Can We Characterize Tasks Without Labels or Features?

Bram Wallace*  
Cornell University  
bw462@cornell.edu

Ziyang Wu*  
Cornell University  
zw287@cornell.edu

Bharath Hariharan  
Cornell University  
bharathh@cs.cornell.edu

Abstract

The problem of expert model selection deals with choosing the appropriate pretrained network (“expert”) to transfer to a target task. Methods, however, generally depend on two separate assumptions: the presence of labeled images and access to powerful “probe” networks that yield useful features. In this work, we demonstrate the current reliance on both of these aspects and develop algorithms to operate when either of these assumptions fail. In the unlabeled case, we show that pseudolabels from the probe network provide discriminative enough gradients to perform nearly-equal task selection even when the probe network is trained on imagery unrelated to the tasks. To compute the embedding with no probe network at all, we introduce the Task Tangent Kernel (TTK) which uses a kernelized distance across multiple random networks to achieve performance over double that of other methods with randomly initialized models. Code is available at https://github.com/BramSW/task_characterization_cvpr_2021/.

1. Introduction

Transfer learning is key to the success and popularity of computer vision. The features from a convolutional neural network (CNN) trained on one task can be incredibly useful across a broad variety of tasks [36, 32, 34, 24]. Even larger performance gains can be obtained by selecting a pretrained model more specially suited to the task at hand. This behavior has been studied in past work [6], but only recently has attention turned to how to determine which specialized model is appropriate for a given task [2].

The key to such model selection is characterizing tasks and their relationships. What does one need to characterize a task? *A priori*, it seems that we need to characterize two things:

1. The input, characterizing which requires a dataset of images, and features to represent them, and

2. The output, characterizing which requires labels for the dataset.

Current approaches to characterizing tasks synthesize both sources of information. Task2Vec [2], for example, trains a linear head on top of a pretrained feature extractor and uses the Fisher information associated with the resulting model to generate a vectorized embedding. Decisions such as choosing the best pretraining task for a target task can then effectively be made using retrieval-like techniques with this embedding.

But how much of these accurate decisions come from characterizing the input domain alone, and how much comes from knowledge of the precise task? This question has important practical considerations. For example, suppose we want to choose a pretrained representation for analyzing x-ray images. We may not yet know what we want to recognize in x-ray images. In fact, we may want a pretrained representation suitable for any kind of x-ray image analysis, even those we haven’t conceived yet. In such cases we are interested in characterizing only the general problem domain, and do not have particular labels (yet) that we are interested in. This raises the question: *Can we characterize tasks without labels?*

As our first contribution, we answer this question in the affirmative. To address this problem, we introduce PseudoTask: a modification of the Task2Vec algorithm that replaces labels with pseudolabels output by an image classifier trained in a generic source domain. While these pseudolabels are definitely incorrect due to domain misalignment, they prove discriminative enough to be quite useful in characterizing tasks. Empirically, we find that PseudoTask embeddings are *as accurate as supervised Task2Vec embeddings*, indicating that one can characterize tasks effectively *even without labels*.

Key to this performance, as also to the performance of Task2Vec, is the inductive bias provided by the pre-trained feature representation. However, this inductive bias may prove harmful as the task domains move farther away from the domain where the feature extractor is pretrained [36], making it risky to rely so heavily on such feature representations. This raises a second question: *can we characterize*
tasks without features?

We answer this too in the affirmative. We design a method called Task Tangent Kernel (TTK) that measures task similarity using the gradients of randomly initialized networks. TTK does not use pretrained probe networks at all. Despite this lack of inductive bias, it provides useful selections as well, at less than half the error of PseudoTask and Task2Vec with random feature extractors.

In sum, this paper introduces two new techniques for characterizing tasks that lift some of the restrictive assumptions of prior work (Figure 2):

1. We introduce **PseudoTask**, a new way of characterizing tasks without labels, allowing one to characterize problem domains in general. We find PseudoTask performs almost the same as Task2Vec, indicating that labels are in fact not necessary.

2. To avoid the potentially mismatched inductive bias of pretrained feature extractors, we introduce **Task Tangent Kernels**, which characterizes tasks effectively even without such feature extractors.

2. Related Work

Previous work has studied why pretrained models transfer so well [43, 19], the tradeoffs between specialization and scale in pretraining [6] and how concurrent multitask learning can benefit performance [41]. Task2Vec [2], the work that this paper builds off, studies the problem of automated model selection, as does [15]. [31, 1, 22] recommend algorithms for various problems using “Active Testing”, intelligently adapting exploration based on results. Such types of approaches are inherently more limited computationally than embedding-based methods. [37] predicts per-image performance for single models, while [8, 26] characterize performance per-dataset. [25, 42] deal with the problem of model recommendation for action recognition and object detection respectively, but require some degree of performance evaluation on the new task in order to provide a recommendation.

Transfer learning has been increasingly optimized, with recent advances detailed in [18]. The problem of expert selection has analogs to image retrieval, examples of which include [30, 4]. Other works measuring distances between domains include evaluating the biases between semantically similar datasets [33] and measuring temporal domain shifts in datasets [17].

Our PseudoTask framework draws heavily on pseudolabeling for semi-supervised learning, such as in [21]. Parts of our training setup are very similar to that of self-training, such as for few-shot transfer or semi-supervised learning [29, 39]. PseudoTask can be considered a form of self-supervised training such as [9, 27, 5, 11, 12]. The Task Tangent Kernel is inspired by Neural Tangent Kernel literature, originating with [14] and developed in [23, 3, 28].

3. Problem setup

A task \( T = (X_T, Y_T) \) consists of a domain of images \( X_T = \{x_i\}_{i=1}^n \) and corresponding labels \( Y_T = \{y_i\}_{i=1}^n \). An expert \( \Theta_T = (\phi_T, h_T) \) is a dedicated neural network, consisting of a feature extractor \( \phi_T \) and a head \( h_T \) that is trained on a task \( T \). Suppose that we have a bank of tasks \( T_1, \ldots, T_N \), and have already trained a corresponding bank of experts \( \Theta_{T_1}, \ldots, \Theta_{T_N} \). Then, when we encounter a new task \( T' \), instead of training a model from scratch, we may want to choose a pretrained expert \( \Theta_{T'} \), use its fixed feature extractor \( \phi_{T'} \) and train a linear head to solve the new task; see Figure 1. The goal of expert selection is to pick the best pretrained expert for the target task.

4. Background: Task2Vec

Our work builds on Task2Vec [2], a recent approach that tries to characterize tasks and embed them in a useful way. Task2Vec tries to characterize both the input domain, as well as the semantic information carried by the labels. The key intuition behind Task2Vec is that we can try to solve the task with a moderately effective but generic off-the-shelf feature extractor (e.g., trained on Imagenet) and then see which parameters of the feature extractor most impact the performance. Task2Vec posits that tasks which are sensitive to the same set of feature extractor parameters are likely to be “similar” to each other, especially in terms of what they demand out of pretrained features.

Concretely, Task2Vec trains a linear layer on top of the off-the-shelf feature extractor (called a “probe”) for the task in question. It then computes the Fisher Information Matrix (FIM), which is known to measure the sensitivity of the loss to the parameters of the model. Denoting by \( p_w(y|x) \) the output distribution of trained model \( p_w \) with weights \( w \), and by \( \tilde{p}(x) \) the data distribution, the FIM is defined as:

\[
F = \mathbb{E}_{x,y} \log p_w(y|x) \nabla_x \log p_w(y|x) \nabla_y \log p_w(y|x)^T
\]

(1)

Task2Vec estimates \( F \) using a variational approach that
amounts finding the optimal weights \( \hat{w} \) and precision matrix \( \Lambda \) that minimize the following objective:

\[
L(\hat{w}, \Lambda) = \mathbb{E}_{w \sim N(\hat{w}, \Lambda)}[L_{CE}(X_T, Y_T, p_w)] + \\
\beta KL(N(0, \Lambda)|N(0, \lambda^2 I))
\]

(2)

Here \( L_{CE} \) is the cross entropy loss over the dataset and \( \lambda \) is a hyperparameter. Achille et al. prove that the solution to this is \( \Lambda = F + \frac{\lambda^2}{2\beta} I \). Task2Vec finally takes the diagonal of \( F \) and averages together the values for different parameters of the same filter to produce the embedding.

When using this embedding, symmetric distances such as cosine distance, denoted \( d_{sym} \), do not yield satisfactory performance when retrieving experts. This is because there is an inherent asymmetry to the expert selection problem: a task with a large dataset and thousands of classes will yield a good expert for a similar task with only two classes and a small dataset, but not vice versa. Therefore, there is a large benefit to making the distance function asymmetric to account for the complexity of the task. The Asymmetric Task2Vec distance \( (d_{asym}) \) is defined as:

\[
d_{asym}(t_A \rightarrow t_B) = d_{sym}(t_A, t_B) - \alpha d_{sym}(t_A, t_0)
\]

(3)

Here \( t_0 \) is the “trivial” task of ImageNet classification. This formulation makes complex tasks that are very different from ImageNet relatively closer to everything else. The intuition is that a more composite task has a higher chance of being a relevant expert given the same degree of symmetric similarity. \( \alpha \)'s value varies by architecture. In the original work, \( \alpha = 0.3 \) is reported as optimal when training with a ResNet-34[13]. In our experiments with ResNet-18s we found \( \alpha = 0.15 \) to yield best performance. See Supplementary for further discussion of \( \alpha \).

**Limitations:** The Task2Vec formulation requires a fully specified task to have a labeled dataset, as well as a pretrained probe network to be available. We next address these limitations using our proposed alternatives below.

### 5. Characterization Without Labels: Pseudo-Task

**Motivation:** The first step in Task2Vec is to train a linear classifier with a probe network’s features for the task in question. This uses labels, which in turn provide the semantics of the task. But often we may want to characterize entire problem domains (e.g., x-ray images) without having a specific task in mind. In such cases, labeled datasets may be unavailable.

Even for well-specified tasks, the amount of labeled data required to characterize the task using the Task2Vec approach can also be substantial, precluding applications like few-shot learning where one may want to choose experts and take decisions with only one or two labels per class. To alleviate this issue, we present a self-supervised task characterization, which we dub “PseudoTask”.

**5.1. Method**

What information can we use to characterize a problem domain with no labels? Self-training approaches have recently shown the usefulness of training a network in one domain to match the predictions of a teacher from a source domain even when the two domains share no classes at all[29, 39]. Furthermore, unsupervised contrastive learning approaches such as MoCo and SimCLR[12, 5], demonstrate the emergence of semantics simply by learning to distinguish individual images.

Both approach types point to the striking power of pseudolabels, and motivate us to create PseudoTask, where pseudolabels are used as a substitute for ground-truth labeling. Concretely, PseudoTask follows the originally presented framework of Task2Vec with a key modification: instead of real labels, we use soft labels from a pretrained classifier (ImageNet or Places365).

Given a domain (task) \( T = (X_T) \) consisting of unlabeled images \( X_T = \{x_i\}_{i=1}^N \) and a pre-trained classifier \( \Theta \), we follow Algorithm 1 to compute our embedding. Note that when computing the Task2Vec embedding (second for loop), the soft predictions are computed dynamically on randomly augmented versions \( (x') \) of the images \( x \).

**Asymmetric PseudoTask:** The final adjustment we found necessary in the self-supervised algorithm was the measure of asymmetry. In the original work, a bias term of \(-\alpha d_{sym}(t_{source}, t_0)\) was used where \( t_0 \) is the “trivial” ImageNet
Indeed, we find that if the feature extractor is far from optimal, Task2Vec fails at effective characterization (Sec. 7.4). We therefore need an alternative approach that does not rely so heavily on a suitable feature representation.

The derivatives of random neural networks We want to declare two tasks to be similar if and only if a model trained on one produces a good feature extractor for the other, or alternatively, the optimal feature extractors for the two tasks are close to each other. A brute force approach to measuring task distance might thus be to separately train models for each task from the same initialization and look at how far the optima are in parameter space.

Of course, this is prohibitively expensive and obviously defeats the point, since we wanted to avoid training a separate model for the target task anyway. But what if we don’t train these models the whole way? Can we instead just train these models for very few epochs or steps and then evaluate how far they are? Concretely, imagine we start the training for both tasks using the same initialization, and take a single step. If the optimal models for the two tasks are close to each other, one might imagine that the very first update will also be close. If we repeat this for multiple initializations and find that the first update for the two tasks are always close, then one might conclude that the tasks are “similar”. Standard optimization procedures rely on gradient descent, so this first update corresponds to the gradient of the randomly initialized model. Thus, we hypothesize that a measure of task similarity could be computed by calculating the expected similarity between the gradients of the tasks at the same random initialization.

The Neural Tangent Kernel, which was first proposed in [14], describes the convergence behavior of neural networks in the limit of infinite width. A side-effect of this analysis is a kernel function between data points that comes close to mimicking the behavior of trained neural networks, but itself requires no training. This kernel takes the form:

\[ k(x, x') = E_\theta \left( \frac{\partial f(\theta, x)}{\partial \theta}, \frac{\partial f(\theta, x')}{\partial \theta} \right) \]

where \( x \) and \( x' \) are data points (e.g., images), and \( \theta \) is the parameters of a randomly initialized neural network drawn from a fixed distribution (typically Gaussian).

Our work essentially adapts the above kernel to operate on tasks instead of points. Based on the results with NTK, we reason that computing the NTK kernel over tasks instead of individual data points (by simply averaging the gradients of points in a task) thus provides a measure of similarity between the tasks.

We concretize this intuition as follows. Suppose we are given two tasks \( T_1 = (X_1, Y_1) \) and \( T_2 = (X_2, Y_2) \) consisting of images \( X_k = \{ x_i^{(k)} \}_{i=1}^{n_k}, k = 1, 2 \) and corresponding labels \( Y_k = \{ y_i^{(k)} \}_{i=1}^{n_k}, k = 1, 2 \).
Suppose \( L \) is a loss function such that a given feature extractor \( \phi \), \( L(\phi, X, Y) \) measures how well the feature extractor is able to separate out the classes in the task \( (X, Y) \). We randomly initialize \( N \) feature extractors \( \{\phi_i\}_{i=1}^{N} \) with parameters \( \{\theta_i\}_{i=1}^{N} \). For each feature extractor, for each task, we compute the gradient of the loss with respect to the feature vector parameters:

\[
g_{i}^{(k)} = \frac{\partial L(\phi_i, X_k, Y_k)}{\partial \theta_i} \quad (6)
\]

We then define a kernel between the two tasks as the average cosine distance between the two gradients:

\[
k(T_1, T_2) = \frac{1}{N} \sum_{i=1}^{N} \frac{(g_{i}^{(1)})(g_{i}^{(2)})}{\|g_{i}^{(1)}\| \|g_{i}^{(2)}\|} \quad (7)
\]

Because we are computing this task kernel using the gradients, we call this kernel the task tangent kernel. We only use the last residual block of the ResNet to compute the embedding as it contains the most channels.

**Loss function:** A key component here is the loss function \( L \). Typical loss functions such as cross entropy operate on predictions rather than features, which presents difficulty given the permutation-variant nature of a randomly initialized classifier head. We use the Maximal Coding Rate Reduction loss (MCR2)\[^{40}\]. This loss measures how close same-class features are, and how spread out the dataset is in feature space. Specifically, we embed the images \( X \) using the feature extractor \( \phi \) yielding embeddings \( Z \in \mathbb{R}^{d \times m} \) (\( d \) being the feature dimensionality, \( m \) the total number of data points). Let the set of embeddings of class \( j \) be denoted by \( Z_j \). Then the loss is defined as:

\[
L(\phi, X, Y) = -R(Z) + R_{\text{class}}(Z) \quad (8)
\]

\[
R(Z) = \frac{1}{2} \log \det \left( I + \frac{d}{m \epsilon^2} ZZ^\top \right) \quad (9)
\]

\[
R_{\text{class}}(Z) = \sum_{j} \frac{m_j}{2m} \log \det \left( I + \frac{d}{m_j \epsilon^2} Z_j Z_j^\top \right). \quad (10)
\]

With \( m_j \) as the number of data points with membership in class \( j \) and \( \epsilon \) a “prescribed precision” constant (\( \epsilon^2 = 0.5 \) in our work, see \[^{40}\] for details). The functions \( R \) and \( R_{\text{class}} \) describe the whole-dataset and per-class coding rates, measuring the compactness of all or subsets of features. This loss encourages the dataset as a whole to be non-compact (discriminable) while the class subsets should be highly compact (clustered), bearing resemblance to the supervised contrastive learning objective such as in\[^{16}\].

### 7. Experiments

#### 7.1. Meta-Task and Baselines

We perform experiments on the CUB+iNat meta-task of the Task2Vec paper \[^{2}\], denoted as \( T \). This set consists of 25 species classification tasks from Caltech-UCSD Birds\[^{38}\] and 25 from iNaturalist\[^{35}\]. Tasks are sets of species grouped at either the Order or Family level. For each individual task \( T \in T \), the benchmark requires us to choose an expert from \( T \). We measure the error obtained with this choice, relative to the error of the optimal choice, reporting the average relative error across all tasks.

**Baselines:** The Random baseline selects a task from \( T \) uniformly at random, and uses the corresponding expert. ImageNet Initialization does not use any of the experts available in \( T \), but instead uses a pretrained ImageNet network every time (an option not available to selection algorithms). Average Features uses an ImageNet-pretrained feature extractor to compute the average feature vector of images in the task, which is used as a task embedding (under cosine distance). Other metrics based solely on ImageNet-pretrained features, such as the H-Divergence\[^{17}\] (accuracy of a linear classifier separating domains) between tasks produced similar or worse results. RSA and LEEP are more sophisticated techniques that analyze the features produced by each expert-target pair. While these techniques prove to be quite effective, the computational cost of running inference using each expert quickly can become prohibitive as the number of considered tasks increases.

#### 7.2. Main Results

We present our main findings in Table 1 and Figure 3. Both proposed methods outperform baselines, with PseudoTask achieving performance on par with Task2Vec. Note that LEEP and RSA require computing predictions for each
Figure 3. Violin plot of the error on each target task (x-axis) obtained by training a linear head on top of a model from each of the other 49 tasks. Markers indicate selection algorithm choices. Both of our methods (PseudoTask and Task Tangent Kernel) reliably outperform using an ImageNet feature extractor. The experimental setting is the same as Figure 3 of [2].

expert/task pair; in the case of RSA an entire model must be trained on the target task before the expert initialization is chosen. In contrast, embedding methods only require per-pair vector arithmetic and representations are persistent.

**Task Tangent Kernel** TTK has the most significant handicap, operating solely on the provided task dataset as opposed to other methods which employ networks pretrained on over a million images. Despite this, TTK is still useful, beating the strong baseline of initializing from ImageNet every time (ImageNet is not a permitted expert to select in the benchmark). We see in Section 7.4 that the performance by TTK is far superior to any other method operating off of random initialization and that this benefit stems from using multiple randomized networks, in accordance with theoretical work involving the Neural Tangent Kernel.

**PseudoTask** PseudoTask outperforms the baselines by even more significant margins than TTK. Furthermore, by using Places365 as the initialization for the probe network instead of ImageNet, we are able to halve the mean relative error increase. In doing so we achieve results almost equal to the original supervised method despite no available labels. We present possible reasons for the success of Places365 initialization in Section 7.5.

In Figure 4, we compare the taxonomical distance between tasks to the induced symmetric distances of our methods. PseudoTask (ImageNet) has an even higher correlation with the ground truth taxonomy than the original Task2Vec.

![Figure 4. Average taxonomical distance between tasks in neighborhoods of varying sizes. Taxonomical distance is the how far up the phylogenetic tree (Order, Class, Phylum, Kingdom) the common root of the two tasks is. Black line represents the ground truth average distance in neighborhoods of a given size (calculated at each task in the meta-task). Preservation of taxonomical distance is desirable, demonstrating capture of the semantics of a task.](image)

**Example choices:** Figure 5 demonstrates PseudoTask’s decision-making ability. As an example, the 2nd column “Piciformes” is the task of classifying the 7 different types of woodpeckers in CUB. Walking through the selections:

1. Charadriiformes (CUB) is properly matched to its counterpart from iNat.
2. Cuculiformes (CUB) is properly matched to Passeriformes (iNat), a very large songbird dataset.
Figure 5. Examples of optimal selections vs. those made by PseudoTask. Source tasks (columns) are selected to transfer to target tasks (rows). Representative images of each dataset are shown. A blue bracket indicates the optimal choice, a red bracket the algorithm’s choice. PseudoTask selections are made without the use of labels.

3. Squamata (iNat, lizards and snakes) is “incorrectly” matched to Anura (iNat, frogs). This demonstrates how transferability and semantics are not perfectly correlated: taxonomically, Squamata is much closer to Anura than Lepidoptera.

7.3. Other Datasets

CUB Attributes One possible concern with PseudoTask is that it does not take into account the labels of the task. As such, it might end up choosing experts relevant to the domain, but not necessarily to the task at hand. We test this scenario by forming a variant of the CUB+iNat benchmark, called CUB-bp+iNat, where the CUB labels are the breast pattern of the birds instead of species classes. Per-task embeddings are re-calculated for Task2Vec and TTK, while experts are trained/transfered to obtain per-selection accuracies. Results are shown in Table 2 (L) for selections on the CUB half of the meta-task. This concern does not prove to be a detriment, on the contrary, PseudoTask performs better relative to other methods than in the purely class-based setting. Intriguingly, this suggests that the more important factor in transfer is the domain, rather than the precise semantics of the labels themselves.

Cars To validate our models on varied types of imagery, we create a new benchmark Cars from the Stanford Cars dataset[20]. The tasks are manufacturers (e.g. Audi) that have more than one model of car in the dataset. In Table 2 (R), we observe that PseudoTask performs nearly equally to Task2Vec and notably TTK’s performance exceeds both of them. This confirms our success of adapting the Task2Vec algorithm to function well without labels or initialization.

<table>
<thead>
<tr>
<th>Method</th>
<th>CUB-bp</th>
<th>Cars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>12.6%</td>
<td>28.2%</td>
</tr>
<tr>
<td>ImageNet Initialization</td>
<td>11.4%</td>
<td>-5.9%</td>
</tr>
<tr>
<td>PseudoTask (ImageNet)</td>
<td>9.3%</td>
<td>18.2%</td>
</tr>
<tr>
<td>Task Tangent Kernel</td>
<td>13.4%</td>
<td>14.0%</td>
</tr>
<tr>
<td>Task2Vec (orig)</td>
<td>12.6%</td>
<td>14.2%</td>
</tr>
</tbody>
</table>

Table 2. (L) Relative errors on the CUB half of the CUBbp+iNat. We see that, despite no knowledge of the label shift, PseudoTask performs substantially better than alternatives. (R) Relative errors on the Cars meta-task. Denominators in relative error calculation are buffered by 1 due to presence of zeros (perfect accuracies). We note that ImageNet Init. is a much stronger baseline than in CUB+iNat due to the strength of self-selection in Cars. For 80% of the tasks incorporating any extra data actually hurts performance.

7.4. Varied Initializations

The significance of initialization is demonstrated in Figure 6. As previously noted, PseudoTask performs significantly better with Places365 pretraining. Hypothesized reasons for this improvement are presented in Section 7.5. Task2Vec performs markedly worse when using Places365, but in the Supplementary we show that this is solely due to hyperparameter tuning.

Both Task2Vec and PseudoTask suffer dramatically with random initialization. Task2Vec still is significantly better than random choice, while PseudoTask is worse. We attribute this difference to PseudoTask having the same dependencies on the probe network as Task2Vec while additionally relying on the induced pseudolabels.

---

1 We create the Cars benchmark as the Mixed benchmark of [2] requires attribute labels that are not publicly available.
Figure 6. Error increase percentage for method-initialization pairs (lower is better). With random initialization, Task Tangent Kernel dramatically outperforms the other methods, more than halving the error. Hyperparameters are constant across initializations.

Figure 7. Each data point is a TTK experiment using \( x \) networks with \( 100/x \) batches of size 128 per network. The dashed horizontal line is the performance of Task2Vec on a random initialization. We see that TTK with a single random network performs comparably to Task2Vec, and that a diversity of models is more beneficial than repeated gradient computations on a single network.

By design, TTK does not rely on a probe network and has vastly superior performance compared to other methods without network initialization, where the error is over a factor of 2 lower. We confirm in Figure 7 that the performance of TTK stems from the diversity of models used. On the far left of the figure, only a single network is trained with 100 batches of data; this method is fairly equivalent to Task2Vec in both nature and performance. Performance improves with the number of networks, validating the motivation of the TTK: the expectation over random models can substitute for a powerful feature extractor.

### 7.5. ImageNet vs. Places365

In both Task2Vec and PseudoTask, Places365 initialization ultimately yields superior performance than the ImageNet counterparts. This is quite unintuitive, as generally ImageNet transfers better than Places365\(^{[10, 44]}\) which has made it the standard in transfer learning. Better pseudo-classification performance is not the reason; the Adjusted Mutual Information between the pseudolabels and ground truth is quite small, ImageNet has the larger score at 0.05 demonstrating a lack of label consistency despite the strong selection performance for both models.

This analysis focuses on PseudoTask. The average prediction confidence (post-softmax) of ImageNet and Places365 are 0.19 and 0.18 respectively, despite the former having nearly triple the available classes. These correspond to 19,000\( \times \) and 6,600\( \times \) higher than a uniform random guess. We theorize that the difference in performance stems from ImageNet making higher confidence predictions, even when incorrect, resulting in less informative embeddings.

We record the class predictions from both models for a batch from each task of size \( \min(100, n_{\text{dataset}}) \). ImageNet averages 84 different predictions per class (8% of 1000 total possibilities) while Places365 uses 77 different predictions (21% of 365 total possibilities). Across all tasks, ImageNet predicts 857 classes (85.7% of maximum possible) and Places365 354 (97% of maximum possible).

We hypothesize that this relative softness of prediction from the Places365 model is beneficial for PseudoTask because the objective becomes more akin to contrastive learning (e.g., MoCo or SimCLR\(^{[12, 5]} \)) instead of a one-hot classification problem. Consistently, PseudoTask with hard pseudolabels performs far worse. By softening the classification problem, the network parameters might equalize in discriminative power (and thus gradient), preventing a small subset of terms from dominating the embedding calculation.

We visualize the values of all embeddings in Figure 8. ImageNet has a longer tail of values while Places365 has a concentrated peak at relatively low values for both algorithms. Thus ImageNet yields “spikier” embeddings whose large values will dominate the distance calculations.

### 8. Conclusion

In this work, we designed methods to overcome current limitations of expert selection algorithms: namely performance without labels or model initializations. Our zero-label approach, PseudoTask, achieved nearly equal performance to the previous supervised version, Task2Vec. We demonstrated that both of these methods fail without pre-trained model initialization and created the Task Tangent Kernel which doubles the performance of other methods when pretraining is not available. Promising lines of future work include semi-supervised expert selection, zero-label zero-initialization methods, and extension to other data forms such as shapes or natural language.
References


[16] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2021. 5


[20] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. 7


[32] Gencer Sumbul, Marcela Charfuelan, Begüm Demir, and Volker Markl. Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. 1


[38] Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010. 5

[39] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. Self-training with noisy student improves imagenet classification, 2020. 2, 3


