Appendix

A: In LECO, label new data or relabel the old?

In LECO, the foremost question to answer is whether to label new data or relabel the old (Fig. 2-right). As discussed in the main paper, both labeling protocols have been used in the community. For example, in the context of autonomous driving research, Argoverse labeled new data with new ontology in its updated version (from V1.0 [12] to V2.0 [78]), whereas Mapillary related the old data from its V1.2 [55] to V2.0 [52]. Our extensive experiments convincingly demonstrate that a better strategy is to label new data, simply because doing so provides more (labeled) data.

Data acquisition (for new data) might be costly. However, many real-world applications acquire data continuously regardless of the cost. For example, autonomous vehicle fleets collect data continuously, so do surveillance cameras that record video frames. Therefore, labeling such new data is reasonable in the real world.

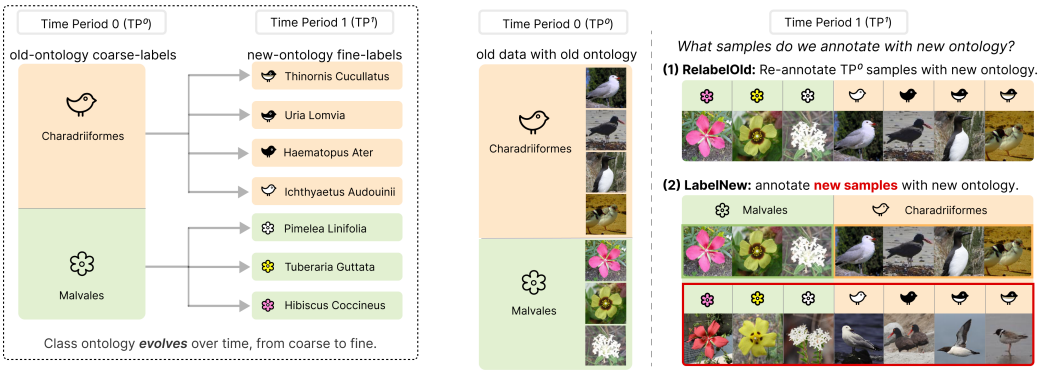


Figure 2: **Left:** Learning with Evolving Class Ontology (LECO) requires training models in time periods (TPs) that refine old ontologies in a coarse-to-fine fashion. This leads to a basic question: for the next TP, should one relabel the old data or label new data? Interestingly, both labeling protocols have been used in the community for large-scale datasets. **Right:** Our extensive experiments 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. Following this labeling protocol and to address LECO, we leverage 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.

B: Rationale behind the selection of levels on iNaturalist

We repurpose the iNaturalist dataset to set up a LECO benchmarking protocol. The dataset has seven levels of taxa, allowing to use such as superclasses in LECO's time periods. To determine the levels to use, we have two principles aiming to have a clean and challenging setup to study LECO. First, we think it is good to have more fine-grained classes in TP1, hence we choose the most fine-grained level "species" in TP1. Second, we think all coarse-classes should be split later to emphasize the difficulty of learning with class-evolution, and TP0's classification task should be challenging as well with more classes. Therefore, we choose the "order" level which has 123 taxa. As reference, the original iNat dataset has seven levels: kingdom (3 taxa), phylum (8), class (29), order (123), family (339), genus (729), and species (810). The long-tailed class distributions in the two TPs are depicted in Fig. 4.

C: Experiments on Mapillary V1.2 $\rightarrow$ V2.0

As we briefly discussed in the main paper, Mapillary is a real-world example where a large dataset evolved over time from V1.2 [55] to V2.0 [52]. We include additional experiments on this real-world dataset versioning scenario. Mapillary is a rich and diverse street-level imagery dataset that was

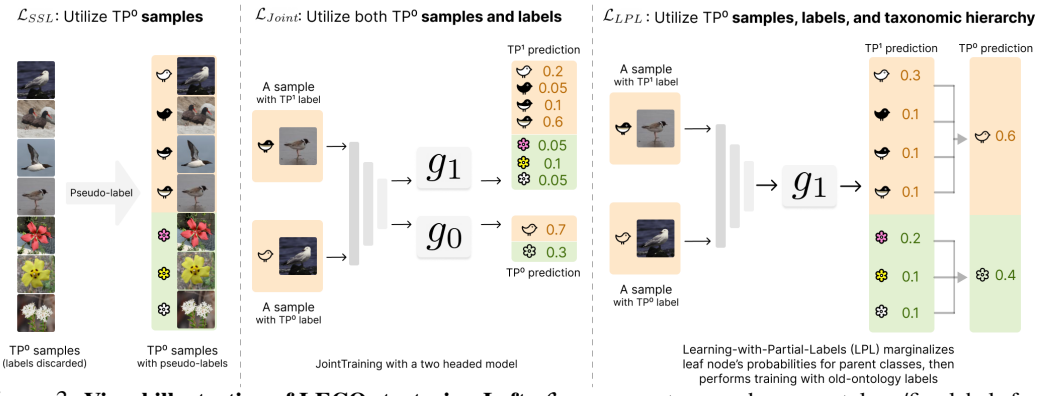
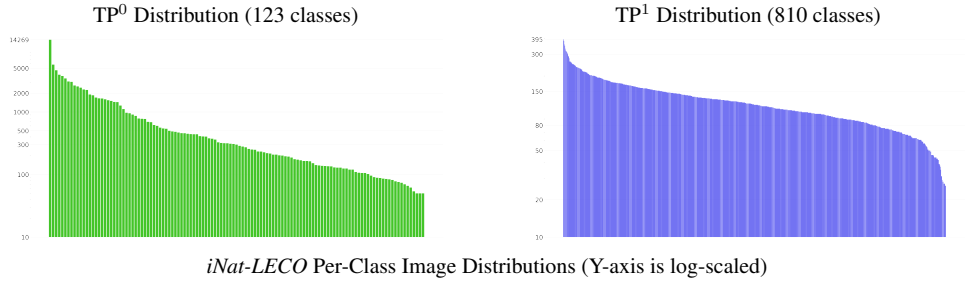


Figure 3: **Visual illustration of LECO strategies**. **Left:** $\mathcal{L}_{SSL}$ generates pseudo new-ontology/fine labels for old-ontology samples via semi-supervised learning (SSL). However, this ignores old-ontology/coarse labels which might serve as coarse-level supervision. We therefore advocate for $\mathcal{L}_{SSL/Filter}$ that filters out pseudo-labels with wrong coarse labels, or $\mathcal{L}_{SSL/Cond}$ that conditions leave's probabilities based on the ground-truth parent/coarse class. **Middle:** $\mathcal{L}_{Joint}$ utilizes the old-ontology labels that come with the samples; to reconcile for different labels at different TPs, it trains a model with multiple classification heads. **Right:** $\mathcal{L}_{LPL}$ (learning-with-partial-labels) further utilizes the taxonomic hierarchy by exploiting the fact that newly added classes in LECO are refined from old classes. It does so by marginalizing leaves' probabilties for their parent classes.



iNat-LECO Per-Class Image Distributions (Y-axis is log-scaled)

Figure 4: **iNat-LECO Per-Class Distributions.** We plot distributions of training images for *iNat-LECO* in log-scale. The x-axis are classes sorted by the number of images. The y-axis shows the number of images per class.

collected for semantic segmentation research in the context of autonomous driving. We repurpose this dataset for setting a LECO benchmark. Briefly, we use the ontology of Mapillary V1.2 in time period 1 (TP1), and V2.0 in TP2. TP1 has 66 classes including a catch-all `background` (aka `void`) class. TP1 (i.e., Mapillary V2.0 [4]) contains 124 classes, out of which 116 are used for evaluation.

Mapillary-LECO benchmark. The original Mapillary contains 18k training images, which is much larger than other mainstream street-level segmentation datasets such as Cityscape [14] (which has 2975 training and 500 validation images). In this work, we repurpose Mapillary to set up a LECO benchmark by splitting training set into two subsets (each containing 2.5k or 9k images) for the two time periods. We use the original validation set of Mapillary with 2k images as our test set. The detailed statistics is shown in Table 7.

Baseline Experiments on Mapillary-LECO. We used the state-of-the-art architecture HRNet-V2-W48 for semantic segmentation with OCR module [72, 76, 81]. We noticed that for semantic segmentation on street-level imagery datasets [14], it is common to start from a Imagenet-pretrained model. Hence we name the weight initialization schemes for *Mapillary-LECO* as:

- (**TrainRandom**) Train an entire network from randomly initialized weights.
- (**TrainScratch**) Initialize the model backbone with ImageNet-pretrained weights.[5]

As we can see from Table 8, **TrainScratch** (from ImageNet pretrained weights) as a popular initialization strategy for segmentation tasks [14, 55] indeed performs better than **TrainRandom**. Therefore,

[4] `https://blog.mapillary.com/update/2021/01/18/vistas-2-dataset.html`
[5] Weights are available at `https://github.com/HRNet/HRNet-Semantic-Segmentation`

Table 7: **Mapillary-LECO statistics**. We repurpose Mapillary [52, 55] V1.2 to V2.0 (a real-world dataset versioning scenario) to set up a LECO benchmark. Since V2.0 contains the same set of images as V1.2, in order to simulate a scenario with new samples, we select the first 5k or all 18k training images, then split them to 2.5k/9k for TP^0 and 2.5k/9k for TP^1. Note that in TP^0, the 66 classes include a catch-all `background` (or `void`) class, which is subsequently refined to 9 fine-grained classes such as `traffic-cone` and `traffic-island`.

dataset	Time Period 0 (TP^0)			Time Period 1 (TP^1)		
	#classes	#train	#test	#classes	#train	#test
Mapillary-LECO	66	2.5k/9k	2k	116	2.5k/9k	2k

for our **FinetunePrev** strategy, we finetune the model checkpoint obtained via **TrainScratch** strategy on TP^0 data. To further bridge the gap to **AllFine**, we experiment other advanced techniques introduced in this paper such as $\mathcal{L}_{SSL}$ and $\mathcal{L}_{Joint}$ as shown in Table 10.

Results of various LECO strategies. Since Mapillary does not reveal the taxonomic hierarchy from V1.2 to V2.0, we use the ground-truth label maps to automatically retrace the ontology evolution. Because Mapillary adopts **RelabelOld** strategy, each input image has both V1.2 and V2.0 label map. We exploit this fact to determine a single parent class in V1.2 for each of the 116 classes in V2.0 following a simple procedure (taking `bird` class in V2.0 as an example):

- First, we use the ground-truth V2.0 label maps to find all pixels labeled as `bird`.
- Then, we locate those pixels in V1.2 label maps, and choose the V1.2 class that occupies the most pixels as `bird`'s parent.

We find that this simple procedure produces a reliable taxonomic hierarchy that can be used to improve the performance via $\mathcal{L}_{LPL}$, $\mathcal{L}_{SSL/Filter}$ or $\mathcal{L}_{SSL/Cond}$, as shown in Table 11. Note that this procedure does not model class merging in Mapillary; however, our solutions still remain effective.

Segmentation Loss. We make use of a standard pixel-level cross-entropy loss function for training on Mapillary. One difference from our previous image classification setup is that both the ontology and semantic label maps changed from V1.2 to V2.0. In general, we find the label maps on V2.0 to be of higher quality. As such, we (1) do not add coarse supervision on TP^1 data ($\mathcal{B}_K$), since this pollutes the quality of annotations on the new data, i.e., $\mathcal{L}_{Joint} = \mathcal{L}_{old}(\hat{\mathcal{B}}_M)$ and $\mathcal{L}_{LPL} = \mathcal{L}_{oldLPL}(\hat{\mathcal{B}}_M)$, and (2) we mask out gradients on pixel regions (around 0.3% of all pixels) where the given V1.2 labels do not align with the parent class of their V2.0 labels for $\mathcal{L}_{old}(\hat{\mathcal{B}}_M)$ and $\mathcal{L}_{oldLPL}(\hat{\mathcal{B}}_M)$.

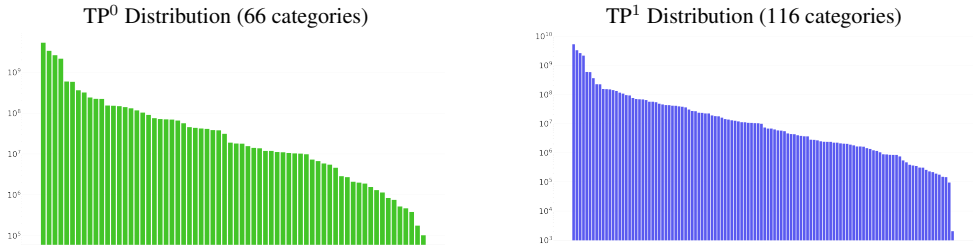
Training and Inference Details. For both **TrainRandom** and **TrainScratch**, we use an initial learning rate of 0.03; for **FinetunePrev**, we use an initial learning rate of 0.003. The L2 weight decay is selected as 0.0005. Other hyperparameters followed the default strategy in HRNet used to achieve the SOTA results on Cityscape. We use SGD with 0.9 momentum and a batch size of 16 for $\mathcal{B}_K$ (or 8 for $\mathcal{B}_K$ and 8 for $\hat{\mathcal{B}}_M$ if we use $\mathcal{L}_{SSL}$ or $\mathcal{L}_{Joint}$). We use linear learning rate decay schedule that sets the learning rate to $\eta(1 - \frac{k}{K})^{0.9}$ where η is the initial learning rate and k/K are the current and total iteration. For data augmentation, an input image and its label map are randomly flipped horizontally and then its longer edge will be scaled with a base size of 2200 pixels multiplied by a scalar factor sampled between 0.5 and 2.1. Finally, a 720x720 region will be randomly cropped for each of the 16 images in mini-batch for training. For inference, we use a sliding window of 720x720 without scaling of the image to determine the final pixel-level prediction. We also perform inference on the flipped image and take the average prediction as final result. Note that it is possible to use more advanced inference techniques such as multi-scaled testing, but we omit them to speed up inference since they are orthogonal to our research goal. We allocate the same training budget for all experiments. In particular, we use a total number of 1600 epochs for 2.5k/9k samples (a single TP), or 800 epochs for 5k/18k samples (two TPs data such as **AllFine** and **LabelNew** with SSL/Joint/LPL techniques). All experiments are conducted on internal clusters with 8 GeForce RTX 3090 cards.

D: Dataset Statistics

To better understand the long-tailed nature of the datasets, we include the per-class image and pixel distributions for *iNat-LECO* in Figure 4 and *Mapillary-LECO* in Figure 5. We note that the long-tailed class distributions make semi-supervised methods less effective, as shown in Table 3, 4, 10, 11, e.g., the pseudo-labeling method of SSL underperforms the simple supervised learning baseline ($\mathcal{L}$ only).

Table 8: **Results of baseline methods for Mapillary-LECO**, analogous to Table 2 in the main paper. We report the mean Intersection-over-Union (mIoU in %). **TrainRandom** trains a model from scratch, as is the protocol in the main paper. Because segmentation tasks typically make use of ImageNet pre-training, we also show results for **TrainScratch** that trains with an ImageNet-pretrained encoder, improving upon **TrainRandom**). Finally, **FinetunePrev** makes use of TP^0's model, which is trained with **TrainScratch**. In Table 10, we explore Joint-Training and SSL, leading to further improvement.

Benchmark	#images / TP	TP^1 Strategy	TP^1 mIoU		
			LabelNew	RelabelOld	AllFine
Mapillary-LECO	2500	TrainRandom	27.24	27.21	27.73
		TrainScratch	28.22	29.28	30.71
		FinetunePrev	30.39	29.05	31.73
Mapillary-LECO	9000	TrainScratch	32.83	33.44	33.82
		FinetunePrev	34.08	35.01	36.20



Mapillary-LECO Per-Class Pixel Distributions (Y-axis is log-scaled)

Figure 5: **Mapillary-LECO Per-Class Distributions.** We plot distributions of training pixels for *iNat-LECO* in log-scale. The x-axis are classes sorted by the number of pixels. The y-axis shows the number of pixels per class (in log scale).

E: Training Details on CIFAR-LECO and iNat-LECO

We include all training details for reproducibility in this section. We adopt SOTA training strategies from recent papers [68–70]. For all experiments, we train the CNN with mini-batch stochastic gradient descent with 0.9 momentum. Same as FixMatch [68], we use the an exponential moving average (EMA) of model parameters for final inference with a decay parameter of 0.999. We use a cosine annealing learning rate decay schedule which sets the learning rate to $\eta \cos \frac{7\pi k}{16K}$ where η is the initial learning rate and k/K are the current and total iteration. By default, we use a strong data augmentation scheme because it provides better generalization performance. In particular, we adopt RandAugment [16] as implemented in this repository.[6] We did a grid search for best learning rate and weight decay using the validation set of each benchmark. The benchmark-specific hyperparameters are detailed below:

CIFAR100-LECO. We use the same model architecture (Wide-ResNet-28-2 [82]) in FixMatch paper for CIFAR100 experiments [68]. We use a batch size of 128 (if using $\mathcal{L}$ only), or we split the batch to 64 for $\mathcal{B}_K$ and 64 for $\hat{\mathcal{B}}_M$. For each experiment, we search for initial learning rate in $[0.6, 0.06, 0.006, 0.0006]$ and report the best mAcc result. We use an L2 weight decay of 5e-4. We run for a total number of 160K iterations and evaluate on the validation set per 1K iterations (equivalent to around 2000 epochs for 10000 images).

iNat-LECO. We use the same model architecture (ResNet50 [28]) as in [70]. We use a batch size of 60 (if using $\mathcal{L}$ only), or we split the batch to 30 for $\mathcal{B}_K$ and 30 for $\hat{\mathcal{B}}_M$. Because it is a long-tailed recognition problem, we search for L2 weight decay as suggested by [2] in $[0.001, 0.0001, 0.00001]$ and found out 0.001 produces the best mAcc. Then for each experiment, we search for initial learning rate in $[0.01, 0.001, 0.0001]$ and report the best mAcc result. We run for a total number of 250K iterations and evaluate on the validation set per 1K iterations (equivalent to around 300 epochs for 50000 images).

[6] https://github.com/kekmodel/FixMatch-pytorch/

Table 9: **Results of baseline methods for LECO** (complete results). For all our experiments except on Mapillary, we run each method five times and report the mean accuracy averaged over per-class accuracies with standard deviations. Because running on Mapillary is very compute-demanding, we run it once and report mean intersection-over-union (mIoU in %) over per-class IoU. For each benchmark, we bold the best mAcc/mIoU including ties that fall within its std. All conclusions we derived in main paper still hold, e.g., **FinetunePrev** is the best learning strategy that outperforms **TrainScratch** (cf. 73.64% vs. 65.69% with **LabelNew** on iNat-LECO), implying that the coarse-to-fine evolved ontology serves as a good learning curriculum. **FreezePrev** significantly underperforms **TrainScratch** (cf. 50.82% vs. 65.69% with LabelNew on iNat-LECO), confirming the difficulty of learning new classes that refine the old. **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 10. The conclusions hold for varied number of training images and generalize to the Mapillary experiments, yet all methods fall short to the **AllFine** (which trains on all data assumed to be labeled with new ontology).

Benchmark	#images / TP	TP^0 mAcc/mIoU	TP^1 Strategy	TP^1 mAcc/mIoU		
				LabelNew	RelabelOld	AllFine
CIFAR-LECO	1000	50.92 ± 0.89	FinetunePrev	**33.53 ± 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
CIFAR-LECO	2000	59.48 ± 0.66	FinetunePrev	**43.44 ± 0.73**	**42.88 ± 0.72**	52.72 ± 0.38
			FreezePrev	33.74 ± 1.46	33.69 ± 1.44	36.57 ± 1.68
			TrainScratch	41.61 ± 0.53	41.77 ± 0.79	52.78 ± 0.35
CIFAR-LECO	10000	77.78 ± 0.27	FinetunePrev	**65.83 ± 0.53**	**65.30 ± 0.32**	71.30 ± 0.41
			FreezePrev	52.64 ± 0.50	52.60 ± 0.52	53.79 ± 0.43
			TrainScratch	**65.40 ± 0.53**	64.98 ± 0.34	71.43 ± 0.24
iNat-LECO	50000	64.21 ± 1.61	FinetunePrev	**73.64 ± 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
Mapillary-LECO	2500	37.01	FinetunePrev	**30.39**	29.05	31.73
			TrainScratch	27.24	27.21	27.73
Mapillary-LECO	9000	44.14	FinetunePrev	34.08	**35.01**	36.20
			TrainScratch	32.83	33.44	33.82

iNat-4TP-LECO. The range of grid search of hyperparameters follows that of **iNat-LECO**, including 250K iterations (equivalent to around 600 epochs for 25000 images) for each TP.

All the classification experiments were performed on a single modern GPU card. Based on the batch size scale, we use different GPU types, e.g., GeForce RTX 2080 Ti for image classification on CIFAR and iNaturalist, and GeForce RTX 3090 Ti on Mapillary.

F: Comprehensive Experiment Results

We now show the complete results on varying sample sizes per TP for all benchmarks. Table 9, 10, 11 in appendix correspond to Table 2, 3, 4 in main paper. All conclusions generalize to varied number of samples and all 3 datasets.

We also have complete results using all combinations of the various SSL/Joint/LPL techniques introduced in main paper. Results can be found on Table 12 (*CIFAR-LECO* and *iNat-LECO*) and Table 13 (*Mapillary-LECO*).

Table 10: **Results of SSL and Joint Training methods** (complete results). Following the notation of Table 9, 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 AllFine uses $\mathcal{L}$ only to train on all the data labeled with new ontology (doubling annotation cost). Using all the three losses together consistently improves mAcc / mIoU 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 AllFine (84.5%)! Similar trends hold for *Mapillary-LECO*.

Benchmark	#images / TP	SSL Alg	TP[1] mAcc/mIoU				
			$\mathcal{L}$ only	$+\mathcal{L}_{SSL}$	$+\mathcal{L}_{Joint}$	$+\mathcal{L}_{Joint}$ $+\mathcal{L}_{SSL}$	AllFine
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	2000	ST-Hard	43.44 ± 0.73	44.87 ± 0.35	47.68 ± 0.70	45.67 ± 0.32	52.72 ± 0.38
		ST-Soft		**48.23 ± 0.55**		**48.72 ± 0.53**	
		PL		43.46 ± 0.61		47.24 ± 0.40	
		FixMatch		44.98 ± 0.59		**48.58 ± 0.71**	
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	
Mapillary-LECO	2500	ST-Hard	30.39	30.06	**31.05**	**31.09**	31.73
Mapillary-LECO	9000	ST-Hard	34.08	32.68	**35.40**	**35.72**	36.20

Table 11: **Results of exploiting taxonomic hierarchy** (complete results). We copy the best results in Table 10 (all obtained by $+\mathcal{L}_{Joint}$ $+\mathcal{L}_{SSL}$) for reference. We can see that $\mathcal{L}_{LPL}$ is less effective but still competitive on *Mapillary-LECO*, presumably because it does not model the class-merging scenarios. Nonetheless, combinations of the strategies, i.e., adding SSL loss along with pseudo-label refinement via either **Filtering** or **Conditioning** still bridge the gap to AllFine.

Benchmark	#images / TP	SSL Alg	TP[1] mAcc				
			$+\mathcal{L}_{Joint}$ $+\mathcal{L}_{SSL}$	$+\mathcal{L}_{LPL}$	$+\mathcal{L}_{LPL}$ $+\mathcal{L}_{SSL/Filter}$	$+\mathcal{L}_{LPL}$ $+\mathcal{L}_{SSL/Cond}$	AllFine
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	2000	ST-Hard	48.72 ± 0.53	49.04 ± 0.47	**49.36 ± 0.56**	48.65 ± 0.53	52.72 ± 0.38
		ST-Soft			49.09 ± 0.46	48.54 ± 0.52	
		PL			**49.25 ± 0.34**	48.91 ± 0.87	
		FixMatch			**49.44 ± 0.22**	48.81 ± 0.40	
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	
Mapillary-LECO	2500	ST-Hard	31.09	31.04	**31.41**	31.23	31.73
Mapillary-LECO	9000	ST-Hard	35.72	35.04	**36.12**	35.47	36.20

Table 12: **Complete results of LECO strategies on classification benchmarks**. We report results on combinations of all LECO stratgies on *CIFAR100-LECO* and *iNat-LECO* with **FinetunePrev** strategy on TP[1]. Major conclusions hold across all benchmarks: (1) Utilizing old-ontology samples and labels, i.e., $\mathcal{L}_{SSL}$ and $\mathcal{L}_{Joint}$, helps bridge the gap to **AllFine**, and (2) exploiting taxonomic hierarchy via $\mathcal{L}_{SSL/Filter}$, $\mathcal{L}_{SSL/Cond}$, or $\mathcal{L}_{LPL}$ further improves the performance.

Setup	TP[0] mAcc	TP[1] mAcc																	
			LabelNew																
		AllFine	No SSL				PL			FixMatch			ST-Hard			ST-Soft		
			$\mathcal{L}$	$\mathcal{L}_{LPL}$	$\mathcal{L}_{Joint}$		$\mathcal{L}$	$\mathcal{L}_{LPL}$	$\mathcal{L}_{Joint}$	$\mathcal{L}$	$\mathcal{L}_{LPL}$	$\mathcal{L}_{Joint}$	$\mathcal{L}$	$\mathcal{L}_{LPL}$	$\mathcal{L}_{Joint}$	$\mathcal{L}$	$\mathcal{L}_{LPL}$	$\mathcal{L}_{Joint}$
CIFAR100-1k	50.92 ± 0.89	42.27 ± 0.83	33.53 ± 0.57	38.47 ± 0.52	37.25 ± 0.66	$+\mathcal{L}_{SSL}$	33.53 ± 0.36	38.70 ± 0.65	37.28 ± 0.67	34.20 ± 0.24	37.56 ± 0.54	37.47 ± 0.53	34.59 ± 0.57	36.78 ± 0.40	35.27 ± 0.64	37.67 ± 0.46	38.80 ± 0.49	38.48 ± 0.67
						$+\mathcal{L}_{SSL/Filter}$	33.33 ± 0.69	38.56 ± 0.44	37.44 ± 0.84	36.02 ± 0.33	37.55 ± 0.69	37.23 ± 0.20	37.93 ± 0.50	38.87 ± 0.54	38.57 ± 0.74	37.59 ± 0.69	38.87 ± 0.52	38.43 ± 0.67
						$+\mathcal{L}_{SSL/Cond}$	37.40 ± 1.19	38.67 ± 0.81	38.00 ± 0.59	36.92 ± 0.64	37.35 ± 0.72	37.08 ± 0.57	37.95 ± 0.34	38.68 ± 0.41	38.20 ± 0.30	38.25 ± 0.62	38.19 ± 0.71	38.45 ± 0.49
CIFAR100-2k	59.48 ± 0.66	52.72 ± 0.38	43.44 ± 0.73	49.04 ± 0.47	47.68 ± 0.70	$\mathcal{L}_{SSL}$	43.46 ± 0.61	48.82 ± 0.30	47.24 ± 0.40	44.98 ± 0.59	49.38 ± 0.29	48.58 ± 0.71	44.87 ± 0.35	46.98 ± 0.33	45.67 ± 0.32	48.23 ± 0.55	49.12 ± 0.50	48.72 ± 0.53
						$\mathcal{L}_{SSL/Filter}$	43.50 ± 0.47	49.25 ± 0.34	47.38 ± 0.60	46.14 ± 0.33	49.44 ± 0.22	48.23 ± 1.21	47.00 ± 0.36	49.36 ± 0.56	49.01 ± 0.73	47.62 ± 0.74	49.09 ± 0.46	48.56 ± 0.50
						$\mathcal{L}_{SSL/Cond}$	48.28 ± 0.27	48.91 ± 0.87	48.45 ± 0.34	48.23 ± 0.93	48.81 ± 0.40	48.57 ± 0.58	48.81 ± 0.57	48.65 ± 0.53	48.60 ± 0.68	48.70 ± 0.58	48.54 ± 0.52	48.86 ± 0.61
CIFAR100-10k	77.78 ± 0.27	71.30 ± 0.41	65.83 ± 0.27	69.31 ± 0.30	68.74 ± 0.21	$\mathcal{L}_{SSL}$	65.86 ± 0.14	69.48 ± 0.24	68.43 ± 0.40	67.13 ± 0.36	69.71 ± 0.49	69.03 ± 0.43	67.11 ± 0.34	68.58 ± 0.51	67.97 ± 0.37	69.86 ± 0.30	68.96 ± 0.27	69.77 ± 0.71
						$\mathcal{L}_{SSL/Filter}$	65.80 ± 0.34	69.20 ± 0.20	68.82 ± 0.23	67.60 ± 0.25	69.61 ± 0.17	69.15 ± 0.41	69.04 ± 0.29	69.90 ± 0.29	69.95 ± 0.28	68.76 ± 0.20	70.10 ± 0.12	69.85 ± 0.38
						$\mathcal{L}_{SSL/Cond}$	68.32 ± 0.26	69.60 ± 0.26	69.09 ± 0.13	68.86 ± 0.52	69.85 ± 0.46	69.41 ± 0.39	70.22 ± 0.38	70.42 ± 0.23	70.05 ± 0.18	70.33 ± 0.15	70.02 ± 0.63	69.93 ± 0.24
iNat-50k	64.21 ± 1.61	84.51 ± 0.46	73.64 ± 0.46	83.62 ± 0.21	82.98 ± 0.33	$\mathcal{L}_{SSL}$	73.56 ± 0.09	83.71 ± 0.44	82.87 ± 0.51	74.57 ± 0.54	84.05 ± 0.26	83.00 ± 0.33	75.23 ± 0.14	79.07 ± 0.48	78.23 ± 0.38	79.64 ± 0.41	84.10 ± 0.54	83.61 ± 0.39
						$\mathcal{L}_{SSL/Filter}$	74.01 ± 0.41	83.72 ± 0.22	82.81 ± 0.27	74.55 ± 0.33	83.91 ± 0.39	83.16 ± 0.33	79.91 ± 0.58	84.23 ± 0.61	83.92 ± 0.25	78.71 ± 0.22	84.12 ± 0.31	83.80 ± 0.60
						$\mathcal{L}_{SSL/Cond}$	81.98 ± 0.12	84.16 ± 0.19	82.99 ± 0.64	82.62 ± 0.21	83.95 ± 0.40	83.66 ± 0.36	84.20 ± 0.29	84.34 ± 0.25	84.37 ± 0.39	83.96 ± 0.47	83.76 ± 0.40	83.95 ± 0.20

Table 13: **Complete results of LECO strategies on** *Mapillary-LECO*. We report results on combinations of all LECO stratgies on *Mapillary-LECO* with **FinetunePrev** strategy on TP[1]. It is worth noting that new labels in V2.0 do not necessarily have surjective mappings to old labels of V1.2. This means, it is non-trivial to apply coarse supervision as partial labels. Moreover, we only perform **ST-Hard** for SSL because other methods are excessively compute-demanding (e.g., ST-SOft requires saving per-class heatmaps for each image to allow making use of pseudo-labels' scores, FixMatch requires learning a separate large-scale model, etc.). Concretely, we (1) do not add coarse supervision for $\mathcal{B}_K$ since it pollutes quality of annotation on new data, and (2) mask out gradient where the given V1.2 labels do not align with the parent class for $\mathcal{L}_{old}(\hat{\mathcal{B}}_M)$ and $\mathcal{L}_{oldLPL}(\hat{\mathcal{B}}_M)$. Major conclusions hold here. We can see that all combinations of various LECO stratgies achieve suprisingly good performances, whereas the best combination bridges 95% of the gap to **AllFine** (gap reduces from 1.34 to 0.08 for 2.5k samples with $\mathcal{L}_{SSL}$ and $\mathcal{L}_{LPL}$, and from 2.12 to 0.08 for 9k samples with $\mathcal{L}_{SSL/Filter}$ and $\mathcal{L}_{LPL}$).

Setup	TP[0] mIoU	TP[1] mIoU						
		AllFine	LabelNew					
			No SSL				ST-Hard	
			$\mathcal{L}$	$\mathcal{L}_{LPL}$	$\mathcal{L}_{Joint}$		$\mathcal{L}$	$\mathcal{L}_{LPL}$	$\mathcal{L}_{Joint}$
Mapillary-2.5k	37.01	31.73	30.39	31.04	31.05	$+\mathcal{L}_{SSL}$	30.06	31.65	31.09
						$+\mathcal{L}_{SSL/Filter}$	30.64	31.41	31.57
						$+\mathcal{L}_{SSL/Cond}$	31.01	31.23	31.50
Mapillary-9k	44.14	36.20	34.08	35.04	35.40	$+\mathcal{L}_{SSL}$	32.68	35.97	35.72
						$+\mathcal{L}_{SSL/Filter}$	34.96	36.12	36.06
						$+\mathcal{L}_{SSL/Cond}$	35.29	35.47	35.70

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] Cohen, N., Gal, R., Meirom, E. A., Chechik, G., and Atzmon, Y. (2022). " this is my unicorn, fluffy": Personalizing frozen vision-language representations. *arXiv preprint arXiv:2204.01694*.

[14] 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*.

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

[16] 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.

[17] De Lange, 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. *IEEE transactions on pattern analysis and machine intelligence*, **44**(7), 3366–3385.

[18] 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*.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

[24] 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.

[25] Goel, A., Fernando, B., Keller, F., and Bilen, H. (2022). Not all relations are equal: Mining informative labels for scene graph generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15596–15606.

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

[27] 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.

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

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

[30] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

[31] Hsieh, C.-Y., Xu, M., Niu, G., Lin, H.-T., and Sugiyama, M. (2019). A pseudo-label method for coarse-to-fine multi-label learning with limited supervision.

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

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

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

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

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

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

[38] Jia, M., Tang, L., Chen, B.-C., Cardie, C., Belongie, S., Hariharan, B., and Lim, S.-N. (2022). Visual prompt tuning. *arXiv preprint arXiv:2203.12119*.

[39] 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*.

[40] 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.

[41] 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.

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

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

[44] 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.

[45] Lei, J., Guo, Z., and Wang, Y. (2017). Weakly supervised image classification with coarse and fine labels. In *2017 14th Conference on Computer and Robot Vision (CRV)*, pages 240–247. IEEE.

[46] Li, W.-H., Liu, X., and Bilen, H. (2022). Cross-domain few-shot learning with task-specific adapters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7161–7170.

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

[48] 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.

[49] 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)*.

[50] 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.

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

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

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

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

[60] 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.

[61] Ristin, M., Gall, J., Guillaumin, M., and Van Gool, L. (2015). From categories to subcategories: large-scale image classification with partial class label refinement. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 231–239.

[62] 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*.

[63] 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.

[64] 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.

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

[66] 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.

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

[68] 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*.

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

[70] 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)*.

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

[72] Sun, K., Xiao, B., Liu, D., and Wang, J. (2019). Deep high-resolution representation learning for human pose estimation. In *CVPR*.

[73] Taherkhani, F., Kazemi, H., Dabouei, A., Dawson, J., and Nasrabadi, N. M. (2019). A weakly supervised fine label classifier enhanced by coarse supervision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6459–6468.

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

[75] 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.

[76] Wang, J., Sun, K., Cheng, T., Jiang, B., Deng, C., Zhao, Y., Liu, D., Mu, Y., Tan, M., Wang, X., Liu, W., and Xiao, B. (2019). Deep high-resolution representation learning for visual recognition. *TPAMI*.

[77] Wang, X., Wu, Z., Lian, L., and Yu, S. X. (2022). Debiased learning from naturally imbalanced pseudo-labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14647–14657.

[78] 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.

[79] 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.

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

[81] Yuan, Y., Chen, X., and Wang, J. (2020). Object-contextual representations for semantic segmentation.

[82] Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

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

[84] 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.

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

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

[87] Zhang, J. O., Sax, A., Zamir, A., Guibas, L., and Malik, J. (2020b). Side-tuning: a baseline for network adaptation via additive side networks. In *European Conference on Computer Vision*, pages 698–714. Springer.

[88] 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.