ETran: Energy-Based Transferability Estimation **Supplementary**

Mohsen Gholami[1,2], Mohammad Akbari[1], Xinglu Wang[1], Behnam Kamranian [1], Yong Zhang[1]
[1]Huawei Technologies Canada Co., Ltd., [2]University of British Columbia
{mohsen.gholami,mohammad.akbari,xinglu.wang,behnam.kamranian,yong.zhang3}@huawei.com

This Appendix provides further details about *ETran* and demonstrates further experimental results. In Section 1 and 2, we further evaluate *ETran* on *target selection* scenario and also language models. In Sections 3-6, we provide a theoretical and experimental analysis of classification, regression, and energy scores of *ETran*. In 7, the fine-tuning procedure and the resulting ground-truth validation scores on all the benchmarks are provided, and in 8 limitations and future work are discussed.

1. Target Selection Scenario

In the main body of the paper, following most of the previous works, we considered that M source models and a target dataset are given and the transferability metrics tried to rank the source models. Some of the previous works including GBC [9] and LEEP [8] define a new scenario, where a single source model and multiple target datasets are given and the transferability metrics are used to rank the target datasets. We call this scenario as *target selection*. Figure 1 shows an overview of the *target selection* scenario.

We use the classification benchmark [10] explained in the main body to evaluate *ETran*'s performance compared with the previous works on the target selection. Table 1 provides Kendall tau (τ_w) of the 11 source classification models. *ETran* obtains an average τ_w of 0.545 over 11 models, while SFDA, PACTran, LogME, and LEEP obtain an average τ_w of 0.376, 0.295, -0.020, and 0.224, respectively. *ETran* outperforms SFDA by a relative improvement of 45% in-terms of τ_w. Table 1 also shows that adding energy score (i.e., S_{en}) to the previous works improves their results on most of the cases (e.g., τ_w of 0.433 vs. 0.376 for SFDA).

2. Evaluation on Language Models

Experiments on other modalities are our future work. In this section we show the preliminary results of our experiments on the language models, where we use the RTE task from GLUE benchmark [11] and 10 popular pre-trained language models from HuggingFace (e.g., BERT, RoBERTa, BART, ALBERT, and DeBERTA). ETran, SFDA, and LogME obtain a τ_w of **0.421**, 0.391, and 0.138, respectively, which shows the superiority of ETran compared to others.

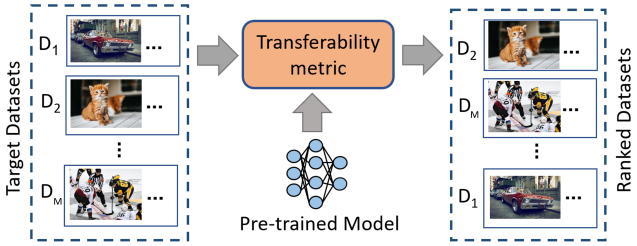

Figure 1: The overall framework of transferability estimation in the target selection scenario. Given M target datasets and a source pre-trained model, the goal is to rank target datasets according to the actual performance of the source model after fine-tuning it on the target dataset.

3. Analysis of *ETran*'s Scores

In this section, we theoretically and experimentally analyze the proposed LDA-based classification and SVD-based regression scores compared with their peers including SFDA-based classification [10] and LogME-based regression [12] scores, respectively.

Before the analysis, we will first recap the intuition for transferability scores. The transferability score measures the compatibility between the extracted features and the ground-truth labels.

More formally, each sample in the target dataset comes from an underlying distribution $\mathcal{D}$. To avoid costly fine-tuning, it is assumed that the source model's backbone is frozen. The extracted feature f and its corresponding labels y come from a feature distribution $\mathcal{F}$, denoting as $(f, y) \sim \mathcal{F}$.

For classification, y is a scalar for the target (ground-truth) class. For regression (specifically for the task of object detection), y means the position and scale for bboxes.

Table 1: The performance of *ETran* compared with previous works for the target selection scenario on the classification benchmark (in terms of Kendall tau τ_w).

Method	Res34	Res50	Res101	Res152	Dens169	Dens121	Dens201	MNas	Google	Inception	Mobilenet	**Average**
LEEP [8]	0.253	0.314	0.330	0.314	0.143	0.157	0.127	0.242	0.159	0.263	0.157	0.224
LogME [12]	-0.387	-0.081	-0.118	-0.101	0.241	-0.226	0.207	0.203	-0.191	0.03	0.203	-0.020
PACTran [4]	0.373	0.467	0.402	0.397	0.260	0.295	0.047	0.243	0.214	0.394	0.154	0.295
SFDA [10]	0.501	0.501	0.484	0.501	0.301	0.314	0.284	0.312	0.211	0.462	0.269	0.376
LEEP+S_{en}	0.333	0.244	0.349	0.410	0.260	0.218	0.296	0.143	0.087	0.191	0.038	0.233
LogME+S_{en}	-0.387	0.053	0.113	0.005	0.306	-0.211	0.302	0.161	-0.259	0.150	0.241	0.043
PACTran+S_{en}	0.350	0.496	0.445	0.392	0.241	0.295	0.106	0.201	0.229	0.239	0.209	0.291
SFDA+S_{en}	0.604	0.624	0.678	0.612	0.276	0.387	0.256	0.271	0.192	0.496	0.371	0.433
ETran (S_{cls}+S_{en})	0.436	0.542	0.525	0.574	0.661	0.525	0.521	0.692	0.449	0.394	0.681	**0.545**

Table 2: Comparing LDA and SFDA on the classification benchmark based on τ_w. The self-challenging mechanism of SFDA diminishes the performance on many datasets.

Method	CIFAR10	VOC	Caltech-101	AirCraft	CIFAR100	Food-101	Pets	Flowers	Cars	DTD	Sun	**Average**
SFDA [10]	**0.849**	0.518	**0.555**	-0.215	0.793	0.427	0.340	**0.590**	**0.312**	**0.633**	**0.722**	**0.502**
LDA (S_{cls})	0.842	**0.521**	0.354	**-0.146**	**0.869**	**0.754**	**0.713**	0.357	-0.006	0.303	0.616	0.470
SFDA + S_{en}	**0.890**	0.606	**0.558**	-0.161	0.856	0.370	0.422	0.406	**0.328**	**0.639**	**0.744**	0.514
LDA (S_{cls}) + S_{en}	0.887	**0.667**	0.440	**-0.091**	**0.900**	**0.829**	**0.713**	**0.580**	0.246	0.303	0.708	**0.562**

Table 3: Comparing LDA and SFDA on object detection benchmarks.

	VOC-FT		COCO		HF	
	Pr(top3)	τ_w	Pr(top3)	τ_w	Pr(top3)	τ_w
SFDA [10]	0.250	0.108	**0.533**	0.104	**1.000**	0.312
LDA (ours)	**0.286**	**0.141**	**0.533**	**0.131**	0.800	**0.376**

Since it is natural that separate weights are leveraged for predicting each component of position and scale, they can be treated independently for transferability score.Thus, for simplicity, y here is a scalar for one component of position and scale.

From distribution $\mathcal{F}$, we have K samples (i.e., bboxes for object detection) and their corresponding labels, i.e., $(f, y) \sim \mathcal{F}^K$. Here, $f \in \mathbb{R}^{K\times\hat{h}}$ is the extracted feature matrix. $y \in \mathbb{R}^K$ is the labels of samples. Given f and y, the transferability score measures the compatibility between the feature and label. We say they are compatible if there exists a mapping from feature space to label space and this mapping is accurate for the feature and label pair from $\mathcal{F}$. To conclude, the transferability score of a source model towards $\mathcal{D}$ is measured by the generalization performance of the mapping on $\mathcal{F}$.

There are two challenges: 1) After fine-tuning the source model, feature distribution $\mathcal{F}$ drift. The transferability score fails to compensate for this. 2) The generalization performance is defined on distribution $\mathcal{F}$, however, only K samples from distribution are available. Thus, it matters whether the estimation of the transferability score is tight or vacuous. Motivated by these two challenges, LDA-based classification and SVD-based regression scores are proposed. We will give a detailed analysis in the following sections.

Note that for the simplicity and generality of our method on different benchmarks, the three transferability scores in Eq. **??** are normalized between [0,1] and equally summed. Based on our initial study, having different weights for the normalized three terms does not significantly affect the final results. It is a common practice to use fixed hyperparameters for generality (i.e., PACTran [4]).

4. Energy Score vs. Classification Score.

As studied in [1, 2, 3, 7], the softmax score for a classifier, Φ with C classes, is defined as:

$$\max_y p(y|x) = \max_y \frac{e^{\Phi^{(y)}(x)}}{\sum_c^C e^{\Phi^{(c)}(x)}} = \frac{e^{\Phi_{max}(x)}}{\sum_c^C e^{\Phi^{(c)}(x)}}. \quad (1)$$

If we take the logarithm of both sides we have:

$$\begin{aligned} \log \max_y p(y|x) &= \Phi_{\max}(x) - \log \sum_c^C e^{\Phi^{(c)}(x)} \\ &= \Phi_{\max}(x) + E(x). \end{aligned} \quad (2)$$

Therefore, the log of softmax confidence score is in fact energy score shifted with the maximum value of logits. Since $\Phi_{max}(x)$ tends to be higher and $E(x)$ (Eq. 5 in the paper) tends to be lower for in-distribution samples, the softmax confidence score is a biased scoring function that is

Table 4: Comparing LogME and SVD-based regression scores on object detection benchmarks.

	VOC-FT		COCO		HF	
	Pr(top3)	τ_w	Pr(top3)	τ_w	Pr(top3)	τ_w
LogME (S_{lmr}) [12]	0.357	0.356	0.400	0.113	0.800	0.400
SVD-reg (S_{reg})	**0.393**	**0.357**	0.400	**0.122**	0.800	**0.512**

no longer proportional to the probability density $p(x)$. Having $E(x)$ from Eq. 6 in the main body of the paper, we can write Eq. 2 as:

$$\log \max_y p(y|x) = -\log p(x) + \underbrace{\Phi_{max}(x) - \log Z}_{\text{not constant, larger for in-dist x}} . \tag{3}$$

Thus, unlike the energy score (as proved in section 3.3 of the paper), the softmax classification score is not well-aligned with $p(x)$ [2, 7], which makes it less reliable for out-of-distribution detection and transferability assessment.

5. LDA-Based Classification Score vs. SFDA

The feature distribution $\mathcal{F}$ shifts from the source to the target dataset after fine-tuning. The features f extracted by the pre-trained models are separable based on the source dataset's classes. However, after fine-tuning, the features are separable based on the target dataset's classes. To mitigate feature distribution shifts, we propose to use an LDA-based classification score. Linear discriminant analysis (LDA) projects the features into a space that is separable w.r.t the target classes. This coincides with the feature from fine-tuned model and thus mitigates the distribution shift.

Compared to our LDA-based score, SFDA [10] has a self-challenging mechanism, which has two drawbacks: 1) The practical computational cost of SFDA is almost twice LDA, which is because the self-challenging mechanism performs Linear Discriminate Analysis twice on all the samples. 2) The self-challenging mechanism also introduces noise on the features, which can negatively affect the deep connection between the discriminative and energy-based models, i.e., the linear alignment of the calculated negative free energy with the likelihood function (as discussed in Section 3.3 of the paper).

As summarized in Table 2, SFDA overally performs a bit better than the LDA-based score (i.e., S_{cls}) on the classification benchmark. However, when integrated with the proposed energy score (i.e., S_{en}), LDA archives a better performance. Table 3 also compares the performance of LDA vs. SFDA on the object detection benchmarks, which shows that LDA outperforms SFDA on three benchmarks in terms of τ_w.

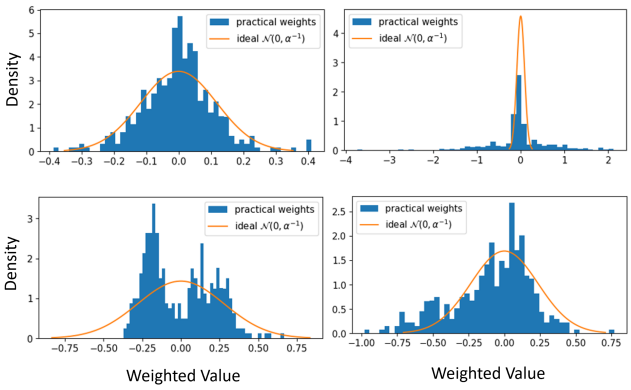

Figure 2: LogME's assumption analysis. **Blue**: the histogram of practical weights. **Orange**: the weights distribution with optimal α. Weight distribution comes from different pairs of source models and target datasets in the VOC-FT benchmark.

6. SVD-Based Regression Score vs. LogME

We will first recap the LogME score, analyze its problem, and then propose our solution. LogME assumes that the weights of a linear model that maps from feature space to label space $\hat{y} = \mathbf{w}^\top f$, have a normal distribution as follows: $\mathbf{w} \sim \mathcal{N}(0, \alpha^{-1} I)$. The prior of weights, α, is optimized on target features, f. Then, the log marginal likelihood (i.e., evidence) of observing f given ground-truth labels y is used to measure the generalization performance of the pre-trained source models [12].

If the assumption $\mathbf{w} \sim \mathcal{N}(0, \alpha^{-1} I)$ matches the actual feature data, the LogME score measures the generalization performance tightly. But if not, LogME will deviate from actual performance. In practice, there are many cases where the assumption of LogME does not hold.

In order to evaluate the assumption of LogME, we calculate the optimal α using LogME and compare $\mathcal{N}(0, \alpha^{-1} I)$ with the practical weight distribution of the last-layer weights of the fine-tuned models in the VOC-FT benchmark. The comparison is illustrated in Figure 2. The first row in Figure 2 shows the successful cases that the practical weights follow Gaussian, where LogME finds the optimal α. However, as shown in the second row, there are cases that the practical weights cannot be described by Gaussian distribution. To this end, if the model's hypothesis space is limited, the derived transferability metric will be a vacuous bound of the model's actual performance.

In our SVD-based regression method, we relax the assumption, resort to SVD-based linear regression, and derive the transferability metric. We verified that this strategy is effective in practice. This simple strategy first finds the optimal mapping by solving best $\mathbf{w}^*$ for $\arg\min_{\mathbf{w}} \|y - f\mathbf{w}\|^2$ and then measures the performance by negative remaining

Table 5: The fine-tuning accuracy (map50) of pre-trained models on VOC-FT benchmark. The best and the second-best pre-trained source models for a given target dataset are shown in bold and underlined, respectively.

Target Datasets \ Source Models	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1	0.28	0.33	0.31	0.36	0.34	0.33	0.35	0.33	0.29	0.33	0.35	0.32	0.35	0.33	0.32	0.32	0.33	**0.36**	0.32
2	0.30	0.33	0.33	0.36	0.34	0.33	0.34	0.33	0.29	0.31	**0.38**	0.33	0.34	0.33	0.34	0.34	0.34	0.37	0.32
3	0.38	0.46	0.43	**0.52**	0.46	0.47	0.49	0.47	0.44	0.42	0.49	0.46	0.47	0.47	0.47	0.43	0.48	0.50	0.46
4	0.33	0.36	0.39	0.41	0.37	0.38	0.40	0.40	0.35	0.33	0.41	0.36	0.40	0.40	0.38	0.36	0.37	**0.42**	0.37
5	0.47	0.50	0.48	0.53	0.51	0.50	0.50	0.48	0.49	0.47	0.51	0.48	0.51	0.49	0.49	0.49	0.49	**0.54**	0.51
6	0.48	0.49	0.49	**0.55**	0.48	0.51	0.52	0.51	0.48	0.46	0.51	0.49	0.51	0.51	0.51	0.46	0.51	0.52	0.49
7	0.45	0.45	0.46	0.51	0.49	0.48	0.49	0.46	0.44	0.44	**0.53**	0.45	0.48	0.50	0.51	0.45	0.51	0.51	0.47
8	0.24	0.31	0.29	0.33	0.31	0.32	0.32	0.29	0.24	0.28	**0.33**	0.29	0.33	0.31	0.31	0.32	0.33	0.33	0.33
9	0.41	0.47	0.46	0.48	0.48	0.50	0.48	0.45	0.40	0.42	0.47	0.42	0.48	0.45	0.49	0.46	0.47	**0.50**	0.47
10	0.27	0.32	0.35	0.36	0.34	0.34	**0.37**	**0.37**	0.28	0.32	0.35	0.32	0.35	0.32	0.35	0.32	0.34	0.36	0.33
11	0.46	0.49	0.50	0.51	0.49	0.48	0.47	0.51	0.47	0.49	0.51	0.46	0.49	0.49	0.50	0.51	0.49	**0.53**	0.49
12	0.44	0.51	0.49	0.49	0.50	0.51	0.50	0.50	0.45	0.48	0.50	0.45	0.50	0.49	0.50	0.48	0.49	**0.52**	0.51
13	0.38	0.44	0.47	**0.49**	0.44	0.44	0.44	0.47	0.38	0.41	0.47	0.44	0.46	0.43	0.46	0.42	0.47	0.46	0.44
14	0.42	0.46	0.43	0.47	0.47	0.47	0.46	0.47	0.41	0.40	**0.49**	0.43	0.47	0.47	**0.49**	0.44	0.47	0.47	0.47
15	0.30	0.34	0.34	0.35	0.35	0.33	0.34	0.33	0.29	0.32	0.37	0.31	0.33	0.33	0.35	0.33	0.35	**0.37**	0.34
16	0.50	0.51	0.51	0.52	0.52	0.50	0.49	0.50	0.49	0.51	0.51	0.49	0.51	0.50	0.51	0.51	0.51	**0.53**	0.52
17	0.45	0.51	0.48	0.51	0.49	0.49	0.49	0.49	0.45	0.48	**0.51**	0.46	0.51	0.50	0.50	0.48	0.49	0.50	0.50
18	0.37	0.41	0.41	**0.44**	0.42	0.43	0.39	0.41	0.36	0.37	0.43	0.41	0.42	0.43	0.43	0.40	0.43	0.43	0.41
19	0.50	0.52	0.51	0.54	0.53	0.57	0.53	0.54	0.50	0.50	0.56	0.51	0.53	0.53	**0.57**	0.49	0.50	0.56	0.56
20	0.59	0.61	0.60	0.63	0.62	0.63	0.62	0.61	0.59	0.61	0.60	0.57	0.62	0.59	0.61	0.60	0.61	**0.64**	0.61
21	0.49	0.55	0.49	0.51	0.50	0.54	0.53	0.52	0.47	0.47	**0.56**	0.49	0.54	0.53	0.48	0.47	0.49	0.55	0.50
22	0.55	0.61	0.56	0.62	0.60	0.63	0.61	0.61	0.54	0.56	0.63	0.60	0.63	0.60	**0.64**	0.59	0.59	0.61	0.61
23	0.51	0.53	0.52	**0.55**	0.53	0.53	0.53	0.52	0.51	0.51	0.54	0.49	0.52	0.52	0.53	0.54	0.52	0.54	0.52
24	0.55	0.54	0.55	0.60	0.55	0.58	0.58	0.56	0.55	0.55	0.59	0.53	0.59	0.57	**0.60**	0.53	0.54	0.58	0.60
25	0.47	0.44	0.47	0.54	0.47	0.50	0.51	0.51	0.42	0.46	0.53	0.46	0.52	0.48	0.52	0.46	0.51	**0.55**	0.51
26	0.65	0.68	0.66	**0.69**	0.67	0.68	0.68	0.67	0.66	0.66	0.68	0.67	**0.69**	0.67	0.67	0.68	0.67	0.68	0.69
27	0.57	0.14	0.56	**0.60**	0.58	0.58	0.58	0.58	0.55	0.57	0.59	0.54	0.57	0.58	0.60	0.58	0.59	0.58	0.57
28	0.58	0.64	0.62	**0.69**	0.63	0.65	0.66	0.65	0.60	0.61	0.68	0.60	0.66	0.63	0.66	0.62	0.64	0.68	0.63

Table 6: The fine-tuning accuracy (map50) of pre-trained models on COCO benchmark. The best and the second-best pre-trained source models for a given target dataset are shown in bold and underlined, respectively.

Source Models \ Target Datasets	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0.344	0.445	0.281	0.177	0.593	0.524	**0.608**	0.425	**0.342**	0.263	**0.430**	0.141	0.305	0.440	0.319
2	**0.373**	0.489	**0.304**	0.194	0.632	**0.556**	0.588	**0.432**	0.334	0.277	0.415	**0.183**	0.343	0.458	0.344
3	0.340	0.481	0.311	0.171	0.573	0.534	0.559	0.412	0.334	0.249	0.403	0.171	0.306	0.452	0.354
4	0.353	0.482	0.278	0.172	0.614	0.523	0.533	0.406	0.311	0.252	0.412	0.161	**0.346**	0.460	0.385
5	0.338	0.462	0.283	0.186	0.600	0.543	0.580	0.385	0.310	0.237	0.392	0.159	0.328	0.450	0.352
6	0.355	**0.512**	**0.304**	0.189	**0.638**	0.547	0.526	0.419	0.310	0.259	0.413	0.181	0.322	0.464	0.358
7	0.353	0.479	0.324	**0.202**	0.636	0.515	0.585	0.407	0.330	0.255	0.412	0.173	0.327	0.473	**0.389**
8	0.355	0.493	0.300	0.181	0.635	0.535	0.555	0.424	0.319	**0.303**	0.422	0.158	0.330	**0.484**	0.384
9	0.365	0.471	0.290	0.186	0.616	0.524	0.548	0.402	0.333	0.248	0.400	0.157	0.400	0.449	0.364

error $-\|y - f\mathbf{w}^*\|$. To prevent overfitting, a more advanced way is to split f into train and test sets by a 7:3 ratio and evaluate performance on the test set.

Considering f is near rank-deficient and may be ill-conditioned (the bottom-level singular value is close to zero), we apply truncated SVD to obtain $\mathbf{w}^*$. With SVD decomposition $f = \mathbf{U}\text{diag}(\mathbf{s})\mathbf{V}^\top$, $\mathbf{w}^* = \mathbf{V}\text{diag}(\hat{\mathbf{s}})^{-1}\mathbf{U}^\top$, where $\text{diag}(\hat{\mathbf{s}})$ is the truncated singular values whose top 80% is preserved. With SVD, we approximately solve the linear regression in a way that it is less sensitive to errors and more effective for ill-conditioned matrices. Moreover, the complexity of our proposed regression score is $\mathcal{O}(n\hat{h}^2)$.

It is more efficient compared to LogME's $\mathcal{O}\left(n\hat{h}^2 + \hat{h}^3 + t(\hat{h}^2 + n\hat{h})\right)$, where t is the iteration for LogME to converge and n is the number of samples in the dataset. Since the tailing singular value is truncated, the practical run-time of our proposed score will be further reduced.

Table 4 shows the performance of LogME (i.e., S_{lmr}) vs. our SVD-based regression (i.e., S_{reg}) on the object detection benchmarks. As seen by the results, the proposed SVD-based regression outperforms LogME on all three bench-

Table 7: The fine-tuning accuracy (map50) of pre-trained models on HuggingFace benchmark. The best and the second-best pre-trained source models for a given target dataset are shown in bold and underlined, respectively.

	NFL	Blood	CSGO	Forklift	Valorant
Yolov5s [5]	0.261	0.902	<u>0.924</u>	0.838	<u>0.982</u>
Yolov5m [5]	**0.314**	0.905	**0.932**	**0.852**	**0.990**
Yolov5n [5]	0.217	<u>0.923</u>	0.908	0.789	0.959
Yolov8s [6]	0.279	0.917	0.886	<u>0.851</u>	0.971
Yolov8m [6]	<u>0.287</u>	**0.927**	0.892	0.846	0.965
Yolov8n [6]	0.209	0.893	0.844	0.838	0.937

marks in terms of τ_w and $Pr(top3)$.

7. Fine-Tuning and Ground-Truth Results

The ground-truth ranking of the pre-trained source models is obtained by fine-tuning each of the source models on all the target datasets. In this section, we provide the details of the fine-tuning procedure for object detection and classification benchmarks.

VOC and COCO. For the VOC and COCO benchmarks, we first train Yolov5s [5] on each of the source models for 300 epochs. The pre-trained source models are then fine-tuned on the train set of the target datasets for 60 epochs. Tables 5 and 6 show the map50 of the fine-tuned models on the validation set of the target datasets. The best and second-best pre-trained source models for a given target dataset are shown in bold and underlined, respectively.

HuggingFace. We use 6 object detection models including: YOLOv5s, YOLO5m, YOLOv5n [5], YOLOv8s, YOLOv8m, and YOLOv8n [6]. All the models were pre-trained on the COCO dataset using the default setting [5, 6]. Table 7 provides the map50 of the source models after fine-tuning on the target datasets.

Classification. The source models were pre-trained on ImageNet and were downloaded from the Pytorch repository. The accuracy of the models after fine-tuning on the target datasets was obtained from SFDA [10]. SFDA [10] provides details of fine-tuning on each of the target datasets. Table 6 in the appendix of the SFDA paper shows the accuracy of the pre-trained models after fine-tuning on each of the target datasets.

8. Limitations

We have briefly discussed the limitations in Section 4.1 of the main body. Fig. 4 of the main body shows two failure cases where the energy score does not always correlate positively with the accuracy of the target dataset. In this section, we further discuss the limitations of our work that need to be addressed in future work including: **1)** In all 4 benchmarks, source models differ either in their architectures or source datasets. It will be comprehensive to further validate considering both.

2) The stability of our method *w.r.t* the small perturbation on the target dataset and source pre-trained models needs to be studied further. **3)** In all scenarios, both source and target tasks were identical, e.g., both aimed for classification tasks. The stability of the method, when source and target tasks differ, should be investigated. **4)** Experiments on other modalities such as language models.

References

[1] Mohammad Akbari, Amin Banitalebi-Dehkordi, and Yong Zhang. EBJR: Energy-based joint reasoning for adaptive inference. *BMVC 2021*, 2021. 2

[2] Mohammad Akbari, Amin Banitalebi-Dehkordi, and Yong Zhang. E-lang: Energy-based joint inferencing of super and swift language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5229–5244, 2022. 2, 3

[3] Mohammad Akbari, Amin BANITALEBI DEHKORDI, Tianxi Xu, and Yong Zhang. System and method for unsupervised multi-model joint reasoning, 2023. US Patent App. 18/074,915. 2

[4] Nan Ding, Xi Chen, Tomer Levinboim, Soravit Changpinyo, and Radu Soricut. Pactran: Pac-bayesian metrics for estimating the transferability of pretrained models to classification tasks. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIV*, pages 252–268. Springer, 2022. 2

[5] Glenn Jocher. YOLOv5 by Ultralytics, 5 2020. 5

[6] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, 1 2023. 5

[7] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in neural information processing systems*, 33:21464–21475, 2020. 2, 3

[8] Cuong Nguyen, Tal Hassner, Matthias Seeger, and Cedric Archambeau. Leep: A new measure to evaluate transferability of learned representations. In *International Conference on Machine Learning*, pages 7294–7305. PMLR, 2020. 1, 2

[9] Michal Pándy, Andrea Agostinelli, Jasper Uijlings, Vittorio Ferrari, and Thomas Mensink. Transferability estimation using bhattacharyya class separability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9172–9182, 2022. 1

[10] Wenqi Shao, Xun Zhao, Yixiao Ge, Zhaoyang Zhang, Lei Yang, Xiaogang Wang, Ying Shan, and Ping Luo. Not all models are equal: Predicting model transferability in a self-challenging fisher space. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIV*, pages 286–302. Springer, 2022. 1, 2, 3, 5

[11] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*, 2018. 1

[12] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. Logme: Practical assessment of pre-trained models for transfer learning. In *International Conference on Machine Learning*, pages 12133–12143. PMLR, 2021. 1, 2, 3