TAGLETS: A System for Automatic Semi-Supervised Learning with Auxiliary Data

Wasu Piriyakulkij [1] **Cristina Menghini** [1] **Ross Briden** [1] **Nihal V. Nayak** [1] **Jeffrey Zhu** [1] **Elaheh Raisi** [2] **Stephen H. Bach** [1]

ABSTRACT

Machine learning practitioners often have access to a spectrum of data: labeled data for the target task (which is often limited), unlabeled data, and *auxiliary data*, the many available labeled datasets for other tasks. We describe TAGLETS, a system built to study techniques for automatically exploiting all three types of data and creating high-quality, servable classifiers. The key components of TAGLETS are: (1) auxiliary data organized according to a knowledge graph, (2) modules encapsulating different methods for exploiting auxiliary and unlabeled data, and (3) a distillation stage in which the ensembled modules are combined into a servable model. We compare TAGLETS with state-of-the-art transfer learning and semi-supervised learning methods on four image classification tasks. Our study covers a range of settings, varying the amount of labeled data and the semantic relatedness of the auxiliary data to the target task. We find that the intelligent incorporation of auxiliary and unlabeled data into multiple learning techniques enables TAGLETS to match—and most often significantly surpass—these alternatives. TAGLETS is available as an open-source system at `github.com/BatsResearch/taglets`.

1 INTRODUCTION

The bottleneck of labeled training data in machine learning has driven research on exploiting alternative, more abundant sources of data such as unlabeled data and *auxiliary data*, i.e., labeled data for possibly related tasks other than the target task. As unlabeled and auxiliary data becomes abundant, labeled training data for a given task has come to represent only a small fraction of the data available to practitioners. As we show, there are underexplored opportunities for taking advantage of all these resources in an integrated, data-centric manner. In this work, we propose a system for automatically exploiting the many annotated datasets either publicly accessible or internally available to organizations, as well as any available unlabeled data, for novel tasks.

In current practice, optimally taking advantage of the full spectrum of available data is not straightforward. Distinct areas of machine learning have grown up around scenarios emphasizing access to different types of data: transfer learning (Pan & Yang, 2010; Yosinski et al., 2014; Weiss et al., 2016; Devlin et al., 2019) for exploiting auxiliary data and semi-supervised learning (Chapelle et al., 2006; Sohn et al., 2020; Chen et al., 2020; Pham et al., 2021) for
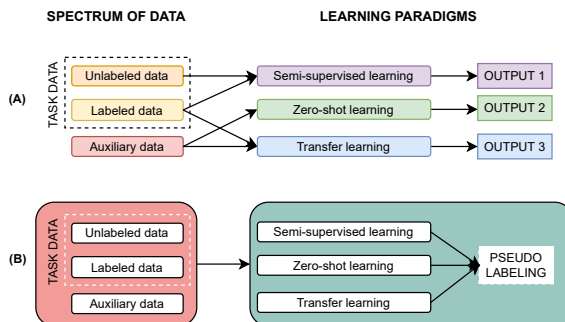

Figure 1. **(A)** *Common practice.* On the left, three categories of data separately passed as input to methods belonging to diverse learning approaches. Each method returns a distinct output. **(B)** *The envisioned approach.* Methods can be trained on all available data. The predictions of such methods are then combined to produce the intermediate pseudo labels for training an end model.

a mix of labeled and unlabeled data. Transfer learning has become the default option in machine learning, and it offers a simple recipe: pre-train a model on a large labeled dataset and then fine-tune it on the target task. This approach is useful for learning widely reusable features of a domain, but is also limiting because it does not take into account the fine-grained similarities between the target task and subsets of the available auxiliary data. Further, it does not take advantage of complementary resources and techniques, such as semi-supervised learning and zero-shot learning (Xian

et al., 2018).

Automatically exploiting auxiliary data, unlabeled data, and limited labeled data for the target task simultaneously requires addressing three main challenges:

1. **Relevant auxiliary data is a needle in a haystack.** Often a huge number of auxiliary datasets are available, but likely only a very small subset is significantly related to any given target task. Further, related examples in auxiliary datasets can be useful for many distinct reasons. In the object classification setting, related datasets might contain examples of visually similar objects, functionally similar objects, objects likely to co-occur in similar settings, and more. How can we automatically find the relevant data quickly?

2. **Best practices vary by amount of training data.** The most accurate methods for learning with limited labeled data vary according to several factors, including how much labeled data is available and how related auxiliary data is to the target task. How can we automatically be robust to these variations?

3. **Pipelines are difficult to serve in production.** In low-latency applications with service-level agreements, it is necessary to make predictions in a fixed amount of time. This requirement limits the complexity of prediction pipelines that are practical. How can we distill all the information in labeled training data, unlabeled data, and auxiliary data into a single, servable model?

In this work, we address these challenges with TAGLETS, a modular end-to-end machine learning system for automatic semi-supervised learning with auxiliary data.

TAGLETS automatically selects relevant subsets of auxiliary data and makes them available to semi-supervised, transfer, and zero-shot learning modules. The modules create an intermediate pseudo labeling, and then TAGLETS distills them into a final, servable model (Figure 1). The system is designed to work with limited labeled data, instead relying on the auxiliary annotated datasets publicly available and internal to organizations. Additionally, unlabeled data from the task distribution is used to further boost performance.

TAGLETS is targeted at scenarios in which non-experts in machine learning need to create classifiers that are deployable at scale. They have access to abundant auxiliary data because much of it is publicly available, and they or their organization might have collected more internally. Use cases include creating classifiers for data science projects and rapid prototyping. We have distilled principles that guided the design of TAGLETS from our experience working with mixtures of labeled, unlabeled, and auxiliary data.

1. **Select the best auxiliary data available.** In some cases, a target task might consist of classes that overlap substantially with those in auxiliary data. In others,

the auxiliary might be only more loosely related. The system should allow models to transfer knowledge from auxiliary data in both cases.

2. **Learn from limited target data.** Different scenarios have access to different amounts of labeled data. The system should be effective when learning from one training example per target class to dozens.

3. **Automatically distill to a servable model.** A user should be able to create a model by providing a minimal semantic description of their target task, and any available labeled and unlabeled data. The system should then automatically produce a high-quality model small enough to be served in production.

We designed TAGLETS to address the above challenges and principles. The key contributions of our work are:

Structured collection of datasets. We introduce the concept of a structured collection of annotated datasets (SCADS). A SCADS gathers all available labeled datasets and connects concepts described in those datasets via a knowledge graph. This repository of linked data is where TAGLETS automatically sources the auxiliary data. SCADS is described in Section 3.1.

Ensemble of methods exploiting auxiliary data. We propose ensembling methods that can exploit a SCADS for robust predictive accuracy. We show how to adapt existing transfer and semi-supervised learning methods to exploit a SCADS in a modular framework. Zero-shot learning based on knowledge graphs further boosts accuracy. We show that ensembling these methods is key to robust accuracy across a range of scenarios covering different amounts of labeled target data and relatedness of the data in the SCADS to the target task. These methods are described in Section 3.2.

End-to-end system. TAGLETS is the first automatic end-to-end machine learning system that simultaneously exploits ecosystems of auxiliary data and models. It achieves state-of-the-art performances on practically useful tasks with limited labeled data. We release it as an open-source package at `github.com/BatsResearch/taglets`.

In this paper, we instantiate TAGLETS for object classification tasks. We use ConceptNet (Speer et al., 2017), a common sense knowledge graph, and ImageNet 21k (Russakovsky et al., 2015b) to create a SCADS. We evaluate TAGLETS by comparing with state-of-the-art transfer and semi-supervised learning methods on four practical object classification tasks. Our study includes experiments with a range of labeled data for target tasks (from 1 shot per class to 20), a range of available auxiliary data, and multiple image embedding backbones. There are several key findings. First, when auxiliary data is closely related to the target task, TAGLETS is most beneficial in the few-shot setting. We find here that TAGLETS improves accuracy over BigTrans-

fer (Kolesnikov et al., 2020) by an average 11.9 percentage points in the one-shot case and 2.13 in the five-shot case. It also improves over the best-performing semi-supervised learning method by an average of 11.54 and 1.96 points in the one-shot and five-shot cases, respectively. In an alternative scenario in which only more distantly related auxiliary data is available, TAGLETS significantly outperforms traditional transfer learning and state-of-the-art semi-supervised learning methods in the one-shot and five-shot cases. In addition, we conduct many ablation experiments to analyze how the multiple components of TAGLETS contribute to its performance. These results demonstrate that automatically selecting auxiliary data, and then exploiting it in combination with unlabeled data and limited labeled data, can lead to large improvements in accuracy across a range of scenarios.

2 RELATED WORK

TAGLETS builds on work in many related areas, including learning with limited labeled data, learning with auxiliary data, and systems for machine learning.

Learning with Limited Labeled Data. There are many subareas of machine learning focused on learning with limited labeled data, including transfer learning (Pan & Yang, 2010; Yosinski et al., 2014; Weiss et al., 2016; Devlin et al., 2019) and semi-supervised learning (Chapelle et al., 2006; Sohn et al., 2020; Chen et al., 2020; Pham et al., 2021). TAGLETS includes modules based on specific types of transfer learning. Multi-task learning (Caruana, 1997) trains a single model to solve multiple tasks simultaneously, with the hope that tasks will benefit from information learned from each other. Zero-shot learning (Xian et al., 2018) trains models to map semantic descriptions of classes to their examples, so that predictions can be made on novel classes without any previously seen examples. Few-shot learning (Finn et al., 2017; Snell et al., 2017; Ravi & Larochelle, 2017; Kolesnikov et al., 2020) emphasizes transfer learning when only a few labeled examples (often one to five per class) are available in the target domain. Early few-shot approaches often use meta-learning, which trains the model by simulating many episodes of few-shot learning. More recently, it has been shown that simply learning a good representation from as much data as possible is often a better alternative (Tian et al., 2020; Chen et al., 2021), and pre-trained models like BigTransfer (Kolesnikov et al., 2020) do well in both few-shot and more conventional transfer learning scenarios.

Other aspects of TAGLETS are connected with specific training strategies. Our use of unsupervised ensembling of different models trained on different parts of the available data can be viewed as a type of bagging (Breiman, 1996). The key difference is that in TAGLETS each ensemble member uses a different learning strategy and types of data (auxiliary, labeled, and unlabeled). These ensemble members can also be viewed as labeling functions in a weak supervision framework such as Snorkel (Ratner et al., 2020), but differ in that that we emphasize automatically creating them from auxiliary data, as opposed to hand engineering them. Knowledge distillation (Hinton et al., 2014; Gou et al., 2021) uses models to train each other, and pseudo labeling can be viewed as a form of knowledge distillation. Another related area is domain adaptation (Ganin et al., 2016; Tzeng et al., 2017), which use unlabeled target data to handle the visual domain shift. There are several few-shot domain adaptation methods (Motiian et al., 2017; Xu et al., 2019) designed for when few target labels are available. However, unlike learning with auxiliary data, in domain adaptation an important assumption is that learning algorithms have access to fully labeled source data with the same label space as the target data.

Other works have considered combining different aspects of these paradigms. Ren et al. (2018); Zhou et al. (2018); Sanodiya et al. (2019); Wei et al. (2019); Yu et al. (2020) and Abuduweili et al. (2021) all consider combining various forms of transfer learning with semi-supervised learning. TAGLETS differs in that it automatically selects subsets of available auxiliary data for learning. Recently, Wortsman et al. (2021) showed that a combination of transfer learning and zero-shot learning can improve over either alone.

Learning with Auxiliary Data. A key aspect of TAGLETS is automatically selecting subsets of auxiliary data to improve accuracy. Learning with auxiliary data has also been considered in prior work. Ge & Yu (2017) use image similarity to select individual examples for transfer learning. Cui et al. (2018) pool image similarity scores to select domains of data for transfer learning. Zhang et al. (2018) uses meta-learning to learn to select auxiliary data. Yan et al. (2020) selects auxiliary data with a mixture of experts trained on the source data. In contrast, TAGLETS selects auxiliary data by organizing it into a knowledge graph and then operating on the graph. This approach avoids the need for training any classifiers or computing per-example similarity.

In addition, work on *data discovery* (Deng et al., 2017; Fernandez et al., 2018) is complementary to our approach. Data discovery methods search for and organize potentially useful datasets from data lakes. We assume that available auxiliary data is organized in a knowledge graph, and data discovery is one way to organize data in this format.

AutoML. TAGLETS emphasizes automatic production of classifiers. Many other works in the field of AutoML have also studied how to automatically produce classifiers and other models, with different focuses. Many systems focus on pipeline optimization (Thornton et al., 2013; Sparks et al., 2015; Feurer et al., 2015; Li et al., 2017; Kotthoff et al., 2017; Shang et al., 2019; Smith et al., 2020). They generally select algorithms and tune hyperparameters, using
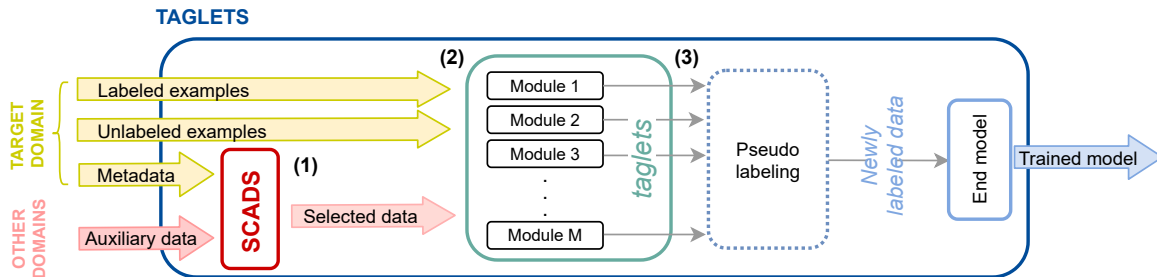
Figure 2. TAGLETS's architecture. **(1)** SCADS organizes and finds auxiliary data related to the target classes. **(2)** Multiple modules are trained to perform pseudo-labeling. **(3)** In the distillation stage, we pseudo-label the unlabeled data and train the *end model* on both the pseudo-labeled and labeled data.

either Bayesian optimization (Snoek et al., 2012) or multi-armed bandits (Slivkins, 2019), which requires labeled validation data. Other systems automate the feature engineering process (Kanter & Veeramachaneni, 2015; Khurana et al., 2016). These approaches are all potentially complementary with TAGLETS, but they require labeled validation data. In contrast, TAGLETS is targeted at few-shot scenarios. We find in our experiments that a single set of modules and hyperparameters works well across tasks, and our focus is instead on automating the process of exploiting the full spectrum of auxiliary, unlabeled, and limited labeled data.

3 TAGLETS ARCHITECTURE

TAGLETS learns a task T and returns a servable classifier $h : \mathcal{X}_T \to \mathcal{Y}_T$ by exploiting ecosystems of datasets and models simultaneously.

The system has access to a spectrum of data:

1. Limited labeled examples $\mathcal{X} = \{(x, y) \in (\mathcal{X}_T, \mathcal{Y}_T)\}$,
2. Task unlabeled data $\mathcal{U} = \{x \in \mathcal{X}_T\}$,
3. Auxiliary data $\mathcal{A} = \{(x, y) \in (\mathcal{X}_S, \mathcal{Y}_S)\}$, from various domains S,

and it consists of the following components (Figure 2):

1. **A structured collection of annotated datasets (SCADS).** We organize auxiliary data in a repository according to a knowledge graph called a SCADS (Section 3.1). We use relations among classes to automatically *select* subsets of task-related auxiliary data $\mathcal{R}$.
2. **Training Modules.** The second component of the system incorporates independent transfer, semi-supervised, and zero-shot learning modules, specifically tailored to exploit task-related auxiliary data (Section 3.2). The output of each module is a classifier, which we call a *taglet*. A taglet takes an input example $x \in \mathcal{X}_T$ and returns a label $y \in \mathcal{Y}_T$. We denote the collection of taglets $\mathcal{T}$.
3. **Distillation stage.** The predictions of different taglets are combined to create *pseudo labels* for the unlabeled

examples, which are used as targets to learn the end model Section 3.3.

3.1 SCADS

The goal of SCADS is to help identify a relevant subset of auxiliary data for a target task. In general, categories described in annotated datasets can overlap or relate to each other. For instance, given two datasets, both gathering images of *postcards*, we need to query them separately to extract all the corresponding examples. Further, when two datasets collect images of related categories, e.g., one contains images of *paper*, and the other of *cards* and *postcards*, we need to automatically identify that these examples are related to associate them. We therefore design SCADS to efficiently retrieve examples from related categories using semantic relations among them. Figure 3 shows an example of a subset of a SCADS.

Construct SCADS. Users can construct a SCADS for their domain by joining all of their data to a *common sense knowledge graph*. Common sense knowledge graphs are powerful resources that make explicit many facts about the world by stating relationships among millions of concepts. We propose using them as the underlying structure of SCADS. In particular, we choose ConceptNet (Speer et al., 2017), a common sense knowledge graph $G = (Q, E)$ whose nodes Q are words and phrases of natural language (i.e., concepts), and edges E express their relations. To build SCADS, we map the classes $c \in \mathcal{Y}_S$ in the auxiliary data and concepts $q \in Q$, such that all the images of the same class are associated to the corresponding node in G (Figure 3 A). We denote $Q_{\mathcal{Y}_S}$ the subset of Q matching with $\mathcal{Y}_S$. SCADS enables end-users to easily install and remove any available annotated dataset. Its extensibility makes possible to unlock the potential of datasets that otherwise could not as easily be used as standalone sources of information. This joining process requires a one-time labor cost, but we find that it is small and can be automated for large datasets like ImageNet-21k using the many links in ConceptNet to external resources like WordNet and Wikipedia.
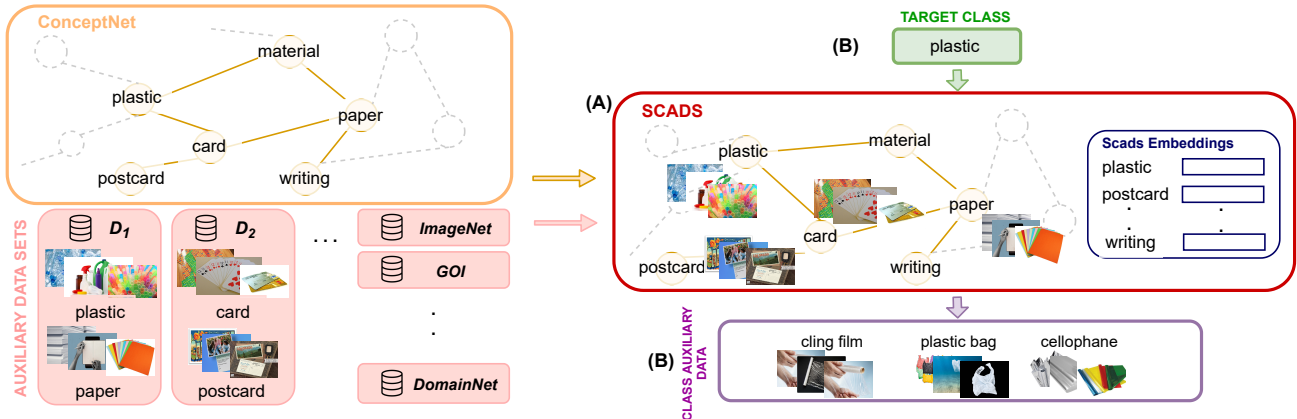
Figure 3. SCADS is built by aligning auxiliary categories with concepts (nodes) in a knowledge graph such as ConceptNet. **(A)** All images of a category belong to the same corresponding concept, and each concept is associated with an embedding, i.e., *scads embedding*, which can be used to determine the relations among concepts. **(B)** Given a query class `plastic`, SCADS retrieves the most similar concepts, according to graph-based similarity.

Select auxiliary data. Joining auxiliary data and Concept-Net defines a rich set of semantic relations on top of examples, and we want to exploit them to select a relevant subset of auxiliary data for the target task. Typical approaches to find data related to the target classes rely on visual similarity (Cui et al., 2018; Yan et al., 2020). However, such methods can be computationally expensive, by involving pairwise calculations. In our setting, we look for task-related data among enormous amounts of auxiliary data (e.g., ImageNet-21k), so visual pairwise-comparisons become intractable. Thus, we leverage SCADS underlying structure and use graph-based semantic similarity between concepts in $Q_{\mathcal{Y}_S}$ and a target class c, to find the subset of concepts mostly related to c. Further, from this subset we query SCADS to get all the corresponding images. With $|Q_{\mathcal{Y}_S}| << |\mathcal{A}|$, our approach is efficient and scales well with increasing amount of auxiliary data in SCADS. In addition, we note that our approach is robust over visual domain shifts, because it does not involve any operation on the images. Computing graph-based similarities requires the definition of *SCADS embeddings*, i.e., a representation of the nodes in SCADS. Since the underlying graph of SCADS is ConceptNet, we use its pre-computed embeddings (Speer et al., 2017), which express both the network topology and word embeddings learned from text. In Appendix A.1, we provide details about the technique to compute these embeddings.

Example 3.1. *Suppose our task is to classify surfaces according to their materials, and consider the target class* `plastic` *(Figure 3 B). From the* scads embeddings *we get the vector* $\hat{e}_{plastic}$ *and use the cosine similarity to find the top-N closest concepts in Q, i.e.,* `cling film,` `plastic bag, cellophane`.

For each of the N similar concepts, SCADS retrieves from the corresponding nodes a set of K images. In this way,

having $|\mathcal{Y}_T| = C$ target classes, we get the set $\mathcal{R}$ consisting of $C \cdot (N \cdot K)$ selected auxiliary examples. SCADS provides flexibility for compute budgets by allowing users to fix the size of the selected auxiliary data $\mathcal{R}$ by setting threshold parameters for the number of task-related concepts N and the number of associated examples K. These thresholds allow the training time of modules (Section 3.2) to remain constant even when the total size of auxiliary data $\mathcal{A}$ is large.

To align target classes with a SCADS, we can use a heuristic match with existing nodes or manual selection. Even in circumstances where a given target class does not align well with any node in ConceptNet, users can manually add nodes and edges to existing concepts.

Example 3.2. *While developing a model to recognize objects in a grocery store (Section 4.1), we found target classes representing specific products without a counterpart in SCADS, e.g.* `oatghurt`*. We handle this scenario by creating a new node* `oatghurt` *and linking it to existing characterizing concepts, e.g.,* yoghurt, carton, *and* oat milk.

In Example 3.2, the target class `oatghurt` is not included in the *scads embedding*. Thus, we compute an *approximation embedding* $\hat{e}_q \approx \sum_{j \in P} w_j \cdot e_j$, where P is the set of approximation terms that share a prefix as long as possible with the given term, and w_i is a term's weight that we set to $\frac{1}{|P|}$. We note that the possibility of modifying the graph underlying SCADS emphasizes the *extensibility* of the component.

3.2 Modules

In TAGLETS, we include four methods in the areas of transfer, semi-supervised and zero-shot learning, and adapt them to fully exploit the knowledge collected in SCADS. We call a method tailored to use SCADS a *module* and its out-

put (i.e., a trained model) a *taglet*. Formally, a module m takes in input data among $\mathcal{X}, \mathcal{U}$, and $\mathcal{R}$, and returns a taglet $t_m : x \mapsto y \in [0,1]^{|\mathcal{Y}_T|}$, such that $\sum_{c=1}^{|\mathcal{Y}_T|} y_c = 1$. Modules are independently trained, and their predictions are ensembled to take advantage of their diversity (Section 3.3 and Sect. 4.4.3). This modular framework is extensible, as other methods can be incorporated on top of the ones we develop here.

3.2.1 Transfer Module

In the scenario of learning with limited labeled data, fine-tuning a pre-trained model ϕ on the target task is the common way of transferring knowledge across domains (Donahue et al., 2014; Yao et al., 2011; Oquab et al., 2014; Razavian et al., 2014). The *transfer module* fine-tunes sequentially on task-related auxiliary and labeled data to leverage the knowledge in SCADS. We train the *transfer module* with an off-the-shelf model ϕ as follows:

1. We extract the set $\mathcal{R}$ of selected auxiliary data from SCADS with the strategy presented in Example 3.1, and note that $|\mathcal{R}| = C \cdot (N \cdot K)$.
2. *Intermediate phase:* We learn the new weights θ' of ϕ fine-tuning its backbone on $\mathcal{R}$, by minimizing the cross entropy loss:

$$\mathcal{L}_{inter} = \frac{1}{|\mathcal{R}|} \sum_{i=1}^{|\mathcal{R}|} \sum_{c=1}^{NC} -y_i^{(c)} \cdot \log \phi_{\theta'}(x_i)^{(c)}. \quad (1)$$

3. We train the final model by fine-tuning $\phi_{\theta'}$ on the labeled examples $\mathcal{X}$, and get the new weights θ''.

$$\mathcal{L}_{target} = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \sum_{c=1}^{C} -y_i^{(c)} \cdot \log \phi_{\theta''}(x_i)^{(c)}, \quad (2)$$

where $y_i^{(c)}$ is 1 if $x \in c$, otherwise 0.

3.2.2 Multi-task Module

An alternative way to transfer knowledge across domains is through multi-task learning (Caruana, 1997) that consists of learning several related tasks simultaneously. The task relatedness has been vaguely defined, and we propose to specify it as the semantic similarity between the classes of the target and auxiliary classification tasks. The *multi-task module* forms an auxiliary classification task out of selected auxiliary data, most semantically similar to the target data, and jointly learns the target task on $\mathcal{X}$ and the auxiliary task on $\mathcal{R}$. In particular, we extract the set of selected auxiliary data $\mathcal{R}$, select a pre-trained model ϕ, and learn its weights θ', by optimizing over a weighted sum of cross entropy losses of the target and and auxiliary tasks:

$$\mathcal{L}_{joint} = \mathcal{L}_{target} + \lambda \mathcal{L}_{aux}, \quad (3)$$

where $\lambda \in \mathbb{R}$ controls the influence of auxiliary task in the joint training, and the losses are:

$$\mathcal{L}_{target} = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \sum_{c=1}^{C} -y_i^{(c)} \cdot \log \phi_{\theta'}(x_i)^{(c)}. \quad (4)$$

$$\mathcal{L}_{aux} = \frac{1}{|\mathcal{R}|} \sum_{i=1}^{|\mathcal{R}|} \sum_{c=1}^{NC} -y_i^{(c)} \cdot \log \phi_{\theta'}(x_i)^{(c)}. \quad (5)$$

3.2.3 FixMatch Module

Since TAGLETS has access to unlabeled data, we include FixMatch (Sohn et al., 2020), an inductive semi-supervised learning algorithm that learns a model ϕ by employing pseudo labeling and consistency regularization. The pseudo labeling procedure uses the model ϕ to label each $x \in \mathcal{U}$ and then trains with the newly labeled instances as targets, assuming the label confidence values are larger than a threshold τ. Additionally, consistency regularization uses unlabeled examples to encourage ϕ to be resistant to information-preserving transformations of input images. Under the regime of limited labeled data, naively applying consistency-based semi-supervised learning algorithms can result in poor performance due to confirmation bias (Wang et al., 2021). We address this by initializing ϕ with a pre-trained model that is further fine-tuned on the auxiliary data $\mathcal{R}$ extracted from SCADS. The *FixMatch module* minimizes the objective:

$$\mathcal{L}_{fix} = \sum_{\substack{u \in \mathcal{U} \\ u_a, u_b = \alpha(u)}} \mathbb{1}(\max(\phi_{\theta'}(u_a)) \geq \tau) H(u_b, \phi_{\theta'}(u_a)),$$

where H is the cross entropy function and α is stochastic function, such as a random rotation, that returns two augmented versions of a single input, i.e., u_a and u_b.

3.2.4 ZSL-KG Module

Finding generalized representations of target classes with minimal data can be hard, especially with only one labeled example per class. To complement the first three modules, we include a zero-shot learning module that makes predictions without any labeled data for the target task. Instead, it uses only the knowledge graph in SCADS. Among the available methods in this subfield, we choose ZSL-KG (Nayak & Bach, 2020), an inductive model for generating class representations from common sense knowledge graphs. The *ZSL-KG module* uses an instance of ZSL-KG Z pre-trained on ConceptNet and ImageNet 1k to learn the representation of target classes. Given a node in a knowledge graph mapped to the target class, ZSL-KG predicts the class' weights vector that can be used in a classification head of an off-the-shelf encoder ϕ. In particular, the module works as follows:

1. We take the graph G underlying SCADS, and the concept q associated to the target class c, and pass them to the pre-trained graph neural network Z, to generate the class representation $z_c = Z(q, G)$.
2. We generate the class representation z_c for all the target classes and plug them into ϕ as the weights of the classification head.

3.3 Distillation stage

We distill the knowledge of individual modules into one servable *end model*. We *ensemble* the predictions of all $t_m \in \mathcal{T}$ on the unlabeled data to produce *pseudo labels* capturing the knowledge of all taglets.

Ensembling. Given an example x:

1. Taglets in $\mathcal{T}$ return probability vectors $\hat{y}^{t_m} \in [0, 1]^C$.
2. We concatenate vectors $\hat{y}^{t_1}, \hat{y}^{t_2}, \cdots, \hat{y}^{t_{|\mathcal{T}|}}$ into a vote matrix $V \in [0, 1]^{|\mathcal{T}| \times C}$.
3. We aggregate the prediction vectors in V and compute the *soft*-pseudo-label:

$$p_x = \frac{1}{|\mathcal{T}|} \sum_{t=1}^{|\mathcal{T}|} V_t, \qquad (6)$$

where the scaling factor $|\mathcal{T}|$ ensures $p_x \in [0, 1]^C$ is a probability vector.

The benefits of the ensemble as a means of exploiting the *taglets*' diversity are discussed in Section 4.4.3.

End model. After we assign to all $x \in \mathcal{U}$ pseudo labels, we use them, along with labeled data $\mathcal{X}$, to train the end model h. We train h fine-tuning a pre-trained model on the pseudo-labeled data $\mathcal{P}$ and $\mathcal{X}$, by optimizing the soft cross entropy loss:

$$\mathcal{L}_{final} = \frac{1}{|\mathcal{P} \cup \mathcal{X}|} \sum_{j=1}^{|\mathcal{P} \cup \mathcal{X}|} \sum_{c=1}^{C} -p_j^{(c)} \cdot \log h_{\theta'}(x_j)^{(c)}. \quad (7)$$

We emphasize the importance of producing a single, servable model when service-level agreements are required.

4 EXPERIMENTAL EVALUATION

In this section, we evaluate TAGLETS on four different image classification tasks with the goal of showing (1) the predictive accuracy of TAGLETS under different circumstances, (2) the benefits of SCADS, and (3) the benefits of combining multiple learning techniques. The code to reproduce the results presented in the reminder of the section is openly available at https://github.com/BatsResearch/piriyakulkij-mlsys22-code.

4.1 Datasets

We refrain from using standard semi-supervised learning datasets, such as ImageNet and CIFAR derivatives, which are general, natural image classification datasets. Instead, we turn to datasets designed for specific applications, not necessarily describing general visual domains. This choice is driven by two main reasons: (1) massive general, natural image datasets, including Google Open Images (Kuznetsova et al., 2020) and ImageNet-21k (Deng et al., 2009) are publicly available, so practitioners can rely on them to get enough data to solve target tasks in a general, natural domain, and (2) TAGLETS is designed to solve tasks for specific applications.

Flickr Material Database (Sharan et al., 2014) is a benchmark for material recognition and captures the natural range of material appearances. It has 1,000 color photographs of surfaces uniformly belonging to 10 common material categories like *fabric, foliage*, and *glass*. To reduce the chances that simple, low-level information (e.g., color) can be used to distinguish material categories, the dataset intentionally includes diversity in the same class. A practical application is to support waste sorting and recycling.

Office-Home dataset (Venkateswara et al., 2017) consists of 65 object categories, like *keyboard, desk lamp*, and *fan*, found typically in office and home settings. The images are from four different domains, among which we consider *clipart* including clipart images, and *product* gathering images without background, and build the **OfficeHome-Clipart** and the **OfficeHome-Product** datasets. The two contain a minimum of 39 and 38 images per class respectively. This dataset requires recognizing daily life objects in non-natural visual domains. A practical application is understanding images on the Web, such as product catalogs and illustrations.

Grocery Store dataset (Klasson et al., 2019) consists of natural images of grocery items taken with a smartphone camera in different grocery stores. The dataset contains 5,125 natural images from 42 coarse-grained classes of fruits, vegetables, and carton items, with the minimum number of images per class being 18. The dataset highlights the extensability of TAGLETS because two target classes, i.e., `oathgurt` and `soygurth`, do not appear as concepts in ConceptNet. To address this issue, we modify SCADS by adding new nodes and manually connect them and existing nodes (see Example 3.2). s A practical application is assistive technology for people with vision impairments.

4.2 Baselines

We compare TAGLETS to state-of-the-art transfer and semi-supervised learning methods.

Fine-tuning. Our system is designed to transfer knowledge across its components, starting from auxiliary data related to
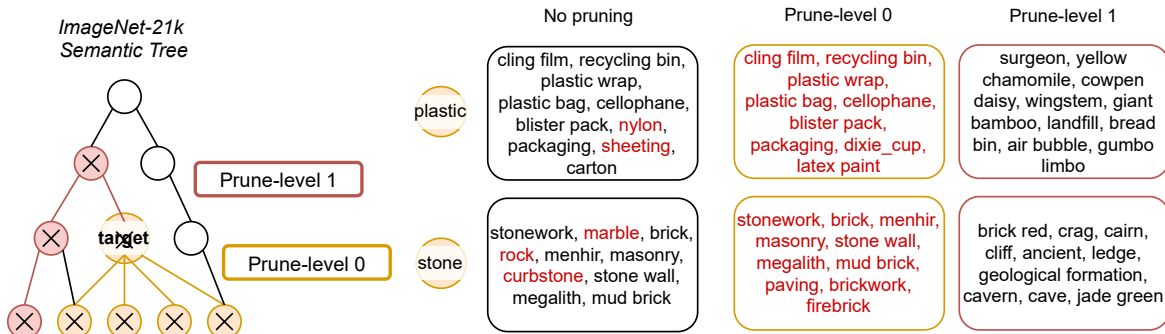
Figure 4. Two examples that demonstrate how pruning affects the retrieval of concepts related to a target class. Given the target class `plastic`, we find the list of ten most related concepts in SCADS, without pruning. We highlight in red the concepts that are pruned from SCADS when we remove `plastic` and its children from the ImageNet-21k semantic tree (prune level 0). We do the same for the next level of pruning. At higher level of pruning, the remaining concepts are more general or more distantly related to the target task.

the target classes. We compare TAGLETS with **BigTransfer** (**BiT**) (Kolesnikov et al., 2020) a model pre-trained on ImageNet-21k (Deng et al., 2009) that uses a ResNet-50-v2 as backbone (He et al., 2016a), and stands out for its performances when fine-tuned with few labeled examples per class. To study situations in which a model pre-trained on all available auxiliary data is not available, we also compare with **ResNet-50** (He et al., 2016b) pre-trained on ImageNet-1k (Russakovsky et al., 2015b), a standard image encoder.

Distilled Fine-tuning. Since TAGLETS has a distillation stage that uses unlabeled data, we also run distilled versions of fine-tuning. In particular, we first fine-tune the backbone of either BiT (ImageNet-21k) or ResNet-50 (ImageNet-1k) on the available labeled examples, then produce pseudo labels for $\mathcal{U}$, and eventually fine-tune either BiT or ResNet-50 using the newly labeled examples as targets.

Semi-supervised Methods. We compare TAGLETS against three state-of-the-art semi-supervised learning algorithms: FixMatch (Sohn et al., 2020), Meta Pseudo Labels (Pham et al., 2021), and SimCLRv2 (Chen et al., 2020). While these methods typically use randomly initialized ResNet encoders, we use pre-trained encoders in our experiments as they give better performance with the limited amount of both labeled and unlabeled data. **FixMatch** combines consistency regularization and pseudo labeling, as explained in Section 3.2.3. For fair comparisons, we implement FixMatch using either ResNet-50 (ImageNet-1k) or BiT (ImageNet-21k) as pretrained encoders. **Meta Pseudo Labels** trains a student network on unlabeled data using pseudo labels generated from a teacher network. Unlike in traditional approaches to pseudo labeling, feedback from the student's performance is also used to adapt the teacher. After teacher-student training, the student is fine-tuned on labeled data to reduce the impact of confirmation bias. Similar to FixMatch, the teacher network uses either ResNet-50 (ImageNet-1k) or BiT (ImageNet-21k) as a pretrained en-

coder. The student network, however, only uses ResNet-50 (ImageNet-1k) as a pretrained encoder to prevent overfitting. **SimCLRv2** uses data augmentation techniques combined with a contrastive loss function to pre-train an encoder with little to no supervision. This approach has exhibited strong performance when trained on large unlabeled datasets, such as ImageNet-1k. However, in our experiments, we found that the performance of SimCLRv2 deteriorates significantly when trained on smaller datasets. Consequently, we do not include this method in our results.

4.3 System and Experiments Set Up

System. On top of ConceptNet, we use ImageNet-21k as the auxiliary dataset in SCADS because it offers a comprehensive overview of categories covering common knowledge helpful to solve our tasks. We point out that our choice can be combined with other annotated datasets potentially useful for the target task. Throughout the experiments, we vary the backbones of the modules (i.e., pre-trained models ϕ) choosing either ResNet-50 trained on ImageNet-1k or BiT trained on ImageNet-21k. The two options represent how the backbone can be either pre-trained on parts or all of the auxiliary data, though TAGLETS's users can also plug in any backbone of their choices that is not necessarily pretrained on data installed on SCADS. For each backbone, the hyperparameters of TAGLETS are fixed throughout our experiments (see Appendix A.3 for details). The good quality of predictions obtained across datasets without hyperparameter tuning highlights the robustness of the system.

Experiments. For each of the datasets, we randomly generate a train and a test set, if the latter is not predetermined (Appendix A.2). In the train set, randomly pick and label 1, 5, or 20 examples (i.e., shots) per class, and use the rest as unlabeled examples. We run the system on three different train-test splits on each dataset (Appendix A.2). Due to

Method	Backbone	OfficeHome-Product			OfficeHome-Clipart		
		1-shot	5-shot	20-shot	1-shot	5-shot	20-shot
Fine-tuning	BiT (ImageNet-21k)	57.28 ± 5.20	84.41 ± 1.17	$\mathbf{90.67 \pm 1.10}$	30.36 ± 2.54	63.44 ± 2.10	$\mathbf{77.13 \pm 1.34}$
Fine-tuning (Distilled)	BiT (ImageNet-21k)	58.21 ± 4.56	85.03 ± 0.22	$\mathbf{92.00 \pm 0.38}$	31.74 ± 1.96	62.92 ± 1.99	$\mathbf{78.26 \pm 0.88}$
FixMatch	BiT (ImageNet-21k)	49.03 ± 2.54	83.03 ± 2.97	$\mathbf{92.21 \pm 0.44}$	30.67 ± 5.44	64.77 ± 1.75	$\mathbf{79.95 \pm 2.30}$
Meta Pseudo Label	BiT (ImageNet-21k)	59.95 ± 0.44	81.53 ± 1.01	$\mathbf{89.90 \pm 0.88}$	31.74 ± 2.1	62.92 ± 2.13	76.05 ± 1.72
TAGLETS	BiT (ImageNet-21k)	$\mathbf{70.92 \pm 1.67}$	$\mathbf{86.31 \pm 1.01}$	$\mathbf{92.36 \pm 3.18}$	$\mathbf{46.97 \pm 2.46}$	$\mathbf{67.64 \pm 3.34}$	78.62 ± 2.76
Fine-tuning	ResNet-50 (ImageNet-1k)	32.51 ± 3.83	63.44 ± 7.45	81.13 ± 0.22	16.26 ± 2.87	43.49 ± 0.58	68.56 ± 2.24
Fine-tuning (Distilled)	ResNet-50 (ImageNet-1k)	32.72 ± 3.67	66.87 ± 5.60	83.85 ± 2.68	15.85 ± 0.66	44.26 ± 2.34	$\underline{72.00 \pm 2.32}$
FixMatch	ResNet-50 (ImageNet-1k)	39.28 ± 3.34	72.87 ± 1.45	85.03 ± 1.81	20.51 ± 0.96	49.69 ± 3.31	66.56 ± 1.23
Meta Pseudo Label	ResNet-50 (ImageNet-1k)	47.38 ± 2.98	76.67 ± 1.75	$\underline{87.38 \pm 1.75}$	22.77 ± 3.99	57.25 ± 4.64	75.17 ± 1.22
TAGLETS	ResNet-50 (ImageNet-1k)	$\underline{67.64 \pm 3.61}$	$\underline{83.54 \pm 2.89}$	$\underline{88.77 \pm 2.89}$	$\underline{43.44 \pm 4.46}$	$\underline{63.90 \pm 1.17}$	75.54 ± 4.34
TAGLETS prune-level 0	ResNet-50 (ImageNet-1k)	$\underline{65.33 \pm 1.23}$	$\underline{81.23 \pm 0.00}$	$\underline{88.62 \pm 1.38}$	$\underline{41.49 \pm 1.54}$	62.21 ± 1.89	75.79 ± 3.11
TAGLETS prune-level 1	ResNet-50 (ImageNet-1k)	57.28 ± 1.89	76.21 ± 1.54	$\underline{86.67 \pm 0.58}$	36.87 ± 2.17	55.33 ± 4.19	72.72 ± 0.96

Table 1. Accuracy (%) of TAGLETS and baselines on *OfficeHome Product* and *Clipart* (`Split 0`) for increasing number of labeled examples (1, 5, and 20 shots), different backbones (ResNet-50 trained on ImageNet-1k and BiT trained on ImageNet-21k) and levels of pruning (no-pruning, 0 and 1). We bold and underline the best performances, and those within their 95% confidence intervals, carried out using **BiT** and ResNet backbones, respectively. Measurements are repeated for three seeds and we report the 95% confidence intervals.

space constraints, we report results of one split and gather the others in Appendix A.4. The trends are consistent across all three splits.

Pruning. We *prune* SCADS to simulate the scenario in which only more distantly related auxiliary data is available, and investigate if the relatedness between auxiliary data and target classes is a crucial factor to boost TAGLETS performance. Pruning SCADS consists in progressively removing concepts closely related to the target classes. In particular, we take the semantic tree H defined over the categories in ImageNet-21k, i.e., the WordNet hierarchy (Fellbaum, 1998), and for the target class c remove: c and all its descendants to obtain **prune-level 0**. Further, we remove c's parent and the parent's subtree for **prune-level 1**. See Figure 4.

4.4 Results

We evaluate TAGLETS, addressing the following questions:

1. How does the system's accuracy compare with state-of-the-art methods, across varying amounts of labeled examples? (Sect. 4.4.1)
2. How does auxiliary data improve the modules' performances? (Sect. 4.4.2)
3. How does module ensembling affect the overall system's performance? (Sect. 4.4.3)

In Section 1, we summarized the accuracy of TAGLETS by averaging the performances over all the tasks and on the three splits. In this section, we discuss results on `split 0` which, according to the 95% confidence intervals, are consistent with those of `split 1` and `split 2` that we report in Appendix A.4.

4.4.1 Accuracy with Limited Labeled Data

Table 1 and Table 2 show that TAGLETS achieves the best accuracy in 1-shot and 5-shots compared to the existing state-of-the-art methods. TAGLETS significantly improves over transfer learning methods by 11.9% and 2.7%, on average, in the 1 and 5-shots settings, using BiT as the backbone. Our system is also competitive with the best-performing transfer learning baseline (Distilled Fine-tuning on BiT) in the 20-shots setting. Similarly, TAGLETS outperforms semi-supervised state-of-the-art algorithms by 12.2% (1-shot) and 2.0% (5-shots), and it is on average 0.12% lower in the 20-shot case.

So far, we have looked at TAGLETS assuming that we have access to backbones pre-trained on all the auxiliary data. However, when auxiliary data is large and does not include datasets typically used for transfer learning, it is unrealistic to pre-train backbones on all of it. Thus, we investigate TAGLETS using ResNet-50 pre-trained on ImageNet-1k as backbone. In this scenario, results show that TAGLETS outperforms distilled BiT by 5.4 points in the 1-shot setting, demonstrating its capability to exploit auxiliary data, even when they are not used to train the initial backbone. In the 5-shot and 20-shot cases, TAGLETS performs worse than distilled BiT (-1.2 and -6.2, respectively), but the drop is still modest considering the small amount of auxiliary data we actually use. We also see that TAGLETS usually outperforms all baselines with ResNet-50 backbones, often by large margins. In cases where it does not, it is among the best performers with one exception: MetaPseudoLabel has a higher score on the 20-shot Flickr Material Dataset task.

Method	Backbone	Grocery Store Dataset		Flickr Material Dataset		
		1-shot	5-shot	1-shot	5-shot	20-shot
Fine-tuning	BiT (ImageNet-21k)	51.32 ± 3.23	85.16 ± 2.41	52.67 ± 2.35	70.07 ± 1.52	$\mathbf{85.40 \pm 0.99}$
Fine-tuning (Distilled)	BiT (ImageNet-21k)	52.42 ± 1.03	86.06 ± 1.81	57.53 ± 10.95	73.27 ± 1.03	$\mathbf{86.80 \pm 1.31}$
FixMatch	BiT (ImageNet-21k)	48.58 ± 13.41	87.61 ± 4.82	19.07 ± 31.91	60.67 ± 5.03	$\mathbf{84.40 \pm 2.28}$
Meta Pseudo Label	BiT (ImageNet-21k)	54.51 ± 6.22	84.02 ± 1.30	52.73 ± 5.82	69.80 ± 2.24	79.33 ± 2.24
TAGLETS	BiT (ImageNet-21k)	$\mathbf{61.60 \pm 2.90}$	$\mathbf{88.91 \pm 1.07}$	$\mathbf{68.07 \pm 5.76}$	$\mathbf{75.20 \pm 1.72}$	84.53 ± 2.29
Fine-tuning	ResNet-50 (ImageNet-1k)	30.25 ± 4.08	70.60 ± 3.25	28.20 ± 3.48	47.20 ± 3.94	60.47 ± 8.52
Fine-tuning (Distilled)	ResNet-50 (ImageNet-1k)	30.64 ± 4.33	74.63 ± 2.44	29.20 ± 7.32	49.80 ± 3.44	66.27 ± 4.51
FixMatch	ResNet-50 (ImageNet-1k)	32.14 ± 4.72	70.77 ± 0.75	37.73 ± 13.97	61.00 ± 6.72	74.13 ± 1.52
Meta Pseudo Label	ResNet-50 (ImageNet-1k)	34.68 ± 3.11	75.16 ± 1.57	37.99 ± 13.52	68.00 ± 3.47	77.13 ± 1.25
TAGLETS	ResNet-50 (ImageNet-1k)	$\underline{51.76 \pm 3.48}$	$\underline{84.35 \pm 0.92}$	$\underline{55.87 \pm 4.05}$	$\underline{68.27 \pm 2.45}$	74.53 ± 2.74
TAGLETS prune-level 0	ResNet-50 (ImageNet-1k)	46.32 ± 5.01	81.88 ± 1.27	$\underline{53.40 \pm 4.42}$	67.60 ± 3.48	$\underline{74.47 \pm 1.88}$
TAGLETS prune-level 1	ResNet-50 (ImageNet-1k)	37.68 ± 6.64	75.75 ± 1.84	$\underline{55.00 \pm 8.18}$	66.00 ± 5.73	$\underline{72.93 \pm 1.60}$

Table 2. Accuracy (%) of TAGLETS and baselines on *Grocery Store* and *Flicker Material Datasets* (`Split 0`) for increasing number of labeled examples (1, 5, and 20 shots), different backbones (ResNet-50 trained on ImageNet-1k and BiT trained on ImageNet-21k) and levels of pruning (no-pruning, 0 and 1). We bold and underline the best performing models, and those within their 95% confidence intervals, carried out using **BiT** and <u>ResNet</u> backbones, respectively. Measurements are repeated for three seeds, and we report the 95% confidence intervals.

4.4.2 The Benefits of SCADS

In this section we study the importance of selecting task-related auxiliary data, and its influence over the performance of individual modules. In Table 1 and Table 2, the performance of TAGLETS when SCADS is pruned at levels 0 and 1 (Section 4.3) offers several insights. First, transferring knowledge from closely related data improves TAGLETS's performance, indeed pruning SCADS causes drops in accuracy. Second, the performance at prune levels 0 and 1 are still often as good or better than the baselines with ResNet-50 backbone, showing that TAGLETS can take advantage of auxiliary data even when it is more distantly related.

For a more in-depth understanding of the importance of selected auxiliary data, we investigate the performance of the training modules at different levels of pruning. Figure 5 shows that the modules benefit individually from the use of task-related auxiliary data. However, we see diminishing gains when the number of labeled data increases. Also, the Transfer and FixMatch modules significantly outperform their baseline counterparts in the no-pruning and 0-level pruning (and often in the 1-level pruning too), showing that our modifications to exploit auxiliary data are beneficial.

4.4.3 The Benefits of Ensembling

In Figure 6, we quantify the benefit of combining multiple learning techniques. We observe that the system benefits from the modules ensemble for all the considered scenarios (i.e., varying labeled examples and pruning levels). There is always an improvement of at least 7% on the average accuracy of the training modules. We note that, in the 1-shot and 5-shots cases, the ensemble boosts the accuracy of the best
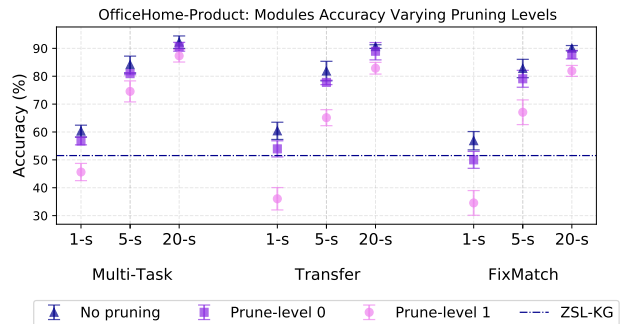

Figure 5. The plot shows the %-accuracy of each module at different levels of pruning (shapes) and for varying amount of labeled examples (x-axis), on the *OfficeHome-Product* dataset. The error bars indicate 95% confidence intervals on three seeds. All modules have ResNet-50 backbone. The performances of the ZSL-KG module are invariant since it is not re-trained.

performing module by at least 3.8 and 1.1 percentage points, respectively. This suggest that the ensemble helps to boost the system performance by exploiting modules' diversity. Additionally, we see that the efficacy of the ensembling is not corrupted when the auxiliary data is not closely related to the target classes, i.e., for different levels of pruning. This indicates that the ensembling procedure is crucial to limit the influence of irrelevant auxiliary data in TAGLETS.

Finally, we investigate how the accuracy of the end model and the ensemble differ. Aggregated results over splits and datasets show that the change in accuracy between the ensemble and the end model is variable. Indeed, the end model performances range between -5 and 4 points on the
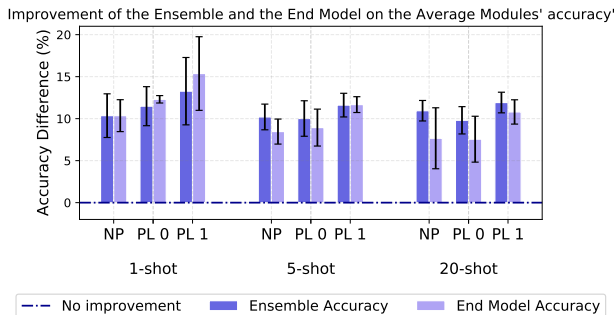
Figure 6. For the *OfficeHome-Product* dataset, we fix the number of labeled examples (1, 5, and 20 shots) and plot the improvements of the ensemble and the end model on the average accuracy of the training modules, for different pruning levels. The results refer to TAGLETS with ResNet-50 backbone and the 95% confidence intervals are computed over three different seeds.

ensemble accuracy (average -0.22). Overall, we believe that the occasional loss in accuracy is a fair price to pay to make more compact classifiers. In Appendix A.4, we gather the plots highlighting the difference of the performances before and after the end model for each dataset and split.

We run additional ablations to understand if each module contributes to the overall performances of TAGLETS. Figure 7 shows that cutting any module reduces accuracy in at least half of the considered cases. This suggests that each module injects diversity into the ensemble that helps the system to be robust across a large variety of settings.

5 Conclusion

We introduce TAGLETS, a novel system for semi-supervised learning with auxiliary data, under the limited labeled data regime. TAGLETS integrates auxiliary and unlabeled data into multiple learning techniques, and it distills their knowledge into one servable model. Our extensive experiments on datasets designed for real applications show that TAGLETS significantly outperforms state-of-the-art algorithms in transfer and semi-supervised learning.

In this work, we emphasized the importance of exploiting auxiliary data and introduced SCADS to organize it. However, at this stage, the component is agnostic to the data domain, which can potentially be an obstacle for learning. This limitation leaves the door open for future works that can address the issue of multiple domains using SCADS.
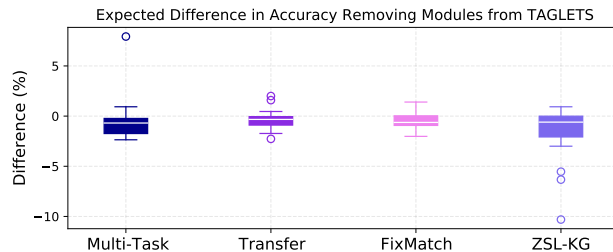
Acknowledgements

Figure 7. Distributions of average differences in accuracy when a module is removed from TAGLETS. Negative values indicate that removing the module deteriorates the overall accuracy of the system. The distribution includes observations on all datasets and backbones in the 1 and 5-shots settings (`split 0`).

References

Abuduweili, A., Li, X., Shi, H., Xu, C.-Z., and Dou, D. Adaptive consistency regularization for semi-supervised transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Breiman, L. Bagging predictors. *Machine Learning*, 24(2): 123–140, 1996.

Caruana, R. Multitask learning. *Machine Learning*, 28: 41–75, 1997.

Chapelle, O., Schölkopf, B., and Zien, A. *Semi-Supervised Learning*. MIT Press, 2006.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020.

Chen, Y., Liu, Z., Xu, H., Darrell, T., and Wang, X. Meta-baseline: Exploring simple meta-learning for few-shot

learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9062–9071, 2021.

Cui, Y., Song, Y., Sun, C., Howard, A., and Belongie, S. Large scale fine-grained categorization and domain-specific transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Deng, D., Fernandez, R. C., Abedjan, Z., Wang, S., Stonebraker, M., Elmagarmid, A., Ilyas, I. F., Madden, S., Ouzzani, M., and Tang, N. The data civilizer system. In *Conference on Innovative Data Systems Research (CIDR)*, 2017.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Meeting of the North American Association for Computational Linguistics (NAACL)*, 2019.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31st International Conference on Machine Learning*, 2014.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

Fellbaum, C. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

Fernandez, R. C., Abedjan, Z., Koko, F., Yuan, G., Madden, S., and Stonebraker, M. Aurum: A data discovery system. In *International Conference on Data Engineering (ICDE)*, 2018.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, 2015.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, 2017.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., and Lempitsky, V. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 2016.

Ge, W. and Yu, Y. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In Leibe, B., Matas, J., Sebe, N., and Welling, M. (eds.), *Computer Vision – ECCV 2016*, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016b.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning Workshop*, 2014.

Kanter, J. M. and Veeramachaneni, K. Deep feature synthesis: Towards automating data science endeavors. *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015.

Khurana, U., Turaga, D. S., Samulowitz, H., and Parthasrathy, S. Cognito: Automated feature engineering for supervised learning. *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, 2016.

Klasson, M., Zhang, C., and Kjellström, H. A hierarchical grocery store image dataset with visual and semantic labels. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big transfer (bit): General visual representation learning. In Vedaldi, A., Bischof, H., Brox, T., and Frahm, J.-M. (eds.), *Computer Vision – ECCV 2020*, 2020.

Kotthoff, L., Thornton, C. J., Hoos, H. H., Hutter, F., and Leyton-Brown, K. Auto-weka 2.0: Automatic model selection and hyperparameter optimization in weka. *Journal of Machine Learning Research*, 2017.

Kuznetsova, A., Rom, H., Alldrin, N. G., Uijlings, J. R. R., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., Duerig, T., and Ferrari, V. The open images dataset v4. *International Journal of Computer Vision*, 2020.

Li, L., Jamieson, K. G., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. S. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 2017.

Marcel, S. and Rodriguez, Y. Torchvision: The machine-vision package of torch. In *International Conference on Multimedia*, 2010.

Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. Efficient estimation of word representations in vector space. In *ICLR*, 2013a.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, 2013b.

Motiian, S., Jones, Q., Iranmanesh, S. M., and Doretto, G. Few-shot adversarial domain adaptation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Nayak, N. V. and Bach, S. H. Zero-shot learning with common sense knowledge graphs. arXiv:2006.10713 [cs.LG], 2020.

Oquab, M., Bottou, L., Laptev, I., and Sivic, J. Learning and transferring mid-level image representations using convolutional neural networks. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22 (10):1345–1359, 2010.

Pham, H., Dai, Z., Xie, Q., Luong, M.-T., and Le, Q. V. Meta pseudo labels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2021.

Ratner, A. J., Bach, S. H., Ehrenberg, H. E., Fries, J., Wu, S., and Ré, C. Snorkel: Rapid training data creation with weak supervision. *The VLDB Journal*, 29(2):709–730, 2020.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.

Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: An astounding baseline for recognition. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations (ICLR)*, 2018.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015a.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015b.

Sanodiya, R. K., Mathew, J., Saha, S., and Thalakottur, M. D. A new transfer learning algorithm in semi-supervised setting. *IEEE Access*, 7:42956–42967, 2019.

Shang, Z., Zgraggen, E., Buratti, B., Kossmann, F., Eichmann, P., Chung, Y., Binnig, C., Upfal, E., and Kraska, T. Democratizing data science through interactive curation of ml pipelines. In *Proceedings of the 2019 International Conference on Management of Data*, 2019.

Sharan, L., Rosenholtz, R., and Adelson, E. H. Accuracy and speed of material categorization in real-world images. *Journal of Vision*, 2014.

Slivkins, A. Introduction to multi-armed bandits. *ArXiv*, 2019.

Smith, M. J., Sala, C., Kanter, J. M., and Veeramachaneni, K. The machine learning bazaar: Harnessing the ml ecosystem for effective system development. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020.

Snell, J., Swersky, K., and Zemel, R. S. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Snoek, J., Larochelle, H., and Adams, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, 2012.

Sohn, K., Berthelot, D., Li, C.-L., Zhang, Z., Carlini, N., Cubuk, E. D., Kurakin, A., Zhang, H., and Raffel, C. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *ArXiv*, abs/2001.07685, 2020.

Sparks, E. R., Talwalkar, A. S., Franklin, M. J., Jordan, M. I., and Kraska, T. Tupaq: An efficient planner for large-scale predictive analytic queries. *ArXiv*, 2015.

Speer, R. and Chin, J. An ensemble method to produce high-quality word embeddings. *CoRR*, 2016.

Speer, R., Chin, J., and Havasi, C. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Thornton, C., Hutter, F., Hoos, H. H., and Leyton-Brown, K. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.

Tian, Y., Wang, Y., Krishnan, D., Tenenbaum, J. B., and Isola, P. Rethinking few-shot image classification: a good embedding is all you need? 2020.

Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Venkateswara, H., Eusebio, J., Chakraborty, S., and Panchanathan, S. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Wang, X., Gao, J., Long, M., and Wang, J. Self-tuning for data-efficient deep learning. In *International Conference on Machine Learning (ICML)*, 2021.

Wei, W., Meng, D., Qian Zhao, Z. X., and Wu, Y. Semi-supervised transfer learning for image rain removal. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Weiss, K., Khoshgoftaar, T. M., and Wang, D. A survey of transfer learning. *Journal of Big Data*, 3(1):1–40, 2016.

Wortsman, M., Ilharco, G., Li, M., Kim, J. W., Hajishirzi, H., Farhadi, A., Namkoong, H., and Schmidt, L. Robust fine-tuning of zero-shot models. arXiv:2109.01903 [cs.CV], 2021.

Xian, Y., Lampert, C. H., Schiele, B., and Akata, Z. Zero-shot learning—A comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 41(9):2251–2265, 2018.

Xu, X., Zhou, X., Venkatesan, R., Swaminathan, G., and Majumder, O. d-sne: Domain adaptation using stochastic neighborhood embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

Yan, X., Acuna, D., and Fidler, S. Neural data server: A large-scale search engine for transfer learning data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Yao, B., Khosla, A., and Fei-Fei, L. Combining randomization and discrimination for fine-grained image categorization. In *CVPR 2011*, 2011.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? In *Neural Information Processing Systems (NeurIPS)*, 2014.

Yu, Z., Chen, L., Cheng, Z., and Luo, J. Transmatch: A transfer-learning scheme for semi-supervised few-shot learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Zhang, Y., Tang, H., and Jia, K. Fine-grained visual categorization using meta-learning optimization with sample selection of auxiliary data. In *European Conference on Computer Vision (ECCV)*, 2018.

Zhou, H.-Y., Oliver, A., Wu, J., and Zheng, Y. When semi-supervised learning meets transfer learning: Training strategies, models and datasets. In *Technical Report*, 2018.

A   APPENDIX

A.1   SCADS Embeddings details

We learn *SCADS embeddings* using the *expanded retrofitting technique* introduced by (Faruqui et al., 2015; Speer & Chin, 2016), which expresses both the network topology (i.e., relations among concepts) and word embeddings learned from text, i.e., word2vec (Mikolov et al., 2013a;b). In a nutshell, for each concept q, the method learns an embedding $\hat{e}_q$ such that it is close to the original word vector e_q and to the neighborhood $\mathcal{N}$ of q in the graph. Formally, the objective function $\Psi(Q)$ to minimize is:

$$\Psi(Q) = \sum_{i=i}^{n} \left[\alpha_i \left\| e_i - \hat{e}_i \right\|^2 + \sum_{(i,j)\in\mathcal{N}} \beta_{ij} \left\| \hat{e}_i - e_j \right\|^2 \right] \tag{8}$$

where α_i indicates the importance of the old word embedding, and β_{ij} is the edge weight. Following (Speer et al., 2017), we set $\alpha = 0$ to handle out-of-vocabulary concepts.

A.2   Experimental setting details

As explained in Section 4.3, for each of the experimental datasets, except Grocery Store Dataset since it already has a pre-determined test set, we first partition the dataset into a train-test split by randomly take a fixed number of images per class from the pool of data to use as the test set and assign the rest for the train set. The number of test images per class is 5, 10 and 10 for FMD and OfficeHome-Product, and OfficeHome-Clipart respectively. Next, we randomly choose a fixed number of images per class (1, 5, or 20) in the train partition to be labeled and use the rest of the train images as unlabeled images. We do not report the 20-shot results of Grocery Store Dataset because the minimum number of images per class for this dataset is 18. We use the same seed for both partitioning a dataset into train-test split and subsequently choosing train images in the dataset to be labeled, and we report the results of three different seeds in Tables 3 and 4. The results are reported as 95% confidence interval over 3 different training seeds, which affects the initialization of linear layers we append to the backbones and the data shuffling during training).

A.3   Training details

For data augmentation during training, we apply random resized crop, where the cropping scale is randomly chosen between 0.8 and 1, and random horizontal flip. All images are resized to 224x224 and normalized with ImageNet1k's statistics, both are community standards. For all modules, except ZSL-KG since it is a zero-shot module, the classifier head(s) appended to ResNet-50 (ImageNet-1k) is a single fully-connected layer, while following (Kolesnikov et al., 2020), the classifier head(s) appended to BiT (ImageNet-

21k) is a 2-D convolutional layer with kernel size equal to 1x1. The training details of individual modules follow.

Transfer Module. The batch size is set to 256. The optimizer is SGD with a learning rate of 0.003 and a momentum of 0.9. When using ResNet-50 (ImageNet-1k), we first fine-tune the model on the selected auxiliary data for 5 epochs, and then we fine-tune the model on the target data for 40 epochs with the learning rate decayed by 0.1 at epoch 20 and 30. When using BiT (ImageNet-21k), we fine-tune the model on the selected auxiliary data for 2000 steps, and then we fine-tune the model on the target data for 500 steps. Linear learning rate warmup is used for the first 100 and 500 steps when the training steps are 500 and 2000 respectively, and then we decay the learning rate at step (200, 300, 400) and (900, 1300, 1700, 2000) by 0.1 when the training steps are 500 and 2000 respectively.

Multi-task Module. The batch size is set to 128. The optimizer is SGD with a learning rate of 0.003 and a momentum of 0.9. When using ResNet-50 (ImageNet-1k), we train the model for 8 epochs conditioned on the selected auxiliary data, and we decay the learning rate by 0.1 at epoch 4 and 6. When using BiT (ImageNet-21k), we train the model for 2000 steps. Linear learning rate warmup is used for the first 500 steps, and then we decay the learning rate at step 900, 1300, and 1700 by 0.1.

FixMatch Module.   We first pretrain the ResNet-50 (ImageNet-1k) or BiT (ImageNet-21k) backbone on selected auxiliary data for five epochs using an SGD optimizer. During pretraining, we set the learning rate to 0.003, momentum is set to 0.9, and the batch size is set to 256. After pretraining, we follow the training procedures outlined in (Sohn et al., 2020). We use an SGD optimizer with Nesterov momentum. The optimizer's learning rate is set to 0.0005, and its momentum is set to 0.9. The model is trained on labeled and unlabeled data for 30 epochs when using ResNet-50 (ImageNet-1k) and 500 epochs when using BiT (ImageNet-21k). Both backbones use a batch size of 128. After each batch, the learning rate is set according to the cosine learning decay rate $\eta \cos\left(\frac{7\pi k}{16K}\right)$, where η is the initial learning rate, k is the current batch number, and K is the total number of batches seen during training.

ZSL-KG Module. In our experiments, we follow the object classification implementation from Nayak & Bach (2020) [1]. We pretrain ZSL-KG by minimizing the L2 distance between the learned class representations and the weights of the fully connected layer from a pretrained classifier:

$$\mathcal{L}_Z = \frac{1}{n} \sum_{i}^{n} \left(w_i - z_i \right)^2 \tag{9}$$

where w_i is the pretrained weight from the classifier and z_i

---

[1]https://github.com/BatsResearch/nayak-arxiv20-code

Method	Backbone	OfficeHome-Product			OfficeHome-Clipart		
		1-shot	5-shot	20-shot	1-shot	5-shot	20-shot
Split 1							
Fine-tuning	BiT (ImageNet-21k)	62.56 ± 2.10	84.72 ± 0.88	91.85 ± 0.38	31.85 ± 4.63	65.95 ± 0.58	$\mathbf{80.67 \pm 0.58}$
Fine-tuning (Distilled)	BiT (ImageNet-21k)	62.56 ± 2.54	$\mathbf{85.74 \pm 0.58}$	$\mathbf{92.51 \pm 0.44}$	31.18 ± 2.46	65.44 ± 0.96	78.51 ± 1.89
FixMatch	BiT (ImageNet-21k)	56.87 ± 3.11	83.95 ± 0.96	91.64 ± 1.17	28.87 ± 9.40	64.56 ± 0.58	81.59 ± 1.23
Meta Pseudo Label	BiT (ImageNet-21k)	61.28 ± 1.12	80.97 ± 2.60	90.77 ± 1.38	31.64 ± 0.96	62.67 ± 1.72	76.15 ± 1.01
TAGLETS	BiT (ImageNet-21k)	$\mathbf{74.67 \pm 1.45}$	$\mathbf{85.85 \pm 0.66}$	91.95 ± 0.88	$\mathbf{43.64 \pm 5.52}$	$\mathbf{68.05 \pm 1.45}$	80.41 ± 3.67
Fine-tuning	ResNet-50 (ImageNet-1k)	34.72 ± 1.10	64.97 ± 0.88	84.41 ± 2.49	17.03 ± 1.89	43.85 ± 3.27	69.64 ± 3.11
Fine-tuning (Distilled)	ResNet-50 (ImageNet-1k)	35.95 ± 2.46	65.44 ± 3.85	85.79 ± 0.58	16.72 ± 1.81	45.13 ± 2.49	70.92 ± 2.13
FixMatch	ResNet-50 (ImageNet-1k)	43.74 ± 0.80	74.31 ± 1.67	85.90 ± 0.58	22.51 ± 1.17	51.38 ± 3.03	67.03 ± 0.22
Meta Pseudo Label	ResNet-50 (ImageNet-1k)	51.69 ± 2.89	76.76 ± 1.38	89.75 ± 0.88	23.48 ± 1.59	56.10 ± 3.18	$\underline{75.40 \pm 1.96}$
TAGLETS	ResNet-50 (ImageNet-1k)	$\underline{70.92 \pm 3.13}$	$\underline{82.67 \pm 1.10}$	$\underline{90.72 \pm 0.96}$	$\underline{43.28 \pm 0.96}$	$\underline{64.77 \pm 2.51}$	$\underline{75.64 \pm 2.68}$
TAGLETS prune-level 0	ResNet-50 (ImageNet-1k)	$\underline{68.87 \pm 2.46}$	$\underline{81.59 \pm 0.58}$	90.51 ± 1.81	$\underline{42.82 \pm 1.45}$	60.10 ± 2.10	$\underline{76.05 \pm 2.17}$
TAGLETS prune-level 1	ResNet-50 (ImageNet-1k)	64.00 ± 3.40	77.59 ± 1.23	88.67 ± 0.80	37.69 ± 2.29	53.23 ± 2.39	74.00 ± 1.75
Split 2							
Fine-tuning	BiT (ImageNet-21k)	57.49 ± 1.54	$\mathbf{82.51 \pm 1.23}$	$\mathbf{90.56 \pm 0.58}$	32.87 ± 3.43	62.87 ± 1.10	$\mathbf{78.15 \pm 1.32}$
Fine-tuning (Distilled)	BiT (ImageNet-21k)	56.62 ± 3.99	$\mathbf{82.77 \pm 1.75}$	$\mathbf{90.77 \pm 0.66}$	32.92 ± 5.88	62.87 ± 2.10	$\mathbf{78.10 \pm 0.44}$
FixMatch	BiT (ImageNet-21k)	48.15 ± 1.91	$\mathbf{82.41 \pm 1.81}$	$\mathbf{90.46 \pm 0.76}$	9.59 ± 6.09	63.23 ± 2.65	$\mathbf{79.23 \pm 1.38}$
Meta Pseudo Label	BiT (ImageNet-21k)	57.79 ± 0.58	78.61 ± 1.52	88.21 ± 0.58	26.46 ± 3.90	54.31 ± 2.12	75.12 ± 0.79
TAGLETS	BiT (ImageNet-21k)	$\mathbf{70.00 \pm 0.76}$	$\mathbf{84.67 \pm 2.30}$	$\mathbf{90.56 \pm 1.72}$	$\mathbf{45.79 \pm 4.19}$	$\mathbf{66.62 \pm 2.13}$	$\mathbf{79.08 \pm 1.32}$
Fine-tuning	ResNet-50 (ImageNet-1k)	34.36 ± 3.45	63.54 ± 3.50	82.10 ± 0.96	18.92 ± 1.99	41.69 ± 2.39	70.72 ± 3.55
Fine-tuning (Distilled)	ResNet-50 (ImageNet-1k)	35.49 ± 3.83	65.90 ± 3.25	84.62 ± 0.66	20.26 ± 0.44	43.85 ± 4.63	72.10 ± 2.97
FixMatch	ResNet-50 (ImageNet-1k)	43.38 ± 3.65	72.15 ± 0.76	84.56 ± 2.10	24.05 ± 1.81	49.38 ± 3.40	69.85 ± 2.39
Meta Pseudo Label	ResNet-50 (ImageNet-1k)	50.87 ± 2.10	74.87 ± 2.76	88.20 ± 1.45	21.28 ± 1.22	44.41 ± 1.45	70.41 ± 2.10
TAGLETS	ResNet-50 (ImageNet-1k)	$\underline{67.54 \pm 0.00}$	$\underline{84.62 \pm 1.38}$	$\underline{89.90 \pm 1.23}$	$\underline{43.95 \pm 2.10}$	$\underline{63.18 \pm 1.45}$	$\underline{78.31 \pm 3.03}$
TAGLETS prune-level 0	ResNet-50 (ImageNet-1k)	65.33 ± 0.58	82.67 ± 1.92	$\underline{89.64 \pm 0.22}$	$\underline{43.59 \pm 1.96}$	59.33 ± 1.10	77.18 ± 2.54
TAGLETS prune-level 1	ResNet-50 (ImageNet-1k)	60.56 ± 1.59	78.00 ± 1.53	$\underline{88.82 \pm 0.80}$	39.59 ± 3.45	55.08 ± 1.99	$\underline{75.49 \pm 1.17}$

Table 3. Accuracy (%) of TAGLETS and baselines on *OfficeHome Product* and *Clipart* (`Split 1 and 2`) for increasing number of labeled examples (1, 5, and 20 shots), different backbones (ResNet-50 trained on ImageNet-1k and BiT trained on ImageNet-21k) and levels of pruning (no-pruning, 0 and 1). We make bold and underline the best performances, and all those within their 95% confidence intervals, carried out using **BiT** and <u>ResNet</u> backbones, respectively. Measurements are repeated for three seeds and we report the 95% confidence intervals.

is the class representation from the ZSL-KG module for the i-th class. Following Nayak & Bach (2020), we train ZSL-KG on the ConceptNet graph. To train the ZSL-KG module with the ConceptNet graph, we use the fully connected layer weights from ResNet101 pretrained on ILSVRC2012 (Russakovsky et al., 2015a) obtained from Torchvision (Marcel & Rodriguez, 2010). The module is trained for 1000 epochs on a random 950/50 class split where 950 classes are used for training while the rest are used for validation. During the pretraining, we use the Adam optimizer with a learning rate of $1e - 03$ and weight decay of $5e - 04$. The model checkpoint with the least loss on the validation classes is used in SCADS.

End Model The batch size is set to 256. When using ResNet-50 (ImageNet-1k), we train the model for 30 epochs using Adam optimizer with learning rate 0.0005 and weight decay 0.0001. The learning rate is decayed by 0.1 at epoch 20. When using BiT (ImageNet-21k), we train the model for 500 steps. The optimizer is SGD with a learning rate of 0.003 and a momentum of 0.9. Linear learning rate warmup is used for the first 100 steps, and then we decay the learning rate at step 200, 300, and 400 by 0.1.

Meta Pseudo Labels As mentioned in Section 4.2, Meta Pseudo Labels is comprised of a teacher model and a student model. The teacher has either a ResNet-50 (ImageNet-1k) or BiT (ImageNet-21k) backbone, and the student has ResNet-50 (ImageNet-1k) backbone. During training, both the student and teacher use an SGD optimizer with momentum set to 0.9. The teacher's initial learning rate is set to 0.0005, and the student's initial learning rate is set to 0.001 for all datasets except for Grocery Store, which uses a learning rate of 0.0001 to mitigate early convergence. The student and teacher are trained for 500 steps using a batch size of 128. The learning rate of both models are set using according to the cosine learning decay rate $\frac{\eta}{2}\left(1 + \cos\frac{\pi k}{K}\right)$, where η is the initial learning rate, k is the current batch number, and K is the total number of batches seen during training. After teacher-student training, the student is fine-tuned for 30 epochs using SGD with momentum set to 0.9 and a fixed learning rate of 0.003.

Method	Backbone	Grocery Store Dataset		Flickr Material Dataset		
		1-shot	5-shot	1-shot	5-shot	20-shot
Split 1						
Fine-tuning	BiT (ImageNet-21k)	60.52 ± 4.55	84.02 ± 1.06	42.47 ± 1.88	76.27 ± 3.86	$\mathbf{87.40 \pm 0.86}$
Fine-tuning (Distilled)	BiT (ImageNet-21k)	60.56 ± 5.02	85.42 ± 2.61	45.07 ± 2.45	$\mathbf{77.93 \pm 0.29}$	$\mathbf{87.67 \pm 1.15}$
FixMatch	BiT (ImageNet-21k)	48.58 ± 13.41	87.61 ± 4.82	12.60 ± 11.19	70.73 ± 10.84	86.47 ± 2.07
Meta Pseudo Label	BiT (ImageNet-21k)	59.52 ± 3.96	85.58 ± 0.75	44.73 ± 5.35	75.20 ± 3.44	80.73 ± 2.45
TAGLETS	BiT (ImageNet-21k)	$\mathbf{66.26 \pm 5.60}$	$\mathbf{87.87 \pm 0.95}$	$\mathbf{60.80 \pm 3.97}$	77.60 ± 4.24	$\mathbf{86.67 \pm 1.03}$
Fine-tuning	ResNet-50 (ImageNet-1k)	34.86 ± 4.11	67.23 ± 0.80	25.33 ± 2.83	41.53 ± 2.91	61.93 ± 3.86
Fine-tuning (Distilled)	ResNet-50 (ImageNet-1k)	36.31 ± 1.70	72.22 ± 2.91	26.80 ± 1.31	40.80 ± 6.88	64.80 ± 3.48
FixMatch	ResNet-50 (ImageNet-1k)	32.69 ± 1.85	71.60 ± 4.17	28.80 ± 4.24	58.80 ± 4.74	75.20 ± 1.31
Meta Pseudo Label	ResNet-50 (ImageNet-1k)	38.19 ± 5.33	75.51 ± 0.47	38.44 ± 10.20	65.13 ± 5.35	$\underline{78.07 \pm 2.50}$
TAGLETS	ResNet-50 (ImageNet-1k)	$\underline{56.45 \pm 1.90}$	$\underline{82.43 \pm 1.76}$	$\underline{51.87 \pm 3.38}$	$\underline{68.53 \pm 2.24}$	73.53 ± 1.43
TAGLETS prune-level 0	ResNet-50 (ImageNet-1k)	49.99 ± 1.85	80.24 ± 2.07	$\underline{48.53 \pm 2.29}$	62.20 ± 4.74	73.00 ± 2.28
TAGLETS prune-level 1	ResNet-50 (ImageNet-1k)	42.91 ± 9.81	73.35 ± 3.25	$\underline{50.07 \pm 5.80}$	66.67 ± 3.76	72.93 ± 1.03
Split 2						
Fine-tuning	BiT (ImageNet-21k)	60.08 ± 1.66	83.61 ± 0.71	41.07 ± 6.96	69.40 ± 2.17	$\mathbf{85.07 \pm 1.52}$
Fine-tuning (Distilled)	BiT (ImageNet-21k)	62.72 ± 2.66	$\mathbf{84.10 \pm 1.14}$	42.53 ± 6.21	70.93 ± 2.74	$\mathbf{84.53 \pm 1.03}$
FixMatch	BiT (ImageNet-21k)	60.16 ± 6.51	$\mathbf{87.08 \pm 3.30}$	12.13 ± 8.32	$\mathbf{72.40 \pm 1.31}$	$\mathbf{83.60 \pm 2.77}$
Meta Pseudo Label	BiT (ImageNet-21k)	62.12 ± 3.12	82.19 ± 0.21	48.86 ± 8.90	$\mathbf{74.06 \pm 2.01}$	78.79 ± 1.31
TAGLETS	BiT (ImageNet-21k)	$\mathbf{68.26 \pm 3.29}$	85.88 ± 0.56	$\mathbf{59.60 \pm 9.48}$	73.47 ± 2.87	$\mathbf{84.27 \pm 1.88}$
Fine-tuning	ResNet-50 (ImageNet-1k)	31.99 ± 5.14	65.78 ± 4.85	28.00 ± 4.07	44.40 ± 1.72	58.80 ± 5.16
Fine-tuning (Distilled)	ResNet-50 (ImageNet-1k)	33.17 ± 2.66	69.59 ± 3.50	27.60 ± 10.57	46.07 ± 7.03	62.73 ± 5.27
FixMatch	ResNet-50 (ImageNet-1k)	37.64 ± 1.50	68.10 ± 2.86	35.80 ± 12.41	57.40 ± 5.23	69.73 ± 2.55
Meta Pseudo Label	ResNet-50 (ImageNet-1k)	39.46 ± 0.97	70.03 ± 1.35	38.66 ± 2.57	65.66 ± 5.31	$\underline{75.46 \pm 2.99}$
TAGLETS	ResNet-50 (ImageNet-1k)	$\underline{57.45 \pm 5.66}$	$\underline{81.57 \pm 1.04}$	$\underline{53.13 \pm 6.29}$	65.87 ± 1.74	72.80 ± 2.28
TAGLETS prune-level 0	ResNet-50 (ImageNet-1k)	$\underline{55.36 \pm 6.00}$	79.38 ± 2.47	$\underline{51.67 \pm 3.31}$	64.53 ± 2.99	71.80 ± 3.48
TAGLETS prune-level 1	ResNet-50 (ImageNet-1k)	47.03 ± 8.44	74.98 ± 1.36	46.00 ± 15.30	62.00 ± 7.81	69.93 ± 2.24

Table 4. Accuracy (%) of TAGLETS and baselines on *Grocery Store* and *Flicker Material Datasets* (Split 1 and 2) for increasing number of labeled examples (1, 5, and 20 shots), different backbones (ResNet-50 trained on ImageNet-1k and BiT trained on ImageNet-21k) and levels of pruning (no-pruning, 0 and 1). We make bold and underline the best performances, and all those within their 95% confidence intervals, carried out using **BiT** and <u>ResNet</u> backbones, respectively. Measurements are repeated for three seeds and we report the 95% confidence intervals.

A.4 Additional results

In this section, we collect the experimental results on `split 1` and `split 2` of the datasets (Section 4.3). Tables 3 and 4 report the trends discussed in Section 4.4.1: (1) TAGLETS outperforms the baselines in the 1 and 5 shots settings, and competitive in the 20-shots case. (2) The system is robust to the amount on auxiliary data its backbone is pre-trained., i.e., TAGLETS with ResNet-50 (ImageNet-1k) achieves better scores than distilled fine tuning of BiT (ImageNet-21k).

Figure 8 and the pruning results in Tables 3 and 4 confirm that the more the selected auxiliary data are far from the target classes, the more the accuracy of the individual modules decreases. Moreover, as we increase the size of labeled data, the importance of relatedness between the auxiliary data and the target classes reduces. Results on the other splits of the datasets suggest the same observations.

In Figure 9, we show that the ensemble generates improvements over the average modules accuracy in all the datasets. Consistently with findings in Section 4.4.3, the relatedness of the auxiliary data to the target classes (i.e., levels of pruning) is not correlated to the effects of ensemble on the system accuracy.

B ARTIFACT APPENDIX

B.1 Abstract

Our artifact provides the source code of TAGLETS, installation instructions, and the necessary scripts to run a demo. The target task of the demo is CIFAR-10 with limited labeled data, and we solve it accessing to CIFAR-100 as the auxiliary data. To replicate the results in the paper, readers must download the datasets used for the experiments, which we refer to in Appendix B.3.4.
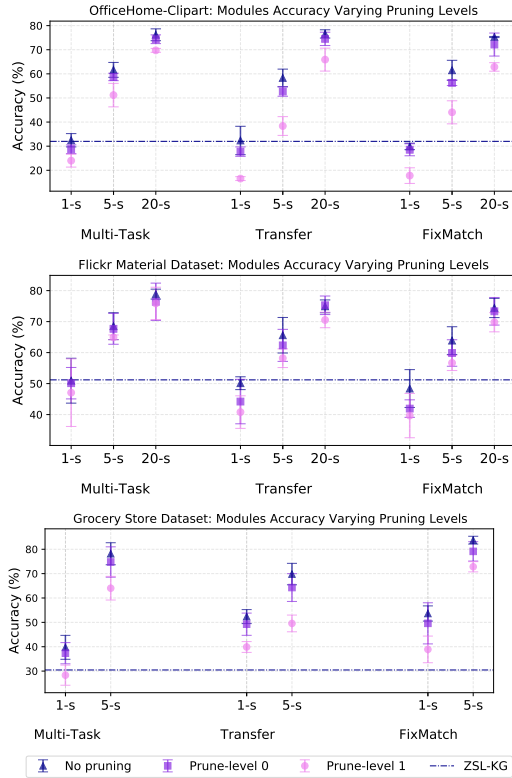
Figure 8. (`Split 0`) The plot shows the %-accuracy of each module at different levels of pruning (shapes) and for varying amount of labeled examples (x-axis), on the *OfficeHome-Clipart*, *Flick Material* and *Grocery Store* datasets. The error bars indicate 95% confidence intervals on three seeds. All modules have ResNet-50 backbone. We note that the performances of ZSL-KG module are invariant to pruning, since it is used as pre-trained model on ConceptNet.
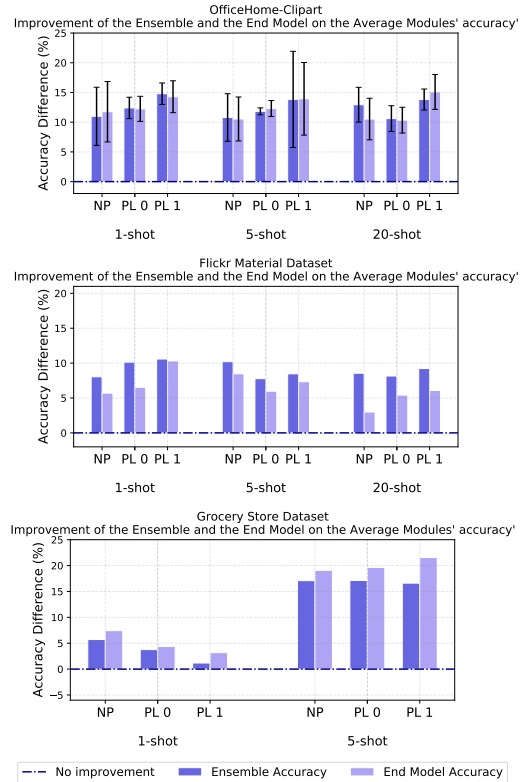

Figure 9. (`Split 0`) For the *OfficeHome-Clipart, Flickr Material*, and *Grocery Store* datasets, we fix the number of labeled examples (1, 5, and 20 shots) and plot the improvements of the ensemble and the end model on the average accuracy of the training modules, for different pruning levels. The results refer to TAGLETS with ResNet-50 backbone and the 95% confidence intervals are computed over three different seeds.

B.2 Artifact check-list (meta-information)

- **Algorithm:** Yes
- **Data set:** CIFAR-10 (demo), CIFAR-100 (demo), Office-Home (experiments), Grocery Store Dataset (experiments), Flickr Material Dataset (experiments), ImageNet21k (experiments)
- **Run-time environment:** Debian GNU/Linux with (optional) CUDA 9 and CUDNN
- **Hardware:** CPU or GPU (recommended)
- **Metrics:** Top-1 accuracy
- **Output:** Log files with accuracy
- **Experiments:** We provide instructions to install the system and run a demo in the README file of the repository indicated in the following sections.
- **How much disk space required (approximately)?** 15 Gb
- **How much time is needed to complete experiments (approximately)?** 2 hours with 1 NVIDIA V100 GPUs/45 minutes with 4 NVIDIA V100 GPUs (demo)
- **Publicly available?** Yes
- **Code licenses (if publicly available)?** Apache License 2.0
- **Archived (provide DOI)?** 10.5281/zenodo.6080805

B.3 Description

B.3.1 How delivered

Two open-sourced versions:

- `taglets.zip` on Zenodo.
  https://doi.org/10.5281/zenodo.6080805
- `BatsResearch/taglets`, public git repository, latest version.
  https://github.com/BatsResearch/taglets

B.3.2 Hardware dependencies

The CPUs or GPUs in use should support Python 3.7 and Pytorch 1.7.

B.3.3 Software dependencies

- Python (3.7)

- PyTorch (1.7)

B.3.4   Data sets

In the demo, we use CIFAR-10 and CIFAR-100 datasets. The GitHub repository includes the bash scripts to download them. You can download the datasets used in the paper here:

- *OfficeHome*: `https://www.hemanthdv.org/officeHomeDataset.html`
- *Grocery Store Dataset*: `https://github.com/marcusklasson/GroceryStoreDataset`
- *Flickr Material Dataset*: `https://people.csail.mit.edu/lavanya/fmd.html`

B.4   Installation

TAGLETS is implemented in Python. Once the software requirements are satisfied (Appendix B.3.3), you can run the following script to install the system. We note that these instruction are also reported in the GitHub README file.

Clone this repository:

```
git clone

https://github.com/BatsResearch/taglets.git
```

Before installing TAGLETS, we recommend creating and activating a new virtual environment which can be done by the following script:

```
python -m venv taglets_venv
source taglets_venv/bin/activate
```

You also want to make sure 'pip' is up-to-date:

```
pip install --upgrade pip
```

Then, to install TAGLETS and download related files, run:

```
bash setup.sh
```

B.5   Experiment workflow

We invite the reader to follow the workflow presented in the GitHub README file to run the demo. We recommend running it with GPUs to speed up the training.

B.6   Evaluation and expected result

For the demo, please follow the instructions on the README to run it. The fine-tuning baseline for the demo task is 41.5% accuracy. The demo is expected to significantly outperforms this baseline.

To replicate results on this paper, you first have to download the datasets used in the paper, including OfficeHome, Grocery Store, Flickr Material, and ImageNet-21k. We have provided the script to download the SCADS file used in the paper (installed with ImageNet-21k). Then, set the hyper-parameters to be the same as the ones used in this paper. Finally, run TAGLETS with all four modules used in this paper. You can expect the same results as presented in Tables 1 and 2 of the main body of the paper.

B.7   Experiment customization

TAGLETS can be used as a Python package where you can import it and interact with its interface. The two key classes of TAGLETS that users will interact with are `Controller` and `Task`.

In `Task`, there are the following argumentts:

- `input_shape` (required): image size
- `batch_size` (optional, default to 128): batch size
- `wanted_num_related_class` (optional, default to 10): number of auxiliary classes per target class

You can also choose a backbone of your choice to be used in TAGLETS by using the `set_initial_model` and `set_model_type` methods of the `Task` class.

In `Controller`, you can specify the modules you want to use in TAGLETS by setting the parameter `modules`.

You can refer to the demo on how these parameters can be specified.

B.8   Notes
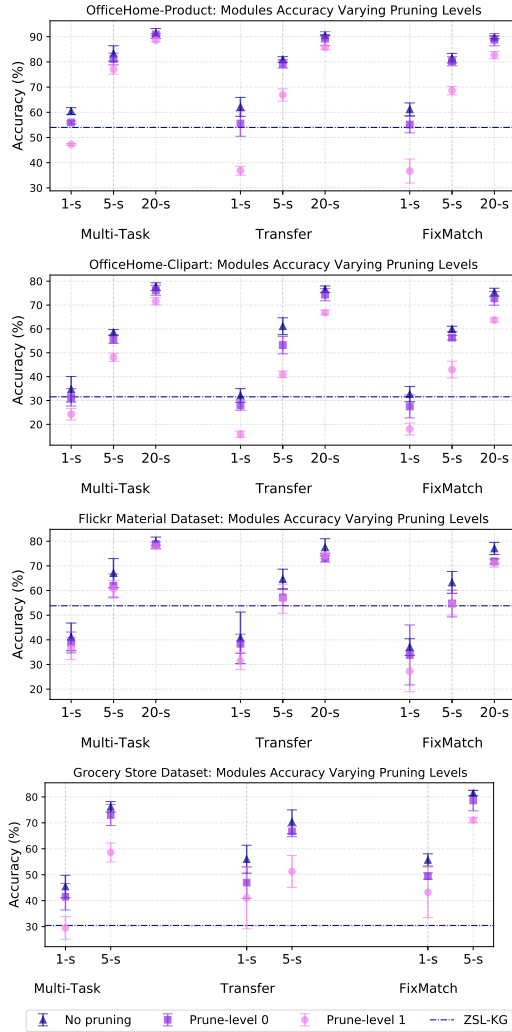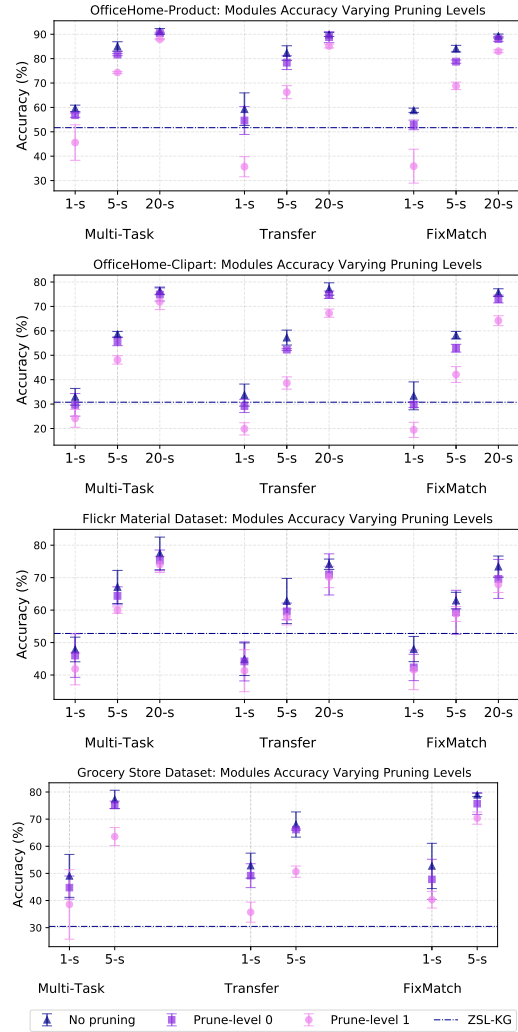
For more questions, please file issues on GitHub.

Figure 10. (Split 1) The plot shows the %-accuracy of each module at different levels of pruning (shapes) and for varying amount of labeled examples (x-axis), on the *OfficeHome-Product*, *OfficeHome-Clipart*, *Flick Material* and *Grocery Store* datasets. The error bars indicate 95% confidence intervals on three seeds. All modules have ResNet-50 backbone. We note that the performances of ZSL-KG module are invariant to pruning, since it is used as pre-trained model on ConceptNet.

Figure 11. (Split 2) The plot shows the %-accuracy of each module at different levels of pruning (shapes) and for varying amount of labeled examples (x-axis), on the *OfficeHome-Product*, *OfficeHome-Clipart*, *Flick Material* and *Grocery Store* datasets. The error bars indicate 95% confidence intervals on three seeds. All modules have ResNet-50 backbone. We note that the performances of ZSL-KG module are invariant to pruning, since it is used as pre-trained model on ConceptNet.
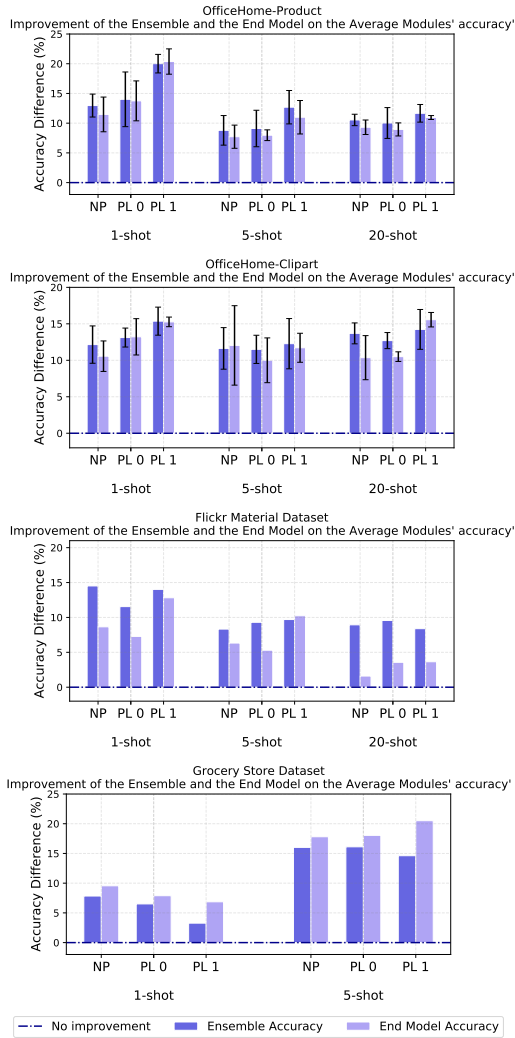
Figure 12. (`Split 1`) For the *OfficeHome-Product*, *OfficeHome-Clipart*, *Flickr Material*, and *Grocery Store* datasets, we fix the number of labeled examples (1, 5, and 20 shots) and plot the improvements of the ensemble and the end model on the average accuracy of the training modules, for different pruning levels. The results refer to TAGLETS with ResNet-50 backbone and the 95% confidence intervals are computed over three different seeds.
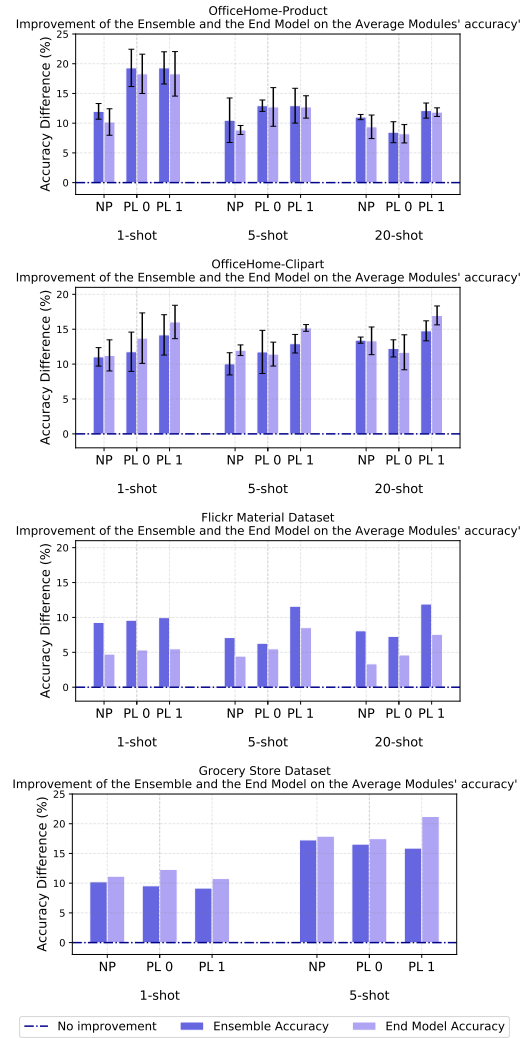
Figure 13. (`Split 2`) For the *OfficeHome-Product*, *OfficeHome-Clipart*, *Flickr Material*, and *Grocery Store* datasets, we fix the number of labeled examples (1, 5, and 20 shots) and plot the improvements of the ensemble and the end model on the average accuracy of the training modules, for different pruning levels. The results refer to TAGLETS with ResNet-50 backbone and the 95% confidence intervals are computed over three different seeds.