Learning with an Evolving Class Ontology

Anonymous Author(s)
Affiliation
Address
email

Abstract

Lifelong learners must recognize concept vocabularies that evolve over time. This is often studied under the formalism of class-incremental learning, which continuously grows the vocabulary of concepts over time. A common yet underexplored scenario is learning with class labels that refine/expand old classes. For example, humans learn to recognize `dog` before dog breeds. In practical settings, dataset *versioning* often introduces refinement to ontologies, such as autonomous vehicle benchmarks that refine a previous `vehicle` class into `school-bus` as autonomous operations expand to new cities. This paper formalizes a protocol for studying the problem of *Learning with Evolving Class Ontology* (LECO). LECO requires learning classifiers in distinct time periods (TPs); each TP introduces a new ontology of "fine" labels that refines old ontologies of "coarse" labels (e.g., dog breeds that refine the previous `dog`). LECO explores such questions as whether to annotate new data or relabel the old, how to leverage coarse labels, and whether to finetune the previous TP's model or train from scratch. To answer these questions, we leverage insights from class-incremental learning, semi-supervised learning, and learning with partial labels. We validate them under the LECO protocol through the lens of image classification on CIFAR and iNaturalist. Extensive experiments lead to some surprising conclusions; while the current status quo in the field is to relabel existing datasets with new class ontologies (such as, COCO-to-LVIS or Mapillary1.2-to-2.0), LECO demonstrates that a far better strategy is to annotate *new* data with the new ontology. However, this produces an aggregate dataset with inconsistent old-vs-new labels, and thus complicating learning. We adopt methods from semi-supervised and partial-label learning literature to show that such strategies can surprisingly be made near-optimal, in the sense of approaching the "oracle" upper bound that learns on the aggregate dataset exhaustively labeled with the most recent ontology.

1 Introduction

We humans, as lifelong learners, learn to recognize the ontology of concepts which is being refined and expanded gradually over time. For example, we learn to recognize `dog` and then dog breeds (refining the class `dog`). The class ontology often evolves from coarse concepts to fine-grained ones but can sometimes involve introducing new classes. We call this *Learning with Evolving Class Ontology* (LECO).

Motivation. LECO is common when deploying machine-learning solutions in practice. One often trains and maintains machine-learned models with class labels (or a class ontology) that are refined over time periods (TPs) (Fig. 1). Such ontology evolution can be caused by new requirements in applications, such as autonomous vehicles that expand operations to new cities. This is well demonstrated in many contemporary large-scale datasets that release updated versions by refining and expanding classes, such as Mapillary [44, 47], Argoverse [12, 65], KITTI [4, 21] and iNaturalist [28–

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

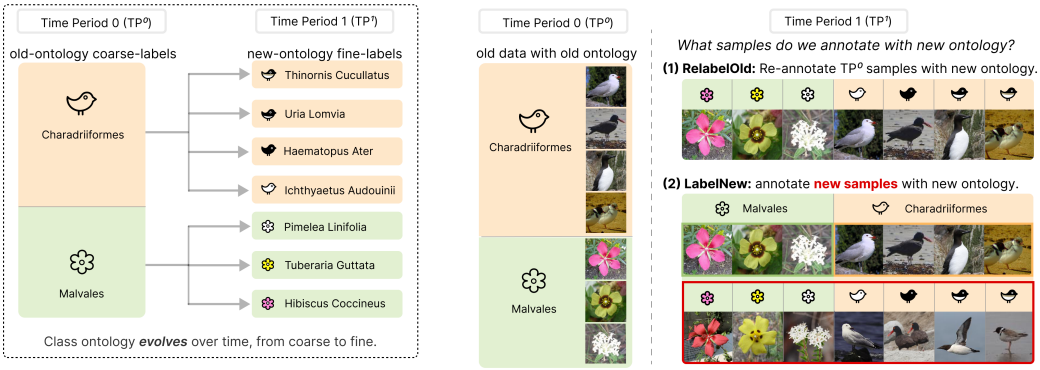

Figure 1: **Left:** Learning with Evolving Class Ontology (LECO) requires training models in time periods (TPs) that refine old ontologies in a coarse-to-fine fashion. This leads to a basic question: for the next TP, should one relabel the old data or label new data? Interestingly, both labeling protocols have been used in the community for large-scale datasets. **Right:** To study this question, we repurpose CIFAR100 and iNaturalist [28–32, 60, 64] (shown here) for LECO, and provide the (perhaps obvious) answer – one should always annotate *new* data with the *new* ontology rather than re-annotating the *old*. One reason is that the former produces more labeled data. But this produces an aggregate dataset with inconsistent old-vs-new labels. We make use of insights from semi-supervised learning and learning-with-partial-labels to learn from such heterogenous annotations, approaching the upper bound of learning from an oracle aggregate dataset with all new labels.

32, 60, 64]. For example, Mapillary-V1.2 [47] defined the `road` class, and Mapillary-V2.0 [44] refined it with fine labels including `parking-aisle` and `road-shoulder`; Argoverse-V1.0 [12] defined the `bus` class, and Argoverse-V2.0 [65] refined it with fine classes including `school-bus`. From a biological perspective, lifelong learning via a curriculum of concepts is widely studied in cognition, behavior and psychology [37, 49, 58]. While curriculum learning often studies the problem of designing a schedule of concept exposure so as to maximize learning [5, 19, 54], LECO's curriculum is *given* by evolving application demands.

Prior art. LECO requires learning new classes continuously in TPs, so it is closely related to class-incremental learning (CIL) [16, 39, 63, 71]. They differ in important ways. First, classic CIL assumes *brand-new* labels have no relations with the old [39, 41, 52], while LECO's new labels are fine-grained ones that expand the old / "coarse" labels (cf. Fig. 1). Interestingly, CIL can be seen as a special case of LECO by making use of a "catch-all" background class [35] (often denoted as `void` or `unlabeled` in several existing datasets [13, 20, 41, 44, 47]), which can be *refined* over time to include specific classes that were previously ignored. Second, CIL restricts itself to a small memory buffer for storing historical data to focus on information retention and catastrophic forgetting [10, 39]. However, LECO stores all historical data (since large-scale human annotations typically come at a much higher cost than that of the additional storage) and is concerned with the most-recent classes of interest. Most related to LECO are approaches that explore the relationship of new classes to old ones; Sariyildiz *et al.* [55] study concept similarity but remove common superclasses in their study, while Abdelsalam *et al.* [1] explore CIL with a goal to *discover* inter-class relations on data labeled (as opposed to assuming such relationships can be derived from a given ontology hierarchy). To explore LECO, we set up its benchmarking protocol (Section 3) and study extensive methods modified from those of related problems (Sections 4, 5, and 6).

Technical insights. LECO studies learning strategies and answers questions as fundamental as whether one should label new data or relabel the old data[1]. Interestingly, both protocols have been used in the community for large-scale dataset versioning: Argoverse [12, 65] annotates new data, while Mapillary [44, 47] relabels its old data. Our experiments provide the (perhaps obvious) answer – one should always annotate *new* data with the *new* ontology rather than reannotating the *old* data. The simple reason is that the former produces more labeled data. However, this leads to the obvious question: how can one leverage the old data annotated with the old ontology to help improve new-

[1]We assume the labeling cost is same for either case. One may think re-annotating old data is cheaper, because one does not need to curate new data and may be able to redesign an annotation interface that exploits the old annotations (e.g., show old `vehicle` labels and just ask for fine-grained make-and-model label). But in practice, in order to prevent carry-over of annotation errors and to simplify the annotation interface, most benchmark curators tend to relabel from scratch (regardless of on old or new data). Examples of relabeling old-data from scratch include Mapillary V1.2 [47] → V2.0 [44], and COCO [40] → LVIS [24].

Table 1: **Datasets statistics**. To explore LECO, we repurpose two datasets: CIFAR100 [36] and iNaturalist [60, 64]. We set two time periods (TPs): TP^0 and TP^1 define their own ontologies of class labels, which are expanded from coarse to fine, from TP^0 to TP^1. See Section 3 for details.

dataset	Time Period 0 (TP^0)			Time Period 1 (TP^1)		
	#classes	#train	#test	#classes	#train	#test
CIFAR-LECO	20	1k/10k	10k	100	1k/10k	10k
iNat-LECO	123	50k	4k	810	50k	4k

ontology models? We find that joint-training [39] on both new and old ontologies works quite well, if performed in a certain way. We tackle this through variants of semi-supervised learning (SSL). Concretely, we generate new-ontology labels for the old data using pseudo labeling, a popular building block of state-of-the-art SSL algorithms [38, 50, 59, 66]. For the old data that has coarse labels from the old-ontology, we use it as "coarse supervision" to reconcile conflicts between the pseudo fine labels and the true coarse labels (Section 6). There is another obvious question: should one finetune the previous TP's model or train from scratch? One may think the former has no benefit (because it accesses the same training data) and might suffer from local minima. We find it actually works better, echoing curriculum learning [5, 19, 54]. In summary, under the LECO protocol of the iNaturalist benchmark (Section 3), (1) labeling new data outperforms relabeling the old: 73.3% vs. 70.9% in accuracy; (2) jointly training on both old and new data boosts accuracy to 83.8%; (3) taking advantage of the relationship between old and new labels (via Learning-with-Partial-Labels and SSL) further improves to 84.0%, close to an upper-bound (84.7%) that trains on *all* available data re-annotated with new ontology. We find consistent results across other datasets (CIFAR100).

Contributions. We make three major contributions. First, we motivate the problem LECO and define its benchmarking protocol. Second, we extensively study approaches to LECO by modifying methods from related problems, including semi-supervised learning, class-incremental learning, and learning with partial labels. Third, we draw consistent conclusions and technical insights, as previewed above.

2 Related Work

Class-incremental learning (CIL) [16, 39, 63, 71] and LECO are closely related as both learn on vocabularies that grow in time periods (TPs). They have important difference as detailed in the last section. First, the typical CIL setup assumes new labels have no relations with the old [1, 39, 41], while LECO's new / "fine" labels refine or expand the old / "coarse" labels (cf. Fig. 1). Second, CIL purposely sets a small memory buffer to store history data and evaluates accuracy over both old and new classes. Its setup emphasizes information retention or catastrophic forgetting [10, 39]. However, LECO preserves all history data (whose labels usually come at a high cost) and concern the classes of interest in the current TP. Therefore, to approach LECO, we do not apply CIL methods that focuses on information retention via weight consolidation [34, 68] or knowledge distillation [39]. Instead, we repurpose simple CIL methods for LECO including finetuning [3, 67], joint training [11, 39, 42, 43], and freezing the backbone [3, 18, 57]. We find the former two work well in LECO although they are usually not treated as typical CIL methods (because they require a large buffer to store history data).

Semi-supervised learning (SSL) learns over both labeled and unlabeled data. State-of-the-art SSL methods follow a self-training paradigm [46, 56, 66] that uses a model trained over labeled data to pseudo label the unlabeled examples, which are then used together with labeled data for training. For example, Lee *et al.* [38] use confident predictions as target labels for unlabeled data, and Rizve *et al.* [53] further incorporate low-probability predictions as negative pseudo labels. Some others improve SSL using self-supervised learning techniques [22, 69], which force predictions to be similar for different augmentations of the same data [6, 7, 59, 66, 70]. Inspired by SSL, in a TP of LECO, we first ignore the coarse labels of old examples and use SSL methods to psuedo-label them with new-ontology. We further propose to exploit the old-ontology to filter the inconsistent pseudo labels.

Learning with partial labels (LPL) tackles the case when some examples are fully labeled while others are only partially labeled (e.g., in the form of coarse-grained labels as opposed to fine-grained labels) [9, 14, 23, 45, 48, 72]. For example, Grandvalet *et al.* [23] formalize the LPL problem in the probabilistic framework and proposes a minimum entropy based solution. Nguyen and Caruana [48] propose an efficient algorithm to learn classifiers from partial labels within the max-margin

framework. Cour *et al.* [14] address desirable properties of partial labels that allow effective learning. In a TP of LECO, the old data from previous TPs can be used as partially-labeled examples as they contain only coarse labels. Therefore, we expect LPL methods to improve model training in LECO. Furthermore, rather than simply exploring LPL methods, we also use state-of-the-art semi-supervised learning techniques to better exploit partial labels [50]. Concretely, we generate pseudo fine-grained labels for the old data, and filter the pseudo labels that conflict the true coarse labels.

3 Problem Setup of Learning with Evolving Class Ontology (LECO)

Notations. There are T time periods (TPs). At TP^t, we have data distributions $\mathcal{D}^t = \mathbf{X} \times \mathbf{Y}^t$, where $\mathbf{X}$ is the input domain[2] and $\mathbf{Y}^t$ is the label set. The evolution of class ontology is modeled as a tree structure: the ith class $\mathbf{y}_i^t \in \mathbf{Y}^t$ is a tree node that has a unique parent $\mathbf{y}_j^{t-1} \in \mathbf{Y}^{t-1}$. However, $\mathbf{y}_j^{t-1}$ can be split into multiple fine classes in TP^t. Therefore, $|\mathbf{Y}^t| > |\mathbf{Y}^{t-1}|$. TP^t concerns classification performance w.r.t label set $\mathbf{Y}^t$.

Benchmarks. We define benchmarks using established datasets (Fig. 1): CIFAR100 [36] and iNaturalist [60, 64]. CIFAR100 is released under the MIT license and iNaturalist is publicly available for non-commercial research and educational purposes. CIFAR100 have classes related to person. This might be a concern related to fairness and privacy. Therefore, we cautiously proceed our research and will release code under the MIT License without re-distributing the data. See Table 1 for dataset statistics. For each benchmark, we sample data from the corresponding dataset to construct two time periods (TPs): TP^0 and TP^1 define their own ontologies of class labels, $\mathbf{Y}^0$ and $\mathbf{Y}^1$, respectively. For *CIFAR-LECO*, we use the two-level hierarchy offered by the original CIFAR100 dataset [36]: using the 20 superclasses as the ontology in TP^0, each of them is split into 5 subclasses as new ontology in TP^1. We randomly sample 1k/2k/10k without replacement from the training set of CIFAR100 as the training images for each TP, and use the original validation set (10k samples) as our test-set to benchmark methods. For *iNat-LECO*, we used the most recent version of Semi-iNat-2021 [62] that comes with rich taxonomic labels at seven-level hierarchy. We choose the third level (i.e., "order") with 123 categories for TP^0, and the seventh ("species") with 810 categories for TP^1. Note that classes in iNat-LECO follow long-tailed distributions. For both benchmarks, we sample 20% data from the training set (in each TP) as the validation set for model selection and hyperparameter tuning.

The benchmarks do not construct more TPs. We focus on exploring learning strategies of LECO by improving accuracies in TP^1. We believe conclusions still hold with more TPs but leave the exploration as future work. Moreover, in TP^1, the LECO benchmark allows one to annotate either old or new data using new ontology. It assumes the same annotation cost in either way. It further allows one to leverage all the old data, unlike typical continual learning that sets a small buffer size to store history data. Using all the labeled data is realistic, because, for example, safety-critical systems (e.g., autonomous vehicles) must not trade off recognition accuracy with parsimonious memory buffer.

Remark. Our work considers ontology evolution as that old classes are refined/split into new fine-grained classes. Other class changes are possible in the real world such as merging classes and renaming a class. For these, trival solutions exist such as merging fine-grained classification scores and changing one-hot labels. Therefore, we focus on the more interesting scenario, class splitting, as the ontology evolution.

Metric. Our benchmarks study LECO through the lens of image classification. Therefore, we evaluate methods using mean accuracy (mAcc) averaged over per-class accuracies. This treats all classes equally, avoiding the bias towards common classes due to the natural long-tailed class distribution (cf. iNaturalist [60, 64]), see data distribution of iNat-LECO in the supplement. We do not use specific algorithms to address the long-tailed recognition problem but instead adopt the recently technique to improve performance by tuning weight decay [2]. We run each method five times using random seeds, and report their averaged mAcc with standard deviations.

4 Baseline Approaches to LECO

Because LECO and class-incremental learning (CIL) both learn with new classes, we first repurpose CIL (and continual learning) methods as baselines to approach LECO.

[2]The data distribution may not be necessarily static where LECO can still happen, as discussed in Section 1. We emphasize evolving class ontology and so simplify the problem by setting a static data distribution.

Preliminaries. Following prior art [33, 39], we train convolutional neural networks (CNNs) and treat a network as a feature extractor f (parametrized by θ_f) plus a classifier g (parameterized by θ_g). The feature extractor f consists of all the layers below the penultimate layer of ResNet [25], and the classifier g is the linear classifier followed by softmax normalization. Specifically, in TP^t, an input image x is fed into f to obtain its feature representation $z = f(x; \theta_f)$. The feature z is then fed into g for the softmax probabilities, $q = g(z; \theta_g)$, w.r.t classes in $\mathbf{Y}^t$. We train CNNs by minimizing the Cross Entropy (CE) loss using mini-batch SGD. At TP^t, to construct a training batch $\mathcal{B}^t$, we randomly sample K examples, i.e., $\mathcal{B}^t = \{(x_1, y_1^t), \cdots, (x_K, y_K^t)\}$. The CE loss on $\mathcal{B}^t$ is

$$\mathcal{L}(\mathcal{B}^t) = -\sum_{(x_k, y_k^t) \in \mathcal{B}^t} \mathcal{H}(y_k^t, g(f(x_k))) \tag{1}$$

where $\mathcal{H}(p, q) = -\sum_c p(c) log(q(c))$ is the entropy between two probability vectors.

Annotation Strategies. Closely related to LECO is the problem class-incremental learning (CIL) or more generally, continual learning (CL) [39, 51, 52, 68, 71]. However, unlike typical CL benchmarks, LECO does not artificially limit the buffer size for storing history samples. Indeed, the expense of hard drive buffer is less of a problem compared to the high cost of data annotation. Therefore, LECO embraces all the history labeled data, regardless of being annoated using old or new ontologies. This leads to a fundamental question: whether to annotate new data or re-label the old, given a fixed labeling budget N at each TP (cf. Fig. 1)?

- (**LabelNew**) Sample and annotate N new examples using new ontology, which trivially infers the old-ontology labels. We further have the history data as labeled with old ontology.
- (**RelabelOld**) Re-annotate the N history data using new ontology. However, these N examples are all available data although they have both old- and new-ontology labels.

The answer is perhaps obvious – **LabelNew** – because it produces more labeled data. Furthermore, **LabelNew** allows one to exploit the old data to boost performance (Section 5). This significantly reduces the performance gap with an **UpperBound** (as known as the cumulative strategy in prior continual learning method [42]), which trains over both old and new data annotated with new-ontology (at a doubled cost of annotation).

Training Strategies. We consider CL baseline methods as LECO baselines. When new tasks (i.e., classes of $\mathbf{Y}^t$) arrive in TP^t, it is desirable to transfer model weights trained for $\mathbf{Y}^{t-1}$ in TP^{t-1} [39, 52]. To do so, we initialize weights θ_{f^t} of the feature extractor f^t with $\theta_{f^{t-1}}$ trained in TP^{t-1}, i.e., $\theta_{f^t} := \theta_{f^{t-1}}$. Then we can

- (**FreezePrev**) Freeze f^t to extract features, and learn a classifier g^t over them.
- (**FinetunePrev**) Finetune f^t and learn a classifier g^t altogether.

Some other CL methods [51] do not adapt from earlier weights but train a feature extractor *from scratch* per TP. Therefore, we are inspired to

- (**TrainScratch**) Train an entire network from scratch with random weights.
- (**RandFeature**) Extract feature representations with a randomly initialized feature extractor f^t and only learn a classifier g^t.

Results. We experiment with different combinations of data annotation and training strategies for *CIFAR-LECO* and *iNat-LECO* in Table 2. We adopt standard training techniques including SGD with momentum, cosine annealing learning rate schedule, weight decay, RandAugment [15], etc. We ensure the maximum training epochs (2000/300 on CIFAR/iNat respectively) to be large enough for convergence. We also search for the learning rate and weight decay [2] via validation. Please refer to the caption of Table 2 for salient conclusions.

5 Improving LECO with the Old-Ontology Data

The previous section shows that **FinetunePrev** is the best training strategy, and **LabelNew** and **RelabelOld** achieve similar performance with the former being not exploiting the old data. In this section, we stick to **FinetunePrev** and **LabelNew** and study how to exploit the old data to improve LECO. To do so, we repurpose methods from semi-supervised learning and incremental learning.

Notation: We refer to the new-ontology (fine) labeled samples at TP^t as $S^t \subset \mathbf{X} \times \mathbf{Y}^t$, and old-ontology (coarse) labeled samples accumulated from previous TPs as $S^{1:t-1} = S^1 + \cdots + S^{t-1}$.

Table 2: **Results of baseline methods for LECO**. For all our experiments, we run each method five times and report the averaged mean per-class accuracy (mAcc in %) with standard deviations. For each benchmark, we bold the best mAcc including ties that fall within it's std. **FinetunePrev** is the best learning strategy that outperforms **TrainScratch** (cf. 73.64% vs. 65.69% with **LabelNew** on iNat-LECO). This implies that the coarse-to-fine evolved ontology serves as a good learning curriculum. **FreezePrev**, a common continual learning strategy, underperforms **TrainScratch** (cf. 50.82% vs. 65.69% with LabelNew on iNat-LECO), presumably because the representation learned with old ontology (coarse labels) are insufficient for new ontology (fine labels). **RandFeature** performs poorly simply because the random feature extractor does not produce useful features. **LabelNew** and **RelabelOld** achieve similar performance with the former using only new data but not the old altogether. If using both old and new data, we boost performance as shown in Table 3. The conclusions hold for varied number of training images, yet all methods have a significant gap to **UpperBound** (which trains on both old and new examples and assumes they have new-ontology labels), suggesting a large room for improvement.

Benchmark	#images / TP	TP^0 mAcc	TP^1 Strategy	TP^1 mAcc		
				LabelNew	RelabelOld	UpperBound
CIFAR-LECO	1000	50.92 ± 0.89	FinetunePrev	$\mathbf{33.53 \pm 0.57}$	32.85 ± 0.47	42.27 ± 0.83
			FreezePrev	24.13 ± 0.67	24.70 ± 0.41	27.82 ± 0.43
			TrainScratch	30.88 ± 1.05	31.02 ± 0.68	41.74 ± 0.56
			RandFeature	8.14 ± 0.29	8.61 ± 0.18	9.53 ± 0.24
CIFAR-LECO	10000	77.78 ± 0.27	FinetunePrev	$\mathbf{65.83 \pm 0.53}$	$\mathbf{65.30 \pm 0.32}$	71.30 ± 0.41
			FreezePrev	52.64 ± 0.50	52.60 ± 0.52	53.79 ± 0.43
			TrainScratch	$\mathbf{65.40 \pm 0.53}$	64.98 ± 0.34	71.43 ± 0.24
			RandFeature	10.93 ± 0.17	11.11 ± 0.34	11.48 ± 0.31
iNat-LECO	50000	64.21 ± 1.61	FinetunePrev	$\mathbf{73.64 \pm 0.46}$	71.26 ± 0.54	84.51 ± 0.66
			FreezePrev	50.82 ± 0.68	54.60 ± 0.49	54.08 ± 0.31
			TrainScratch	65.69 ± 0.46	65.78 ± 0.56	73.10 ± 0.80
			RandFeature	1.07 ± 0.06	1.02 ± 0.05	1.08 ± 0.09

5.1 Exploiting the Old-Ontology Data via Semi-Supervised Learning (SSL)

State-of-the-art SSL methods aim to increase the amount of labeled data by pseudo-labeling the unlabeled examples [61]. Therefore, a natural thought to utilize the old examples is to *ignore their old-ontology labels* and use SSL methods. We describe several representative SSL methods below.

Each input sample x, if not stated otherwise, undergoes a strong augmentation (RandAugment [15]) denoted as $\mathcal{A}(x)$ before fed into f. For all SSL algorithms, we construct a batch $\mathcal{B}_K$ of K samples from S_t and a batch $\hat{\mathcal{B}}_M$ of M samples from $S^{1:t-1}$. For brevity, we hereafter ignore the superscript (TP index). On new-ontology labeled samples, we use Eq. (1) to obtain $\mathcal{L}(\mathcal{B}_K)$, and sum it with an extra SSL loss on $\hat{\mathcal{B}}_M$, denoted as $\mathcal{L}_{SSL}(\hat{\mathcal{B}}_M)$. We have four SSL losses below.

Following [61], we use "Self-Training" to refer to a teacher-student distillation procedure [26]. Note that we adopt the **FinetunePrev** strategy and so both teacher and student models are initialized from the previous TP's weights. We first have two types of Self-Training:

- **ST-Hard** (Self-Training with one-hot labels): We first train a teacher model ($f_{teacher}$ and $g_{teacher}$) on the training set S^t, then use the teacher to "pseudo-label" history data in $S^{1:t-1}$. Essentially, for a $(x_m, y_m) \in S^{1:t-1}$, the old-ontology label y_m is ignored and a one-hot hard label is produced as $q_h = \arg\max g_{teacher}(f_{teacher}(\mathcal{A}(x_m)))$. Finally, we train a student model ($f_{student}$ and $g_{student}$) with CE loss:

$$\mathcal{L}_{SSL}(\hat{\mathcal{B}}_M) = \sum_{(x_m,\cdot)\in\hat{\mathcal{B}}_M} \mathcal{H}(q_h, g_{student}(f_{student}(\mathcal{A}(x_m))) \tag{2}$$

- **ST-Soft** (Self-Training with soft labels): The teacher is trained in same way as **ST-Hard**, but the difference is that pseudo-labels are softmax probability vectors $q_s = g_{teacher}(f_{teacher}(\mathcal{A}(x_m)))$ instead. The final loss is the KL divergence [50, 61]:

$$\mathcal{L}_{SSL}(\hat{\mathcal{B}}_M) = \sum_{(x_m,\cdot)\in\hat{\mathcal{B}}_M} \mathcal{H}(q_s, g_{student}(f_{student}(\mathcal{A}(x_m))) \tag{3}$$

In contrast to the above SSL methods that train separate teacher and student models, the following two train a single model along with the pseudo-labeling technique.

- **PL (Pseudo-Labeling)** [38] pseudo-labels image x_m as $q = g(f(\mathcal{A}(x_m)))$ using the current model. It also filters out pseudo-labels that have probabilities less than 0.95.

$$\mathcal{L}_{SSL}(\hat{\mathcal{B}}_M) = \sum_{(x_m,\cdot)\in\hat{\mathcal{B}}_M} \mathbb{1}[\max q > 0.95]\mathcal{H}(q, g(f(\mathcal{A}(x_m))) \tag{4}$$

- **FixMatch** [59] imposes a consistency regularization into PL by enforcing the model to produce the same output across weak augmentation ($a(x_m)$, usually standard random flipping and cropping) and strong augmentation ($\mathcal{A}(x_m)$, in our case is weak augmentation followed by RandAugment [15]). The pseudo-label is produced by weakly augmented version $q_w = g(f(a(x_m)))$. The final loss is:

$$\mathcal{L}_{SSL}(\hat{\mathcal{B}}_M) = \sum_{(x_m,\cdot)\in\hat{\mathcal{B}}_M} \mathbb{1}[\max q_w > 0.95]\mathcal{H}(q_w, g(f(\mathcal{A}(x_m))) \tag{5}$$

5.2 Exploiting the Old-Ontology Data via Joint Training

The above SSL approaches discard the old-ontology labels, which could be a coarse supervision in training. A simple approach to leverage this coarse supervision is **Joint Training**, a classic incremental learning method [16, 39] that trains jointly on both new and old tasks. Interestingly, **Joint Training** is usually the upperbound performance in traditional incremental learning benchmarks [39]. Therefore, we perform **Joint Training** to learn a shared feature extractor f and independent classifiers $(g^1, \cdots, g^t)$ w.r.t both new and old ontologies. We define the following loss at TP^t using old-ontology data from TP^1 to TP^{t-1} :

$$\mathcal{L}_{old}(\hat{\mathcal{B}}_M) = \sum_{t'\in\{1,\cdots,t-1\}} \sum_{(x_m,y_m^{t'})\in\hat{\mathcal{B}}_M^{t'}} \mathcal{H}(y_m^{t'}, g^{t'}(f(\mathcal{A}(x_m))) \tag{6}$$

where $y_m^{t'}$ is the coarse label of x_m at TP$^{t'}$ and $\hat{\mathcal{B}}_M^{t'}$ is the subset of $\hat{\mathcal{B}}_M$ with label $y_m^{t'}$ available. Importantly, because new-ontology fine labels $\mathbf{y}_i^t \in \mathbf{Y}^t$ refine old-ontology coarse labels, e.g., $\mathbf{y}_j^{t-1} \in \mathbf{Y}^{t-1}$, we can trivially infer *the coarse label for any fine label*. Therefore, we also apply $\mathcal{L}_{old}$ for fine-grained samples in B_K, i.e., $\mathcal{L}_{old}(\mathcal{B}_K)$. We have the final loss for **Joint Training** as $\mathcal{L}_{Joint} = \mathcal{L}_{old}(\mathcal{B}_K) + \mathcal{L}_{old}(\hat{\mathcal{B}}_M)$.

5.3 Results

Table 3 compares the above-mentioned SSL and Joint Training methods. In training, we sample the same amount of old data and new data in a batch. i.e., $|\hat{\mathcal{B}}_M| = |\mathcal{B}_K|$ (64 for *CIFAR-LECO* and 30 for *iNat-LECO*). We assign equal weight to $\mathcal{L}_{SSL}$, $\mathcal{L}_{Joint}$, and $\mathcal{L}$. We do hyperparameter search (weight decay and learning rate) and model selection via validation, and report the accuracies in TP^1 on the test sets. Please refer to Table 3-caption for salient conclusions.

6 Improving LECO with Taxonomic Hierarchy

While the SSL approaches exploit old samples and Joint Training further utilizes their old-ontology labels, none of them fully utilize the given taxonomic relationships between old and new ontologies. The fact that each fine-grained label $\mathbf{y}_i^t \in \mathbf{Y}^t$ refines a coarse label $\mathbf{y}_j^{t-1} \in \mathbf{Y}^{t-1}$ suggests a chance to improve LECO by exploiting such taxonomic relationship between the old and new classes [50, 60].

6.1 Incorporating Taxonomic Hierarchy via Learning-with-Partial-Labels (LPL)

Joint Training in the last section trains separate classifiers for each TP. This is suboptimal because we can simply marginalize fine-classes' probabilities for coarse-classes' probabilities [17]. This technique has been used in other works [27, 60] and is commonly called Learning-with-Partial-Labels (LPL) [27]. Concretely, the relationship between classes in $\mathbf{Y}^t$ and $\mathbf{Y}^{t'}$ with $t > t'$ can be summarized by an edge matrix $\mathbf{E}_{t\to t'}$ with shape $|\mathbf{Y}^t| \times |\mathbf{Y}^{t'}|$. $\mathbf{E}_{t\to t'}[i,j] = 1$ when an edge

Table 3: **Results of SSL and Joint Training methods**. Following the notation of Table 2, we study **LabelNew** annotation strategies that make use of algorithms for learning from old examples *without* exploiting the relationship between old-vs-new labels. Recall that UpperBound uses $\mathcal{L}$ only to train on all the data labeled with new ontology (doubling annotation cost). Both $\mathcal{L}_{Joint}$ and the best $\mathcal{L}_{SSL}$ (ST-Soft) outperform the baseline which uses loss $\mathcal{L}$ only (i.e., LabelNew + FinetunePrev in Table 2). ST-Soft yields the most performance gain. Using all the three losses together consistently improves mAcc for all the benchmarks, confirming the benefit of exploiting the old examples. For instance, on *iNat-LECO*, this boosts the performance to 83.6%, approaching the UpperBound (84.5%)!

Benchmark	#images / TP	SSL Alg	TP[1] mAcc				
			$\mathcal{L}$ only	$+\mathcal{L}_{SSL}$	$+\mathcal{L}_{Joint}$	$+\mathcal{L}_{Joint}$ $+\mathcal{L}_{SSL}$	UpperBound
CIFAR-LECO	1000	ST-Hard	33.53 ± 0.57	34.59 ± 0.57	37.25 ± 0.66	35.27 ± 0.64	42.27 ± 0.83
		ST-Soft		37.67 ± 0.46		**38.48 ± 0.67**	
		PL		33.53 ± 0.36		37.28 ± 0.67	
		FixMatch		34.20 ± 0.24		37.47 ± 0.53	
CIFAR-LECO	10000	ST-Hard	65.83 ± 0.27	67.11 ± 0.34	68.74 ± 0.21	67.97 ± 0.37	71.30 ± 0.41
		ST-Soft		**69.86 ± 0.30**		**69.77 ± 0.71**	
		PL		65.86 ± 0.14		68.43 ± 0.40	
		FixMatch		67.13 ± 0.36		69.03 ± 0.43	
iNat-LECO	50000	ST-Hard	73.64 ± 0.46	75.23 ± 0.14	82.98 ± 0.33	78.23 ± 0.38	84.51 ± 0.66
		ST-Soft		79.64 ± 0.41		**83.61 ± 0.39**	
		PL		73.56 ± 0.09		82.87 ± 0.51	
		FixMatch		74.57 ± 0.54		83.00 ± 0.33	

$y_i^t \to y_j^{t'}$ exists in the taxonomy tree, i.e., $y_j^{t'}$ is the superclass of y_i^t; otherwise $\mathbf{E}_{t\to t'}[i,j] = 0$. Therefore, to obtain prediction probability of a coarse-grained label $y^{t'}$ for a sample x, we sum up the probabilities in $g^t(f^t(x))$ that correspond to fine labels of $y^{t'}$, which can be efficiently done by a matrix vector multiplication: $g^t(f^t(x)) \cdot \mathbf{E}_{t\to t'}$. Formally, we define a loss over a sampled batch $\hat{\mathcal{B}}_M \subset S^{1:t-1}$ as:

$$\mathcal{L}_{OldLPL}(\hat{\mathcal{B}}_M) = \sum_{t' \in \{1,\cdots,t-1\}} \sum_{(x_m, y_m^{t'}) \in \hat{\mathcal{B}}_M^{t'}} \mathcal{H}(y_m^{t'}, g^t(f^t(\mathcal{A}(x_m))) \cdot \mathbf{E}_{t\to t'}) \tag{7}$$

We apply this loss on the fine-grained samples and define the **LPL Loss** $\mathcal{L}_{LPL} = \mathcal{L}_{oldLPL}(\mathcal{B}_K) + \mathcal{L}_{oldLPL}(\hat{\mathcal{B}}_M)$.

6.2 Incorporating Taxonomic Hierarchy via Pseudo-label Refinment

Another strategy to incorporate the taxonomic hierarchy is to exploit coarse labels in the pseudo-labeling process, which has been explored in a different task (semi-supervised few-shot learning) [50]. Prior art shows that the accuracy of pseudo labels is positively correlated with SSL performance [59]. Therefore, at TP^t, in pseudo-labeling an old sample x_m, we use its coarse label $y_m^{t'}$ ($t' < t$) to clean up inconsistent pseudo labels. In particular, we consider

1. **Filtering**: we filter out pseudo-labels that do not align with the ground-truth coarse labels, i.e., removing a pseudo-label $q = g^t(f^t(x))$ if $\mathbf{E}_{t\to t'}[\arg\max_c q_c, y_m^{t'}] = 0$.
2. **Conditioning**: we clean up the pseudo-label q by zeroing out all scores unaligned with the true coarse label: $q[c] := q[c] * \mathbf{E}_{t\to t'}[c, y_m^{t'}]$. We then renormalize $q[c] := q[c]/\sum_i q[i]$.

6.3 Results

Table 4 shows the results of **LPL Loss** and SSL strategies with pseudo-label refinement. We can see that simply adding the **LPL Loss** $\mathcal{L}_{LPL}$ matches the SOTA (achieved by naive SSL and **Joint Training** strategies in Table 3). Also, adding SSL loss along with pseudo-label refinement via either **Filtering** or **Conditioning** further bridges the gap to **UpperBound**. For example, on *iNat-LECO*, the best strategy ($+\mathcal{L}_{LPL} + \mathcal{L}_{SSL/Cond}$ with ST-Hard) is only 0.2% lower than the **UpperBound**

Table 4: **Results of exploiting taxonomic hierarchy**. We copy the best results in Table 3 (all obtained by $+\mathcal{L}_{Joint}$ $+\mathcal{L}_{SSL}$) for reference. Adding supervision via taxonomic hierarchy is effective: (1) simply using $\mathcal{L}_{LPL}$ already matches the previously best performance (by by $+\mathcal{L}_{Joint}$ $+\mathcal{L}_{SSL}$); (2) adding SSL loss along with pseudo-label refinement via either **Filtering** or **Conditioning** further bridges the gap to UpperBound. As a result, for example, on *iNat-LECO*, the best strategy ($+\mathcal{L}_{LPL} + \mathcal{L}_{SSL/Cond}$ with ST-Hard) is only 0.2% lower than the UpperBound!

Benchmark	#images / TP	SSL Alg	TP[1] mAcc				
			$+\mathcal{L}_{Joint}$ $+\mathcal{L}_{SSL}$	$+\mathcal{L}_{LPL}$	$+\mathcal{L}_{LPL}$ $+\mathcal{L}_{SSL/Filter}$	$+\mathcal{L}_{LPL}$ $+\mathcal{L}_{SSL/Cond}$	UpperBound
CIFAR-LECO	1000	ST-Hard	38.48 ± 0.67	38.47 ± 0.52	**38.87 ± 0.54**	**38.68 ± 0.41**	42.27 ± 0.83
		ST-Soft			**38.87 ± 0.52**	38.19 ± 0.71	
		PL			**38.56 ± 0.44**	**38.67 ± 0.81**	
		FixMatch			37.55 ± 0.69	37.35 ± 0.72	
CIFAR-LECO	10000	ST-Hard	69.77 ± 0.71	69.31 ± 0.30	69.90 ± 0.29	**70.42 ± 0.23**	71.30 ± 0.41
		ST-Soft			70.10 ± 0.12	70.02 ± 0.63	
		PL			69.20 ± 0.20	69.60 ± 0.26	
		FixMatch			69.61 ± 0.17	69.85 ± 0.46	
iNat-LECO	50000	ST-Hard	83.61 ± 0.39	83.62 ± 0.21	**84.23 ± 0.61**	**84.34 ± 0.25**	84.51 ± 0.66
		ST-Soft			**84.12 ± 0.31**	83.76 ± 0.40	
		PL			83.72 ± 0.22	**84.16 ± 0.19**	
		FixMatch			83.91 ± 0.39	83.95 ± 0.40	

(84.3% v.s. 84.5%). We acknowledge that the **UpperBound** can be improved further if using LPL loss as well [8] but we use the current version just as a reference of performance. We also tried combining $\mathcal{L}_{SSL/Filter}$ or $\mathcal{L}_{SSL/Cond}$ with the **Joint Training**, and observe similar improvements upon $\mathcal{L}_{SSL} + \mathcal{L}_{Joint}$ (details in the supplement).

7 Discussions and Conclusions

We introduce the problem *Learning with Evolving Class Ontology* (LECO) motivated by the fact that lifelong learners must recognize concept vocabularies that evolve over time. To explore LECO, we set up its benchmarking protocol and study various approaches by repurposing methods of related problems. Extensive experiments lead to several surprising conclusions. First, we find the status-quo for dataset versioning, where old data is relabeled, is not the most effective strategy. One should label new data with new labels, but the mixture of old and new labels is effective only by exploiting semi-supervised methods that exploit the relationship between old and new labels. Surprisingly, such strategies can approach the upper bound of learning from an oracle aggregate dataset exhaustively annotated with new labels.

Societal Impacts. The study of LECO has various positive impacts as it reflects practical development of machine-learning solutions and biological learning paradigms. For example, LECO emphasizes lifelong learning and maintainance of machine-learning systems, which arise in inter-disciplinary research (where a bio-image analysis system needs to learn from a refined taxonomy, cf. iNaturalist), autonomous vehicles (where critical fine-grained classes appear as operations expand to more cities), and the fashion industry (where a growing number of clothes styles appear from season to season). We are not aware of negative societal impacts of LECO over those that arise in traditional image classification.

Future work to address limitations. We notice several limitations and anticipate future work to address them. Firstly, the assumption of constant annotation cost between the old and new data needs to be examined further, but this likely requires the design of novel interfaces for (re)annotation. Secondly, our work did not consider the cost for obtaining new data, which might include data mining and cleaning. In some settings, data is already being continuously collected (e.g., by autonomous fleets). This also suggests that LECO should incorporate unlabeled data. Lastly, we explore LECO on well-known benchmarks with static data distributions over time periods, but in the future, we would like to embrace temporal shifts in the data distribution as well [41].

References

[1] Abdelsalam, M., Faramarzi, M., Sodhani, S., and Chandar, S. (2021). Iirc: Incremental implicitly-refined classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11038–11047.

[2] Alshammari, S., Wang, Y., Ramanan, D., and Kong, S. (2022). Long-tailed recognition via weight balancing. In *CVPR*.

[3] Azizpour, H., Razavian, A. S., Sullivan, J., Maki, A., and Carlsson, S. (2015). Factors of transferability for a generic convnet representation. *IEEE transactions on pattern analysis and machine intelligence*, **38**(9), 1790–1802.

[4] Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J. (2019). Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*.

[5] Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.

[6] Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. (2019a). Mixmatch: A holistic approach to semi-supervised learning. *NeurIPS*, **32**.

[7] Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., and Raffel, C. (2019b). Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*.

[8] Bertinetto, L., Mueller, R., Tertikas, K., Samangooei, S., and Lord, N. A. (2020). Making better mistakes: Leveraging class hierarchies with deep networks. In *CVPR*.

[9] Bucak, S. S., Jin, R., and Jain, A. K. (2011). Multi-label learning with incomplete class assignments. In *CVPR 2011*, pages 2801–2808. IEEE.

[10] Cai, Z., Sener, O., and Koltun, V. (2021). Online continual learning with natural distribution shifts: An empirical study with visual data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8281–8290.

[11] Caruana, R. (1997). Multitask learning. *Machine learning*, **28**(1), 41–75.

[12] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., *et al.* (2019). Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757.

[13] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *CVPR*.

[14] Cour, T., Sapp, B., and Taskar, B. (2011). Learning from partial labels. *The Journal of Machine Learning Research*, **12**, 1501–1536.

[15] Cubuk, E. D., Zoph, B., Shlens, J., and Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703.

[16] Delange, M., Aljundi, R., Masana, M., Parisot, S., Jia, X., Leonardis, A., Slabaugh, G., and Tuytelaars, T. (2021). A continual learning survey: Defying forgetting in classification tasks. *PAMI*.

[17] Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., Li, Y., Neven, H., and Adam, H. (2014). Large-scale object classification using label relation graphs. In *European conference on computer vision*, pages 48–64. Springer.

[18] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR.

[19] Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition*, **48**(1), 71–99.

[20] Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *IJCV*, **111**(1), 98–136.

[21] Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE.

[22] Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P., and Cord, M. (2019). Boosting few-shot visual learning with self-supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8059–8068.

[23] Grandvalet, Y., Bengio, Y., *et al.* (2004). Learning from partial labels with minimum entropy. Technical report, CIRANO.

[24] Gupta, A., Dollar, P., and Girshick, R. (2019). Lvis: A dataset for large vocabulary instance segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5356–5364.

[25] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.

[26] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[27] Hu, P., Lipton, Z. C., Anandkumar, A., and Ramanan, D. (2018). Active learning with partial feedback. *arXiv preprint arXiv:1802.07427*.

[28] iNaturalist 2017 (2017). `https://www.inaturalist.org/stats/2017`.

[29] iNaturalist 2018 (2018). `https://www.inaturalist.org/stats/2018`.

[30] iNaturalist 2019 (2019). `https://www.inaturalist.org/stats/2019`.

[31] iNaturalist 2020 (2020). `https://www.inaturalist.org/stats/2020`.

[32] iNaturalist 2021 (2021). `https://www.inaturalist.org/stats/2021`.

[33] Kang, B., Xie, S., Rohrbach, M., Yan, Z., Gordo, A., Feng, J., and Kalantidis, Y. (2020). Decoupling representation and classifier for long-tailed recognition. In *International Conference on Learning Representations*.

[34] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., *et al.* (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, **114**(13), 3521–3526.

[35] Kong, S. and Ramanan, D. (2021). Opengan: Open-set recognition via open data generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 813–822.

[36] Krizhevsky, A., Hinton, G., *et al.* (2009). Learning multiple layers of features from tiny images.

[37] Krueger, K. A. and Dayan, P. (2009). Flexible shaping: How learning in small steps helps. *Cognition*, **110**(3), 380–394.

[38] Lee, D.-H. *et al.* (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896.

[39] Li, Z. and Hoiem, D. (2017). Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, **40**(12), 2935–2947.

[40] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *ECCV*. Springer.

[41] Lin, Z., Shi, J., Pathak, D., and Ramanan, D. (2021). The clear benchmark: Continual learning on real-world imagery. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

[42] Lomonaco, V. and Maltoni, D. (2017). Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26. PMLR.

[43] Maltoni, D. and Lomonaco, V. (2019). Continuous learning in single-incremental-task scenarios. *Neural Networks*, **116**, 56–73.

[44] Mapillary-Vistas-2.0 (2021). `https://blog.mapillary.com/update/2021/01/18/vistas-2-dataset.html`.

[45] Marszalek, M. and Schmid, C. (2007). Semantic hierarchies for visual object recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE.

[46] McLachlan, G. J. (1975). Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association*, **70**(350), 365–369.

[47] Neuhold, G., Ollmann, T., Rota Bulo, S., and Kontschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999.

[48] Nguyen, N. and Caruana, R. (2008). Classification with partial labels. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–559.

[49] Peterson, G. B. (2004). A day of great illumination: Bf skinner's discovery of shaping. *Journal of the experimental analysis of behavior*, **82**(3), 317–328.

[50] Phoo, C. P. and Hariharan, B. (2021). Coarsely-labeled data for better few-shot transfer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9052–9061.

[51] Prabhu, A., Torr, P. H., and Dokania, P. K. (2020). Gdumb: A simple approach that questions our progress in continual learning. In *ECCV*.

[52] Rebuffi, S.-A., Kolesnikov, A., Sperl, G., and Lampert, C. H. (2017). icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010.

[53] Rizve, M. N., Duarte, K., Rawat, Y. S., and Shah, M. (2021). In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. *arXiv preprint arXiv:2101.06329*.

[54] Sanger, T. D. (1994). Neural network learning control of robot manipulators using gradually increasing task difficulty. *IEEE transactions on Robotics and Automation*, **10**(3), 323–333.

[55] Sariyildiz, M. B., Kalantidis, Y., Larlus, D., and Alahari, K. (2021). Concept generalization in visual representation learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9629–9639.

[56] Scudder, H. (1965). Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, **11**(3), 363–371.

[57] Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.

[58] Skinner, B. F. (1958). Reinforcement today. *American Psychologist*, **13**(3), 94.

[59] Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C. A., Cubuk, E. D., Kurakin, A., and Li, C.-L. (2020). Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *NeurIPS*.

[60] Su, J. and Maji, S. (2021a). Semi-supervised learning with taxonomic labels. In *British Machine Vision Conference (BMVC)*.

[61] Su, J., Cheng, Z., and Maji, S. (2021). A realistic evaluation of semi-supervised learning for fine-grained classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[62] Su, J.-C. and Maji, S. (2021b). The semi-supervised inaturalist challenge at the fgvc8 workshop. *arXiv preprint arXiv:2106.01364*.

[63] Van de Ven, G. M. and Tolias, A. S. (2019). Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

[64] Van Horn, G., Mac Aodha, O., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., and Belongie, S. (2018). The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778.

[65] Wilson, B., Qi, W., Agarwal, T., Lambert, J., Singh, J., Khandelwal, S., Pan, B., Kumar, R., Hartnett, A., Pontes, J. K., *et al.* (2021). Argoverse 2: Next generation datasets for self-driving perception and forecasting.

[66] Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10687–10698.

[67] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *NeurIPS*.

[68] Zenke, F., Poole, B., and Ganguli, S. (2017). Continual learning through synaptic intelligence. In *ICML*.

[69] Zhai, X., Oliver, A., Kolesnikov, A., and Beyer, L. (2019). S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1476–1485.

[70] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2018). mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

[71] Zhang, J., Zhang, J., Ghosh, S., Li, D., Tasci, S., Heck, L., Zhang, H., and Kuo, C.-C. J. (2020). Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1131–1140.

[72] Zweig, A. and Weinshall, D. (2007). Exploiting object hierarchy: Combining models from different category levels. In *2007 IEEE 11th international conference on computer vision*, pages 1–8. IEEE.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We did in Section 7.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We did in Section 7.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We include the github repo in supplement and attach code used for running the experiments in this paper.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report standard deviations after five random runs of each methods in Table 2, 3, and 4.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In the supplement.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] We did in Section 3.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]