Let Go of Your Labels with Unsupervised Transfer

Artyom Gadetsky [*1] **Yulun Jiang** [*1] **Maria Brbić** [1]

Abstract

Foundation vision-language models have enabled remarkable zero-shot transferability of the pre-trained representations to a wide range of downstream tasks. However, to solve a new task, zero-shot transfer still necessitates human guidance to define visual categories that appear in the data. Here, we show that *fully unsupervised transfer* emerges when searching for the labeling of a dataset that induces maximal margin classifiers in representation spaces of different foundation models. We present TURTLE, a fully unsupervised method that effectively employs this guiding principle to uncover the underlying labeling of a downstream dataset without any supervision and task-specific representation learning. We evaluate TURTLE on a diverse benchmark suite of 26 datasets and show that it achieves new state-of-the-art unsupervised performance. Furthermore, TURTLE, although being fully unsupervised, outperforms zero-shot transfer baselines on a wide range of datasets. In particular, TURTLE matches the average performance of CLIP zero-shot on 26 datasets by employing the same representation space, spanning a wide range of architectures and model sizes. By guiding the search for the underlying labeling using the representation spaces of two foundation models, TURTLE surpasses zero-shot transfer and unsupervised prompt tuning baselines, demonstrating the surprising power and effectiveness of unsupervised transfer.

1. Introduction

Transfer learning is a fundamental machine learning paradigm that leverages large-scale pre-training of deep neural networks to improve model performance on downstream tasks with limited resources (Pan & Yang, 2009). Early transfer learning approaches relied on supervised fine-tuning of the entire model to solve a downstream task of interest (Kolesnikov et al., 2020). Recent works (He et al., 2022; Li et al., 2022; Zhou et al., 2022; Oquab et al., 2023; Darcet et al., 2024) have shown that fine-tuning an entire model during transfer brings only marginal gains compared to training a linear classifier on top of the frozen pre-trained backbone (*i.e.*, linear probe). Although these approaches eliminated the need for task-specific fine-tuning of representations, they still require at least a few labeled examples per class to achieve human-level performance on downstream tasks.

Recently, foundation models (Bommasani et al., 2022) have emerged, approaching human-level intelligence on a variety of tasks in the zero-shot setting. In particular, Radford et al. (2021) proposed CLIP, which trains representations by aligning images and their corresponding captions in the joint embedding space. After pre-training, a zero-shot classifier is constructed by embedding the descriptions of visual categories that appear in the data. Subsequent works have successfully adopted this representation learning principle to enable zero-shot transfer in other domains, such as audio signal processing (Elizalde et al., 2023a;b), biomedicine (Lin et al., 2023; Robinson et al., 2023) and symbolic regression (Meidani et al., 2024). Despite the remarkable success of foundation models, zero-shot transfer still requires human instructions to solve a new task. But, can the representations of foundation models be utilized to solve a new task in a *fully unsupervised manner*?

The simplest approach for unsupervised transfer would be to apply off-the-shelf clustering methods (MacQueen, 1967) on top of the pre-trained representations. However, this strategy inevitably leads to a drastic decrease in performance compared to (weakly) supervised and zero-shot transfer (Zhou et al., 2022; Oquab et al., 2023). Recently, Gadetsky & Brbić (2023) introduced HUME, an unsupervised learning framework for inferring the underlying human labeling of a given dataset from pre-trained representations. While HUME has achieved superior performance compared to unsupervised baselines, it still requires task-specific representation learning and does not close the gap between unsupervised and zero-shot transfer.

Here, we present TURTLE, a method that enables unsuper-

[*]Equal contribution [1]EPFL, Lausanne, Switzerland. Correspondence to: Maria Brbić <mbrbic@epfl.ch>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

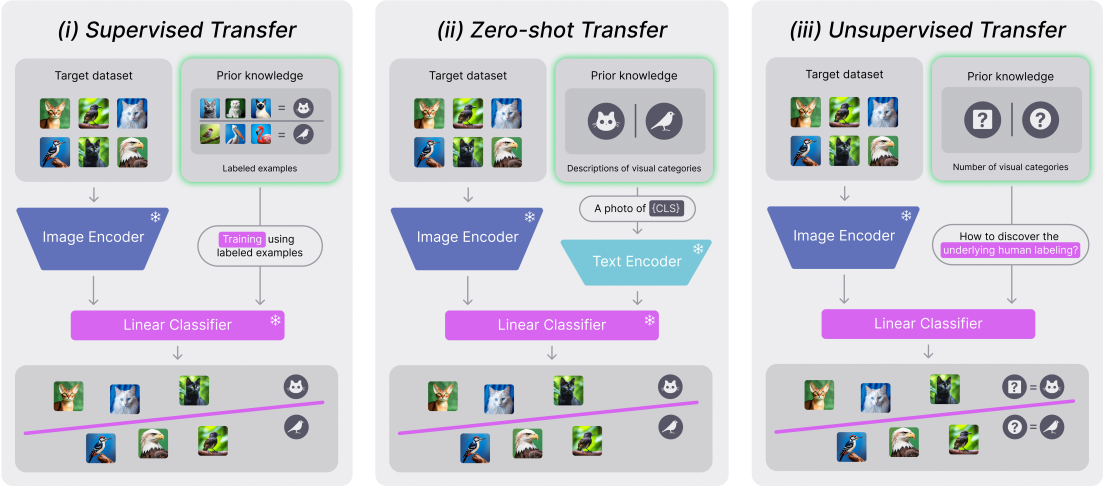

Figure 1. **Types of downstream transfer differ in the amount of available supervision.** Given representation spaces of foundation models, *(i) supervised transfer*, represented as a linear probe, trains a linear classifier given labeled examples of a downstream dataset; *(ii) zero-shot transfer* assumes descriptions of the visual categories that appear in a downstream dataset are given, and employs them via text encoder to solve the task; and *(iii) unsupervised transfer* assumes the least amount of available supervision, *i.e.*, only the number of categories is given, and aims to uncover the underlying human labeling of a dataset.

vised transfer from foundation models. The key idea behind our approach is to search for the labeling of a downstream dataset that maximizes the margins of linear classifiers in the space of single or multiple foundation models to uncover the underlying human labeling. Compared to zero-shot and supervised transfer, unsupervised transfer with TURTLE does not need the supervision in any form (Figure 1). Compared to deep clustering methods (Xie et al., 2016; Chang et al., 2017; Caron et al., 2018; Van Gansbeke et al., 2020; Niu et al., 2022), TURTLE does not require task-specific representation learning that is expensive for modern foundation models.

We study the performance of TURTLE on the extensive evaluation suite spanning 26 datasets and 7 different foundation models. We compare TURTLE to various baselines that differ in the amount of available supervision for the downstream transfer. First, when compared to the recent state-of-the-art unsupervised baselines, TURTLE outperforms these baselines on all the considered datasets, setting the new state-of-the-art unsupervised performance. Compared to zero-shot transfer, TURTLE instantiated with two foundation models surpasses CLIP zero-shot transfer across all studied model sizes, achieving exceptional absolute improvements up to 35% on the studied datasets. Given the same single representation space, TURTLE closely matches the performance of the CLIP zero-shot transfer on 7 out of 8 studied model architectures. In particular, the best TURTLE model, which utilizes the same model size and representation space, outperforms CLIP zero-shot on 13 out of 26 datasets. Finally, when compared to supervised transfer represented by linear probe, TURTLE approaches its performance on 5 out of 26 studied datasets, suggesting that labels may not be needed to infer the underlying human labeling when given sufficiently high-quality representations.

2. Background

In this section, we introduce the problem setting of unsupervised transfer and provide an overview of key concepts that we build upon.

Unsupervised transfer. Let $\mathcal{X} \subseteq \mathbb{R}^d$ be an input space and $\mathcal{D} = \{x_n\}_{n=1}^N,\ x_n \in \mathcal{X}$ be a dataset consisting of N samples and C classes, where C is known a priori. Let $\phi(x) : \mathcal{X} \to \mathbb{R}^q$ denotes a mapping from an input space $\mathcal{X}$ to a q-dimensional representation space of a pre-trained foundation model. The question we aim to answer is how to utilize representations from foundation models to solve a new task in a *fully unsupervised manner*. Thus, by unsupervised transfer we consider the task of inferring the underlying human labeling[1] of a dataset $\mathcal{D}$ without any supervision given representations of foundation models.

Generalization-based learning of human labelings. Gadetsky & Brbić (2023) recently introduced a generalization-based objective that evaluates the generalization ability of linear models on top of representations obtained from pre-trained models. The objective is motivated by a strong generalization ability of linear models

[1]We interchangeably use terms "task" and "labeling" in the context of this paper, since any labeling defines a task. Consequently, we refer to a task as human labeled if it corresponds to the underlying human labeling of a given dataset $\mathcal{D}$.

in representation spaces of foundation models on many human labeled tasks. Equipped with this insight, the goal is to find such labeling that optimizes generalization ability of a linear model over all possible labelings of a given dataset. The quality of a labeling is measured by the ability of a linear model to generalize on a task defined by the given labeling.

In particular, let $\tau : \mathcal{X} \to \{1, \dots, C\}$ denote a labeling function of a dataset. Let $f(x) = w^T \phi(x)$ denote a linear model in the representation space $\phi(x)$ of a foundation model. Given a train-test split $(\mathcal{D}_{tr}, \mathcal{D}_{te})$, one can train the model on a training split $\mathcal{D}_{tr}$ with labeling $\tau(\mathcal{D}_{tr})$ and classification loss function $\mathcal{L}$ to obtain $\hat{f}$. After training, the generalization ability of the model can be assessed by computing the error of $\hat{f}$ on $\mathcal{D}_{te}$. Consequently, the generalization-based objective is defined as follows:

$$\begin{aligned} &\min_{\tau} \sum_{x \in \mathcal{D}_{te}} \mathcal{L}(\hat{f}(x), \tau(x)) \\ &\text{s.t. } \hat{f} = \arg\min_{f} \sum_{x \in \mathcal{D}_{tr}} \mathcal{L}(f(x), \tau(x)), \end{aligned} \tag{1}$$

where minimization is performed over the set of all possible labelings of a dataset $\mathcal{D}$. This leads to a difficult discrete optimization problem. To overcome this limitation, Gadetsky & Brbić (2023) replace minimization w.r.t. a discrete labeling τ with minimization w.r.t. continuous parameters θ of a task encoder $\tau_\theta(x) : \mathcal{X} \to \Delta^{C-1}$, where Δ^{C-1} denotes $(C-1)$-dimensional probability simplex. As a result, careful design of τ_θ becomes crucial since it defines the search space explored by the generalization-based objective (1).

HUME framework. The instantiation of this framework, proposed in HUME (Gadetsky & Brbić, 2023), models τ_θ using a linear model in the representation space $\psi(x)$ obtained via self-supervised pre-training on the target dataset $\mathcal{D}$:

$$\tau_\theta^{\text{HUME}}(x) = \sigma(\theta^T \psi(x)), \tag{2}$$

where $\sigma : \mathbb{R} \to \Delta^{C-1}$ denotes an activation function. This modeling choice corresponds to *restricting the search space* in (1) to a set of labelings which are linearly separable in the representation space $\psi(x)$. In addition, obtaining $\psi(x)$ *requires task-specific representation learning*, *i.e.*, running self-supervised learning on the target dataset $\mathcal{D}$. Since reliable self-supervised pre-training necessitates a large amount of data (Wang & Isola, 2020), this prevents successful unsupervised transfer on downstream tasks with limited resources.

Given the task encoder parametrization $\tau_\theta^{\text{HUME}}$, HUME optimizes the following objective to search for the underlying human labeling:

$$\mathcal{L}^{\text{HUME}}(\theta) = \sum_{x \in \mathcal{D}_{te}} \mathcal{L}_{\text{ce}}(f_{\text{approx}}(x), \tau_\theta^{\text{HUME}}(x)), \tag{3}$$

where $\mathcal{L}_{\text{ce}}$ is the cross-entropy loss function and f_{approx} is an approximate solution to $\hat{f}$ obtained using iterative optimization algorithms. HUME resorts to iterative differentiation (Domke, 2012; Shaban et al., 2019) to solve the resulting bilevel optimization problem, *leading to an expensive overall training procedure*.

3. Analysis of Generalization-Based Objective

To understand inductive biases of the generalization-based objective proposed in (1), we consider this objective in case of binary labelings $\tau(x) : \mathcal{X} \to \{-1, +1\}$ with exponential loss function $\mathcal{L}_{\text{exp}}(f(x), \tau(x)) = \exp(-\tau(x) f(x))$. To simplify the analysis, we assume that the task encoder τ_θ is a linear model in the same representation space $\phi(x)$, *i.e.*, $\tau_\theta(x) = \sigma(\theta^T \phi(x))$, where $\sigma : \mathbb{R} \to [-1; 1]$ is an odd activation function such as $\tanh$. This corresponds to *restricting the search space* in (1) to a set of labelings which are linearly separable in the representation space $\phi(x)$. Additionally, we do not distinguish between train and test splits, *i.e.*, $\mathcal{D}_{tr} = \mathcal{D}_{te} = \mathcal{D}$. We provide a detailed discussion of the aforementioned assumptions in the remarks at the end of this section.

To obtain an approximate solution to $\hat{f}$, we use iterative optimization algorithms. Specifically, let $w_{m+1} = \Xi(w_m, \mathcal{D})$ denote a one step of an optimization algorithm, *i.e.*, $\Xi(w_m, \mathcal{D}) = w_m - \eta \nabla_w \sum_{x \in \mathcal{D}} \mathcal{L}(w_m^T \phi(x), \tau_\theta(x))$ for the gradient descent with a step size η. Similarly, let $w_M = \Xi^{(M)}(w_0, \mathcal{D})$ denote M steps of an optimization algorithm starting from w_0. Eventually, the above specifications result in the following bilevel optimization problem:

$$\mathcal{L}_M^{\text{binary}}(\theta) = \sum_{x \in \mathcal{D}} \exp(-\tau_\theta(x) w_M^T \phi(x)) \tag{4}$$

$$\text{s.t. } w_M = \Xi^{(M)}(w_0, \mathcal{D}), \tag{5}$$

where we refer to (4) and (5) as *inner* and *outer* objectives respectively.

The key observation underlying our main result is that the inner optimization (5) corresponds to the unregularized logistic regression on *separable* data, allowing us to employ the seminal result by Soudry et al. (2018). This work shows that gradient descent, when applied to the task of unregularized logistic regression, outputs iterates that are biased towards the direction of the max-margin hyperplane. Evidently, the task encoder τ_θ generates labelings of $\mathcal{D}$, which, by definition, are linearly separable in the representation space $\phi(x)$. Consequently, w_M will follow the direction of max-margin hyperplane for a given labeling τ_θ. In turn,

the last point to observe is that substituting the iterates in (4), the outer objective is minimized when w_M has a larger margin $\tau_\theta(x) w_M^T \phi(x)$ with respect to τ_θ. Equipped with this intuition, we are now ready to state our main result:

Proposition 3.1. *Given $M \gg 1$, $\theta \neq 0$ and appropriate step size η which ensures convergence, then*

$$\mathcal{L}_M^{binary}(\theta) \geq g(\theta) \|w_{SVM}(\theta)\|_2^2, \tag{6}$$

where $g(\theta) = (M\eta \exp(\|r_M(\theta)\|_2))^{-1}$, the residual $r_M(\theta)$ is bounded with $\lim_{M\to\infty} \|r_M(\theta)\|_2 = 0$, and $w_{SVM}(\theta)$ is the solution of the hard-margin SVM for a given θ:

$$\begin{aligned} w_{SVM}(\theta) = \min_{w} \quad & \|w\|_2^2 \\ \text{s.t.} \quad & \tau_\theta(x_n) w^T \phi(x_n) \geq 1 \quad \forall x_n \in \mathcal{D}. \end{aligned} \tag{7}$$

We defer the proof to Appendix A. This result shows that the generalization-based objective upper bounds the norm of hard-margin SVM fitted to a labeling τ_θ. Consequently, minimizing $\mathcal{L}_M^{\text{binary}}$ will inevitably lead to minimizing the norm (*i.e.*, maximizing the margin) *with respect to a labeling*. As a result, the optimization procedure will yield labelings with large margin of the corresponding classifier. Overall, our result unveils that the maximum margin principle (Vapnik, 1995), widely employed by supervised learning algorithms, emerges as the inductive bias of the generalization-based objective (1).

Remark 3.2. *(Search space restriction).* The result above holds when labelings generated by τ_θ are linearly separable in the representation space $\phi(x)$. This assumption leads to the analysis of the generalization-based objective (1) with the restricted search space. Ji & Telgarsky (2019) showed that in the case of non-separable labelings, gradient descent mirrors the separable case, following the max-margin direction of a maximal linearly separable subset of the data. Therefore, one could expect that the lower bound of the generalization-based objective (1) optimized over the complete search space inherits these properties, reflecting the separable case.

Remark 3.3. *(Train-test split assumption).* The generalization-based objective (1) assumes different train-test splits $(\mathcal{D}_{tr}, \mathcal{D}_{te})$ on the inner-outer levels respectively to obtain an unbiased estimate of the true risk of a model f. In our analysis, we simplify this assumption and employ $\mathcal{D}$ on both levels. Our result shows that minimizing the generalization-based objective in this case leads to maximizing the margin of a linear model with respect to a labeling τ on $\mathcal{D}$. In turn, this will inevitably lead to low error on a held out data given that margin size upper bounds generalization error (Bartlett & Shawe-Taylor, 1999; Gronlund et al., 2020).

Remark 3.4. *(Asymptotic analysis)* Proposition 3.1 is rather informal since it substitutes the asymptotic behaviour of the gradient descent iterates w_M into the outer objective. Although a rigorous analysis of the residual is required to establish exact bounds, these results serve to grasp the inductive bias incorporated in the generalization-based objective designed for the inference of human labelings.

In summary, this result shows that optimizing the generalization-based objective (1) yields labelings that induce maximal margin classifiers in the representation space $\phi(x)$. Our main result is greatly inspired by the seminal works (Soudry et al., 2018; Ji & Telgarsky, 2019) that reveal the implicit bias of gradient descent towards max-margin solution. Likewise, we demonstrate that the generalization-based objective (1) encourages labelings τ such that if one were to subsequently train a max-margin classifier in the representation space $\phi(x)$ to fit a labeling τ, *the margin obtained would be maximal over all possible labelings*.

4. TURTLE Framework

These insights serve us as a guiding principle to develop TURTLE, a general framework for efficient *fully unsupervised transfer* given representations of foundation models.

Optimization objective. Proposition 3.1 provides an important insight on the inductive bias incorporated in the generalization-based objective (1). Indeed, one can search for the underlying human labeling by maximizing the margin of a linear model with respect to a labeling. Pushing the limits of this principle, we propose to search for a labeling by maximizing margins of linear models *in spaces of multiple foundation models at the same time*. Given K foundation models, let $\phi_k(x)$ be a representation space of k-th foundation model. Given labeling defined by a task encoder τ_θ, let w_M^k be k-th linear model trained to fit this labeling in a representation space $\phi_k(x)$. Then, TURTLE's optimization objective is as follows:

$$\begin{aligned} \mathcal{L}_M^{\text{TURTLE}}(\theta) = & \sum_{k=1}^{K} \sum_{x \in \mathcal{D}} \mathcal{L}_{\text{ce}}(w_M^k \phi_k(x); \tau_\theta(x)) \\ & \text{s.t. } w_M^k = \Xi^{(M)}(w_0^k, \mathcal{D}), \forall k, \end{aligned} \tag{8}$$

where, $\Xi^M(w_0^k, \mathcal{D})$ denotes an iterative optimization algorithm Ξ run for M steps starting from w_0^k. Intuitively, each of the K terms in the loss function encourages τ_θ to maximize margin of k-th linear model in the corresponding representation space ϕ_k. As opposed to the HUME's objective (3), which maximizes margin only in the single space $\psi(x)$, TURTLE provides more effective guidance to the search process.

Task encoder parametrization. The parametrization of a task encoder τ_θ defines the search space of labelings, thus it has a crucial importance on the optimization process. In TURTLE, we employ pre-trained representation spaces of

foundation models to define a task encoder τ_θ. These representations *remain fixed* during the overall training procedure, alleviating the need of task-specific representation learning.

In particular, given K representation spaces $\phi_k(x)$, we define our task encoder τ_θ as follows:

$$\tau_\theta^{\text{TURTLE}}(x) = \frac{1}{K}\sum_{k=1}^{K} \tau_{\theta_k}(x), \quad \text{where } \tau_{\theta_k}(x) = \sigma(\theta_k^T \phi_k(x)), \tag{9}$$

such that $\theta = \{\theta_1, \ldots, \theta_K\}$ denotes all trainable parameters and σ is a softmax activation function. After training, cluster assignments are computed as usual:

$$\arg\max_{c=1,\ldots,C} \left[\tau_\theta^{\text{TURTLE}}(x)\right]_c, \tag{10}$$

where $\left[\tau_\theta^{\text{TURTLE}}(x)\right]_c$ denotes the probability of assigning a sample x to the c-th cluster.

Compared to the HUME framework in (2) which searches for the underlying human labeling only over all linearly separable labelings in the self-supervised representation space $\psi(x)$, TURTLE's parametrization greatly expands the search space. Indeed, modeling τ_θ as a simple ensemble induces the search space which is at least union of all linearly separable labelings in each of the representation spaces of foundation models $\phi_1, \ldots, \phi_K$. One could further suggest employing deeper architectures to model τ_θ, however such modeling choice may give rise to tasks that capture spurious correlations in data and do not necessarily reflect human labelings (Atanov et al., 2022). Therefore, our design choice effectively increases the search space and alleviates the need of task-specific fine-tuning by employing strong representations of foundation models.

Regularization. The task encoder can synthesize degenerate labelings, *i.e.*, assign all samples to a single class (Gadetsky & Brbić, 2023). Although such labelings induce linear classifiers with the largest possible margin in all representation spaces, they are irrelevant. To avoid such trivial solutions, we separately regularize each term of the task encoder:

$$\mathcal{R}(\theta) = \sum_{k=1}^{K} \mathbb{H}(\overline{\tau}_{\theta_k}^k), \tag{11}$$

where $\overline{\tau}_{\theta_k}^k = (|\mathcal{D}|)^{-1}\sum_{x\in\mathcal{D}} \tau_{\theta_k}(x) \in \Delta^{C-1}$ is an empirical label distribution of k-th component τ_{θ_k} and $\mathbb{H}(\cdot)$ is the entropy function of discrete distribution.

Final objective function. Putting (8) and (11) together, TURTLE finally optimizes the following objective function:

$$\min_\theta \mathcal{L}_M^{\text{TURTLE}}(\theta) - \gamma\mathcal{R}(\theta), \tag{12}$$

where we found $\gamma = 10$ is a good default choice for the entropy regularization strength γ. We show robustness to this hyperparameter in Appendix G.

Efficient optimization. The new optimization-based objective (8) is a bilevel optimization problem with the convex inner part. Indeed, given τ_θ, computing w_M^k corresponds to the logistic regression problem on $\mathcal{D}$ with labeling $\tau_\theta(\mathcal{D})$ in the k-th representation space ϕ_k. Learning parameters θ using gradient-based techniques involves computing a total derivative $\frac{d}{d\theta}\mathcal{L}_M^{\text{TURTLE}}$:

$$\frac{d}{d\theta}\mathcal{L}_M^{\text{TURTLE}} = \frac{\partial}{\partial\theta}\mathcal{L}_M^{\text{TURTLE}} + \sum_{k=1}^{K}\left(\frac{\partial w_M^k}{\partial\theta}\right)^T \frac{\partial}{\partial w_M^k}\mathcal{L}_M^{\text{TURTLE}}, \tag{13}$$

where $\frac{\partial w_M^k}{\partial\theta}$ is the Jacobian, which is expensive to compute in practice (Domke, 2012; Shaban et al., 2019). The key observation is that employing the same set of samples $\mathcal{D}$ on both inner and outer levels allows us to discard the second term of the total derivative. Indeed, after training w_M^k on $\mathcal{D}$, one can approximate $\frac{d}{d\theta}\mathcal{L}_M^{\text{TURTLE}} \approx \frac{\partial}{\partial\theta}\mathcal{L}_M^{\text{TURTLE}}$ since w_M^k is an approximate stationary point of the inner problem for a given τ_θ, *i.e.*, $\frac{\partial}{\partial w_M^k}\mathcal{L}_M^{\text{TURTLE}} \approx 0$. Ablin et al. (2020) have shown a strong performance of this estimator in practice for bilevel optimization problems similar to ours. The pseudocode of TURTLE is provided in Algorithm B1 with implementation details in Appendix B.3.

5. Experiments

5.1. Experimental setup

Datasets and evaluation metric. We study the performance of TURTLE on the extensive benchmark of 26 vision datasets (Radford et al., 2021). The detailed description of each dataset is provided in Appendix B.1. We compare our framework with the baselines using accuracy metric and employ Hungarian algorithm (Kuhn, 1955) to match the labeling found by TURTLE (10) to the ground truth labeling of a corresponding dataset. By default, we train TURTLE on the training split of a corresponding dataset and provide the results on the test split. In Appendix H, we additionally show that mimicking deployment regime, *i.e.*, having only test split available for training, does not lead to performance decrease of TURTLE.

Foundation models in TURTLE. We employ CLIP (Radford et al., 2021) representations which span different architectures and model sizes, in particular, 5 different ResNets (R50, R101, R50x4, R50x16 and R50x64) and 3 different Vision Transformers (ViT-B/32, ViT-B/16 and ViT-L/14). We refer to the TURTLE as TURTLE 1-space if it utilizes only a single space CLIP representation ($K = 1$ in (8) and (9)). We refer to the TURTLE as TURTLE 2-spaces

if it utilizes two different foundation models. Namely, we use DINOv2 ViT-g/14 (Oquab et al., 2023) as the second space while the first space is always represented with one of the CLIP variants. Consequently, to specify the particular CLIP architecture when utilizing two representation spaces, *e.g.*, ViT-L/14, we refer to TURTLE as TURTLE 2-spaces ViT-L/14. We precompute all representations for the entire benchmark and keep these representations fixed during the overall training procedure. The detailed description of the used models and other specifications to prepare representations are provided in Appendix B.2.

Baselines. We compare unsupervised transfer using TURTLE to baselines that differ in the amount of supervision they use (Figure 1). First, we compare TURTLE to HUME (Gadetsky & Brbić, 2023), a method that has recently shown state-of-the-art unsupervised learning performance and surpassed traditional deep clustering approaches (Van Gansbeke et al., 2020; Niu et al., 2022; Amrani et al., 2022; Feng et al., 2023). Next, to explore how far can we go with unsupervised transfer, we compare TURTLE in a challenging setting to zero-shot transfer, unsupervised prompt tuning and supervised baselines. All these baselines use some form of supervision compared to TURTLE which is fully unsupervised. We start by comparing TURTLE to the CLIP zero-shot transfer (Radford et al., 2021) that employs descriptions of ground truth classes as a form of supervision. Following (Radford et al., 2021), we perform prompt engineering and ensembling to construct a zero-shot classifier for each dataset. As even stronger baselines, we compare TURTLE to the state-of-the-art unsupervised prompt tuning methods UPL (Huang et al., 2022), POUF (Tanwisuth et al., 2023) and GDA (Wang et al., 2024). These approaches enhance class prototypes defined by the CLIP zero-shot classifier via unsupervised adaptation on the downstream task. Finally, we employ supervised linear probe on top of the CLIP representations to serve as a supervised transfer baseline. Differences between types of transfer are highlighted in Table 1.

Table 1. **Differences between the considered types of downstream transfer.**

	Available Supervision	Training on $\mathcal{D}$
Unsupervised transfer (*ours*)	Number of classes	✓
Zero-shot transfer	Class descriptions	✗
Unsupervised prompt tuning	Class descriptions	✓
Supervised transfer	Labeled samples	✓

Model Selection. Gadetsky & Brbić (2023) showed that generalization-based objective (1) is strikingly well-correlated with human labelings, which we further confirm in Figure B1 on 26 datasets. Notably, this enables *unsupervised hyperparameter search* in TURTLE. For supervised linear probes, we perform standard cross-validation to search for the L2-regularization strength. We refer the reader to Appendix B.3 for the detailed description of our model selection procedures. Code is publicly available at `https://github.com/mlbio-epfl/turtle`.

5.2. Results

Comparison to unsupervised baselines. We start by comparing TURTLE to HUME. Originally, HUME utilized self-supervised representation learning on the given dataset $\mathcal{D}$ to model the task encoder (2). To ensure the fair comparison, we instead employ representation spaces of foundation models for both modeling the task encoder (2) and for representation space used to model f_{approx} in (3). Consequently, both TURTLE and HUME use the same representation spaces, *i.e.*, CLIP ViT-L/14 and DINOv2. Furthermore, we improve the optimization procedure of HUME to enable accelerated convergence. In addition, we compare TURTLE to the K-Means clustering (MacQueen, 1967) on top of concatenated embeddings from both representation spaces employed by TURTLE. The K-Means clustering serves as the simple unsupervised transfer baseline since, like TURTLE, it does not require task-specific representation learning. We refer the readers to Appendix C for the detailed description of the improvements made to HUME as well as the implementation details of the K-Means.

As shown in Figure 2, TURTLE substiantially outperforms HUME on all considered datasets, confirming that maximizing margin in both spaces simultaneously to search for the underlying human labeling (8) and expanding the search space of labelings (9) is the effective design choice. Remarkably, TURTLE leads to 23% and 11% absolute improvements (30% and 18% relative improvement) on the MNIST and Birdsnap datasets, respectively. Furthermore, among other datasets, TURTLE sets the new state-of-the-art unsupervised performance on the ImageNet dataset, achieving 72.9% accuracy and outperforming the previous state-of-the-art (Alkin et al., 2024) by 5.5%.

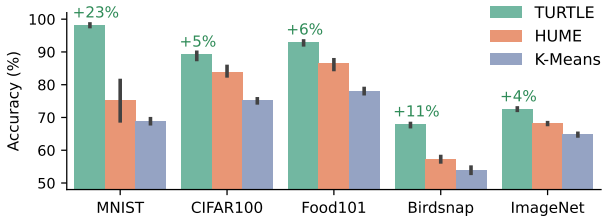

Figure 2. **TURTLE outperforms unsupervised baselines.** Comparison of TURTLE to unsupervised baselines with respect to accuracy. All methods use the CLIP ViT L/14 and DINOv2 representations. Bars represent the average performance with standard deviations computed over three runs.

In addition, we validate optimization efficiency of TURTLE and compare training time between all the considered methods in Figure 3. The results corroborate the use of first-

order hypergradient approximation (13) in TURTLE. Notably, TURTLE achieves $10\times$ speedup compared to HUME, achieving the impressive training time on the ImageNet dataset that takes less than five minutes. Overall, our results show that TURTLE effectively addresses the challenges of unsupervised transfer and outperforms unsupervised baselines by a large margin.

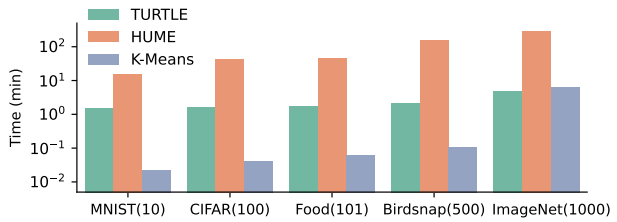

Figure 3. **TURTLE is an efficient method.** Comparison of running time between TURTLE and unsupervised baselines. TURTLE employs efficient first-order optimization procedure, achieving more than **10× speedup** compared to HUME. All methods use CLIP ViT L/14 and DINOv2 representations. Bars represent the average performance over three runs. Standard deviations are negligible (Table C2) and omitted for clarity.

Comparison to zero-shot transfer. We compare TURTLE to the CLIP zero-shot transfer that uses descriptions of ground truth classes as a form of supervision. Remarkably, without using any supervision, TURTLE 2-spaces outperforms the zero-shot transfer of CLIP by a large margin across 26 benchmark datasets for different ViT backbones (Figure 4).

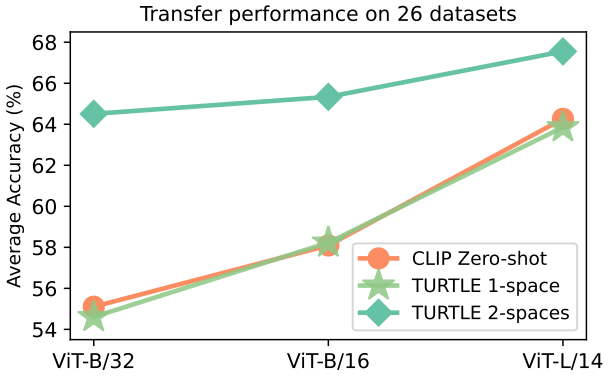

Figure 4. **TURTLE enables unsupervised transfer given representation spaces of foundation models.** Employing the same CLIP representation space, TURTLE closely matches the performance of the corresponding CLIP zero-shot classifier on average over 26 datasets. With the use of an additional representation space, TURTLE outperforms zero-shot transfer, demonstrating exceptional abilities of unsupervised transfer learning.

In particular, TURTLE 2-spaces outperforms CLIP zero-shot by 9%, 7% and 4% absolute improvement (17%, 12% and 5% relative improvement) with ViT-B/32, ViT-B/16 and ViT-L/14 backbones, respectively. Moreover, even TURTLE 1-space matches the performance of CLIP zero-shot across

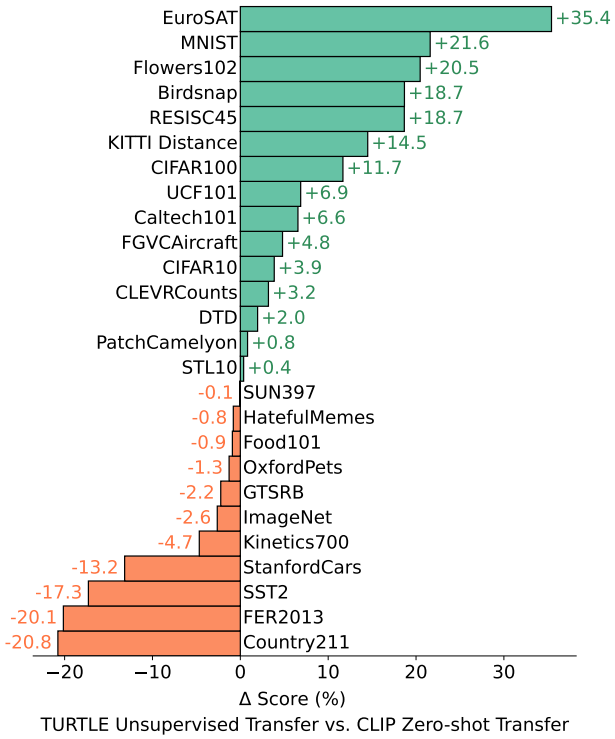

Figure 5. **TURTLE outperforms the CLIP zero-shot classifier on 15 out of 26 datasets.** TURTLE is trained with CLIP ViT-L/14 and DINOv2 representations. CLIP zero-shot utilizes the same CLIP ViT-L/14 architecture. Furthermore, we observe that even with only a single CLIP representation space TURTLE outperforms CLIP on $13/26$ datasets (Figure E1).

all studied ViT models. It is important to note that both CLIP zero-shot and TURTLE 1-space are linear models in the same representation space and *differ only in the amount of supervision* which is available to produce the weights. When comparing performance on individual datasets, TURTLE outperforms CLIP zero-shot transfer on 15 out of 26 datasets with remarkable absolute gains of 35%, 21% and 20% on the EuroSAT, MNIST and Flowers102 datasets, respectively (Figure 5). We provide individual scores for all TURTLE and CLIP zero-shot variants in Appendix D.

Comparison to unsupervised prompt tuning. Next, we compare TURTLE to unsupervised prompt tuning baselines. We follow previous works and use CLIP ResNet-50 representations for all methods. Although being fully unsupervised, TURTLE consistently outperforms all the considered baselines by a large margin (Table 2). Specifically, TURTLE achieves 8% absolute improvement (12% relative improvement) in average accuracy over the best unsupervised prompt tuning baseline. On the Flowers102 and EuroSAT datasets, our framework attains outstanding absolute gains of 27% and 41% (37% and 75% relative improvement), respectively. Overall, these results demonstrate the surprising effectiveness of the unsupervised transfer.

Table 2. **TURTLE 2-spaces outperforms unsupervised prompt tuning methods.** *ZS* column indicates whether method utilizes zero-shot supervision to make predictions. All methods employ CLIP ResNet-50 representations. TURTLE additionally uses DINOv2 representations as the second representation space.

Method	ZS	Pets	Flowers	FGVC	DTD	EuroSAT	Cars	Food	SUN	Caltech	UCF	ImageNet	Avg.
POUF	✓	88.0	66.7	16.7	41.5	42.1	57.4	74.7	58.6	86.9	61.1	55.2	59.0
UPL	✓	88.3	68.9	17.3	46.6	54.8	**62.1**	77.6	64.0	**89.9**	67.2	60.5	63.4
GDA	✓	89.9	72.7	18.7	46.8	49.9	60.8	78.3	63.6	87.5	68.7	61.2	63.5
TURTLE	✗	**90.9**	**99.7**	**25.3**	**57.0**	**95.5**	32.6	**84.1**	**65.7**	88.6	**77.7**	**66.3**	**71.2**

Comparison to supervised transfer. Finally, we compare TURTLE 1-space ViT-L/14 to supervised linear probe in the same representation space. This means that in this setup both models are linear in the representation space of CLIP ViT-L/14 and differ only in the amount of supervision utilized to produce the weights. Supervised linear probe is trained using all available labels. Consequently, we can assume that it represents the maximal transfer learning performance that can be achieved by the unsupervised transfer.

We observe a high positive correlation of 0.87 (p-value $< 10^{-8}$) between unsupervised transfer performance and its fully supervised counterpart (Figure 6). This result indicates that with better supervised linear probe performance, TURTLE's performance may also increase, which we further investigate in the subsequent paragraph. Notably, TURTLE approaches the "optimal" transfer performance on the STL10, CIFAR10, Flowers102, Food101 and HatefulMemes, demonstrating that *labels may not be needed* when given sufficiently high-quality representations, as measured by supervised linear probe. We perform similar analysis for TURTLE 2-spaces and observe stronger correlation, leading to reduced gap between TURTLE 2-spaces and supervised linear probe (Figure E2).

Ablation of different representation spaces on ImageNet. Results from the previous paragraph speculate that incorporating stronger representations may lead to the increased performance of unsupervised transfer. To validate this, we run TURTLE with pairs of different representation spaces on the ImageNet-1000 dataset (Deng et al., 2009). Results in Figure 7 show a positive correlation of 0.74 (p-value $< 10^{-8}$) between unsupervised transfer performance and the quality of representations measured by supervised linear probe. The obtained result confirms that employing stronger representations for a given dataset leads to the improved performance of TURTLE. Consequently, TURTLE can further improve performance by exploiting continual progress in the development of foundation models. Furthermore, given high positive correlation between TURTLE's accuracy and the generalization-based objective (Figure B1), TURTLE can be utilized as the proxy to measure the quality of given representations in the absence of labels for the downstream task.

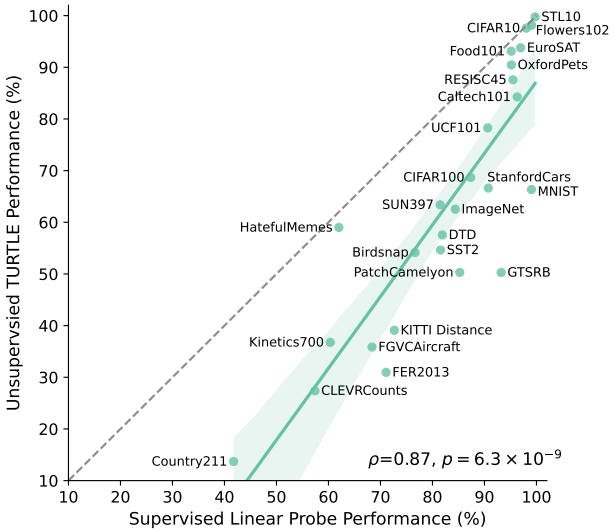

Figure 6. **Unsupervised transfer performance of TURTLE is correlated with supervised linear probe performance.** Dashed line $y = x$ denotes the "optimal" unsupervised transfer. The performance of TURTLE and supervised linear probe shows a strong correlation ($\rho = 0.87, p = 6.3 \times 10^{-9}$ of two-sided Pearson correlation coefficient). On 5 datasets TURTLE approaches the performance of the "optimal" unsupervised transfer (≤ 3 point difference).

6. Related Work

(Weakly) supervised transfer. (Weakly) supervised transfer approaches require at least some amount of supervision to perform downstream transfer. For instance, BigTransfer (Kolesnikov et al., 2020) showed that supervised fine-tuning of the entire model after large-scale pre-training successfully transfers knowledge in both fully supervised and few-shot regimes. Recent advances in self-supervised learning (He et al., 2022; Li et al., 2022; Zhou et al., 2022; Oquab et al., 2023; Darcet et al., 2024) have demonstrated that a linear probe suffices to achieve competitive performance compared to the fine-tuning the entire model. Despite the strength of these approaches, they necessitate labeled examples to perform downstream transfer.

Zero-shot transfer. Foundation models such as CLIP (Radford et al., 2021) have recently enabled zero-shot transfer, which relies only on a set of human instructions such as de-

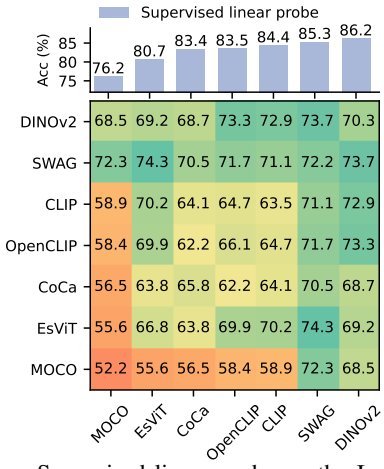

Figure 7. **Top**: Supervised linear probe on the ImageNet-1000 dataset for 7 different representation spaces. **Bottom**: Heat map represents unsupervised performance of TURTLE on ImageNet-1000. Secondary diagonal cells correspond to TURTLE 1-space, while off-diagonal cells refer to TURTLE 2-spaces with the pair of corresponding representation spaces. The performance of TURTLE indicates a strong positive correlation with the performance of supervised linear probe ($\rho = 0.74, p = 1.4 \times 10^{-9}$ of two-sided Pearson correlation coefficient).

scriptions of visual categories that appear in the data rather than a set of labeled examples. Despite the success of zero-shot transfer in different domains (Elizalde et al., 2023a;b; Lin et al., 2023; Robinson et al., 2023; Meidani et al., 2024), collecting zero-shot annotations still requires expert domain knowledge which can be hard to get in many real-world applications. In contrast to the zero-shot transfer approaches, TURTLE enables *fully unsupervised transfer*, effectively alleviating the need of any human guidance.

Deep clustering. Deep clustering methods (Xie et al., 2016; Chang et al., 2017; Caron et al., 2018; Van Gansbeke et al., 2020; Niu et al., 2022) aim to jointly perform deep representation learning and clustering on a target dataset. Recent state-of-the-art approaches (Van Gansbeke et al., 2020; Niu et al., 2022) rely on time-consuming three-stage procedures that involve self-supervised representation learning, clustering and fine-tuning via self-labeling respectively. In contrast to the deep clustering approaches, TURTLE alleviates the need for laborious task-specific representation learning by employing representation spaces of pre-trained foundation models. Furthermore, compared to deep clustering methods that heavily depend on image augmentations to induce semantically meaningful clusters, TURTLE builds upon the seminal maximum margin principle that is effortlessly applicable beyond image data modality. Consequently, our approach offers an efficient and effective way to perform fully unsupervised transfer from foundation models.

Maximum margin clustering. Our work has revealed that optimizing the generalization-based objective proposed in Gadetsky & Brbić (2023) results in the search for a labeling that maximizes the margin of a maximal margin classifier over all possible labelings of a dataset. The first attempt to employ maximum margin principle to perform clustering dates back to Maximum Margin Clustering (MMC) (Xu et al., 2004). Later works extended this framework to multi-class clustering (Xu & Schuurmans, 2005; Wang et al., 2010), multi-view clustering (Zhao et al., 2009), or focused on improving the scalability (Zhang et al., 2007; Wang et al., 2010). Compared to TURTLE, which employs efficient first-order gradient optimization techniques, the aforementioned approaches rely on the expensive discrete optimization techniques. Furthermore, each of the approaches adopts maximum margin principle in its own way to enable multi-class or multi-space scenario, while TURTLE provides a unified framework for any number of classes and representation spaces.

Implicit bias of optimization algorithms. Understanding the implicit bias of optimization algorithms plays a crucial role in modern machine learning. The seminal work by Soudry et al. (2018) showed that the gradient descent, when applied to the task of unregularized logistic regression on separable data, converges to the direction of the maximal margin hyperplane without explicitly enforcing such margin maximization. Later, Ji & Telgarsky (2019) extended the analysis and demonstrated a similar behavior of gradient descent in the case of non-separable data. In our work, we employ the aforementioned findings to study the inductive bias of the generalization-based objective. Surprisingly, we reveal that it yields labelings that maximize the margin of a maximal margin classifier with respect to labeling. As a result, this insight allows us to develop TURTLE, a method that enables fully unsupervised transfer given representations of foundation models.

7. Conclusion

In this work, we have shown that the representations of foundation models can be utilized to solve a new task in a fully unsupervised manner. The key insight behind our approach is to search for a labeling that induces maximal margin classifiers in the representation spaces of foundation models. We utilize this insight to develop TURTLE, a general framework for effective unsupervised transfer given representations of different foundation models. Through extensive evaluation, we found that TURTLE, being fully unsupervised, achieves competitive performance compared to zero-shot transfer by employing only a single representation space. Furthermore, utilizing an additional representation space results in remarkable gains over zero-shot transfer. Given the flexibility of our framework, the results also suggest that TURTLE can deliver even better unsupervised transfer performance by taking advantage of new more powerful foundation models that will emerge in the future.

Acknowledgements

We thank Chanakya Ekbote, Shuo Wen and Tingyang Yu for valuable suggestions that helped to improve the clarity of the manuscript. We also thank Nikita Doikov for fruitful discussions regarding efficient bilevel optimization techniques. We gratefully acknowledge the support of EPFL and ZEISS.

Impact Statement

Although the main goal of our work is to advance the field of Machine Learning, the proposed framework relies on representation spaces of foundation models. These models inherit biases embedded in the data on which they were trained on (Bommasani et al., 2022). Consequently, the extensive evaluation and alignment is recommended when deploying TURTLE to critical use-cases such as medicine.

References

Ablin, P., Peyré, G., and Moreau, T. Super-efficiency of Automatic Differentiation for Functions Defined as a Minimum. In *International Conference on Machine Learning*, 2020.

Alkin, B., Miklautz, L., Hochreiter, S., and Brandstetter, J. MIM-Refiner: A Contrastive Learning Boost from Intermediate Pre-Trained Representations. *arXiv preprint arXiv:2402.10093*, 2024.

Amrani, E., Karlinsky, L., and Bronstein, A. Self-supervised Classification network. In *European Conference on Computer Vision*, 2022.

Atanov, A., Filatov, A., Yeo, T., Sohmshetty, A., and Zamir, A. Task Discovery: Finding the Tasks that Neural Networks Generalize on. In *Advances in Neural Information Processing Systems*, 2022.

Bartlett, P. and Shawe-Taylor, J. Generalization Performance of Support Vector Machines and Other Pattern Classifiers. In *Advances in Kernel Methods: Support Vector Learning*, 1999.

Berg, T., Liu, J., Woo Lee, S., Alexander, M. L., Jacobs, D. W., and Belhumeur, P. N. Birdsnap: Large-Scale Fine-grained Visual Categorization of Birds. In *Computer Vision and Pattern Recognition*, 2014.

Bolte, J., Pauwels, E., and Vaiter, S. One-Step Differentiation of Iterative Algorithms. In *Advances in Neural Information Processing Systems*, 2023.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., et al. On the Opportunities and Risks of Foundation Models. *arXiv preprint arXiv:2108.07258*, 2022.

Bossard, L., Guillaumin, M., and Van Gool, L. Food-101–Mining Discriminative Components with Random Forests. In *European Conference on Computer Vision*, 2014.

Caron, M., Bojanowski, P., Joulin, A., and Douze, M. Deep clustering for unsupervised learning of visual features. In *European Conference on Computer Vision*, 2018.

Carreira, J., Noland, E., Hillier, C., and Zisserman, A. A Short Note on the Kinetics-700 Human Action Dataset. *arXiv preprint arXiv:1907.06987*, 2019.

Chang, J., Wang, L., Meng, G., Xiang, S., and Pan, C. Deep Adaptive Image Clustering. In *International Conference on Computer Vision*, 2017.

Chen, X., Xie, S., and He, K. An Empirical Study of Training Self-Supervised Vision Transformers. In *International Conference on Computer Vision*, 2021.

Cheng, G., Han, J., and Lu, X. Remote Sensing Image Scene Classification: Benchmark and State of the Art. In *Proceedings of the IEEE*, 2017.

Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., and Jitsev, J. Reproducible Scaling Laws for Contrastive Language-Image Learning. In *Computer Vision and Pattern Recognition*, 2023.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. Describing Textures in the Wild. In *Computer Vision and Pattern Recognition*, 2014.

Coates, A., Ng, A., and Lee, H. An Analysis of Single-layer Networks in Unsupervised Feature Learning. In *International Conference on Artificial Intelligence and Statistics*, 2011.

Dagréou, M., Ablin, P., Vaiter, S., and Moreau, T. A Framework for Bilevel Optimization that Enables Stochastic and Global Variance Reduction Algorithms. In *Advances in Neural Information Processing Systems*, 2022.

Darcet, T., Oquab, M., Mairal, J., and Bojanowski, P. Vision Transformers Need Registers. In *International Conference on Learning Representations*, 2024.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A Large-scale Hierarchical Image Database. In *Computer Vision and Pattern Recognition*, 2009.

Domke, J. Generic Methods for Optimization-Based Modeling. In *International Conference on Artificial Intelligence and Statistics*, 2012.

Elizalde, B., Deshmukh, S., Al Ismail, M., and Wang, H. CLAP: Learning Audio Concepts from Natural Language Supervision. In *International Conference on Acoustics, Speech and Signal Processing*, 2023a.

Elizalde, B., Deshmukh, S., and Wang, H. Natural Language Supervision for General-Purpose Audio Representations. *arXiv preprint arXiv:2309.05767*, 2023b.

Fei-Fei, L., Fergus, R., and Perona, P. Learning Generative Visual Models From Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories. In *Conference on Computer Vision and Pattern Recognition Workshop*, 2004.

Feng, W., Tao, K., Rufeng, Z., and Huaping, L. Self-supervised Learning by Estimating Twin Class Distribution. In *IEEE Transactions on Image Processing*, 2023.

Finn, C., Abbeel, P., and Levine, S. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *International Conference on Machine Learning*, 2017.

Gadetsky, A. and Brbić, M. The Pursuit of Human Labeling: A New Perspective on Unsupervised Learning. In *Advances in Neural Information Processing Systems*, 2023.

Geiger, A., Lenz, P., and Urtasun, R. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Computer Vision and Pattern Recognition*, 2012.

Goodfellow, I. J., Erhan, D., Carrier, P. L., Courville, A., Mirza, M., Hamner, B., Cukierski, W., Tang, Y., Thaler, D., Lee, D.-H., et al. Challenges in Representation Learning: A Report on Three Machine Learning Contests. In *Neural Network*, 2015.

Gronlund, A., Kamma, L., and Larsen, K. G. Near-Tight Margin-Based Generalization Bounds for Support Vector Machines. In *International Conference on Machine Learning*, 2020.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., and Girshick, R. Masked Autoencoders are Scalable Vision Learners. In *Computer Vision and Pattern Recognition*, 2022.

Helber, P., Bischke, B., Dengel, A., and Borth, D. Eurosat: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification. In *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

Huang, T., Chu, J., and Wei, F. Unsupervised Prompt Learning for Vision-Language Models. *arXiv preprint arXiv:2204.03649*, 2022.

Ji, K., Yang, J., and Liang, Y. Bilevel Optimization: Convergence Analysis and Enhanced Design. In *International Conference on Machine Learning*, 2021.

Ji, Z. and Telgarsky, M. The Implicit Bias of Gradient Descent on Nonseparable Data. In *Conference on Learning Theory*, 2019.

Johnson, J., Hariharan, B., Van Der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., and Girshick, R. Clevr: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In *Computer Vision and Pattern Recognition*, 2017.

Kiela, D., Firooz, H., Mohan, A., Goswami, V., Singh, A., Ringshia, P., and Testuggine, D. The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes. In *Advances in neural information processing systems*, 2020.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations*, 2015.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big Transfer (BiT): General Visual Representation Learning. In *European Conference on Computer Vision*, 2020.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. 3D Object Representations for Fine-grained Categorization. In *International Conference on Computer Vision Workshops*, 2013.

Krizhevsky, A. and Hinton, G. Learning Multiple Layers of Features from Tiny Images. *Technical Report, University of Toronto*, 2009.

Kuhn, H. W. The Hungarian Method for the Assignment Problem. In *Naval Research Logistics Quarterly*, 1955.

Kwon, J., Kwon, D., Wright, S., and Nowak, R. D. A Fully First-Order Method for Stochastic Bilevel Optimization. In *International Conference on Machine Learning*, 2023.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based Learning Applied to Document Recognition. In *Proceedings of the IEEE*, 1998.

Li, C., Yang, J., Zhang, P., Gao, M., Xiao, B., Dai, X., Yuan, L., and Gao, J. Efficient Self-supervised Vision Transformers for Representation Learning. In *International Conference on Learning Representations*, 2022.

Lin, W., Zhao, Z., Zhang, X., Wu, C., Zhang, Y., et al. PMC-CLIP: Contrastive Language-Image Pre-training using Biomedical Documents. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2023.

Liu, B., Ye, M., Wright, S., Stone, P., and Liu, Q. Bome! Bilevel Optimization Made Easy: A Simple First-Order Approach. In *Advances in Neural Information Processing Systems*, 2022.

Lorraine, J., Vicol, P., and Duvenaud, D. Optimizing Millions of Hyperparameters by Implicit Differentiation. In *International Conference on Artificial Intelligence and Statistics*, 2020.

MacQueen, J. B. Some Methods for Classification and Analysis of MultiVariate Observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, 1967.

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

Meidani, K., Shojaee, P., Reddy, C. K., and Farimani, A. B. SNIP: Bridging Mathematical Symbolic and Numeric Realms with Unified Pre-training. In *International Conference on Learning Representations*, 2024.

Nilsback, M.-E. and Zisserman, A. Automated Flower Classification over a Large Number of Classes. In *Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, 2008.

Niu, C., Shan, H., and Wang, G. SPICE: Semantic pseudo-labeling for image clustering. *IEEE Transactions on Image Processing*, 31:7264–7278, 2022.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al. DINOv2: Learning Robust Visual Features Without Supervision. In *Transactions on Machine Learning Research*, 2023.

Pan, S. J. and Yang, Q. A Survey on Transfer Learning. In *IEEE Transactions on Knowledge and Data Engineering*, 2009.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. Cats and Dogs. In *Computer Vision and Pattern Recognition*, 2012.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning Transferable Visual Models from Natural Language Supervision. In *International Conference on Machine Learning*, 2021.

Raschka, S., Patterson, J., and Nolet, C. Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. In *Information*. MDPI, 2020.

Robinson, L., Atkinson, T., Copoiu, L., Bordes, P., Pierrot, T., and Barrett, T. Contrasting Sequence with Structure: Pre-training Graph Representations with PLMs. In *Advances in Neural Information Processing Systems AI for Science Workshop*, 2023.

Salimans, T. and Kingma, D. P. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. In *Advances in Neural Information Processing Systems*, 2016.

Santurkar, S., Tsipras, D., and Madry, A. Breeds: Benchmarks for Subpopulation Shift. In *International Conference on Learning Representations*, 2021.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C. W., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., Schramowski, P., Kundurthy, S. R., Crowson, K., Schmidt, L., Kaczmarczyk, R., and Jitsev, J. LAION-5b: An Open Large-scale Dataset for Training Next Generation Image-text Models. In *Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

Scieur, D., Gidel, G., Bertrand, Q., and Pedregosa, F. The Curse of Unrolling: Rate of Differentiating Through Optimization. In *Advances in Neural Information Processing Systems*, 2022.

Shaban, A., Cheng, C.-A., Hatch, N., and Boots, B. Truncated Back-propagation for Bilevel Optimization. In *International Conference on Artificial Intelligence and Statistics*, 2019.

Singh, M., Gustafson, L., Adcock, A., de Freitas Reis, V., Gedik, B., Kosaraju, R. P., Mahajan, D., Girshick, R., Dollár, P., and Van Der Maaten, L. Revisiting Weakly Supervised Pre-training of Visual Perception Models. In *Computer Vision and Pattern Recognition*, 2022.

Soomro, K., Zamir, A. R., and Shah, M. UCF101: A Dataset of 101 Human Actions Classes from Videos in the Wild. *arXiv preprint arXiv:1212.0402*, 2012.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The Implicit Bias of Gradient Descent on Separable Data. In *Journal of Machine Learning Research*, 2018.

Stallkamp, J., Schlipsing, M., Salmen, J., and Igel, C. Man vs. Computer: Benchmarking Machine Learning Algorithms for Traffic Sign Recognition. In *Neural Networks*, 2012.

Tanwisuth, K., Zhang, S., Zheng, H., He, P., and Zhou, M. POUF: Prompt-oriented Unsupervised Fine-tuning for Large Pre-trained Models. In *International Conference on Learning Representations*, 2023.

Van Gansbeke, W., Vandenhende, S., Georgoulis, S., Proesmans, M., and Van Gool, L. SCAN: Learning to Classify Images without Labels. In *European Conference on Computer Vision*, 2020.

Vapnik, V. The Nature of Statistical Learning Theory. In *Springer*, 1995.

Veeling, B. S., Linmans, J., Winkens, J., Cohen, T., and Welling, M. Rotation Equivariant CNNs for Digital Pathology. In *Medical Image Computing and Computer Assisted Intervention*, 2018.

Vicol, P., Lorraine, J. P., Pedregosa, F., Duvenaud, D., and Grosse, R. B. On Implicit Bias in Overparameterized Bilevel Optimization. In *International Conference on Machine Learning*, 2022.

Wang, F., Zhao, B., and Zhang, C. Linear Time Maximum Margin Clustering. In *IEEE Transactions on Neural Networks*, 2010.

Wang, T. and Isola, P. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *International Conference on Machine Learning*, 2020.

Wang, Z., Liang, J., Sheng, L., He, R., Wang, Z., and Tan, T. A Hard-to-Beat Baseline for Training-free CLIP-based Adaptation. In *International Conference on Learning Representations*, 2024.

Xiao, J., Ehinger, K. A., Hays, J., Torralba, A., and Oliva, A. Sun Database: Exploring a Large Collection of Scene Categories. In *International Journal of Computer Vision*, 2016.

Xie, J., Girshick, R., and Farhadi, A. Unsupervised Deep Embedding for Clustering Analysis. In *International Conference on Machine Learning*, 2016.

Xu, L. and Schuurmans, D. Unsupervised and Semi-supervised Multi-class Support Vector Machines. In *AAAI Conference on Artificial Intelligence*, 2005.

Xu, L., Neufeld, J., Larson, B., and Schuurmans, D. Maximum Margin Clustering. In *Advances in Neural Information Processing Systems*, 2004.

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., and Wu, Y. CoCa: Contrastive Captioners are Image-text Foundation Models. In *Transactions on Machine Learning Research*, 2022.

Zhang, K., Tsang, I. W., and Kwok, J. T. Maximum Margin Clustering Made Practical. In *International Conference on Machine Learning*, 2007.

Zhao, B., Kwok, J., and Zhang, C. Multiple Kernel Clustering. In *International Conference on Data Mining*, 2009.

Zhou, J., Wei, C., Wang, H., Shen, W., Xie, C., Yuille, A., and Kong, T. iBOT: Image BERT Pre-training with Online Tokenizer. In *International Conference on Learning Representations*, 2022.

A. Proof of Proposition 3.1

Here, we first provide the simplified version of the main results from Soudry et al. (2018) for completeness and then present the proof of Proposition 3.1. For clarity, we overload notation for x_n and consider x_n is already represented in a representation space $\phi(x)$, *i.e.*, $x_n = \phi(x_n)$. Given binary labeling function $\tau(x) \in \{-1, +1\}$ of the dataset $\mathcal{D} = \{x_n\}_{n=1}^N$, let $\mathcal{L}(w)$ be the exponential loss function:

$$\mathcal{L}(w) = \sum_{n=1}^{N} \exp(-\tau(x_n) w^T x_n) \tag{14}$$

Assumption A.1. (*Linear separability*) The dataset $\mathcal{D}$ is linearly separable: $\exists w_* \in \mathbb{R}^d$ s.t. $\tau(x_n) w_*^T x_n > 0$ for all $x_n \in \mathcal{D}$.

We consider minimizing (14) using gradient descent with a step size η:

$$w_m = w_{m-1} - \eta \nabla_w \mathcal{L}(w_{m-1}) \tag{15}$$

Let w_{SVM} denote the *primal* solution to the hard margin SVM problem:

$$\begin{aligned} w_{\text{SVM}} = \min_{w} \quad & \|w\|_2^2 \\ \text{s.t.} \quad & \tau(x_n) w^T x_n \geq 1 \quad \forall x_n \in \mathcal{D}. \end{aligned} \tag{16}$$

Let α_{SVM} denote the *dual* solution to the hard margin SVM problem:

$$\begin{aligned} \alpha_{\text{SVM}} = \max_{\alpha} \quad & \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \tau(x_i) \tau(x_j) x_i^T x_j \\ \text{s.t.} \quad & \alpha_n \geq 0 \quad \forall n, \end{aligned} \tag{17}$$

where primal and dual variables are related as $w_{\text{SVM}} = \sum_{n=1}^{N} (\alpha_{\text{SVM}})_n \tau(x_n) x_n$ (Vapnik, 1995).

Assumption A.2. (*Non-degenerate dataset*) Support vectors $S = \{x_n \in \mathcal{D} | \tau(x_n) w_{\text{SVM}}^T x_n = 1\}$ span the data, *i.e.*, $\text{rank}(\mathcal{D}_S) = \text{rank}(\mathcal{D})$, where $\mathcal{D}_S$ is a matrix whose columns are $x_n \in S$. Furthermore, for each $x_n \in S$, the corresponding dual variables are strictly positive, *i.e.*, $(\alpha_{\text{SVM}})_n > 0$, and the rest are zero.

After above specifications, the simplified version of the seminal result by Soudry et al. (2018) is:

Proposition A.3. *(Implicit Bias of Gradient Descent, Soudry et al. (2018)) For almost any non-degenerate (Assumption A.2) dataset which is linearly separable (Assumption A.1), any starting point w_0 and step size $\eta < 1/\mathcal{L}(w_0)$, the gradient descent iterates (15) will behave as:*

$$w_m = w_{SVM} \log m + \tilde{w} + r_m, \tag{18}$$

where w_{SVM} is the max-margin vector (16), $\tilde{w}$ is a solution to:

$$\forall x_n \in S \ : \ \eta \exp(-\tau(x) \tilde{w}^T x_n) = (\alpha_{SVM})_n, \tag{19}$$

and the residual r_m is bounded with $\lim_{m \to \infty} \|r_m\|_2 = 0$

Equipped with this result, we analyze the generalization-based objective:

$$\mathcal{L}_M^{\text{binary}}(\theta) = \sum_{x \in \mathcal{D}} \exp(-\tau_\theta(x) w_M^T \phi(x)) \tag{20}$$

$$\text{s.t. } w_M = \Xi^{(M)}(w_0, \mathcal{D}), \tag{21}$$

where $\tau_\theta(x) = \sigma(\theta^T x)$ is the task encoder with an odd activation function such as $\tanh$ and $w_M = \Xi^{(M)}(w_0, \mathcal{D})$ denotes M steps of gradient descent with step size η and labeling defined by τ_θ. Without loss of generality, we can assume $\|x_n\|_2 \leq 1, \forall x_n \in \mathcal{D}$. Given the above specifications, we are now ready to state our main result:

Proposition A.4. *(Lower bound for the generalization-based objective) Following the assumptions of Proposition A.3, given $M \gg 1$ and $\theta \neq 0$, we have:*

$$\mathcal{L}_M^{binary}(\theta) \geq g(\theta)\|w_{SVM}(\theta)\|_2^2, \tag{22}$$

where $g(\theta) = (M\eta \exp(\|r_M(\theta)\|_2))^{-1}$, the residual $r_M(\theta)$ is bounded with $\lim_{M\to\infty} \|r_M(\theta)\|_2 = 0$, and $w_{SVM}(\theta)$ is the solution of the hard-margin SVM for a given θ:

$$\begin{aligned} w_{SVM}(\theta) = \min_{w} \quad & \|w\|_2^2 \\ \textit{s.t.} \quad & \tau_\theta(x_n) w^T x_n \geq 1 \quad \forall x_n \in \mathcal{D}. \end{aligned} \tag{23}$$

Proof. The key observation is that the task encoder $\tau_\theta(x)$ generates linearly separable labelings, allowing us to apply Proposition A.3 and substitute the explicit form of iterates w_M into the outer objective (20). Indeed, for example, $w_* = \theta$ satisfies Assumption A.1, *i.e.*, $\tau_\theta(x_n)\theta^T x_n > 0$ for all $x_n \in \mathcal{D}$ and $\theta \neq 0$. Thus, substituting the iterates w_M into the outer objective leads to:

$$\mathcal{L}_M^{\text{binary}}(\theta) = \sum_{n=1}^{N} \exp(-\tau_\theta(x_n) \left(w_{\text{SVM}}(\theta) \log M + \tilde{w}(\theta) + r_M(\theta)\right)^T x_n), \tag{24}$$

where we explicitly indicate that $w_{\text{SVM}}(\theta)$, $\tilde{w}(\theta)$ and $r_M(\theta)$ depend on the parameters θ of the task encoder τ_θ. Let $\mathcal{L}_n(\theta)$ be n-th term of the sum and S_θ be the indices of support vectors, *i.e.*, $n \in \{1, \dots, N\}$ s.t. $\tau_\theta(x_n) w_{\text{SVM}}^T x_n = 1$. Then, due to the non-negativity of $\exp(\cdot)$, we have:

$$\mathcal{L}_M^{\text{binary}}(\theta) \geq \sum_{n \in S_\theta} \mathcal{L}_n(\theta). \tag{25}$$

Considering single term $\mathcal{L}_n(\theta),\ n \in S_\theta$ and opening the brackets, we obtain:

$$\mathcal{L}_n(\theta) = \underbrace{\exp(-\log M \cdot \tau_\theta(x_n) w_{\text{SVM}}^T(\theta) x_n)}_{\mathcal{L}_1} \underbrace{\exp(-\tau_\theta(x_n)\tilde{w}(\theta)^T x_n)}_{\mathcal{L}_2} \underbrace{\exp(-\tau_\theta(x_n) r_M(\theta)^T x_n)}_{\mathcal{L}_3}. \tag{26}$$

Inspecting (26) separately for each term $\mathcal{L}_i$, we obtain: *(i)* $\mathcal{L}_1 = \frac{1}{M}$ since $n \in \mathcal{S}_\theta$; *(ii)* $\mathcal{L}_2 = \frac{(\alpha_{\text{SVM}}(\theta))_n}{\eta}$ by (19); and *(iii)* $\mathcal{L}_3 \geq \exp(-\|r_M(\theta)\|_2)$ by Cauchy–Schwarz inequality given that $\|\tau_\theta(x_n)x_n\|_2 \leq 1$. Combining this with (25), finally we obtain:

$$\mathcal{L}_M^{\text{binary}}(\theta) \geq (M\eta \exp(\|r_M(\theta)\|_2))^{-1} \sum_{n \in S_\theta} \alpha_n(\theta) = (M\eta \exp(\|r_M(\theta)\|_2))^{-1} \|w_{\text{SVM}}(\theta)\|_2^2, \tag{27}$$

where the last equality comes from the fact that:

$$\begin{aligned} \|w_{\text{SVM}}(\theta)\|_2^2 &= w_{\text{SVM}}(\theta)^T w_{\text{SVM}}(\theta) = w_{\text{SVM}}(\theta)^T \sum_{n \in S_\theta} (\alpha_{\text{SVM}}(\theta))_n \tau_\theta(x_n) x_n = \\ &= \sum_{n \in S_\theta} (\alpha_{\text{SVM}}(\theta))_n \tau_\theta(x_n) w_{\text{SVM}}(\theta)^T x_n = \sum_{n \in S_\theta} (\alpha_{\text{SVM}}(\theta))_n \cdot 1 = \sum_{n \in S_\theta} (\alpha_{\text{SVM}}(\theta))_n, \end{aligned}$$

concluding the proof. □

B. Experimental Details

B.1. Datasets

We evaluate our framework on 26 vision datasets studied in Radford et al. (2021). These datasets cover a wide range of vision tasks, including general object classification datasets CIFAR10 (Krizhevsky & Hinton, 2009), CIFAR100 (Krizhevsky & Hinton, 2009), STL10 (Coates et al., 2011), ImageNet (Deng et al., 2009), Caltech101 (Fei-Fei et al., 2004); fine-grained object classification datasets Food101 (Bossard et al., 2014), Flowers (Nilsback & Zisserman, 2008), Birdsnap (Berg et al., 2014), Stanford Cars (Krause et al., 2013), FGVC Aircraft (Maji et al., 2013), Oxford Pets (Parkhi et al., 2012); handwritten digits classification dataset MNIST (LeCun et al., 1998); texture classification dataset DTD (Cimpoi et al., 2014); scene classification dataset SUN397 (Xiao et al., 2016); the facial emotion recognition dataset FER2013 (Goodfellow et al., 2015); the satellite image classification datasets EuroSAT (Helber et al., 2019), RESISC45 (Cheng et al., 2017); the German Traffic Sign Recognition Benchmark (GTSRB) (Stallkamp et al., 2012); the KITTI Distance dataset (Geiger et al., 2012); the metastatic tissue classification dataset PatchCamelyon (PCam) (Veeling et al., 2018); action recognition datasets UCF101 (Soomro et al., 2012), Kinetics700 (Carreira et al., 2019); the CLEVR counting dataset (Johnson et al., 2017); the Hateful Memes dataset (Kiela et al., 2020); the country classification dataset Country211 (Radford et al., 2021) and the Rendered SST2 dataset (Radford et al., 2021). For CLEVR, we take 2000 random samples as training split and 500 random samples as test split. For two video datasets UCF101 and Kinetics700, we take the middle frame of each video clip as the input of the pre-trained models. Details of each dataset are provided in Table B1. We use *accuracy* as the evaluation metric for all the datasets. Finally, it's worth noting that TURTLE could also be applied to the tasks in various modalities besides vision or even in cross-modalities scenarios, provided that the pre-trained representations are available.

Table B1. **Benchmark suite of 26 datasets.** We use *accuracy* as the evaluation metric for all datasets.

Dataset	Number of Classes	Train size	Test size
Food101 (Bossard et al., 2014)	101	75,750	25,250
CIFAR10 (Krizhevsky & Hinton, 2009)	10	50,000	10,000
CIFAR100 (Krizhevsky & Hinton, 2009)	100	50,000	10,000
Birdsnap (Berg et al., 2014)	500	37,221	2,500
SUN397 (Xiao et al., 2016)	397	19,850	19,850
StanfordCars (Krause et al., 2013)	196	8,144	8,041
FGVC Aircraft (Maji et al., 2013)	100	6,667	3,333
DTD (Cimpoi et al., 2014)	47	3,760	1,880
OxfordPets (Parkhi et al., 2012)	37	3,680	3,669
Caltech101 (Fei-Fei et al., 2004)	102	3,060	6,084
Flowers (Nilsback & Zisserman, 2008)	102	2,040	6,149
MNIST (LeCun et al., 1998)	10	60,000	10,000
FER2013 (Goodfellow et al., 2015)	7	28,709	3,589
STL10 (Coates et al., 2011)	10	5,000	8,000
EuroSAT (Helber et al., 2019)	10	10,000	5,000
RESISC45 (Cheng et al., 2017)	45	25,200	6,300
GTSRB (Stallkamp et al., 2012)	43	26,640	12,630
KITTI Distance (Geiger et al., 2012)	4	5,985	1,496
Country211 (Radford et al., 2021)	211	42,200	21,100
PatchCamelyon (Veeling et al., 2018)	2	294,912	32,768
UCF101 (Soomro et al., 2012)	101	9,537	3,783
Kinetics700 (Carreira et al., 2019)	700	536,485	33,966
CLEVR Counts (Johnson et al., 2017)	8	2,000	500
HatefulMemes (Kiela et al., 2020)	2	8,500	500
The Rendered SST2 (Radford et al., 2021)	2	7,792	1,821
ImageNet (Deng et al., 2009)	1000	1,281,167	50,000

B.2. Representations

TURTLE is compatible with any pre-trained representations. This paper presents the comprehensive evaluation of TURTLE on a wide range of representation spaces that vary on the pre-training datasets, model architectures and training objectives. Specifically, we consider CLIP ResNets (RN50, RN101, RN50x4, RN50x16, RN50x64) and CLIP Vision Transformers (ViT-B/32, ViT-B/16, ViT-L/14) pre-trained on WebImageText-400M (Radford et al., 2021) for training TURTLE 1-space. These models are pre-trained on the same dataset, scaling with number of the parameters. For TURTLE 2-spaces, we incorporate DINOv2 ViT-g/14 pre-trained on LVD-142M (Oquab et al., 2023) as the second representation space. Moreover, we also include SWAG ViT-H/14 pre-trained on IG-3.6B (Singh et al., 2022), CoCa ViT-L/14 (Yu et al., 2022) pre-trained on LAION-2B (Schuhmann et al., 2022) [2], OpenCLIP ViT-L/14 pre-trained on LAION-2B (Cherti et al., 2023), MOCOv3 ViT-B/16 pre-trained on ImageNet-1000 (Chen et al., 2021) and EsViT Swim-B pre-trained on ImageNet-1000 (Li et al., 2022) to study whether incorporating stronger representations on the given dataset may lead to the increased performance of unsupervised transfer with TURTLE. For all the models, we precompute the representations with standard image preprocessing pipelines and do not use any data augmentations during training of TURTLE. The details of pre-trained representations are provided on Table B2.

Table B2. **Representation spaces used in TURTLE.** "Weakly Supervised" indicates that the model is pre-trained with text supervision, such as caption or hashtag of the image.

Model	Architecture	Parameters	Trained on	Weakly Supervised
CLIP (Radford et al., 2021)	RN50	100M	WebImageText-400M	✓
	RN101	120M		
	RN50x4	180M		
	RN50x16	290M		
	RN50x64	620M		
	ViT-B/32	150M		
	ViT-B/16	150M		
	ViT-L/14	430M		
OpenCLIP (Cherti et al., 2023)	ViT-L/14	430M	LAION-2B	✓
SWAG (Singh et al., 2022)	ViT-H/14	630M	IG-3.6B	✓
CoCa (Yu et al., 2022)	ViT-L/14	640M	LAION-2B	✓
MOCOv3 (Chen et al., 2021)	ViT-B/16	86M	ImageNet-1000	✗
EsViT (Li et al., 2022)	Swin-B	88M	ImageNet-1000	✗
DINOv2 (Oquab et al., 2023)	ViT-g/14	1.1B	LVD-142M	✗

[2]Since the original paper does not release the models, we use the reproduced version from the OpenCLIP project, which could be found at https://github.com/mlfoundations/open_clip.

B.3. Implementation Details

Efficient alternating optimization. TURTLE contains a bilevel objective that measures the loss of the task encoder using the training loss of a linear classifier trained on the task produced by the task encoder. The hyper-gradient of the task encoder is $\nabla_\theta \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \theta} + (\frac{\partial w}{\partial \theta})^T \frac{\partial \mathcal{L}}{\partial w}|_{w=w^*}$, where the Jacobian $\frac{\partial w}{\partial \theta}$ is generally expensive to obtain. Existing works usually estimate the hyper-gradient via unrolling or approximation based on the implicit function theorem, e.g., see Finn et al. (2017); Lorraine et al. (2020); Ji et al. (2021); Kwon et al. (2023); Dagréou et al. (2022); Bolte et al. (2023); Liu et al. (2022). However, these methods might be inefficient and suboptimal in practice (Scieur et al., 2022). Fortunately, in the TURTLE framework, one could avoid the estimation of $(\frac{\partial w}{\partial \theta})^T \frac{\partial \mathcal{L}}{\partial w}$ given the fact that $\frac{\partial \mathcal{L}}{\partial w}|_{w=w^*} \approx 0$. Thus, the gradient of the task encoder is simplified to $\nabla_\theta \mathcal{L} = \frac{\partial \mathcal{L}}{\partial \theta}|_{w=w^*}$. This inspires us to train the task encoder via *alternating optimization*, which has been shown efficient for the min-min optimization problems (Ablin et al., 2020). At each iteration, we first fix the task encoder and train the linear classifier for M steps to find its approximate optima. Note that one could choose to re-initialize the linear classifier every time (*cold-start*), or just start from the values of last iteration (*warm-start*), which might introduce different implicit bias as noted by Vicol et al. (2022). After that, we update the task encoder based on the loss of the linear classifier. The training is efficient since no second-order gradient is needed in this process. The pseudo-code of TURTLE is provided in Algorithm B1.

Algorithm B1 TURTLE for Unsupervised Transfer

Input: Dataset $\mathcal{D}$, number of classes C, number of iterations T, representation spaces $\phi_1(\cdot), ..., \phi_K(\cdot)$, task parameters $\theta = \{\theta_1, ..., \theta_K\}$, linear classifiers $w^1, ..., w^K$, learning rate η, optimization operator $\Xi(\cdot)$, number of adaptation steps M, entropy regularization weight γ // $\Xi(\cdot)$ can be any iterative operator, e.g., gradient descent
Randomly initialize $\theta_1, ..., \theta_K$ and $w_0^1, ..., w_0^K$
for $t = 1$ to T **do**
  Sample mini-batch from dataset $X \sim \mathcal{D}$
  Generate task from task encoder $\tau_\theta \leftarrow \frac{1}{K} \sum_{k=1}^{K} \sigma(\theta_k \phi_k(X))$
  Update linear classifiers for M steps $\forall k \in [K] : w_M^k \leftarrow \Xi^{(M)}(w_0^k, X)$
  Update task parameters $\forall k \in [K] : \theta_k \leftarrow \theta_k - \eta \frac{\partial}{\partial \theta_k} \left[\mathcal{L}_{ce}(w_M^k \phi_k(X), \tau_\theta) + \gamma \mathcal{R}(\overline{\tau}_\theta) \right]$ // partial derivative $\frac{\partial}{\partial \theta_k}$
  if warm-start **then** update start points $\forall k \in [K], w_0^k \leftarrow w_M^k$ // cold-start keeps the initial w_0^k
end for
Output: Task parameters $\theta = \{\theta_1, ..., \theta_K\}$

Training details. We precompute the feature representations for all datasets before the training. We use Weight Normalization (Salimans & Kingma, 2016) to parameterize the task encoder since we found it helpful for the convergence. ADAM (Kingma & Ba, 2015) optimizer is used for the training of both linear classifier and task encoder. We use 10000 as the default batch-size. For datasets smaller than 10000, we train the model with full-batch at each iteration. Overall, we found TURTLE is robust to the choice of the batch-size. We update the linear classifier for $M = 10$ steps at each iteration and train the task encoder for $T = 6000$ iterations in total. If not specifically mentioned, we set the entropy regularization parameter $\gamma = 10$ for all experiments. We show robustness of TURTLE to this hyperparameter in Appendix G. For each dataset, we do a grid search over 5 different learning rates for both task encoder and linear classifier with $\eta \in \{0.01, 0.005, 0.001, 0.0005, 0.0001\}$, respectively. We combine each pair of learning rates with the choice of warm-start or cold-start, and finally get set of 50 triplets to search over for each dataset. We use *cross-validation* to select hyper-parameters, as described below. Following Gadetsky & Brbić (2023); Van Gansbeke et al. (2020), we use Hungarian algorithm (Kuhn, 1955) to match the labeling found by TURTLE and the ground truth labeling to compute the clustering accuracy. If not specified, we train our model on the training split and report the clustering accuracy on the test split. In Section H, we also consider the setting of both training and evaluating TURTLE on the test split, mimicking low data regime.

Cross-validation for task selection. For each dataset, we obtain 50 tasks after grid search, *i.e.*, each corresponds to the hyperparameter triplet. We use 10-fold cross-validation to select the best task. The cross-validation regards the learned task as "pseudo-labels" and measures the *generalization error* of a linear classifier trained on these "pseudo-labels". Specifically, we randomly split the dataset into 10 folds. In each round, a linear classifier is trained on 9 folds and tested on the rest fold. The final cross-validation score is the average test accuracy over all rounds. Importantly, this process relies solely on the learned tasks and does *not* need any information about the ground-truth labels. For TURTLE trained on multiple representations, we do cross-validation on each representation space separately and average the final scores. The task with the highest cross-validation score is selected as the final output of TURTLE. Figure B1 shows the performance of the learned

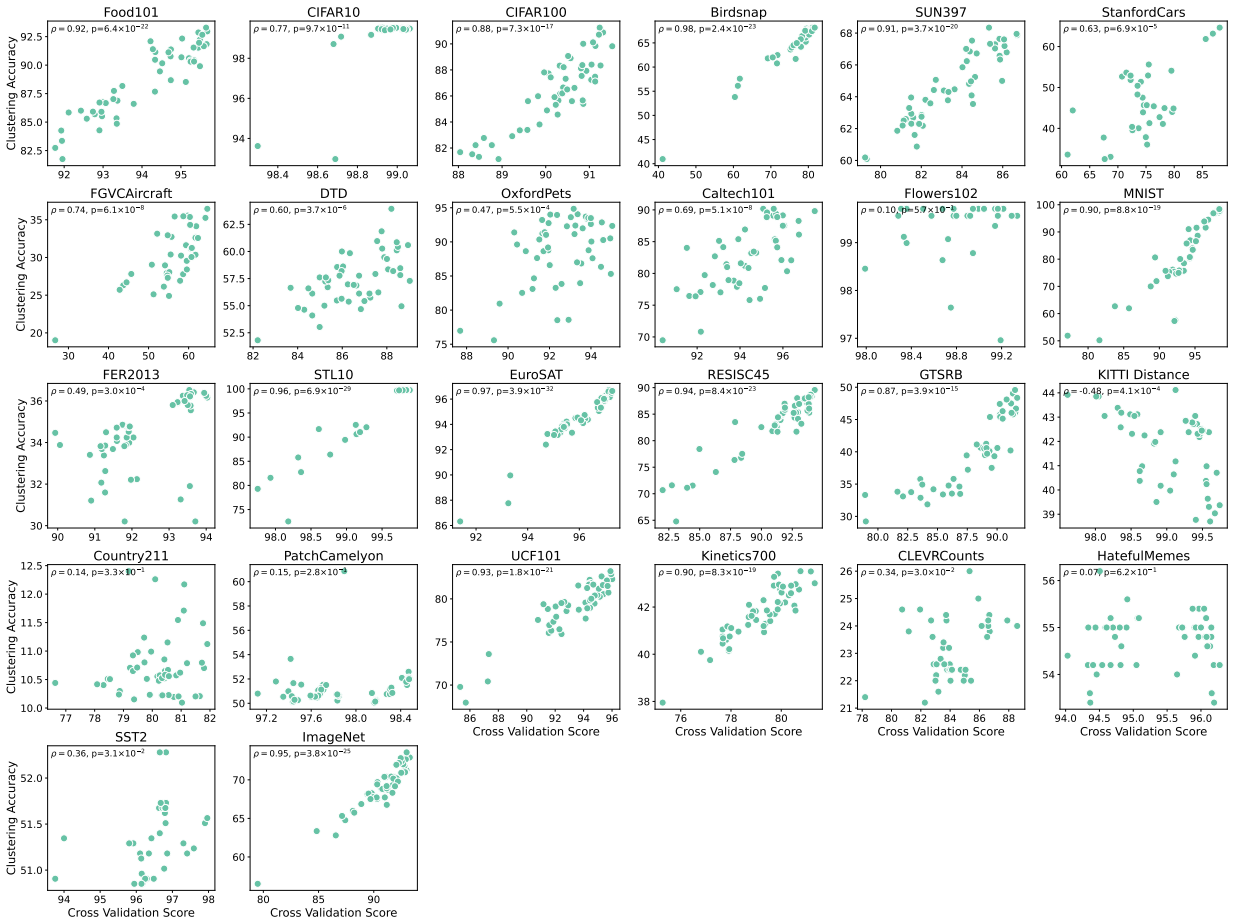

Figure B1. **Task selection via cross-validation**. We use TURTLE 2-spaces CLIP ViT-L/14 and DINOv2 to produce the tasks. We show the cross-validation score and corresponding clustering accuracy of the tasks learned by TURTLE with 50 different hyperparameters for each dataset. The cross-validation score is well correlated with the clustering accuracy ($\rho = 0.61$ of two-sided Pearson correlation coefficient averaged over 26 datasets).

tasks obtained by TURTLE 2-spaces CLIP ViT-L/14 and DINOv2 and their corresponding cross-validation scores over 26 datasets. As indicated by the plot, the cross-validation score is well correlated with the clustering accuracy with an average of $\rho = 0.61$ two-sided Pearson correlation coefficient over 26 datasets. Moreover, among 20 datasets, cross-validation successfully identifies the best or near-best task (*i.e.*, with less than 1.5 point difference of clustering accuracy). The result of cross-validation also empirically verifies the effectiveness of the generalization-based objective and suggests that the labelings with low generalization error tend to be more aligned with human labeled tasks, confirming the original findings of Gadetsky & Brbić (2023) on the wide suite of datasets.

Linear probe. Supervised linear probe is a widely used method to evaluate the quality of representation learning (Radford et al., 2021; Oquab et al., 2023). It trains a linear classifier on the train split on top of the representations extracted from the pre-trained models and then evaluates the performance on the test split. We use the `cuML.LogisticRegression` (Raschka et al., 2020) for linear probe evaluation in our paper [3]. The linear classifier is trained with L-BFGS optimizer for maximum of 1000 iterations. The `cuML` library allows for GPU acceleration and, thus, it is much faster than `sklearn.linear_model.LogisticRegression` counterpart, especially on large datasets such as ImageNet. To determine the strength of L2 norm regularization, we randomly take 20% of the training split for validation and search over 96 values in the log-space ranging from 10^{-6} to 10^{6}. The selection process takes a few minutes on small datasets, and around 8 hours on ImageNet, with a single NVIDIA A100 GPU. After that, we train the model with the best regularization strength on the entire training split and report the classification accuracy on the test split.

[3]This library is available at https://github.com/rapidsai/cuml.

C. Details on Unsupervised Baselines and Numerical Results

K-Means clustering. We apply K-Means (MacQueen, 1967) clustering on top of pre-trained features as a simple baseline that does not require task-specific representation learning. Similarly to the linear probe, we also use the implementation from `CuML` library for the GPU acceleration (i.e., `CuML.KMeans`). For each dataset and the corresponding representation, we train K-Means with maximum 1000 iterations (`max_iter=1000`) and 10 random initializations (`n_init=10`) on the train split, and report the clustering accuracy on the test split. In the case when multiple representations are used, we first L2 normalize representation from each pre-trained model, and then apply K-Means clustering on top of the concatenation of all L2 normalized features.

HUME. HUME (Gadetsky & Brbić, 2023) is the recent state-of-the-art unsupervised learning baseline that introduced the instantiation of the generalization-based objective (3). Specifically, it learns task-specific representations on the target dataset to model the task encoder, and then measures the generalization error of a linear classifier in the representation space of a foundation model. We use the original source code [4] for the implementation of HUME, with modifications to improve the speed and performance. In particular, we replace task-specific representations with a pre-trained foundation model since we empirically found it yields better performance. Besides, we remove the variance reduction used in the original HUME and sample only the single mini-batch at every iteration (*i.e.*, the same as TURTLE), since we found it significantly reduces the computational cost and does not influence the final performance. We update the linear classifier with $M = 300$ steps at each iteration, and train the task encoder with $T = 6000$ iterations in total. The default batch-size is set to 10000. Moreover, we follow the same hyperparameter selection procedure of TURTLE to select the inner/outer learning rates and warm-start/cold-start for HUME.

Comparison of TURTLE to HUME and K-Means. For a fair comparison, we train TURTLE, HUME and K-Means using the same representation spaces, i.e., CLIP ViT-L/14 and DINOv2 ViT-g/14. Given that HUME's task encoder parametrization uses only the single space, we run HUME with the task encoder modeled using CLIP or DINOv2 (denoted as HUME *CLIP* and HUME *DINOv2* respectively), and measure the generalization error using the rest representation space. For each method, we report the training time in minutes and the clustering accuracy averaged over 3 random seeds. For each random seed, we perform the hyperparameter selection for HUME and TURTLE as described in the corresponding subsection above. Table C1 and Table C2 show the obtained results on 5 datasets. Overall, the results indicate that TURTLE outperforms HUME and K-Means on all the considered datasets, highlighting the effectiveness of design choices made in TURTLE. For instance, combining multiple representation spaces for modeling the task encoder in TURTLE brings substantial gains compared to HUME. Namely, TURTLE achieves remarkable 28% and 23% absolute improvement (40% and 30% relative improvement) over HUME *DINOv2* and HUME *CLIP* respectively on the MNIST dataset. Furthermore, efficient first-order optimization techniques used in TURTLE allow for fast optimization, taking just 5 minutes even on large-scale datasets such as ImageNet.

Table C1. **Accuracy of TURTLE and unsupervised baselines.** The results are averaged with standard deviations computed over 3 runs.

Method	MNIST	CIFAR100	Food101	Birdsnap	ImageNet
K-Means	68.9 ± 0.6	75.1 ± 0.5	78.0 ± 0.7	54.0 ± 0.8	64.8 ± 0.3
HUME *CLIP*	75.3 ± 5.0	71.2 ± 2.2	86.5 ± 1.3	45.8 ± 1.8	65.2 ± 0.9
HUME *DINOv2*	69.7 ± 5.9	83.9 ± 1.2	85.3 ± 1.7	57.3 ± 0.7	68.1 ± 0.2
TURTLE	$\mathbf{98.0 \pm 0.4}$	$\mathbf{89.1 \pm 1.0}$	$\mathbf{92.8 \pm 0.5}$	$\mathbf{67.8 \pm 0.4}$	$\mathbf{72.4 \pm 0.3}$

Table C2. **Training time (in minutes) of TURTLE and unsupervised baselines.** The results are averaged with standard deviations computed over 3 runs. The standard deviation for K-Means and TURTLE is negligible.

Method	MNIST	CIFAR100	Food101	Birdsnap	ImageNet
K-Means	0.02 ± 0.0	0.04 ± 0.0	0.1 ± 0.0	0.1 ± 0.0	6.4 ± 0.0
HUME *CLIP*	15.2 ± 2.4	44.1 ± 0.5	45.5 ± 1.3	156.8 ± 1.2	285.4 ± 4.1
HUME *DINOv2*	11.1 ± 0.3	31.7 ± 0.4	31.0 ± 0.3	96.7 ± 0.6	185.6 ± 7.9
TURTLE	1.6 ± 0.0	1.6 ± 0.0	1.7 ± 0.0	2.1 ± 0.0	4.8 ± 0.0

[4] Code could be found at https://github.com/mlbio-epfl/hume.

D. Complete List of Individual Numerical Results

Table D1. **Complete list of numerical results.** Results of supervised linear probe, CLIP zero-shot transfer, K-Means clustering and TURTLE unsupervised transfer on 26 datasets and 9 representation spaces (CLIP RN50, RN101, RN50x4, RN50x16, RN50x64, ViT-B/32, ViT-B/16, ViT-L/14 and DINOv2 ViT-g/14). The results for K-Means 2-spaces and TURTLE 2-spaces are obtained using DINOv2 and the corresponding CLIP model.

		Food101	CIFAR10	CIFAR100	Birdsnap	SUN397	Stanford Cars	FGVC Aircraft	DTD	Oxford Pets	Caltech101	Flowers102	MNIST	FER2013	STL10	EuroSAT	RESISC45	GTSRB	KITTI	Country211	PCam	UCF101	Kinetics700	CLEVR	HatefulMemes	Rendered SST2	ImageNet	Average
Linear Probe	DINOv2	94.9	99.5	93.9	88.9	78.7	91.3	87.8	85.0	96.6	93.8	99.7	98.7	66.8	99.6	97.3	95.3	80.1	76.9	26.1	87.5	91.5	59.2	52.0	52.4	56.1	86.2	82.1
	RN50	86.4	88.8	70.4	57.4	73.3	77.9	49.2	76.4	88.4	91.6	95.3	98.3	63.4	97.3	94.2	91.0	84.5	72.9	25.5	82.5	80.6	44.8	51.6	58.6	72.2	73.1	74.8
	RN101	88.9	91.1	73.5	61.2	75.0	84.0	52.6	76.4	90.8	93.6	95.2	98.5	65.2	98.2	94.5	92.2	84.3	71.7	26.6	82.8	82.7	48.0	52.0	55.4	73.9	75.9	76.3
	RN50x4	91.3	90.5	73.2	66.5	76.7	86.0	57.3	79.6	92.0	94.5	97.2	98.6	67.3	98.1	94.4	92.9	86.4	73.9	30.2	83.1	84.2	50.2	51.8	60.0	76.1	78.1	78.1
	RN50x16	93.3	92.1	75.0	72.5	79.1	88.9	63.4	79.6	93.6	94.9	97.6	98.9	68.5	98.8	95.3	94.0	89.1	74.5	34.9	83.6	86.2	53.8	49.0	59.4	79.0	81.4	79.9
	RN50x64	94.7	94.1	78.5	77.9	81.0	90.6	68.4	81.9	94.3	96.6	98.7	99.0	71.2	99.1	95.7	94.8	91.7	75.5	40.7	83.7	89.3	57.5	54.8	58.2	81.4	83.7	82.0
	ViT-B/32	88.6	95.1	80.2	59.9	76.1	80.9	50.4	76.5	89.4	94.2	96.1	98.9	66.3	98.6	95.2	92.5	86.7	71.4	27.3	83.1	83.6	48.6	50.8	57.0	70.4	75.7	76.7
	ViT-B/16	92.7	95.9	82.5	69.3	78.3	86.4	58.8	79.0	93.1	91.2	97.7	98.9	68.0	99.1	95.6	93.9	88.3	73.0	32.4	83.2	86.7	53.6	55.4	58.8	75.1	79.8	79.5
	ViT-L/14	95.2	98.1	87.4	76.6	81.5	90.7	68.4	81.9	95.2	96.3	99.1	99.0	71.1	99.8	97.0	95.5	93.2	72.7	41.8	85.3	90.6	60.4	57.4	62.0	81.6	84.4	83.1
CLIP Zero-Shot	RN50	80.6	71.5	42.0	34.5	59.8	54.3	16.6	41.2	85.8	76.7	66.2	58.1	39.3	94.2	40.3	53.5	35.1	10.8	15.4	61.4	63.1	30.8	20.8	56.2	55.6	59.8	50.9
	RN101	83.6	80.8	48.8	35.8	59.2	61.1	18.6	43.5	86.9	77.5	65.4	50.9	44.0	96.8	30.8	58.7	37.6	33.4	16.9	58.2	61.2	33.2	24.8	52.8	64.2	62.3	53.4
	RN50x4	86.9	79.4	49.8	39.3	62.6	67.0	20.5	48.5	88.8	78.3	69.9	48.7	48.3	96.6	31.5	60.5	36.2	31.6	20.4	53.5	64.6	35.5	19.4	51.2	66.8	66.2	54.7
	RN50x16	90.6	81.4	52.6	45.8	65.0	73.3	27.2	53.2	90.1	81.0	72.0	67.8	55.7	97.8	41.9	64.7	39.8	31.3	24.4	62.3	68.7	39.9	20.2	51.4	67.6	70.7	59.1
	RN50x64	92.1	85.1	60.9	50.6	67.1	75.9	30.0	53.4	93.7	83.0	76.0	85.7	60.8	98.3	57.9	70.8	48.0	36.2	29.8	55.5	73.0	43.8	25.2	59.6	70.7	73.9	63.7
	ViT-B/32	83.9	89.9	65.1	37.1	62.9	59.7	19.4	43.8	87.3	81.6	66.5	48.6	48.0	97.1	44.2	61.0	32.7	29.6	17.2	60.7	64.1	35.1	23.2	52.0	58.5	63.4	55.1
	ViT-B/16	88.8	90.8	68.3	42.1	65.2	64.7	24.0	45.0	89.1	81.0	71.5	51.9	52.9	98.2	54.1	65.6	43.5	21.7	22.9	53.9	68.9	39.4	23.6	54.4	60.6	68.4	58.1
	ViT-L/14	93.1	95.6	78.2	49.4	68.0	77.8	31.6	55.3	93.6	83.3	79.1	76.2	56.3	99.4	61.2	70.9	50.6	24.9	31.9	51.2	75.4	47.7	20.8	55.0	68.9	75.5	64.3
K-Means 1-space	DINOv2	72.3	66.7	70.3	47.1	57.4	25.4	20.3	50.1	78.0	78.5	98.7	46.8	33.7	40.8	64.7	66.6	23.8	50.1	8.9	59.8	68.7	34.4	19.8	50.4	52.2	62.3	51.8
	RN50	47.2	54.9	25.9	29.8	42.3	31.8	20.6	41.2	39.9	67.3	69.2	51.2	25.5	89.8	54.4	55.2	27.7	47.3	8.8	64.9	52.1	17.4	27.4	58.0	53.9	33.3	43.7
	RN101	54.1	69.3	34.9	34.0	46.9	40.0	22.9	43.9	49.5	70.9	70.5	48.9	26.7	96.9	51.8	61.6	26.4	48.6	9.4	64.2	59.7	21.2	27.6	57.2	55.0	39.4	47.4
	RN50x4	62.6	60.5	30.6	38.2	49.1	42.7	25.4	47.3	53.5	75.4	76.5	45.5	27.6	83.8	52.8	61.4	25.3	48.1	10.4	61.0	59.7	21.8	26.4	57.6	55.8	41.9	47.7
	RN50x16	70.2	66.4	34.7	42.8	49.0	47.3	30.0	49.2	59.3	72.2	78.5	59.2	25.6	94.7	59.4	69.5	35.4	49.1	11.0	64.3	65.0	24.0	23.4	56.6	55.1	45.8	51.4
	RN50x64	75.9	67.9	36.8	46.2	51.7	49.8	33.9	49.5	56.0	78.5	82.5	67.0	26.4	94.7	60.4	73.2	35.8	49.0	11.2	63.3	67.2	25.4	23.6	57.8	55.0	50.9	53.4
	ViT-B/32	58.7	75.8	40.9	31.8	50.4	35.1	22.1	43.7	42.6	77.2	76.3	57.5	26.0	94.5	63.3	66.0	32.7	48.4	9.4	63.2	61.3	22.4	26.6	57.6	54.6	37.9	49.1
	ViT-B/16	72.2	78.2	46.9	38.5	52.4	42.5	27.9	47.3	49.7	78.8	81.6	54.9	27.7	94.9	73.2	72.8	34.6	50.1	10.8	63.2	64.1	25.3	24.8	59.2	55.2	43.1	52.7
	ViT-L/14	82.5	83.5	51.9	46.1	55.4	52.2	32.9	50.5	72.2	82.4	89.5	66.6	31.2	86.4	71.7	73.7	45.9	48.8	12.3	63.3	74.0	32.5	26.4	60.8	51.7	51.8	57.5
K-Means 2-spaces	RN50	71.5	67.5	69.4	46.5	56.9	25.4	20.2	50.1	78.0	80.2	99.1	46.8	33.7	49.4	64.7	66.7	23.6	50.1	9.1	59.8	70.8	34.5	19.4	50.4	52.2	64.2	52.3
	RN101	70.1	67.5	69.9	48.0	56.7	24.6	20.1	50.1	78.0	78.5	97.6	46.8	33.7	40.8	64.6	65.1	24.4	50.1	9.1	59.8	69.3	34.7	19.4	50.4	52.2	62.4	51.7
	RN50x4	70.7	67.5	70.9	47.4	57.1	24.8	19.6	50.2	78.0	80.2	98.6	46.8	33.7	49.5	64.7	65.4	23.6	50.1	9.1	59.8	71.0	34.3	19.8	50.4	52.2	62.0	52.2
	RN50x16	72.2	67.5	72.6	46.8	56.8	25.1	19.4	50.2	77.1	79.8	99.5	46.9	33.7	49.2	64.6	66.7	23.2	50.1	9.2	59.8	72.1	34.6	19.4	50.4	52.2	63.1	52.4
	RN50x64	71.7	63.7	69.9	47.4	56.5	25.4	20.1	48.8	78.5	78.9	98.0	46.9	33.6	50.2	64.7	62.0	24.4	50.1	9.2	59.8	69.2	34.5	19.4	50.4	52.2	62.6	51.8
	ViT-B/32	69.7	76.4	69.3	48.4	57.5	24.8	19.9	50.1	80.1	76.0	98.5	46.9	33.6	55.0	64.8	65.3	23.4	50.0	9.1	59.8	67.8	34.0	20.4	50.4	52.3	62.5	52.5
	ViT-B/16	71.5	76.4	71.8	48.3	55.5	26.2	20.7	47.2	79.0	79.0	97.1	46.9	33.7	55.5	64.9	63.0	24.4	50.1	9.3	59.8	71.7	34.5	20.6	50.6	52.2	62.6	52.8
	ViT-L/14	78.8	85.9	75.1	53.0	60.6	35.6	21.9	49.0	79.3	87.4	95.6	69.0	33.1	84.8	84.7	73.9	31.3	50.5	9.9	61.7	73.3	37.7	23.0	50.6	52.1	64.9	58.6
TURTLE 1-space	DINOv2	78.9	99.3	87.1	66.7	60.3	31.2	23.5	55.2	82.2	81.4	99.0	69.0	32.5	72.3	93.8	73.5	23.4	41.6	9.0	50.7	74.5	35.1	22.4	52.6	51.6	69.1	59.1
	RN50	65.0	57.2	30.8	34.1	50.0	37.0	23.0	48.5	51.2	75.7	82.7	62.1	29.5	96.6	73.3	67.9	34.3	41.4	8.9	68.8	61.4	21.6	26.6	57.0	53.5	41.1	50.0
	RN101	74.9	71.6	40.2	38.2	54.6	46.3	24.2	52.4	66.7	83.1	88.1	56.6	26.4	98.0	69.1	74.2	36.4	39.6	9.5	71.0	65.5	25.1	26.4	57.8	55.0	48.3	53.8
	RN50x4	80.7	67.7	37.7	42.4	57.9	52.9	26.7	56.4	74.3	83.6	91.6	57.9	26.7	97.8	68.4	77.3	37.6	39.6	10.1	54.7	65.2	26.3	25.6	58.6	55.2	52.2	54.8
	RN50x16	87.0	75.4	38.7	47.5	58.5	58.7	30.3	55.3	82.0	85.4	91.9	63.2	28.1	98.6	81.2	84.9	43.5	38.7	11.4	50.8	70.1	28.6	26.2	57.2	53.5	56.4	57.8
	RN50x64	88.3	77.0	42.9	53.7	61.6	64.1	34.0	54.0	85.0	83.7	95.3	78.4	27.9	99.1	80.7	83.4	43.8	41.0	12.3	50.6	73.8	30.5	26.4	58.0	54.2	60.4	60.1
	ViT-B/32	72.4	85.9	45.8	35.2	58.1	42.3	24.0	49.1	62.1	81.5	85.2	80.9	30.3	98.3	69.8	82.2	39.6	40.7	9.8	50.5	67.4	25.4	24.6	57.6	55.2	46.8	54.6
	ViT-B/16	82.4	94.1	54.4	43.1	58.8	52.8	28.6	54.7	71.1	83.6	94.2	73.0	29.1	99.1	82.5	85.6	36.3	39.4	11.1	51.0	71.3	29.2	24.0	58.0	54.1	53.3	58.3
	ViT-L/14	93.1	97.6	68.7	54.2	63.4	66.6	35.9	57.6	90.5	84.3	98.1	66.3	31.0	99.8	93.8	87.6	50.3	39.1	13.7	50.3	78.3	36.8	27.4	59.0	54.6	62.5	63.9
TURTLE 2-spaces	RN50	84.1	96.8	83.0	67.2	65.7	32.6	25.3	57.0	90.9	88.6	99.7	90.8	34.0	99.7	95.5	85.4	32.4	44.2	9.4	50.6	77.7	35.6	25.0	56.0	50.0	66.3	63.2
	RN101	86.5	98.6	81.1	63.2	64.8	32.8	25.1	61.1	90.7	89.1	99.7	97.1	33.8	99.7	94.8	86.1	36.6	39.0	9.6	50.3	79.0	37.4	24.6	56.2	51.3	67.9	63.7
	RN50x4	87.6	98.5	79.4	66.3	65.3	47.1	24.3	62.9	93.0	88.5	99.7	97.1	33.9	99.7	95.4	83.6	36.4	41.5	10.1	51.1	77.9	38.0	25.4	54.8	51.3	69.8	64.6
	RN50x16	90.2	98.7	80.7	62.9	65.6	50.5	25.5	55.9	92.3	88.9	99.7	97.5	32.7	99.7	94.8	86.4	43.4	41.0	10.6	50.5	77.4	39.9	24.4	55.6	50.0	70.8	64.8
	RN50x64	89.9	99.0	85.3	68.0	66.6	58.8	31.7	61.2	93.3	88.8	99.7	97.4	34.7	99.7	95.1	89.8	41.5	44.8	11.2	50.2	80.7	40.4	25.4	51.0	50.5	72.1	66.4
	ViT-B/32	86.4	99.1	84.6	64.0	65.2	46.0	27.7	56.0	92.8	89.0	99.7	97.2	34.2	99.7	95.6	88.0	38.3	42.0	10.0	52.8	75.6	37.6	23.2	52.6	51.2	68.6	64.5
	ViT-B/16	90.2	99.2	89.0	66.6	65.0	49.8	25.4	57.3	93.0	89.1	99.7	98.0	33.6	99.8	96.0	89.0	35.0	41.4	10.5	52.1	79.5	39.7	25.6	52.2	51.6	70.2	65.3
	ViT-L/14	92.2	99.5	89.9	68.1	67.9	64.6	36.5	57.3	92.3	89.8	99.6	97.8	36.2	99.7	96.6	89.6	48.4	39.4	11.1	52.0	82.3	43.0	24.0	54.2	51.6	72.9	67.6

E. Additional Results on 26 Vision Datasets

We show all experimental results of supervised transfer with linear probe, CLIP zero-shot transfer, and TURTLE unsupervised transfer on 26 vision datasets in Table D1. The linear probe performance of DINOv2 ViT-g/14 is also included for reference. As indicated in the table, TURTLE achieves strong unsupervised transfer performance across various datasets and models. For example, as illustrated in Figure E1, TURTLE 1-space CLIP ViT-L/14 surpasses the corresponding CLIP zero-shot transfer on 13 out of 26 datasets. When trained with multiple representations (*i.e.*, using CLIP models and DINOv2 ViT-g/14), TURTLE 2-spaces achieves superior performance on most datasets compared to TURTLE 1-space. Remarkably, on datasets such as MNIST and CIFAR100, the absolute improvement is 31% and 21%, indicating the effectiveness of TURTLE in combining the knowledge of multiple foundation models. Furthermore, as shown in Figure 5, TURTLE trained with CLIP ViT-L/14 and DINOv2 ViT-g/14 outperforms CLIP zero-shot on 15 out of 26 datasets.

In addition, we compare the performance of TURTLE to supervised linear probe using single representation space in Figure 6. It can be seen that there exists a strong positive correlation between the performance of unsupervised transfer and supervised linear probe. Figure E2 provides the analysis of TURTLE 2-spaces trained using CLIP ViT-L/14 and DINOv2 ViT-g/14, indicating that the performance of TURTLE 2-spaces is also strongly correlated with the average linear probe performance. Overall, these results suggest that TURTLE could potentially benefit from the improved quality of representations, as measured by supervised linear probe.

Finally, it's worth noting that TURTLE 2-spaces might underperform TURTLE 1-space on some datasets, as shown in Table D1. We hypothesize that the discrepancy might stem from the suboptimality of DINOv2 representations for the tasks heavily related to semantic comprehension, such as semantic analysis (Rendered SST, HatefulMemes), traffic sign recognition (GTSRB), geolocation (Country211) and object counting (CLEVR). Since DINOv2 is pre-trained with self-supervised objective (Oquab et al., 2023), the learned features might not be directly transferable to these semantic-intensive tasks. Such trend could also be observed by the linear probe performance, where CLIP ViT-L/14 outperforms DINOv2 ViT-g/14 by a large margin on the Rendered SST2, CLEVR, Country211, GTSRB and FER2013. Therefore, incorporating DINOv2 representations might not yield optimal results for these specific datasets.

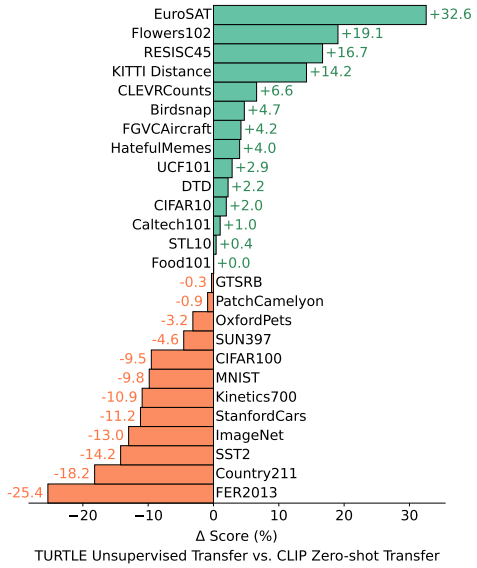

Figure E1. **Using the same representation space, TURTLE outperforms CLIP zero-shot classifier on 13 out of 26 datasets.** TURTLE is trained with CLIP ViT-L/14 and does not require any supervision. CLIP zero-shot utilizes the same architecture, but requires the additional text encoder and description of visual categories.

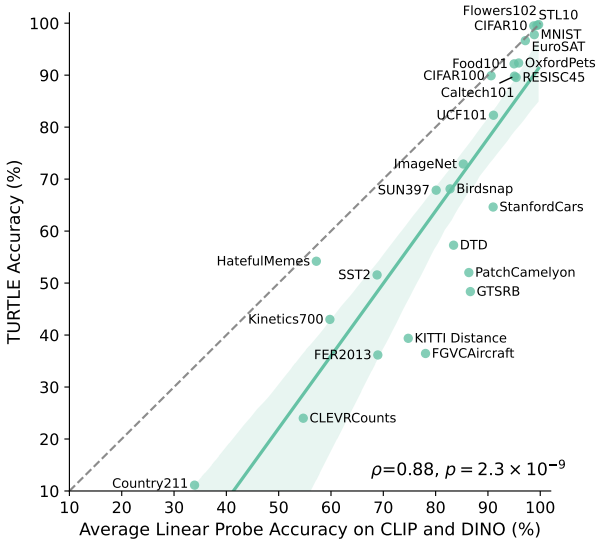

Figure E2. **Unsupervised transfer learning performance is correlated with supervised linear probe performance.** The performance of TURTLE 2-spaces is strongly correlated with the average performance of linear probe using CLIP ViT-L/14 and DINOv2 ViT-g/14 ($\rho = 0.88, p = 2.3 \times 10^{-9}$ for two-sided Pearson correlation coefficient).

F. Results with Scaling Parameters.

We provide the complete results for TURTLE on each dataset for multiple architectures and sizes scaling in the number of parameters in Table D1. Specifically, we consider 8 CLIP models trained in Radford et al. (2021) with scaling model sizes of ResNets (RN50, RN101, RN50x4, RN50x16, RNx64) and Vision Transformers (ViT-B/32, ViT-B/16, ViT-L/14). For each CLIP representation, we train TURTLE 1-space with the aforemention representation spaces, and TURTLE 2-spaces with DINOv2 ViT-g/14 used as the second representation space.

Figure F1 and Figure F2 show the average performance of TURTLE trained with different CLIP ResNets and CLIP Vision Transformers, and compare it with the performance of the CLIP zero-shot transfer. As indicated by the plots, the performance of TURTLE 1-space and TURTLE 2-spaces smoothly improves as the model compute increases. Moreover, although being fully unsupervised, TURTLE 1-space achieves competitive performance (*i.e.*, with less than 1 point difference) compared to the zero-shot transfer for RN50, RN101, RN50x4, ViT-B/32, ViT-B/16 and ViT-L/14. Using RN50x16 and RN50x64, the performance TURTLE 1-space becomes a little bit worse than zero-shot transfer. However, with the additional DINOv2 representations, TURTLE 2-spaces consistently outperforms zero-shot transfer for all the models by a large margin. For example, TURTLE 2-spaces CLIP ViT-L/14 is on average 4% better than zero-shot transfer. For completeness, Figure F3 and Figure F4 also provide the performance of TURTLE and CLIP zero-shot transfer on each individual dataset.

Overall, the performance of TURTLE follows a similar scaling trend as the zero-shot transfer (Radford et al., 2021). Furthermore, TURTLE can effectively combine the knowledge of multiple foundation models to further improve the performance of unsupervised transfer.

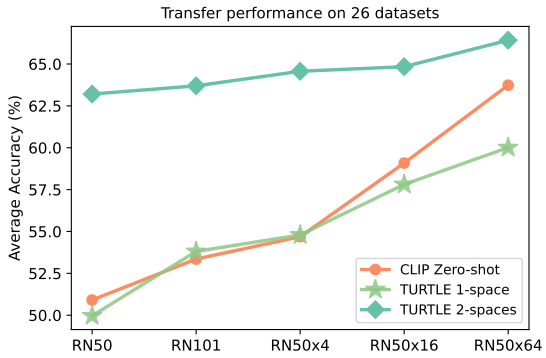

Figure F1. **Performance of TURTLE and CLIP zero-shot averaged on 26 vision datasets using ResNets.** TURTLE 2-spaces uses DINOv2 ViT-g/14 as the second representation space.

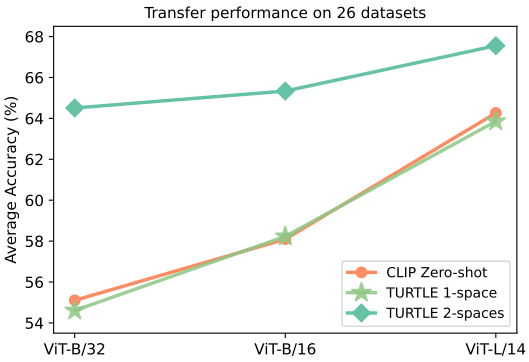

Figure F2. **Performance of TURTLE and CLIP zero-shot averaged on 26 vision datasets using Vision Transformers.** TURTLE 2-spaces uses DINOv2 ViT-g/14 as the second representation space.

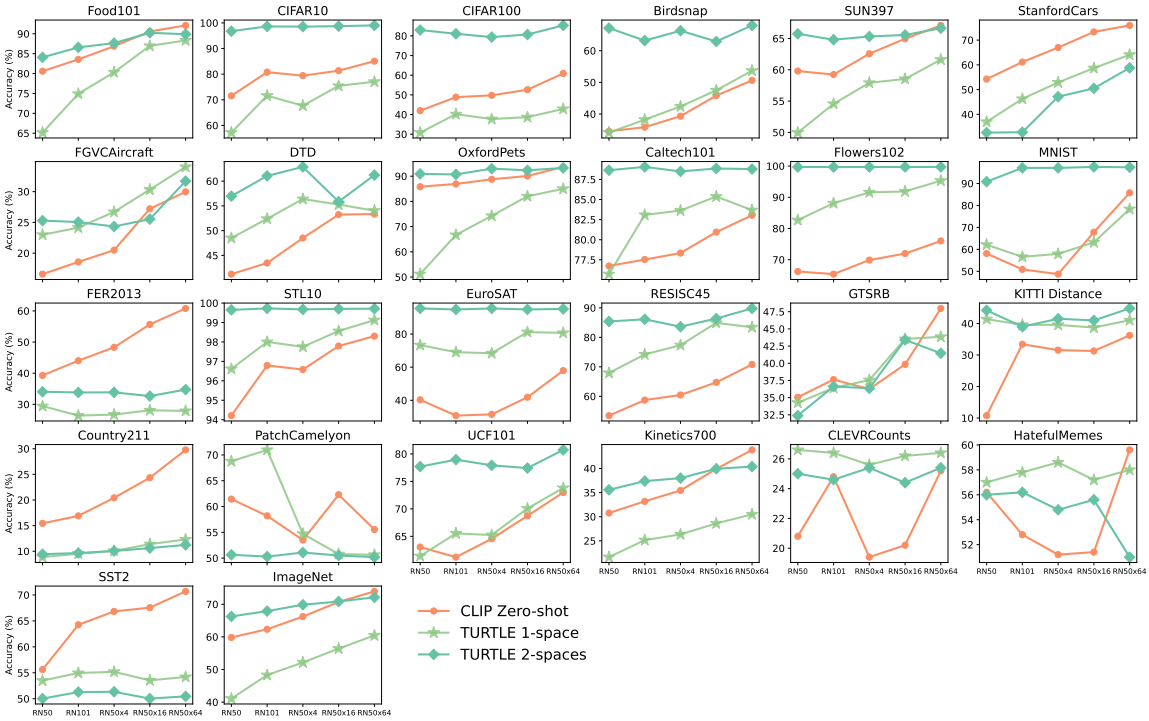

Figure F3. **Per dataset performance of TURTLE and CLIP zero-shot using ResNets.** TURTLE 2-spaces uses DINOv2 ViT-g/14 as the second representation space.

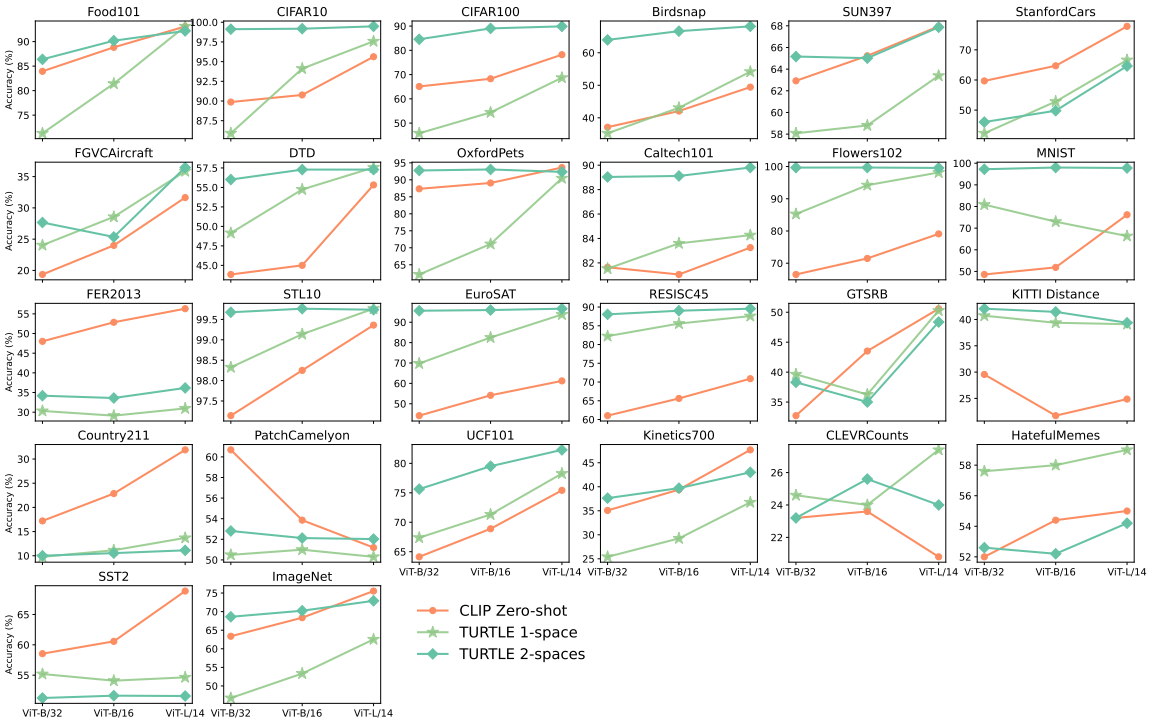

Figure F4. **Per dataset performance of TURTLE and CLIP zero-shot using Vision Transformers.** TURTLE 2-spaces uses DINOv2 ViT-g/14 as the second representation space.

G. Imbalanced Dataset and Entropy Regularization

Following Xu et al. (2004); Van Gansbeke et al. (2020); Gadetsky & Brbić (2023), we use entropy regularization (11) to prevent the task encoder from producing trivial solutions, *i.e.*, assigning all the samples to a single class. By default we set the regularization strength to $\gamma = 10$ for all the experiments. Note that the optimal solution of (11) is to produce a labeling with the equal number of samples for each class. However, some of the datasets are not class balanced. In this case, a strong entropy regularization might hurt the learning process. To understand the effect of the entropy regularization, we show the average performance of TURTLE with $\gamma \in \{0, 1, 3, 5, 10\}$ separately on the imbalanced datasets (Birdsnap, FER2013, GTSRB, KITTI, HatefulMemes), and the rest 21 balanced datasets in Figure G1. The results indicate that the entropy regularization is generally helpful since $\eta = 0$ might lead to trivial solutions. Furthermore, for the balanced datasets, TURTLE is robust to the choice of the regularization hyperparameter. While for the imbalanced datasets, a properly chosen regularization parameter could further improve the performance.

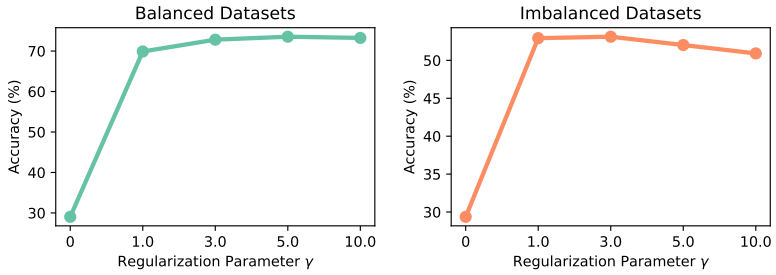

Figure G1. **Ablation of the entropy regularization.** We show the average performance for class imbalanced datasets (Birdsnap, FER2013, GTSRB, KITTI, HatefulMemes) and class balanced datasets (the rest 21 datasets) for the different entropy regularization strength.

H. TURTLE Trained and Evaluated on Test Split

In previous experiments, we train TURTLE on the training split $\mathcal{D}_{tr}$ and evaluate the clustering accuracy on the test split $\mathcal{D}_{te}$. In this section, to study the performance of TURTLE in low data regime, we consider the setting when training and evaluating TURTLE directly on the test split. Figure H1 compares the performance of TURTLE trained on $\mathcal{D}_{tr}$ and TURTLE trained on $\mathcal{D}_{te}$ on the 26 datasets. Both settings are evaluated on the test split. As shown in the plot, TURTLE trained on $\mathcal{D}_{te}$ achieves nearly identical performance as TURTLE trained on $\mathcal{D}_{tr}$ for 24 out of 26 datasets, except Caltech101 and Flowers102. We found the discrepancy might be attributed to the fact that the Caltech101 and Flowers102 have balanced training split but imbalanced test split. Overall, the results suggest that TURTLE does not require a large amount of data to perform successful unsupervised transfer.

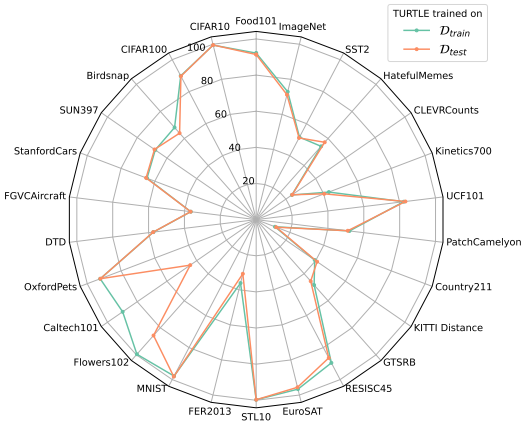

Figure H1. **TURTLE trained on test split achieves similar performance as TURTLE trained on training split for** 24 **out** 26 **of datasets**. Results are both evaluated on the test split. The discrepancy of Caltech101 and Flowers102 is because that they are balanced on training split but imbalanced on test split.

I. Additional Analysis on Fine-grained Classification

In previous sections, we have evaluated the performance of TURTLE on 26 datasets, including 6 fine-grained classification datasets: Food101, Flowers102, Birdsnap, StanfordCars, FGVCAircraft and OxfordPets. According to the results from Table D1 and Figure 5, TURTLE outperforms CLIP zero-shot transfer on 3 datasets (Flowers102, Birdsnap and FGVCAircraft) and performs comparably on 2 datasets (Food101 and OxfordPets). The results indicate that TURTLE remains effective for the task of fine-grained classification.

To further study the dependence between number of classes and the performance of TURTLE, we perform the additional experiments on 4 datasets from the BREEDS benchmark (Santurkar et al., 2021). These datasets are the subsets of ImageNet that contain both coarse and fine-grained labels. We run TURTLE for each dataset to infer the coarse and fine grained labels separately by specifying the ground truth number of classes for each case. The statistics for each dataset and TURTLE's performance are provided in Table I1.

We observe that TURTLE performs worse on the fine-grained classification compared to the coarse-grained classification on LIVING-17. The result is expected since fine-grained classification is considered to be a more difficult task. However, on ENTITY-13, ENTITY-30 and NONLOVING-26, the performance of the fine-grained classification is better than the performance of the coarse-grained classification. We hypothesize that this might be due to the high intra-variance of coarse classes, which was also reported in the previous works on unsupervised image classification (Van Gansbeke et al., 2020). In conclusion, the results suggest that the performance of TURTLE is not largely affected by the granularity of the dataset, but rather by the quality of the representations as indicated by Figure 6 and Figure 7.

Table I1. **BREEDS benchmark and performance of TURTLE.** TURTLE column represents the performance of TURTLE on the given dataset with coarse or fine-grained taxonomy.

Dataset	# Coarse classes	# Fine classes	Train size	Test size	TURTLE	
					Coarse	Fine
ENTITY-13	13	260	334,712	13,000	73.1	85.5
ENTITY-30	30	240	307,828	12,000	79.5	82.4
LIVING-17	17	68	88,400	3,400	96.0	87.0
NONLIVING-26	26	104	132,765	5,200	76.9	78.9