Continual Long-Tailed Recognition: *Merge Tail Classes Today, Separate them Tomorrow*

Yanan Li
Zhejiang Lab
ynli.zju@gmail.com

Shaden Alshammari
MIT
shaden@mit.edu

Jiyong Jin
Zhejiang Lab
jinjy.cn@gmail.com

Shu Kong
Texas A&M University
aimerykong@gmail.com

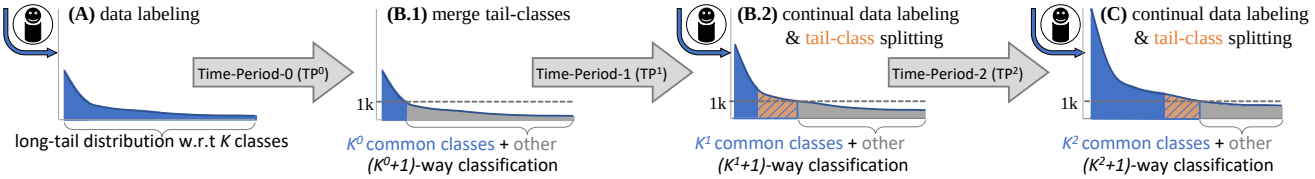

Figure 1. We illustrate a *Continual* practice to address *Long-Tailed Recognition* (CLTR). **(A)** Machine-learned systems often suffer from the long-tailed data distributions w.r.t categorical labels, i.e., many classes "in the tail" have too few examples (e.g., <1k) to train *and* validate their classifiers [5,54]. To address the tail, real-world applications take two insufficiently well-known strategies: **(B.1)** temporarily merge tail-classes (e.g., into a catch-all `other` [20]) and train `other`-classifier, and **(B.2)** continually label more data and train classifiers for tail-classes that have >1k examples. **(C)** This progresses in the life-cycle of machine-learned systems. We explore CLTR as a problem of requiring learning classifiers in distinct time periods (TPs). For example, TP^t defines a (K^t+1)-way classification task, where K^t classes include some tail-classes which have >1k training data now but were merged as `other` in TP^{t-1}; TP^t's `other` class contains the remaining tail-classes that still have <1k training data. Generally, CLTR uses multiple superclasses to merge semantically similar tail-classes (Sec 3). CLTR explores open questions (Sec 4 and 5) such as how to train better by exploiting all class labels, although TP^t focuses on (K^t+1)-way classification, whether it is better to train (K^t+1)-way model from scratch or finetune TP^{t-1}'s model.

Abstract

Real-world data tend to follow long-tailed distributions w.r.t categorical labels, and tail-classes have too few examples to train and evaluate their classifiers. To address the tail, real-world applications such as autonomous vehicles (AVs) often take two insufficiently well-known strategies: (1) temporarily merge semantically similar tail-classes as superclasses which have enough and balanced data to train and evaluate, and (2) continually label more data and train individual classifiers for the previously merged tail-classes when they have enough data. For example, AVs first learn a `wheeled-device` *classifier to recognize tail-classes such as* `stroller` *and* `wheelchair`*, and later learn their own classifiers when they have sufficient data. We study this common yet underexplored continual practice for long-tailed recognition (CLTR). CLTR defines different classification tasks in distinct time periods (TPs); each TP requires learning for classes including some tail-classes that were merged into superclasses in the previous TP.*

CLTR explores open questions such as whether to exploit tail-class labels in training although they are merged as superclasses altogether, whether to finetune the previous TP's model which might be less discriminative for the current TP's task. We answer these questions by leveraging insights from long-tailed recognition and feature learning. We develop a novel method that uses a Mean-Shift module along with Supervised Contrastive loss to learn features with all class labels, and learn a post-hoc classifier for current TP's classification. Experiments validate that our method not only achieves state-of-the-art performance, but also significantly expedites feature learning and allows for fast finetuning in subsequent TPs.

1. Introduction

Real-world data tend to follow long-tail distributions w.r.t categorical labels [44, 56, 61], motivating the widely studied problem of long-tailed recognition (LTR). While the LTR literature emphasizes training on classes "in the tail" that have insufficient training data, it largely overlooks the fact that such classes also have too few examples to validate classifiers [5, 10, 55, 63]. Consequently, many benchmarks either ignore such tail-classes [15, 55, 63] or provide artificially large validation sets for hyperparameter tuning and model selection [2, 8, 40] (cf. suppl).

Motivation. Real-world applications must not ignore tail-classes. For example, autonomous vehicles (AVs) must not ignore rare vulnerable objects like `wheelchairs` and `strollers` [5, 10, 54]; video surveillance must not ignore various kinds of rarely-occurring `terrorist-attacks`

and `criminal-activities` [46]. We are inspired to address LTR in a continual perspective, termed *Continual Long-Tailed Recognition* (CLTR). CLTR temporarily merges tail-classes into superclasses so to have enough and balanced data for training superclasses' classifiers [10, 54], continually labels more data, and when some tail-classes have enough labeled data, separates them from the previous superclasses and trains their own classifiers (Fig. 1). For example, the AV dataset nuScenes [5] merges tail-classes `child`, `construction-worker`, `police-officer` into `pedestrian`; Argoverse V1 has 15 classes [10] but V2 expands to 30 [54] by separating V1's superclasses (e.g., V2's `school-bus` is from V1's `bus`).

Why CLTR? Although CLTR is a real-world practice in many applications, one may ask why CLTR should be regarded as an interesting research problem because it largely balances class distributions and hence nullifies the LTR challenge. First, CLTR is important because it formulates an underexplored practice to address LTR in the real world. It acknowledges the fact that many tail-classes do not even have enough data for validation, a fact neglected in the current LTR setups. Second, CLTR offers new challenges and exploration space to answer interesting questions, e.g., whether to exploit tail-class labels in training although they are merged as superclasses in a TP, whether to finetune the previous TP's model which might make previously-merged tail-classes indistinguishable, etc. Third, by exploring CLTR, we derive a rather simple method (detailed below) that achieves state-of-the-art and allows for fast training and finetuning. This has practical significance for real-world applications.

Technical Insights. As CLTR defines a $(K^t{+}S^t)$-way classification task (Fig. 1) in TP^t, a naive method is to directly learn a $(K^t{+}S^t)$-way classifier by merging relevant tail-classes into the S^t superclasses. However, this does not make use of tail-class labels, which are presumably useful for learning better models. To exploit such, one may learn the K-way classifier and post-process it for $(K^t{+}S^t)$-way classification, e.g., by summing softmax scores of the tail-classes as the S^t superclasses. Learning a K-way classifier is a standard LTR problem that must strike a balance between all classes and does not serve the $(K^t{+}S^t)$-way classification at the current TP. Inspired by recent LTR works that decouple feature learning and classifier learning [2, 25], we propose a simple method that achieves the state-of-the-art (cf. Fig. 2). It first learns features with a Supervised Contrastive loss (SupCon) [29] using all labels, and then the $(K^t{+}S^t)$-way classifier atop of the learned features. Such feature learning enables finetuning the previous TP's model and achieves fast convergence in new TPs. Further, motivated to accelerate feature learning, we incorporate a Mean-Shift grouping module [9, 13, 14, 21, 31] into SupCon, achieving better performance and speeding up learning.

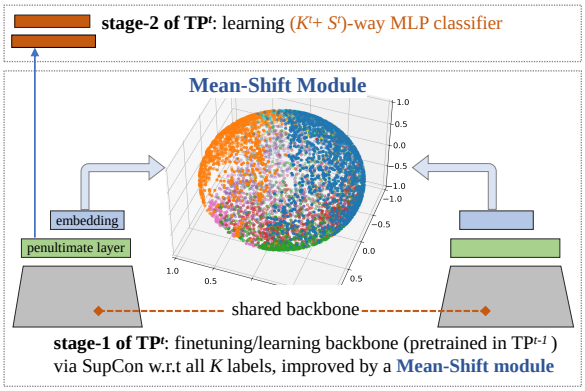

Figure 2. Our approach to CLTR has two training stages: (1) feature learning and (2) classifier learning. To learn features, we use the Supervised Contrastive learning (SupCon) [29] w.r.t all K labels, producing more generalizable features than those learned by the cross entropy loss [2, 25]. Importantly, in SupCon, we incorporate a differentiable Mean-Shift (MS) module [31], which performs gradient ascent on the features. The MS module exacerbates hard examples (Fig. 3) during training, leading to expedited training and better performance. Importantly, in subsequent TPs, finetuning the previous TP's SupCon model with the MS module significantly improves convergence (Fig. 6). For classifiers, it generally works the best by learning a multilayer perceptron (MLP) at the penultimate layer of the feature backbone using class-balanced loss [17]. Our method achieves state-of-the-art for CLTR.

Contributions. We make three major contributions. First, we formulate the problem Continual Long-Tailed Recognition (CLTR) and define its benchmarking protocol. Second, we extensively study approaches to CLTR by modifying methods from related problems, including feature learning and long-tailed recognition. Third, we develop a rather simple approach that achieves the state-of-the-art and allows for fast finetuning the previous TP's model, showing practical significance for real-world applications.

2. Related Work

Long-Tailed Recognition (LTR) emphasizes per-class accuracy of a classifier trained over long-tailed distributed data w.r.t K classes [8, 25, 57, 61]. Existing LTR methods aim to balance various aspects. *Data re-balancing* resamples the training data to achieve a more balanced distribution across classes [41, 47], such as over-sampling rare-classes [11, 22] and undersampling common-classes [19]. *Loss reweighting* assigns weights to the classes [8, 17, 24, 27, 28, 60] or training examples [27, 36, 45, 48], aiming to achieve "balanced" gradients during training. Recent work convincingly shows that learning more discriminative features is crucial to LTR [25, 62], allowing simple *weight balancing* [2] to rival state-of-the-art methods [6, 16, 52]. Perhaps surprisingly, for model tuning, most LTR benchmarks use artificially larger validation sets than training sets of

tail-classes (cf. suppl). This hides the real-world issue that if a tail class does not have enough data for training, it does not have enough data for validation too. Our work confronts this issue with the proposed CLTR protocol, *merging tail-classes as superclasses for training and evaluation, and separating them later when they have enough labeled data.*

Class Merging is common in dataset curation for real-world applications such as autonomous vehicles (AVs). The AV dataset nuScenes [5] merges some tail-classes into superclasses because (quoted) "some of these (classes) only have a handful of annotations" and incorrect predictions within each superclass do not affect the performance for motion planning (for now) [4]. For example, it merges `child`, `construction-worker`, `police-officer` into `pedestrian`. Other AV datasets also contain such superclasses, e.g., Argoverse [10] has `other-mover`, `large-vehicle` and `animal`; Mapillary [43] has `other-vehicle` and `other-rider`; Cityscapes [15] has `dynamic` and `static`. Many non-AV datasets also merge classes. For example, the iNaturalist [51] creates a catch-all class `Animalia` to include diverse species. In terms of merging classes, recent work [32] shows that even simply merging tail-classes into a single catch-all `other` can improve the robustness of a trained model. Our work is inspired by the *class merging* and formulates it as a practical research problem CLTR to address the long-tailed issue.

Class Splitting means a (super)class is continually divided into fine-grained subclasses. It is well demonstrated by *dataset versioning*, reflecting new application requirements changed/refined over time. For example, Mapillary V1.2 has 66 classes [43] and expands to 124 classes in V2.0 [42]; Argoverse expands its 15 classes in V1 [10] to 30 classes in V2 [54] because it has more labeled data for tail-classes to be split out from previous superclasses. Datasets that have not versioned yet also anticipate future versioning. For example, SemanticKITTI [3] states "`other-structure`, `other-vehicle`, and `other-object`... might be used to distinguish these (fine-grained) categories further in future". These datasets hints another practice to address LTR, i.e., separating superclasses into tail-classes with sufficient data labeled continually. Our CLTR explicitly formulates this practice.

Class-Incremental Learning (CIL) and CLTR require learning new classes in distinct time periods (TPs) [18, 35, 50, 59]. Differently, CLTR's "new" classes are pre-defined tail-classes that are merged into superclasses in the previous TP; CIL's new classes are never-before-seen that have no relation to previous classes [1, 35, 39]. CIL emphasizes catastrophic forgetting with a limited buffer [7, 30, 35, 58], assuming abundant labeled data for both old and new classes; CLTR focuses on addressing tail-classes that have insufficient data. Compared to CIL, CLTR is a new problem.

3. Continual Long-Tailed Recognition

In this section, we formulate the problem Continual Long-Tailed Recognition (CLTR), illustrated by Fig. 1.

3.1. Formulation and Benchmarking Protocol

Problem formulation. In CLTR, data are labeled w.r.t K predefined classes and follow a long-tailed distribution w.r.t class labels. CLTR requires learning classifiers in distinct time periods (TPs). The t-th TP (denoted TP^t) has N^t new labeled examples in set $\mathcal{D}^t = \left\{(\mathbf{x}_i^t, y_i^t)\right\}_{i=1}^{N^t}$, where $\mathbf{x}_i^t$ and $y_i^t \in \{1, \dots, K\}$ are the i-th image and its ground-truth label, respectively. TP^t is concerned with the $(K^t{+}S^t)$-way classification task, where K^t is the number of common classes that have $\geq$*thresh* training data and S^t is the number of superclasses, each of which merges some semantically similar tail-classes that have $<$*thresh* training data.[1] That said, TP^t has a mapping function $\tilde{y}_i^t = g^t(y_i^t)$ from $y_i^t \in \{1, \dots, K\}$ labels to $\tilde{y}_i^t$, one of the $(K^t{+}S^t)$ labels. When training in TP^t, one can accumulate all historical training data, i.e., $\mathcal{D}^{0:t} = \cup_{i=1}^t \mathcal{D}^i$.

Validation. Each TP has a validation set (val-set) for hyperparameter tuning and model selection. The val-set follows the same long-tailed class distribution as the train-set.

Test-set and evaluation metrics. We use a held-out test-set for benchmarking. In TP^t that defines the $(K^t{+}S^t)$-way classification, we compute the accuracy averaged over $(K^t{+}S^t)$ classes. We also report the mean of per-TP accuracies as a summary number for easy comparison.

3.2. Discussions

Contrast to Class Incremental Learning (CIL). CLTR requires learning classifiers in TPs similar to CIL. Unlike a CIL's TP which provides data from novel classes, CLTR annotates data with the predefined K class labels. While CLTR's TP^t defines the specific $(K^t{+}S^t)$-way classification task, it is a design choice whether to use all labels in training. Moreover, CIL typically limits a buffer size to store history data with a hidden assumption that the data are abundant for all classes. Yet, CLTR has long-tailed class distributed data that many tail-classes have insufficient data, hence we do not limit the buffer to store data. A common pursuit for both is to efficiently adapt the previous TP's model to the current TP. CIL approximates the efficiency measure using a limited buffer; we explore this by analyzing performance as a function of training epochs (Fig. 6).

Human-in-the-loop labeling is indispensable in CTLR to produce more labeled data, similar to what is needed in continual learning. Data labeling can be done in different strategies, e.g., top-down labeling using the predefined K classes and bottom-up labeling that annotates data

[1] Real applications decide merging rules, e.g., the resulting superclasses should have $>$*thresh* data and be more balanced than the raw K classes.

with an open vocabulary. This work assumes the former and a uniform data sampling. This is a quite common practice in many datasets such as Sport1M [26], PASCAL VOC [20], SemanticKITTI [3], SUN [55], COCO [37], Cityscapes [15], and Mapillary [43]. Therefore, all data follow the same long-tailed distribution w.r.t K classes.

Novel classes. In contrast to CIL, CLTR does not have data of novel classes in TPs because of the top-down labeling policy. This may be a limitation of the current CLTR setup because the opposite bottom-up labeling might produce novel classes. We argue that, as data follow a long-tailed distribution, novel classes emerge only in the tail, which are rarely encountered. So to catch all rarely-seen and novel classes, we can define an `other` superclass [32].

4. Methodology

To better understand CLTR's challenges and difficulties, we establish baseline methods by modifying LTR methods rather than developing sophisticated ones. We present methods for initial training in TP^0 (Sec 4.1) and continual training in subsequent TPs (Sec 4.2). Further, we propose a novel method in Sec 4.3 to expedite training.

4.1. Initial Training in TP^0

In TP^t with t=0, we study methods below that have different choices as to whether to use all K labels for training.

(K^t+S^t)-way method learns a (K^t+S^t)-way classifier from TP^0's data and concerned classes. For each training example $(\mathbf{x}, y)$, we reassign the label $\tilde{y}$ of interest in TP^0 with a designed mapping function $\tilde{y} = g^t(y)$. In particular, we train a neural network $f(\cdot;\boldsymbol{\theta}^t)$ parameterized by $\boldsymbol{\theta}^t$ that takes as input $\mathbf{x}$ and outputs a softmax vector $\mathbf{s} = f(\mathbf{x};\boldsymbol{\theta}^t) \in \mathbb{R}^{K^t+S^t}$. We use a cross-entropy loss to train $f(\cdot;\boldsymbol{\theta}^t)$. This method directly optimizes for TP^0's task but does not exploit all (tail-)class labels. Hypothetically, using all labels might improve training.

K-way with post-process. A simple way to exploit all labels is to train a K-way classification model and then process the K-way softmax probabilities towards (K^t+S^t)-way classification. We train the K-way model using the cross-entropy loss. To process the probabilities, we tested: (1) simply taking the argmax y' of the K-way softmax scores and reassign the predicted label using the mapping function, i.e., $\tilde{y}' = g^t(y')$; (2) summing scores of the tail-classes according to the S^t superclasses; and (3) averaging scores of the tail-classes according to the S^t superclasses. While (3) is probabilistically meaningful, we find that others can empirically work better due to uncalibrated scores. We report the best numbers of these methods after validation. Regardless of post-process techniques, training the K-way classifier is the standard LTR problem, which must strike a balance between imbalanced classes, thus producing suboptimal performance for CLTR. Inspired by [2, 25], we next present two-stage methods that use all labels for feature learning followed by classifier learning.

Two-stage methods decouple feature learning and classifier learning in two training stages. For feature learning, we train a model $\mathcal{F}(\cdot;\boldsymbol{\Theta})$ with parameters $\boldsymbol{\Theta}$ that produces an embedding feature $\mathbf{z} \in \mathbb{R}^d$ for a given input image $\mathbf{x}$, i.e., $\mathbf{z} = \mathcal{F}(\mathbf{x};\boldsymbol{\Theta})$. In particular, we train $\mathcal{F}(\cdot;\boldsymbol{\Theta})$ using all K labels and the Supervised Contrastive loss (SupCon) [29], which shows superior performance over cross-entropy loss for downstream tasks. Below is the SupCon loss:

$$\mathcal{L} = \sum_{(\mathbf{x}_i, y_i)\in\mathcal{D}^0} \frac{-1}{|P(y_i)|} \sum_{p\in P(y_i)} \log \frac{\exp\left(\mathbf{z}_i^T \mathbf{z}_p/\tau\right)}{\sum\limits_{a\in A(\mathbf{x}_i)} \exp\left(\mathbf{z}_i^T \mathbf{z}_a/\tau\right)}$$

where $P(y_i)$ is the set of all positive examples given label y_i within the training batch, and $A(\mathbf{x}_i)$ denotes all samples in the batch excluding $\mathbf{x}_i$. τ is a scalar temperature parameter set as 0.07 in our work.

Once $\mathcal{F}(\cdot;\boldsymbol{\Theta})$ is trained, we train a (K^t+S^t)-way classifier ($t = 0$ in TP^0) over it. For classifier training, we use either the cross-entropy loss or the class-balanced loss [17], with the latter mitigating class imbalance issue. We study the following classifiers:

- **LIN**: a linear classifier for (K^t+S^t)-way classification.
- **MLP**: a two-layer multilayer perceptron (MLP) as the (K^t+S^t)-way classifier [53].

We can either freeze the feature backbone to train classifiers or train the whole model. We find that the latter performs better (see details in the supplement). We also study which layer's features to use for classifier learning. We find that using the penultimate layer tends to outperform the last layer. Fig. 2 sketches our method.

4.2. Continual Learning in Subsequent TPs

After training in TP^{t-1} ($t = 1, \ldots, T$), we study training strategies in TP^t. There are three options depending on how and whether to use the model previously trained in TP^{t-1}.

- **Fixing Backbone** ("fix") freezes TP^{t-1}'s backbone and only trains the (K^t+S^t)-way classifier.
- **Train-from-scratch** ("sc") trains the whole model and the (K^t+S^t)-way classifier with a random initialization. That said, it does not use the TP^{t-1}'s model.
- **Finetuning Backbone** ("ft") finetunes model trained in TP^{t-1} and learns a classifier simultaneously for (K^t+S^t)-way classification.

Importantly, an open question in "ft" is what models to finetune. For the K-way method, we can finetune the TP^{t-1}'s model either with or without the classifier for TP^t. For the (K^{t-1}+S^{t-1})-way and the two-stage methods, we can finetune the backbone $\mathcal{F}^{t-1}(\cdot;\boldsymbol{\Theta})$ right after the first training stage using SupCon or after the second-stage training. We tested all the choices and found that the last performs consistently the best.

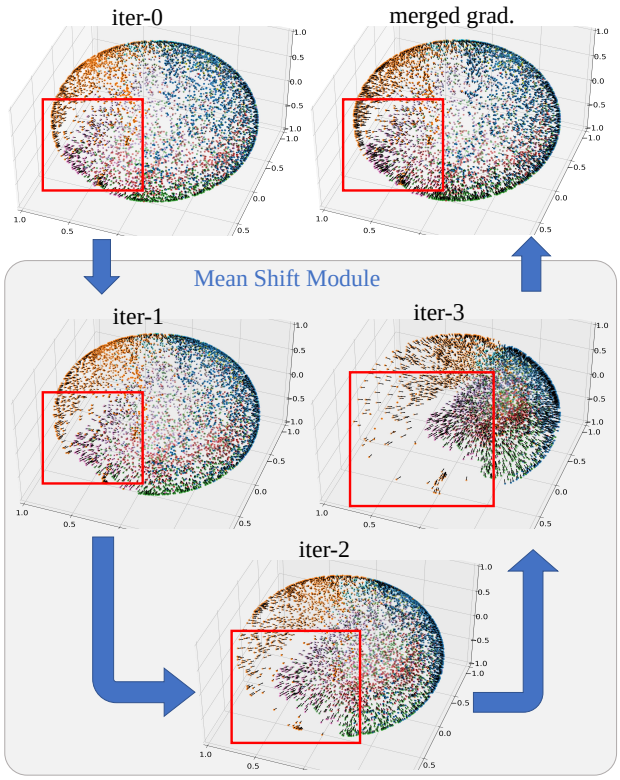

Figure 3. Mean-Shift (MS) helps Supervised Contrastive (SupCon) learning by making hard examples even harder. To demonstrate this, we train a Res18 model to output 3-dim features using SupCon and MS on the CIFAR10-CLTR (IF 0.1, in TP^0, cf. details in the Experiment). Given a checkpoint, we visualize the features of training data as points on a sphere, color them using their ground-truth labels, and plot their gradients using arrows with length and direction indicating the magnitude and derivative, respectively. Iter-0 shows gradients computed by a normal SupCon loss. Clearly, running MS amplifies gradients on the points which are clustered (by MS) into the incorrect data groups (see red box). That said, the MS module makes these points even harder for SupCon, forcing training to focus on these hard examples. Our loss merges the gradients from all the MS iterations. In ths way, hard examples have much larger gradients than easy ones (zoom in for better visualization). Fig. 5 shows that MS greatly improves SupCon w.r. speed and accuracy.

Moreover, "sc" should perform no worse than "ft" because it accesses all data and labels. However, "sc" is slow and ft can be much faster because the TP^{t-1}'s backbone might be a good initialization or fast adapted in TP^t. Motivated by fast adaptation, we study the following to make use of a Mean-Shift module in SupCon, showing improved classification accuracy and training convergence (Fig. 6).

4.3. Improve Learning via Mean-Shift

Motivated by the desired fast training / adaptation and inspired by [31], we propose to use a Mean Shift module in feature learning with SupCon. The literature shows

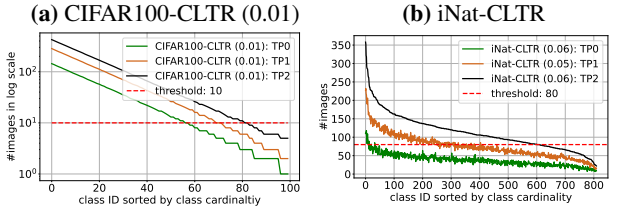

Figure 4. Sorted per-class cardinalities in the three TPs of two benchmarks. Horizontal lines are the *threshold* set before hand. In CIFAR100-CLTR (0.01), we use a single `other` superclass to catch tail-classes that have $<$ *threshold* training examples in each TP. In iNat-CLTR, we group species labels according to the seven levels of taxonomic labels (e.g., kingdom, phylum, class, family, order, genus, and species). Refer to the supplement for details.

that Mean Shift improves feature learning because its mode seeking mechanism clusters some hard examples in incorrect groups w.r.t the ground-truth class labels; this produces larger gradients for these hard examples and hence making better use of them to enhance learning [31] (Fig. 3).

Given a training batch consisting of B examples and its d-dimensional embedding features $\mathbf{Z} \in \mathbb{R}^{d\times B}$, an iteration of Mean Shift updates the features as below [9]:

$$\begin{aligned}\mathbf{Z} &\leftarrow \mathbf{Z} + \eta(\mathbf{Z}\mathbf{K}\mathbf{D}^{-1} - \mathbf{Z}),\\ &\leftarrow \mathbf{Z}(\eta\mathbf{K}\mathbf{D}^{-1} + (1-\eta)\mathbf{I})\end{aligned} \tag{1}$$

where $\mathbf{K} = \exp(\delta\mathbf{Z}^T\mathbf{Z})$ is the kernel matrix. $\mathbf{D} = diag(\mathbf{K}^T\mathbf{I})$ is the diagonal matrix of total affinities, referred to as the degree when $\mathbf{K}$ is viewed as a weighted graph adjacency matrix. We set $\delta = 0.1$ and $\eta = 0.5$ in our work.

We illustrate why Mean Shift improves model training in Fig. 3. Quantitatively, we show how much Mean Shift improves learning the initial model in TP^0 in Fig. 5, and how much it improves finetuning in the subsequent TP^1 w.r.t learning speed and accuracy in Fig. 6.

Remark. Incorporating Mean Shift in deep learning is not new. [31] uses Mean Shift to learn per-pixel embedding features to group pixels into object instances; [33] uses Mean Shift to generate pseudo-labels for self-supervised contrastive learning. However, to the best of our knowledge, our work makes the first attempt to use Mean Shift for supervised feature learning with SupCon. We show that it significantly expedites feature learning and finetuning while maintaining the state-of-the-art performance for CLTR.

5. Experiments

We conduct experiments to justify the benefits of two-stage methods and the use of Mean Shift module in feature learning along with SupCon, as described in Sec 4. We start with datasets and implementation in Sec 5.1, carry out ablation study in Sec 5.2 and analyze benchmarking results in Sec 5.3. We include our open-source code in the supplement and will release it to the public.

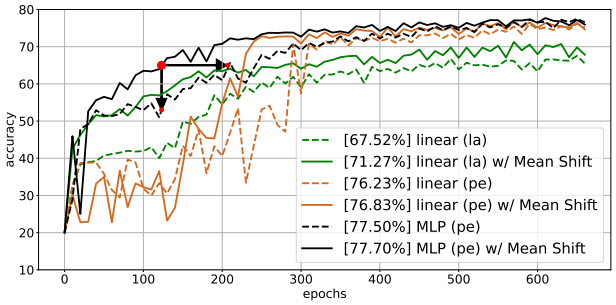

Figure 5. **Mean-Shift (MS) improves SupCon w.r.t learning speed and accuracy.** We train a Res18 backbone using SupCon on CIFAR10-CLTR (0.1) in TP^0 (cf. Experiments for details). Keywords "la" and "pe" mean the last and penultimate layer. At each epoch, we freeze the backbone checkpoint and train (K^0+1)-way classifiers. A linear classifier on 128d features at the last layer achieves 71.27% with MS, significantly better than 67.52% without. Further, we follow [53] to learn MLP classifiers at the penultimate layer (512d features). Using MS significantly expedites training: to achieve 65% (red marker), it takes 123 epochs with MS otherwise 207 epochs; it also achieves the best performance (77.70%) with MS, slightly better than without (77.50%).

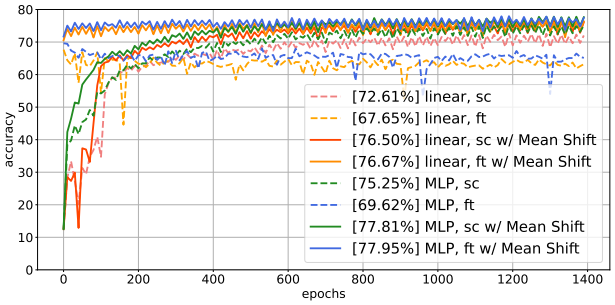

Figure 6. **Finetuning with Mean-Shift improves training speed and accuracy in TP^1.** We follow the same setup as in Fig. 5 but train in TP^1. We compare training-from-scratch (sc) and finetuing (ft), with and without the Mean Shift module. We present three salient conclusions. (1) Naively finetuning the previous TP's backbone yields significantly worse performance than train-from-scratch (67.65% vs. 72.61% using linear classifiers) presumably because such a backbone serves as a less effective local minima in SupCon finetuning. (2) Applying Mean-Shift module to finetuning boosts the linear classifier's accuracy to 76.67%, surpassing train-from-scratch (72.61%)! (3) The above conclusions hold for the stronger MLP classifier, which performs the best (77.95%).

5.1. Datasets and Implementations

Datasets. We construct CLTR benchmarks using publicly available datasets including CIFAR10 [34], CIFAR100 [34], and semi-iNaturalist [49, 51], which have 10, 100, and 810 classes respectively. For CIFAR10 and CIFAR100, we first follow [17] to prepare long-tailed versions with imbalance factors (IF) 0.1, 0.02, and 0.01. We use their official validation set as our test-set. From the official train-set, we randomly sample 15% examples of each class as our val-set, and evenly split the rest into three subsets as three TPs. With a benchmark name like CIFAR10-CLTR (0.01), we mean the benchmark is derived from CIFAR10 with IF=0.01. For CIFAR based benchmarks, we use a single `other` class to catch tail-classes that have $<$*threshold* training examples. We set the *thresholds* as (600, 300, 300) and (45, 10, 10) for CIFAR10-CLTR (0.1, 0.02, 0.01) and CIFAR100-CLTR (0.1, 0.02, 0.01), respectively.

The semi-iNaturalist dataset has a long-tailed class distribution. We merge all its labeled subsets and sample examples to form our val-set and three training subsets corresponding to three TPs. Interestingly, this dataset has seven levels of labels: kingdom, phylum, class, order, family, genus, and species. We set the *threshold* as 80 and merge the species labels at different levels to form multiple superclasses. We name this benchmark iNat-CLTR.

We show the cardinality distributions of the classes of the CIFAR100-CLTR (0.01) and iNat-CLTR in Fig. 4. We refer the readers to the supplement for more details about benchmarks. As the conclusions hold across all the benchmarks, we move the experiment results on CIFAR10-CLTR (0.02) and CIFAR100-CLTR (0.02) to the supplement.

Implementations. We implement all methods with the PyTorch toolbox. We use Res18/50 [23] on the CIFAR and iNat benchmarks. In the training stage of SupCon, we train 2,000 epochs with a batch size of 400 and a learning rate of 0.5 to learn the feature backbone. While in the training stage of classifiers, we train 300 epochs with a batch size of 128 / 400 on CIFAR and iNat benchmarks. We use the SGD optimizer with momentum 0.9, weight decay $1e^{-4}$, and the cosine learning rate scheduler. We use the validation set for model selection and hyperparameter tuning. We conduct all experiments with two NVIDIA RTX A6000 GPUs.

5.2. Ablation Study

We conduct a comprehensive ablation study on the CIFAR10-CLTR (0.1) benchmark. Particularly, we analyze the important components of the methods described in Sec 4 (1) for training the initial model in TP^0, and (2) for continually training in subsequent TPs. Table 1 lists detailed results, and below we explain the most important conclusions.

Ablation study in TP^0. The macro-column "TP^0" in Table 1 lists comparisons, and Fig. 5 shows how these methods perform during training. We summarize three salient conclusions. First, two-stage methods that adopt SupCon for feature learning followed by classifier learning significantly outperform K-way and (K^t+S^t)-way methods. For example, using a linear classifier atop of SupCon trained feature backbone (SupCon LIN (la)) achieves 69.36%, much better than K-way (61.91%) and (K^t+S^t)-way (62.62%) models. Second, applying the Mean Shift module to SupCon training slightly improves accuracy (e.g., 76.96% vs. 75.59% by SupCon LIN (pe) with vs. without Mean Shift). Im-

Table 1. **Ablation Study CIFAR10-CLTR (0.1)** w.r.t top-1 accuracy (%). We average the results on all (K^t+S^t) classes across all TPs for easy comparison. The keywords "la" and "pe" mean the last and penultimate layer, "LIN" and "MLP" mean the linear and MLP classifiers. We make the best numbers bolded and underline the best results with the finetuning (ft) methods. We summarize the salient conclusions below. (1) Two-stage methods significantly outperform the K-way and (K^t+S^t)-way methods for both initial training in TP^0 and continual learning in subsequent TPs, e.g., 77.87% by SupCon MLP (pe) vs. 61.91% / 62.62% by K-way and (K^t+S^t)-way methods in TP^0. (2) Applying Mean Shift to SupCon yields better or on-par performance compared to SupCon only (e.g., 71.40% vs. 69.36% using SupCon LIN (la) with / without Mean Shift), and significantly expedite training (3) In subsequent TPs, finetuning (ft) the previous TP's model achieves on-par performance compared to training from scratch (sc), which should be thought of as an upperbound. Importantly, ft significantly accelerates training, as demonstrated by Fig. 6. (4) With the two-stage methods, training MLP classifiers at the penultimate layer (pe) performs the best, better than linear classifiers and those on the last layer (la).

	TP^0			TP^1				TP^2				
method	(K^0+S^0)	K^0	S^0	mode	(K^1+S^1)	K^1	S^1	mode	(K^2+S^2)	K^2	S^2	*avg.*
K-way + post-proc.	61.91	64.78	50.45	fix	54.79	56.69	41.50	fix	54.16	56.30	37.00	56.95
				sc	59.57	62.26	40.77	sc	64.19	66.26	47.60	61.89
				ft	59.60	62.54	38.97	ft	61.00	63.31	42.50	60.84
(K^t+S^t)-way	62.62	63.70	58.30	fix	51.69	54.27	33.63	fix	66.71	67.87	57.35	59.54
				sc	66.94	68.99	52.60	sc	74.81	75.74	67.35	68.12
				ft	67.25	69.31	52.77	ft	73.87	74.48	69.05	67.91
SupCon LIN (la)	69.36	76.22	41.90	fix	62.70	66.26	37.77	fix	63.51	63.46	63.85	65.19
				sc	66.12	68.23	51.40	sc	69.09	68.03	77.65	68.19
				ft	64.11	66.24	49.17	ft	67.40	66.46	74.90	66.96
SupCon LIN (la) + MeanShift	71.40	76.55	50.82	fix	65.68	67.51	52.80	fix	65.77	65.15	70.70	67.62
				sc	69.60	70.34	64.43	sc	74.11	72.76	84.90	71.70
				ft	67.80	69.13	58.53	ft	69.07	68.31	75.15	69.42
SupCon LIN (pe)	75.59	81.73	51.07	fix	69.25	70.91	57.57	fix	68.66	68.17	72.55	71.17
				sc	73.07	74.19	65.23	sc	74.13	73.32	80.55	74.26
				ft	71.65	72.71	64.23	ft	72.39	71.39	80.45	73.21
SupCon LIN (pe) + MeanShift	76.96	81.28	**59.70**	fix	70.77	71.49	65.73	fix	69.87	69.02	76.60	72.53
				sc	76.46	76.79	74.20	sc	80.13	79.58	84.60	77.85
				ft	73.80	74.20	70.97	ft	76.45	75.64	82.95	75.74
SupCon MLP (pe)	**77.87**	82.62	58.83	fix	71.35	71.94	67.23	fix	70.39	69.21	79.85	73.20
				sc	76.28	77.13	70.33	sc	77.36	76.61	83.35	77.17
				ft	72.93	73.67	67.77	ft	76.25	75.36	83.35	75.68
SupCon MLP (pe) + MeanShift	77.42	**83.00**	55.10	fix	72.20	72.73	68.53	fix	71.03	69.95	79.70	73.55
				sc	**77.66**	**78.17**	**74.07**	sc	**81.23**	**80.54**	**86.80**	**78.77**
				ft	74.85	75.10	73.10	ft	78.33	78.45	77.35	76.87

portantly, the Mean Shift significantly accelerates feature learning as shown in Fig. 5. Third, training an MLP classifier atop of the penultimate layer of the feature backbone performs the best. This is also reported in the literature [53].

Ablation study in TP^t ($t = 1, \ldots, T$). We study how different training strategies ("sc", "ft", and "fix") perform in subsequent TPs. These methods differ in whether/how to use the feature backbone trained in the previous TP. In fact, "sc" can be thought of as an upper bound because it accesses all training data (from all prior and current TPs). The macro-columns "TP^1" and "TP^2" in Table 1 list detailed comparisons. We further analyze how these methods perform over training epochs in Fig. 6. We summarize three salient conclusions. First, the two-stage methods generally outperform the K-way and (K^t+S^t)-way methods when equipped with proper classifiers such as linear and MLP on the penultimate layers. Second, over the (K^t+S^t)-way method, "fix" performs the worst because the feature backbone learned in the previous TP (for classes including the catch-all `other`) is insufficient to discriminate fine-grained tail classes separated from the previous `other`. However, with SupCon, "fix" performs well because SupCon uses K class labels in feature training, producing a sufficiently good feature representation even in the previous TP which has less training data. Third, applying Mean Shift to SupCon in "ft" significantly expedites convergence as convincingly shown by Fig. 6. This suggests that the Mean Shift module makes better use of hard examples from more labeled data to tweak previously trained feature backbone towards a more discriminative one.

5.3. Benchmarking Results

Given the good trade-off of finetuning ("ft") between accuracy and training/finetuning speed (Table 1), we adopt "ft" in subsequent TPs. Table 2 lists benchmarking results, along with Table 1 on the CIFAR10-CLTR (0.1). We re-

Table 2. **Benchmarking results on more datasets** w.r.t top-1 accuracy (%). The keywords "la" and "pe" mean the last and penultimate layer, "LIN" and "MLP" mean the linear and MLP classifiers. We average the results on all (K^t+S^t) classes across all TPs for easy comparison. Salient conclusions summarized in Table 1 generally hold across benchmarks. Importantly, conclusions generalize from the CIFAR based benchmarks that use a single `other` to catch tail-classes to multiple superclasses as in iNat-CLTR. It is worth noting that linear classifiers sometimes outperform MLP classifiers, e.g., with the two-stage methods that learn features with SupCon and Mean Shift: LIN (la) outperforms MLP (la) on iNat-CLTR, i.e., 67.75% vs. 66.68%. We refer the readers to Section 5.3 for more analyses.

	CIFAR10-CLTR (0.01)				CIFAR100-CLTR (0.1)				CIFAR100-CLTR (0.01)				iNat-CLTR			
	TP^0	TP^1	TP^2	avg	TP^0	TP^1	TP^2	avg	TP^0	TP^1	TP^2	avg	TP^0	TP^1	TP^2	avg
K-way + post-proc.	59.59	56.52	54.38	56.83	29.33	33.54	34.51	32.46	21.22	23.71	26.53	23.82	48.27	56.50	61.04	55.27
(K^t+S^t)-way	56.55	50.65	47.23	51.48	24.08	32.57	32.08	29.58	21.64	25.40	24.55	23.86	44.11	55.10	60.82	53.34
SupCon + LIN (la)	61.18	61.78	57.20	60.05	28.36	31.87	32.74	30.99	20.48	22.45	21.47	21.47	55.64	69.53	73.35	66.17
SupCon + LIN (la) + MS	64.13	61.82	56.36	60.77	29.29	32.36	32.78	31.48	22.91	23.60	22.85	23.12	**58.81**	**70.63**	**73.83**	**67.75**
SupCon + LIN (pe)	68.72	65.18	62.05	65.32	31.11	34.56	**39.24**	34.97	23.04	27.43	**27.37**	**25.95**	56.27	65.20	70.62	60.03
SupCon + LIN (pe) + MS	69.24	70.28	63.61	67.71	31.91	34.92	38.84	**35.22**	23.36	27.23	27.08	25.89	56.92	65.61	71.74	64.75
SupCon + MLP (pe)	70.06	67.29	63.25	66.87	31.47	34.96	32.86	33.10	23.31	25.22	24.74	24.42	57.72	66.93	71.81	64.54
SupCon + MLP (pe) + MS	**70.87**	**71.40**	**68.56**	**70.28**	**32.18**	**35.25**	34.59	34.00	**23.38**	**25.95**	25.51	24.95	58.60	67.57	72.92	66.68

fer the reader to the supplement for more results on more benchmarks. We present three salient conclusions below.

Two-stage methods are clearly better than K-way and (K^t+S^t)-way methods. For example, on iNat-CLTR, SupCon + MLP (pe) achieves 57.72%, nearly 10% higher than the K-way method (48.27%) and >13.0% higher than the (K^t+S^t)-way method (44.11%). This demonstrates the importance of learning discriminative features, particularly using all K labels with the SupCon loss.

Mean Shift helps both initial training and finetuning. Recall that Mean Shift significantly improves training/finetuning speed (Fig. 5 and 6). It also improves the final classification accuracy in most cases. For example, on iNat-CLTR, SupCon + MLP (pe) achieves 58.60% and 57.72% with and without Mean Shift. Please refer to Fig. 3 for the reason why Mean Shift helps feature learning.

While MLP tends to outperform linear classifiers, the latter sometimes rivals the former. For example, with the two-stage methods that learn features with SupCon and Mean Shift: LIN (la) outperforms MLP (la) on iNat-CLTR, 67.75% vs. 66.68%; LIN (pe) outperforms MLP (pe) on CIFAR100-CLTR (0.1), 35.22% vs. 34.00%. We argue that the linear classifier with SupCon and Mean Shift should be a baseline for its simplicity and competitive performance.

6. Discussions and Limitations

Societal Impacts. Because real-world data tends to follow long-tailed distributions, our work has multiple positive societal impacts. For example, addressing the long-tail proves an important direction for studying bias and fairness in recognition [12]. However, any system that makes it easier to train a fair classifier in long-tailed classes also makes it possible for a malicious agent to train a system that automatically discriminates against a certain subgroup for which only little training data is available. This is potentially a negative societal impact.

Limitations. We notice two limitations related to the current CLTR setup and methodology. First, the current setup does not incorporate unlabeled data, which naturally follows the same long-tailed distribution as labeled data. Unlabeled data allows for self-supervised pretraining and semi-supervised learning. Second, the models explored in this work are based on convolutional network architecture. Although our methods are orthogonal to architectural design, it is still an open question whether other architectures (e.g., transformers) might make a difference. We address the above in future work. Third, our current CLTR assumes uniform sampling for data labeling, in practice, how to mine valuable data is an open question

Future Work. Addressing the above limitations is in future work. Moreover, as CLTR aims to address the long-tailed issue continuously over time, data distribution can reasonably shift over time. Therefore, formulating such is a realistic research direction. Lastly, application requirements might also change over time, reflected as new classes added to the ontology. This is similar to CIL. Formulating the case of adding new classes [38] is another valuable direction in address the long-tailed issue in an open world.

7. Conclusion

We formulate a new problem, Continual Long-Tailed Recognition (CLTR), which is a practical but underexplored problem when solving the long-tailed data distribution issue w.r.t class labels. We design the CLTR benchmarking protocol and extensively explore approaches to it. We find that two-stage methods work best that train feature backbones and classifiers in two separate stages. Furthermore, in feature learning using Supervised Contrastive (SupCon) loss, we find that applying a Mean Shift module significantly expedites learning and adaptation in subsequent TPs while maintaining the state-of-the-art performance.

References

[1] Mohamed Abdelsalam, Mojtaba Faramarzi, Shagun Sodhani, and Sarath Chandar. Iirc: Incremental implicitly-refined classification. In *CVPR*, 2021. 3

[2] Shaden Alshammari, Yu-Xiong Wang, Deva Ramanan, and Shu Kong. Long-tailed recognition via weight balancing. In *CVPR*, 2022. 1, 2, 4

[3] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *ICCV*, 2019. 3, 4

[4] Luca Bertinetto, Romain Mueller, Konstantinos Tertikas, Sina Samangooei, and Nicholas A. Lord. Making better mistakes: Leveraging class hierarchies with deep networks. In *CVPR*, 2020. 3

[5] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 1, 2, 3

[6] Jiarui Cai, Yizhou Wang, and Jenq-Neng Hwang. Ace: Ally complementary experts for solving long-tailed recognition in one-shot. In *ICCV*, 2021. 2

[7] Zhipeng Cai, Ozan Sener, and Vladlen Koltun. Online continual learning with natural distribution shifts: An empirical study with visual data. In *ICCV*, pages 8281–8290, 2021. 3

[8] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. In *NeurIPS*, 2019. 1, 2

[9] Miguel A Carreira-Perpinán. Generalised blurring mean-shift algorithms for nonparametric clustering. In *CVPR*, 2008. 2, 5

[10] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *CVPR*, 2019. 1, 2, 3

[11] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. 2

[12] Irene Chen, Fredrik D Johansson, and David Sontag. Why is my classifier discriminatory? *NeurIPS*, 2018. 8

[13] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799, 1995. 2

[14] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. 2

[15] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 1, 3, 4

[16] Jiequan Cui, Zhisheng Zhong, Shu Liu, Bei Yu, and Jiaya Jia. Parametric contrastive learning. In *ICCV*, 2021. 2

[17] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019. 2, 4, 6

[18] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *PAMI*, 2021. 3

[19] Chris Drummond, Robert C Holte, et al. C4. 5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on learning from imbalanced datasets II*, volume 11, pages 1–8. Citeseer, 2003. 2

[20] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 1, 4

[21] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on information theory*, 21(1):32–40, 1975. 2

[22] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International Conference on Intelligent Computing*, pages 878–887. Springer, 2005. 2

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6

[24] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Deep imbalanced learning for face recognition and attribute prediction. *PAMI*, 42(11):2781–2794, 2019. 2

[25] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. In *ICLR*, 2020. 2, 4

[26] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 4

[27] Salman Khan, Munawar Hayat, Syed Waqas Zamir, Jianbing Shen, and Ling Shao. Striking the right balance with uncertainty. In *CVPR*, 2019. 2

[28] Salman H Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A Sohel, and Roberto Togneri. Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems*, 29(8):3573–3587, 2017. 2

[29] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *NeurIPS*, 2020. 2, 4

[30] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 3

[31] Shu Kong and Charless C Fowlkes. Recurrent pixel embedding for instance grouping. In *CVPR*, 2018. 2, 5

[32] Shu Kong and Deva Ramanan. Opengan: Open-set recognition via open data generation. In *ICCV*, 2021. 3, 4

[33] Soroush Abbasi Koohpayegani, Ajinkya Tejankar, and Hamed Pirsiavash. Mean shift for self-supervised learning. In *ICCV*, 2021. 5

[34] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[35] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2017. 3

[36] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 2

[37] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 4

[38] Zhiqiu Lin, Deepak Pathak, Yu-Xiong Wang, Deva Ramanan, and Shu Kong. Learning with an evolving class ontology. In *NeurIPS*, 2022. 8

[39] Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The clear benchmark: Continual learning on real-world imagery. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 3

[40] Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X Yu. Large-scale long-tailed recognition in an open world. In *CVPR*, 2019. 1

[41] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *ECCV*, 2018. 2

[42] Mapillary-Vistas-2.0. https://blog.mapillary.com/update/2021/01/18/vistas-2-dataset.html. 2021. 3

[43] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. 3, 4

[44] William J Reed. The pareto, zipf and other power laws. *Economics letters*, 74(1):15–19, 2001. 1

[45] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *ICML*, 2018. 2

[46] Juan Carlos San Miguel, Jesús Bescós, José M Martínez, and Álvaro García. Diva: a distributed video analysis framework applied to video-surveillance systems. In *International Workshop on Image Analysis for Multimedia Interactive Services*, 2008. 2

[47] Li Shen, Zhouchen Lin, and Qingming Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016. 2

[48] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. In *NeurIPS*, 2019. 2

[49] Jong-Chyi Su and Subhransu Maji. The semi-supervised inaturalist challenge at the fgvc8 workshop, 2021. 6

[50] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 3

[51] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018. 3, 6

[52] Xudong Wang, Long Lian, Zhongqi Miao, Ziwei Liu, and Stella X Yu. Long-tailed recognition by routing diverse distribution-aware experts. In *ICLR*, 2020. 2

[53] Yizhou Wang, Shixiang Tang, Feng Zhu, Lei Bai, Rui Zhao, Donglian Qi, and Wanli Ouyang. Revisiting the transferability of supervised pretraining: an mlp perspective. In *CVPR*, pages 9183–9193, 2022. 4, 6, 7

[54] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. 1, 2, 3

[55] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 1, 4

[56] Lu Yang, He Jiang, Qing Song, and Jun Guo. A survey on long-tailed visual recognition. *International Journal of Computer Vision*, pages 1–36, 2022. 1

[57] Yuzhe Yang and Zhi Xu. Rethinking the value of labels for improving class-imbalanced learning. In *NeurIPS*, 2020. 2

[58] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *ICML*, 2017. 3

[59] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1131–1140, 2020. 3

[60] Songyang Zhang, Zeming Li, Shipeng Yan, Xuming He, and Jian Sun. Distribution alignment: A unified framework for long-tail visual recognition. In *CVPR*, 2021. 2

[61] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *arXiv preprint arXiv:2110.04596*, 2021. 1, 2

[62] Boyan Zhou, Quan Cui, Xiu-Shen Wei, and Zhao-Min Chen. Bbn: Bilateral-branch network with cumulative learning for long-tailed visual recognition. In *CVPR*, 2020. 2

[63] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. 1