 body keypoints.

We use ResNet34 [14] as the model backbone due to its good performance. Following [23, 30], we investigate how well our transferability estimators correlate (using Pearson correlation) with the *test* MSE of the target model obtained from actual transfer learning. We consider three transfer learning algorithms: (1) **head re-training**: We fix all layers of the source model up until the penultimate layer and re-train the last fully-connected (FC) layer; (2) **half fine-tuning**: We fine-tune the last convolutional block and all the FC layers of the source model (around half of the parameters in the network), while keeping all other layers fixed; and (3) **full fine-tuning**: We fine-tune the whole source model using the target training set. More details of our experiment settings are in Appendix B.1.

For our methods, we consider $\lambda = 0$ (**LinMSE0** and **LabMSE0**) and $\lambda = 1$ (**LinMSE1** and **LabMSE1**). The effects of λ on our algorithms are investigated in Appendix B.4. We compare with two recent SotA baselines for regression: **LogME** [35] and **TransRate** [16]. Besides their usual versions, we also consider the label-based versions where these estimators are computed from dummy labels and target labels (**LabLogME** and **LabTransRate**). As in [16], we divide the target label values into 5 equal-sized bins to compute TransRate and LabTransRate.

General Transfer Between Two Different Domains. We first consider the general case where source models are trained on OpenMonkey and then transferred to CUB-200-2011. Specifically, we

Table 2: Correlation coefficients when transferring between tasks with shared inputs. The last row shows the average running time (in milliseconds) on CUB-200-2011.

Dataset	Transfer setting	Label-based method				Feature-based method			
		LabLogME	LabTransRate	LabMSE0	LabMSE1	LogME	TransRate	LinMSE0	LinMSE1
Open Monkey	Head re-training	0.890	0.666	0.973*	0.973*	0.646	0.711	0.949	**0.981***
	Half fine-tuning	0.615	0.340	0.754*	0.754*	0.391	0.488	**0.893***	0.787
	Full fine-tuning	0.569	0.269	0.705*	0.705*	0.352	0.439	**0.852***	0.739
CUB-200-2011	Head re-training	0.547	0.008	0.916*	0.916*	0.889	0.029	0.921	**0.932***
	Half fine-tuning	0.400	0.006	0.536*	0.536*	0.560	0.006	**0.628***	0.607
	Full fine-tuning	**0.120***	0.001	0.056	0.056	0.099	0.100*	0.097	0.093
Average running time (ms)		3.55	4.11	2.87	**2.58***	112.99	94.57	107.48	27.41*

train a source model for each of the 17 OpenMonkey keypoints and transfer them to each of the 15 CUB-200-2011 keypoints, resulting in a total of 255 final models. Since each keypoint consists of x and y positions, all source and target tasks in this experiment have two dimensional labels. The actual MSEs of these models are computed on the respective test sets and then used to calculate the Pearson correlation coefficients with the transferability estimators. In this experiment, the label-based estimators are computed from the dummy labels and the actual target labels.

Table 1 shows the results for this experiment.[1] From the results, TransRate and LabTransRate perform poorly in this regression setting, while our methods are equal or better than LogME and LabLogME in all cases, especially when using $\lambda = 1$ (LinMSE1) or dummy labels (LabMSE0 and LabMSE1). We observe that adding a regularizer in LabMSE1 does not significantly change its correlation coefficient compared to LabMSE0. This is because we scaled the labels to $[0, 1]$ as a pre-processing step, so a regularizer with $\lambda = 1$ has negligible effect on the scores. It is also surprising that LabMSE0 and LabMSE1 are better than LinMSE0 and LinMSE1. One possible explanation for this phenomenon is that the dummy labels (i.e., body parts of monkeys) give more information about the target labels (i.e., body parts of birds) than the extracted features.

Transfer Between Tasks with Shared Inputs. In this experiment, we consider the setting where the source and target tasks have the same inputs. Since each image in our datasets contains multiple keypoints, we can use any two different keypoints on the same dataset as source and target tasks. In total, we can construct 210 source-target pairs for CUB-200-2011 and 272 pairs for OpenMonkey. The labels for all source and target tasks are also 2-dimensional real values. We repeat the previous experiment with these source-target pairs for each dataset. The main difference in this experiment is that we use the *true* source labels (instead of dummy labels) when computing LabLogME, LabTransRate, LabMSE0, and LabMSE1. Thus, these estimators can be computed without any source models in this setting, and hence incurring very low computational costs.

Results for this experiment are in Table 2. Here both versions of TransRate perform poorly on CUB-200-2011, while TransRate is slightly better than LogME on OpenMonkey. In the first 5 settings of Table 2, LabMSE0 and LabMSE1 both outperform LabLogME and LabTransRate, while LinMSE0 and LinMSE1 both outperform LogME and TransRate. In these settings, the LinMSE methods achieve the best correlations overall. In the last setting (full fine-tuning on CUB-200-2011), all methods perform poorly. We also report in Table 4- 5 (in the appendix) additional results for each source task where our methods are consistently better than the baselines in most cases.

Efficiency of Our Methods. In the last row of Table 2, we compare the average running time of different methods in the shared inputs setting on CUB-200-2011. We can observe that LabMSE0, LabMSE1, LinMSE0, and LinMSE1 are all faster than the corresponding label-based or feature-based baselines. Overall, LabMSE1 and LinMSE1 achieve the best running time among the label-based and feature-based methods respectively. Additional efficiency results are in Appendix B.2.

4 Conclusion

We proposed Linear MSE and Label MSE, two families of simple but effective transferability estimation methods for regression, and proved novel theoretical bounds for the expected risk of the target models using these estimators. Our experiments showed that the proposed methods are superior to recent, relevant SotA methods in terms of both effectiveness and efficiency.

[1] In our tables, bold numbers indicate the best results in each row, while asterisks (*) indicate the best results among the corresponding label-based or feature-based methods.

References

[1] Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014.

[2] Raman Arora, Amitabh Basu, Poorya Mianjy, and Anirbit Mukherjee. Understanding deep neural networks with rectified linear units. In *ICLR*, 2018.

[3] Kamyar Azizzadenesheli, Anqi Liu, Fanny Yang, and Animashree Anandkumar. Regularized learning for domain adaptation under label shifts. In *ICLR*, 2019.

[4] Yajie Bao, Yang Li, Shao-Lun Huang, Lin Zhang, Lizhong Zheng, Amir Zamir, and Leonidas Guibas. An information-theoretic approach to transferability in task transfer learning. In *ICIP*, 2019.

[5] Shai Ben-David and Reba Schuller. Exploiting task relatedness for multiple task learning. *Learning theory and kernel machines*, 2003.

[6] John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning bounds for domain adaptation. In *NeurIPS*, 2007.

[7] Daniel Bolya, Rohit Mittapalli, and Judy Hoffman. Scalable diverse model selection for accessible transfer learning. In *NeurIPS*, 2021.

[8] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[10] Aditya Deshpande, Alessandro Achille, Avinash Ravichandran, Hao Li, Luca Zancato, Charless Fowlkes, Rahul Bhotika, Stefano Soatto, and Pietro Perona. A linearized framework and a new benchmark for model selection for fine-tuning. *arXiv:2102.00084*, 2021.

[11] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

[12] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[13] Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In *COLT*, 2018.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[15] Jiaji Huang, Qiang Qiu, and Kenneth Church. Exploiting a zoo of checkpoints for unseen tasks. In *NeurIPS*, 2021.

[16] Long-Kai Huang, Junzhou Huang, Yu Rong, Qiang Yang, and Ying Wei. Frustratingly easy transferability estimation. In *ICML*, 2022.

[17] Yandong Li, Xuhui Jia, Ruoxin Sang, Yukun Zhu, Bradley Green, Liqiang Wang, and Boqing Gong. Ranking neural checkpoints. In *CVPR*, 2021.

[18] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. Deep transfer learning with joint adaptation networks. In *ICML*, 2017.

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[20] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *COLT*, 2009.

[21] Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dSprites: Disentanglement testing Sprites dataset, 2017. `https://github.com/deepmind/dsprites-dataset/`.

[22] Cuong N Nguyen, Lam Si Tung Ho, Vu Dinh, Tal Hassner, and Cuong V Nguyen. Generalization bounds for deep transfer learning using majority predictor accuracy. In *ISITA*, 2022.

[23] Cuong V Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. LEEP: A new measure to evaluate transferability of learned representations. In *ICML*, 2020.

[24] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *CVPR*, 2014.

[25] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[26] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPR Workshops*, 2014.

[27] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *ICANN*, 2018.

[28] Yang Tan, Yang Li, and Shao-Lun Huang. OTCE: A transferability metric for cross-domain cross-task representations. In *CVPR*, 2021.

[29] Xinyi Tong, Xiangxiang Xu, Shao-Lun Huang, and Lizhong Zheng. A mathematical framework for quantifying transferability in multi-source transfer learning. In *NeurIPS*, 2021.

[30] Anh T Tran, Cuong V Nguyen, and Tal Hassner. Transferability and hardness of supervised classification tasks. In *ICCV*, 2019.

[31] Nilesh Tripuraneni, Michael Jordan, and Chi Jin. On the theory of transfer learning: The importance of task diversity. In *NeurIPS*, 2020.

[32] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The Caltech-UCSD Birds-200-2011 Dataset. *Technical Report*, 2011. `https://authors.library.caltech.edu/27452/`.

[33] Boyu Wang, Jorge Mendez, Mingbo Cai, and Eric Eaton. Transfer learning via minimizing the performance gap between domains. In *NeurIPS*, 2019.

[34] Yuan Yao, Abhiraj Abhiraj Mohan, Eliza Bliss-Moreau, Kristine Coleman, Sienna M Freeman, Christopher J Machado, Jessica Raper, Jan Zimmermann, Benjamin Y Hayden, and Hyun Soo Park. OpenMonkeyChallenge: Dataset and Benchmark Challenges for Pose Tracking of Non-human Primates. *bioRxiv*, 2021. `http://openmonkeychallenge.com/`.

[35] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. LogME: Practical assessment of pre-trained models for transfer learning. In *ICML*, 2021.

A Proofs and Additional Theoretical Results

A.1 Theoretical Results for Label MSE

We detail the proof of Theorem 1 in this section. Throughout the section, we will assume the space of all target regression heads k, which may have more than one layer, is a superset of all the linear regression models. This assumption is generally true for ReLU networks [2]. First, we prove the following lemma relating the empirical MSE loss $\mathcal{L}(w^*, k^*; \mathcal{D}_t)$ of the transferred target model (w^*, k^*) and the Label MSE.

Lemma 1. *For all $\lambda \geq 0$, we have $\mathcal{L}(w^*, k^*; \mathcal{D}_t) \leq \mathcal{T}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)$.*

Proof. Denote $A^*, B^* = \text{argmin}_{A,B} \left\{ \frac{1}{n_t} \sum_{i=1}^{n_t} \|y_i^t - Az_i - B\|^2 + \lambda \|A\|_F^2 \right\}$. For all k, we have:

$$
\begin{aligned}
\sqrt{\mathcal{L}(w^*, k^*; \mathcal{D}_t)} &\leq \sqrt{\mathcal{L}(w^*, k; \mathcal{D}_t)} && \text{(definition of } k^*\text{)} \\
&= \Big[\frac{1}{n_t} \sum_{i=1}^{n_t} \|y_i^t - k(w^*(x_i^t))\|^2 \Big]^{1/2} && \text{(definition of } \mathcal{L}\text{)} \\
&\leq \Big[\frac{1}{n_t} \sum_{i=1}^{n_t} \|y_i^t - A^* z_i - B^*\|^2 \Big]^{1/2} + \Big[\frac{1}{n_t} \sum_{i=1}^{n_t} \|A^* z_i + B^* - k(w^*(x_i^t))\|^2 \Big]^{1/2} && \text{(triangle inequality)} \\
&\leq \sqrt{\mathcal{T}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)} + \Big[\frac{1}{n_t} \sum_{i=1}^{n_t} \|A^* z_i + B^* - k(w^*(x_i^t))\|^2 \Big]^{1/2} \\
&= \sqrt{\mathcal{T}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)} + \Big[\frac{1}{n_t} \sum_{i=1}^{n_t} \|A^* h^*(w^*(x_i^t)) + B^* - k(w^*(x_i^t))\|^2 \Big]^{1/2}. && \text{(definition of } z_i\text{)}
\end{aligned}
$$

By choosing $k(\cdot) = A^* h^*(\cdot) + B^*$, the second term in the above inequality becomes 0. This implies $\sqrt{\mathcal{L}(w^*, k^*; \mathcal{D}_t)} \leq \sqrt{\mathcal{T}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)}$. □

To derive the generalization bound for $\mathcal{R}(w^*, k^*)$, we also need to prove the uniform bound in Lemma 2 below. Theorem 1 easily follows from Lemma 1 and 2. Without loss of generality, we also assume all input and output data are upper bounded by 1 in ℓ_∞-norm. This assumption can easily be satisfied by a pre-processing step that scales them to $[0, 1]$ in ℓ_∞-norm.

Lemma 2. *For any $\delta > 0$, with probability at least $1-\delta$, for all ReLU feed-forward neural network (w, k) of the target task, we have: $|\mathcal{R}(w, k) - \mathcal{L}(w, k; \mathcal{D}_t)| \leq C(d, d_t, M, H, L, \delta)/\sqrt{n_t}$, where $C(d, d_t, M, H, L, \delta) = 16M^{2L+2}H^{2L}[d_t^2 d\sqrt{L+1+\ln d} + d_t d^2\sqrt{2\ln(4/\delta)}]$.*

Proof. Given a real-valued function class $\mathcal{G}$ and a set of data points $\mathcal{D} = \{u_i\}_{i=1}^n$, the (empirical) Rademacher complexity $\widehat{R}_{\mathcal{D}}(\mathcal{G})$ is defined as: $\widehat{R}_{\mathcal{D}}(\mathcal{G}) = \mathbb{E}_\epsilon\left[\sup_{g\in\mathcal{G}} \frac{1}{n}\sum_{i=1}^n \epsilon_i g(u_i)\right]$, where $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n)$ is a vector uniformly distributed in $\{-1, +1\}^n$.

In our setting, the hypothesis space Φ is the class of L-layer ReLU feed-forward neural networks whose number of hidden nodes and parameters in each layer are bounded from above by H and $M \geq 1$ respectively. For all $(w, k) \in \Phi$ and x such that $\|x\|_\infty \leq 1$, we have $\|k(w(x))\|_\infty \leq dM^{L+1}H^L$.

Define $f_{w,k}(x, y) = y - k(w(x))$ and note that $f_{w,k}(x, y) \in \mathbb{R}^{d_t}$. For any $j = 1, 2, \ldots, d_t$, let $[\cdot]_j$ be the projection map to the j-th coordinate. We consider the following real-valued function classes:

$$\begin{aligned}
\mathcal{F} &= \{\|f_{w,k}\|^2 : (w, k) \in \Phi\},\\
\mathcal{F}_j &= \{[f_{w,k}]_j : (w, k) \in \Phi\},\\
\Phi_j &= \{[k(w(\cdot)]_j : (w, k) \in \Phi\},
\end{aligned}$$

where each element of $\mathcal{F}$ or $\mathcal{F}_j$ is a function with variables (x, y), and each element of Φ_j is a function with variable x. Let $\mathcal{D}_t^x = \{x_i^t\}_{i=1}^{n_t}$ be the set of target inputs. By Theorem 2 of [13], for all $j = 1, 2, \ldots, d_t$, we have $\widehat{R}_{\mathcal{D}_t^x}(\Phi_j) \leq 2d_t M^{L+1}H^L\sqrt{(L+1+\ln d)/n_t}$.

For any $i = 1, 2, \ldots, n_t$, the function $r_i(a) = (a - y_i^t)^2$ mapping from $a \in [-dM^{L+1}H^L, dM^{L+1}H^L]$ to $\mathbb{R}$ is Lipschitz with constant $4dM^{L+1}H^L$. Thus, applying the Contraction Lemma (Lemma 26.9 in [25]), we obtain $\widehat{R}_{\mathcal{D}_t}(\mathcal{F}_j) \leq 4dM^{L+1}H^L\widehat{R}_{\mathcal{D}_t^x}(\Phi_j) \leq 8dd_t M^{2L+2}H^{2L}\sqrt{(L+1+\ln d)/n_t}$.

Therefore, $\widehat{R}_{\mathcal{D}_t}(\mathcal{F}) \leq \sum_{j=1}^{d_t}\widehat{R}_{\mathcal{D}_t}(\mathcal{F}_j) \leq 8dd_t^2 M^{2L+2}H^{2L}\sqrt{(L+1+\ln d)/n_t}$. Using this inequality, the result of Lemma 2 follows from Theorem 26.5 in [25]. □

A.2 Theoretical Results for Setting with Shared Inputs

Recall that $\widehat{\mathcal{T}}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t) := \min_{A,B}\left\{\frac{1}{n}\sum_{i=1}^n \|y_i^t - Ay_i^s - B\|^2 + \lambda\|A\|_F^2\right\}$, and denote $A_\lambda^*, B_\lambda^* := \operatorname{argmin}_{A,B}\left\{\frac{1}{n}\sum_{i=1}^n \|y_i^t - Ay_i^s - B\|^2 + \lambda\|A\|_F^2\right\}$. We first show the following lemma for the empirical loss $\mathcal{L}(w^*, k^*; \mathcal{D}_t)$ of the transferred target model (w^*, k^*).

Lemma 3. *For any $\lambda \geq 0$, we have: $\sqrt{\mathcal{L}(w^*, k^*; \mathcal{D}_t)} \leq \sqrt{\widehat{\mathcal{T}}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)} + \|A_\lambda^*\|_F\sqrt{\mathcal{L}(w^*, h^*; \mathcal{D}_s)}$.*

Proof. For all k, we have:

$$\begin{aligned}
&\sqrt{\mathcal{L}(w^*, k^*; \mathcal{D}_t)} \leq \sqrt{\mathcal{L}(w^*, k; \mathcal{D}_t)} && \text{(definition of } k^*)\\
&= \left[\frac{1}{n}\sum_{i=1}^n \|y_i^t - k(w^*(x_i))\|^2\right]^{1/2} && \text{(definition of } \mathcal{L})\\
&\leq \left[\frac{1}{n}\sum_{i=1}^n \|y_i^t - A_\lambda^* y_i^s - B_\lambda^*\|^2\right]^{1/2} + \left[\frac{1}{n}\sum_{i=1}^n \|A_\lambda^* y_i^s + B_\lambda^* - k(w^*(x_i))\|^2\right]^{1/2} && \text{(triangle inequality)}\\
&\leq \sqrt{\widehat{\mathcal{T}}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)} + \left[\frac{1}{n}\sum_{i=1}^n \|A_\lambda^* y_i^s + B_\lambda^* - k(w^*(x_i))\|^2\right]^{1/2}. && \text{(definition of } \widehat{\mathcal{T}}_\lambda^{\text{lab}})
\end{aligned}$$

Picking $k(\cdot) = A_\lambda^* h^*(\cdot) + B_\lambda^*$, the above inequality becomes:

$$\begin{aligned}
\sqrt{\mathcal{L}(w^*, k^*; \mathcal{D}_t)} &\leq \sqrt{\widehat{\mathcal{T}}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)} + \left[\frac{1}{n}\sum_{i=1}^n \|A_\lambda^*[y_i^s - h^*(w^*(x_i))]\|^2\right]^{1/2}\\
&\leq \sqrt{\widehat{\mathcal{T}}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)} + \|A_\lambda^*\|_F\left[\frac{1}{n}\sum_{i=1}^n \|y_i^s - h^*(w^*(x_i))\|^2\right]^{1/2}\\
&= \sqrt{\widehat{\mathcal{T}}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)} + \|A_\lambda^*\|_F\sqrt{\mathcal{L}(w^*, h^*; \mathcal{D}_s)}.
\end{aligned}$$

□

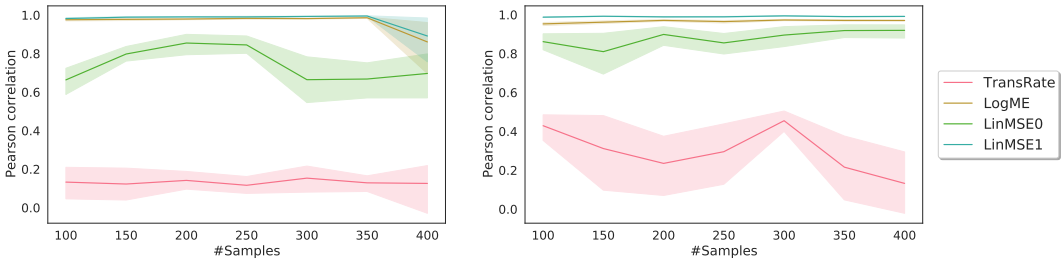


Figure 1: Correlation coefficients with respect to different target training set sizes for 4 transferability estimators on CUB-200-2011 (left) and OpenMonkey (right).

Using Lemma 2 and 3, we obtain the following generalization bound for this setting with shared inputs.

Theorem 2. *For any $\lambda \geq 0$ and $\delta > 0$, with probability at least $1-\delta$, we have:*

$$\sqrt{\mathcal{R}(w^*, k^*)} \leq \sqrt{\widehat{\mathcal{T}}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)} + \|A_\lambda^*\|_F \sqrt{\mathcal{L}(w^*, h^*; \mathcal{D}_s)} + \sqrt{C(d, d_t, M, H, L, \delta)}/n^{\frac{1}{4}}.$$

Proof. Applying Lemma 2 for (w^*, k^*), with probability at least $1-\delta$, we have:

$$\begin{aligned}
\sqrt{\mathcal{R}(w^*, k^*)} &= \sqrt{\mathcal{L}(w^*, k^*; \mathcal{D}_t) + [\mathcal{R}(w^*, k^*) - \mathcal{L}(w^*, k^*; \mathcal{D}_t)]} \\
&\leq \sqrt{\mathcal{L}(w^*, k^*; \mathcal{D}_t)} + \sqrt{|\mathcal{R}(w^*, k^*) - \mathcal{L}(w^*, k^*; \mathcal{D}_t)|} \\
&\leq \sqrt{\widehat{\mathcal{T}}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)} + \|A_\lambda^*\|_F \sqrt{\mathcal{L}(w^*, h^*; \mathcal{D}_s)} + \sqrt{C(d, d_t, M, H, L, \delta)}/n^{\frac{1}{4}}. \qquad \square
\end{aligned}$$

From the theorem, the Label MSE $\widehat{\mathcal{T}}_\lambda^{\text{lab}}(\mathcal{D}_s, \mathcal{D}_t)$ can indirectly tell us information about the expected risk $\mathcal{R}(w^*, k^*)$ without actually training w^*, h^*, and k^*. This bound becomes tighter when n is large or $\mathcal{L}(w^*, h^*; \mathcal{D}_s)$ is small (e.g., when the source model is expressive enough to fit the source data).

B Additional Experiment Details and Results

B.1 More Details of the Experiment Settings

In our experiments, we train our source models from scratch using the MSE loss with the AdamW optimizer [19], which we run for 40 epochs with batch size of 64 and the cosine learning rate scheduler. To obtain good source models, we resize all input images to 256×256 and apply basic image augmentations without horizontal flipping (i.e., affine transformation, Gaussian blur, and color jitter). We also scale all labels into $[0, 1]$ using the width and height of the input images.

For the transfer learning setting with head re-training, we freeze the trained feature extractor and re-train the regression head on the target dataset using the same setting above, except that we run 15 epochs on the CUB-200-2011 dataset and 30 epochs on the OpenMonkey dataset. For half fine-tuning, we unfreeze the last convolution layer and the head classifier since the number of trainable parameters is around half of the total number of parameters. For full fine-tuning, we unfreeze the whole network. In these two fine-tuning settings, we fine-tune for 15 epochs on both datasets.

B.2 Additional Experiment: Evaluations on Small Target Sets

This experiment evaluates the effectiveness of the feature-based transferability estimators (LogME, TransRate, LinMSE0, and LinMSE1) in the small data regime where the number of samples is smaller than the feature dimension. For this experiment, we fix a source task (*Belly* for CUB-200-2011 and *Right eye* for OpenMonkey) and transfer to all other tasks in the corresponding dataset using head re-training. For each target task, instead of using the full data, we randomly select a subset of 100 to 400 images to perform transfer learning and to compute the transferability scores. The actual MSEs of the transferred models are still computed using the full target test sets.

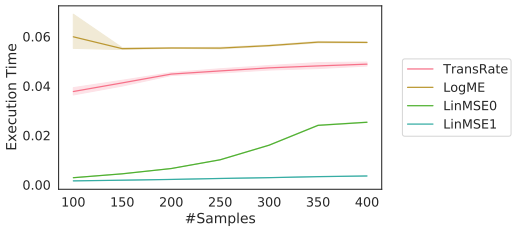


Figure 2: Average running time (milliseconds) with respect to different target training set sizes.

Figure 1 compares the correlations of the 4 methods on different target set sizes between 100 and 400. The results are averaged over 10 runs with 10 different random seeds. From the figure, LogME and LinMSE1 are better than TransRate and LinMSE0. This is expected since LogME and LinMSE1 are designed to avoid overfitting on small data. Both LogME and LinMSE1 are also more stable, but LinMSE1 is slightly better than LogME on all dataset sizes. In Figure 2, we also compare the average running time of the 4 transferability estimators on the CUB-200-2011 dataset. This figure clearly shows that our methods, LinMSE0 and LinMSE1, are more computationally efficient than LogME and TransRate.

B.3 Additional Experiment: Source Task Selection

Source task selection is important for applying transfer learning. It allows us to choose the right source data or model that can offer the best transfer learning performance. In this experiment, we examine the application of our transferability estimation methods for selecting source data on the CUB-200-2011 dataset. We use the shared inputs setting with head re-training in Section 3, but fix one of the tasks as the target and choose the best source task from the rest of the task pool. We repeat this process for all 15 target tasks and measure the top-k matching rate of each transferability measure. The top-k matching rate is defined as $m_{\text{match}}/m_{\text{target}}$, where m_{target} is the total number of target tasks (15 in our case), and m_{match} is the number of times the selected source task gives a target model within the best k models. Here the best k models are determined by the actual test MSE on the target task.

According to our experiment results, the *top*-5 *matching rates* obtained are: 0/15 (LabLogME), 0/15 (LabTransRate), 15/15 (LabMSE0), 15/15 (LabMSE1), 15/15 (LogME), 0/15 (TransRate), 15/15 (LinMSE0), and 15/15 (LinMSE1), while the *top*-3 *matching rates* are: 0/15 (LabLogME), 0/15 (LabTransRate), 0/15 (LabMSE0), 0/15 (LabMSE1), 15/15 (LogME), 0/15 (TransRate), 15/15 (LinMSE0), and 15/15 (LinMSE1). We also observe that all transferability scores perform equally poorly (0/15) on top-1 metric. These results show our methods, LinMSE0 and LinMSE1, are comparable with LogME for source task selection. When the source models are not available (i.e., LogME, TransRate, LinMSE0, and LinMSE1 are not readily available), our methods (LabMSE0 and LabMSE1) are better than LabLogME for source dataset selection, measured by the top-5 metric.

B.4 Additional Experiment: Effects of λ

In this experiment, we investigate the effects of λ on our proposed transferability estimators. We use the shared inputs setting in Section 3 with the CUB-200-2011 dataset and vary the value of λ in the range [0, 100] for both LabMSE and LinMSE. Table 3 reports the results for this experiment with all three transfer learning settings. As noted in Section 3, adding a regularizer does not affect LabMSE much (unless λ is very large) since the labels used to compute LabMSE were already scaled to [0,1]. For head re-training, increasing λ improves the correlation for both LabMSE and LinMSE, with the best correlations achieved at $\lambda = 100$. For half fine-tuning, increasing λ slightly improves the correlation for LabMSE, but decreases the correlation for LinMSE. In this setting, LabMSE is best at $\lambda = 100$, while LinMSE is best at $\lambda = 0$. For full fine-tuning, increasing λ slightly decreases the correlations for both LabMSE and LinMSE, with the best correlations achieved at $\lambda = 0$.

We can observe from these results that varying λ does not significantly change the correlations. In the head re-training and half fine-tuning settings (where we have significant correlations for at least one transferability estimators), for **all** values of λ in Table 3, LabMSE is better than LabLogME and LabTransRate, while LinMSE is better than LogME and LabTransRate. These results show that our proposed methods are generally robust to the value of λ and also better than the baselines for a wide range of λ values.

B.5 Additional Experiment: Usefulness of Theoretical Bounds

Although our theoretical bounds show a relationship between the expected risk of the optimal transferred model and our transferability estimators, these bounds could be loose in practice unless the number of samples is large. To show the usefulness of our bounds in practice, we conduct an experiment to investigate the generalization gap in the general transfer setting with head re-training in Section 3. The generalization gap is defined as the *difference between our transferability score and the MSE of the transferred model*. According to our theorems, this generalization gap is bounded above by the complexity term. We will compare the generalization gap with our transferability score and inspect whether it has any significant correlation with the actual transferred MSE.

From this experiment, the ratios between the transferability score and the generalization gap for our transferability estimators are: 1.6 (LinMSE0), 2.0 (LinMSE1), 2.3 (LabMSE0), and 2.3 (LabMSE1). These results show that the transferability scores dominate the generalization gap in practice. More importantly, there is *no significant correlation* between the generalization gap and the actual transferred MSE. These findings indicate that the complexity term in our bounds may have little effects for transferability estimation, as opposed to the transferability score term that has a strong effect.

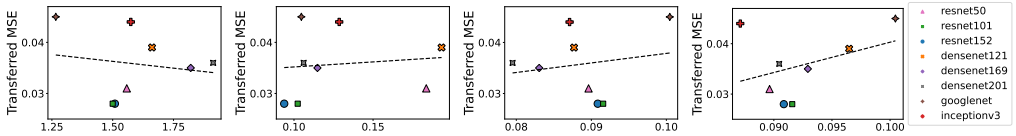
Table 3: Correlation coefficients for different values of λ on the CUB-200-2011 dataset. Bold numbers indicate the best results in each column. Results of the baselines are given in the last 4 rows for comparison.

λ	Head re-training		Half fine-tuning		Full fine-tuning	
	LabMSE	LinMSE	LabMSE	LinMSE	LabMSE	LinMSE
0.0	0.9156	0.9205	0.5359	**0.6275**	**0.0558**	**0.0968**
0.5	0.9156	0.9316	0.5359	0.6069	**0.0558**	0.0921
1.0	0.9156	0.9317	0.5359	0.6071	**0.0558**	0.0929
2.0	0.9156	0.9321	0.5359	0.6079	**0.0558**	0.0941
5.0	0.9156	0.9331	0.5359	0.6092	**0.0558**	0.0956
7.0	0.9156	0.9336	0.5359	0.6095	0.0557	0.0959
10	0.9157	0.9342	0.5359	0.6096	0.0556	0.0961
20	0.9159	0.9358	0.5360	0.6092	0.0554	0.0959
50	0.9172	0.9380	0.5363	0.6072	0.0541	0.0948
75	0.9185	0.9391	0.5364	0.6060	0.0528	0.0943
100	**0.9198**	**0.9398**	**0.5365**	0.6051	0.0516	0.0938
LabLogME	0.547	-	0.400	-	0.120	-
LogME	-	0.889	-	0.560	-	0.099
LabTransRate	0.008	-	0.006	-	0.001	-
TransRate	-	0.029	-	0.006	-	0.100

B.6 Beyond Regression

Although our paper mainly focuses on regression tasks, the main idea of using the penalized MSE of a Ridge regression model for transferability estimation goes beyond regression. In principle, this idea can be applied for transferring between classification tasks (in this case, we should train a linear classifier and use its penalized accuracy as the transferability estimator) or between a classification and a regression task. In this section, we describe an experiment where we transfer from a classification task to a regression task. Particularly, we use 8 source models pre-trained on ImageNet [9] (see the legend of Figure 3) and transfer to a target regression task on the *dSprite* dataset [21] using full fine-tuning. The settings for this experiment are similar to that of [35] where the target task is a regression task with 4-dimensional labels: x and y positions, scale, and orientation. We compute the transferability scores from the extracted features and the labels of the target training set.

From the results in Figure 3, the prediction trends for LogME, LinMSE0, and LinMSE1 are correct, while that of TransRate is not correct. To compare these methods, we fit a linear regression model to the points in each plot and compute its RMSE to these points. The RMSEs for each plot are: 6.12×10^{-3} (LogME), 6.16×10^{-3} (TransRate), 6.10×10^{-3} (LinMSE0), and 5.73×10^{-3} (LinMSE1). These results show that our methods, LinMSE0 and LinMSE1, are better than the LogME and TransRate baselines in this setting.



(a) LogME (b) TransRate (c) LinMSE0 (d) LinMSE1

Figure 3: Actual transferred MSEs vs. transferability scores when transferring from pre-trained classification models to a target regression task. A linear regression model (dashed line) is fitted to the points in each plot.

B.7 Additional Results

Table 4: Correlation coefficients for all source tasks on CUB-200-2011. Each row corresponds to transferring from one source task to all other tasks in the dataset.

Transfer setting	Source task	Label-based method				Feature-based method			
		LabLogME	LabTransRate	LabMSE0	LabMSE1	LogME	TransRate	LinMSE0	LinMSE1
Head re-training	Back	0.743	0.116	**0.956***	**0.956***	0.920	0.273	0.931*	0.931*
	Beak	0.863	0.229	**0.922***	**0.922***	0.878	0.158	0.906	0.907*
	Belly	0.893	0.097	0.970*	0.970*	0.933	0.188	0.932	**0.978***
	Breast	0.914	0.119	**0.935***	**0.935***	0.903	0.279	0.922*	0.922*
	Crown	0.917	0.040	**0.962***	**0.962***	0.913	0.251	0.945	0.946*
	Forehead	0.887	0.076	**0.941***	0.940	0.885	0.221	0.924	0.925*
	Left eye	0.034	0.076	0.913*	0.913*	0.924	0.289	0.945	**0.946***
	Left leg	0.260	0.221	0.935*	0.935*	0.935	0.223	**0.953***	**0.953***
	Left wing	0.260	0.169	0.964*	0.964*	0.980	0.173	**0.994***	**0.994***
	Nape	0.888	0.084	0.922*	0.922*	0.900	0.299	**0.929***	**0.929***
	Right eye	0.625	0.242	0.904*	0.904*	0.921	0.243	**0.948***	**0.948***
	Right leg	0.507	0.047	**0.958***	**0.958***	0.942	0.217	0.954	0.956*
	Right wing	0.521	0.166	0.907*	0.907*	0.934	0.269	0.946	**0.947***
	Tail	0.591	0.392	**0.900***	**0.900***	0.872	0.544	0.880	0.881*
	Throat	0.895	0.123	**0.938***	**0.938***	0.890	0.291	0.924	0.926*
Half fine-tuning	Back	0.714	0.076	0.791*	0.791*	0.835	0.168	**0.911***	**0.911***
	Beak	0.663	0.159	0.831*	0.831*	0.765	0.076	**0.883***	**0.883***
	Belly	0.529	0.233	0.655*	0.655*	0.757	0.309	**0.849***	0.709
	Breast	0.730	0.100	0.802*	0.802*	0.762	0.152	**0.867***	**0.867***
	Crown	0.644	0.068	0.752*	0.752*	0.714	0.165	**0.832***	**0.832***
	Forehead	0.653	0.032	0.804*	0.804*	0.727	0.120	**0.859***	0.858
	Left eye	0.419	0.046	**0.913***	**0.913***	0.812	0.227	0.892*	0.891
	Left leg	0.120	0.094	0.721*	0.721*	0.845	0.150	**0.893***	**0.893***
	Left wing	0.352	0.149	**0.949***	**0.949***	0.859	0.189	0.919*	0.919*
	Nape	0.660	0.055	0.705*	0.705*	0.751	0.181	**0.863***	**0.863***
	Right eye	0.561	0.221	**0.911***	**0.911***	0.786	0.180	0.871*	0.871*
	Right leg	0.267	0.125	0.690*	0.690*	0.810	0.069	**0.861***	0.858
	Right wing	0.407	0.133	0.495*	0.495*	0.515	0.338	0.521	**0.522***
	Tail	0.800	0.117	**0.929***	**0.929***	0.847	0.285	0.924*	0.924*
	Throat	0.767	0.012	0.869*	0.869*	0.811	0.253	**0.900***	0.898
Full fine-tuning	Back	0.710	0.085	0.785*	0.785*	0.829	0.178	**0.906***	**0.906***
	Beak	0.659	0.161	0.826*	0.826*	0.758	0.073	**0.877***	**0.877***
	Belly	0.645	0.273	0.782*	0.782*	0.861	0.365	**0.926***	0.826
	Breast	0.740	0.104	0.811*	0.811*	0.768	0.152	**0.871***	**0.871***
	Crown	0.647	0.073	0.756*	0.756*	0.717	0.157	**0.834***	0.833
	Forehead	0.648	0.037	0.799*	0.799*	0.723	0.111	**0.855***	0.854
	Left eye	0.224	**0.456***	0.297	0.297	0.333*	0.246	0.282	0.284
	Left leg	0.057	0.067	0.659*	0.659*	0.796	0.146	**0.850***	**0.850***
	Left wing	0.342	0.159	**0.954***	**0.954***	0.860	0.195	0.920*	0.920*
	Nape	0.667	0.041	0.713*	0.713*	0.752	0.177	**0.864***	**0.864***
	Right eye	0.549	0.213	**0.915***	**0.915***	0.794	0.199	0.877*	0.877*
	Right leg	0.237	0.377	0.673*	0.673*	0.755	0.431	**0.766***	0.763
	Right wing	**0.254***	0.046	0.237	0.237	0.225	0.093	0.227*	0.226
	Tail	0.803	0.122	**0.930***	**0.930***	0.846	0.288	0.923*	0.923*
	Throat	0.665	0.027	0.801*	0.801*	0.744	0.256	**0.850***	0.848

Table 5: Correlation coefficients for all source tasks on OpenMonkey. Each row corresponds to transferring from one source task to all other tasks in the dataset.

Transfer setting	Source task	Label-based method				Feature-based method			
		LabLogME	LabTransRate	LabMSE0	LabMSE1	LogME	TransRate	LinMSE0	LinMSE1
Head re-training	Right eye	0.894	0.859	0.986*	0.986*	0.992	0.846	0.979	**0.994***
	Left eye	0.895	0.854	0.987*	0.987*	0.915	0.858	0.982	**0.995***
	Nose	0.908	0.849	0.988*	0.988*	0.856	0.837	0.979	**0.996***
	Head	0.941	0.881	0.992*	0.992*	0.872	0.884	0.984	**0.994***
	Neck	0.972	0.862	**0.998***	**0.998***	0.960	0.839	0.984	0.997*
	Right shoulder	0.977	0.837	0.994*	0.994*	0.955	0.811	0.983	**0.995***
	Right elbow	0.963	0.529	0.994*	0.994*	0.890	0.564	0.974	**0.996***
	Right wrist	0.970	0.753	0.993*	0.993*	0.614	0.446	0.969	**0.995***
	Left shoulder	0.972	0.800	0.997*	0.997*	0.772	0.808	0.989	**0.999***
	Left elbow	0.960	0.546	0.994*	0.994*	0.256	0.572	0.972	**0.998***
	Left wrist	0.975	0.597	0.993*	0.993*	0.689	0.544	0.966	**0.996***
	Hip	0.922	0.540	0.988*	0.988*	0.881	0.557	0.817	**0.992***
	Right knee	0.925	0.080	0.975*	0.975*	0.919	0.331	0.949	**0.984***
	Right ankle	0.931	0.411	0.989*	0.989*	0.538	0.371	0.935	**0.993***
	Left knee	0.923	0.160	0.978*	0.978*	0.936	0.209	0.939	**0.990***
	Left ankle	0.916	0.416	0.986*	0.986*	0.8737	0.329	0.928	**0.993***
	Tail	0.936	0.712	**0.993***	**0.993***	0.625	0.662	0.892	0.990*
Half fine-tuning	Right eye	0.795	0.734	0.906*	0.906*	0.772	0.709	**0.960***	0.923
	Left eye	0.797	0.731	0.905*	0.905*	0.789	0.719	**0.959***	0.918
	Nose	0.829	0.736	0.914*	0.914*	0.739	0.721	**0.966***	0.926
	Head	0.835	0.759	0.921*	0.921*	0.754	0.751	**0.962***	0.930
	Neck	0.902	0.793	0.929*	0.929*	0.816	0.765	**0.967***	0.934
	Right shoulder	0.887	0.725	0.924*	0.924*	0.853	0.758	**0.970***	0.932
	Right elbow	0.764	0.250	0.806*	0.806*	0.621	0.602	**0.924***	0.833
	Right wrist	0.806	0.501	0.823*	0.823*	0.209	0.643	**0.920***	0.840
	Left shoulder	0.893	0.718	0.927*	0.927*	0.731	0.774	**0.971***	0.937
	Left elbow	0.782	0.369	0.824*	0.824*	0.153	0.594	**0.943***	0.845
	Left wrist	0.822	0.523	0.828*	0.828*	0.401	0.663	**0.924***	0.839
	Hip	0.030	0.487*	0.233	0.233	0.159	0.359	**0.782***	0.346
	Right knee	0.481	0.429	0.598*	0.598*	0.421	0.067	**0.820***	0.631
	Right ankle	0.357	0.275	0.534*	0.534*	0.019	0.226	**0.797***	0.577
	Left knee	0.467	0.355	0.601*	0.601*	0.491	0.215	**0.846***	0.658
	Left ankle	0.331	0.242	0.530*	0.530*	**0.816***	0.572	0.189	0.303
	Tail	0.231	0.196	0.434*	0.434*	0.335	0.212	**0.638***	0.342
Full fine-tuning	Right eye	0.796	0.711	0.905*	0.905*	0.754	0.693	**0.957***	0.922
	Left eye	0.790	0.733	0.904*	0.904*	0.780	0.713	**0.956***	0.918
	Nose	0.810	0.731	0.911*	0.911*	0.722	0.708	**0.958***	0.928
	Head	0.801	0.737	0.899*	0.899*	0.712	0.718	**0.945***	0.912
	Neck	0.893	0.781	0.930*	0.930*	0.818	0.743	**0.960***	0.934
	Right shoulder	0.896	0.721	**0.992***	0.936	0.865	0.750	0.973*	0.944
	Right elbow	0.689	0.168	0.736*	0.736*	0.549	0.562	**0.880***	0.769
	Right wrist	0.795	0.505	0.805*	0.805*	0.194	0.644	**0.900***	0.822
	Left shoulder	0.872	0.690	0.901*	0.901*	0.212	0.537	**0.910***	0.798
	Left elbow	0.726	0.282	0.774*	0.774*	0.395	0.672	**0.894***	0.798
	Left wrist	0.786	0.487	**0.983***	0.786	0.125	0.382	0.736*	0.284
	Hip	0.016	0.518*	0.173	0.173	0.336	0.141	**0.750***	0.550
	Right knee	0.391	0.518*	0.516	0.516	0.025	0.340	**0.715***	0.481
	Right ankle	0.246	0.396	0.436*	0.436*	**0.981***	0.722	0.977	0.977
	Left knee	0.381	0.338	0.521*	0.521*	0.405	0.303	**0.779***	0.582
	Left ankle	0.244	0.297	0.444*	0.444*	0.100	0.357	**0.744***	0.486
	Tail	0.105	0.299	0.309*	0.309*	0.335	0.212	**0.638***	0.341