Learning to Segment the Tail

Xinting Hu[1], Yi Jiang[2], Kaihua Tang[1], Jingyuan Chen[3], Chunyan Miao[1], Hanwang Zhang[1]
[1]Nanyang Technological University, [2]Alibaba Group, [3]Damo Academy, Alibaba Group
xinting001@e.ntu.edu.sg, jiangyi0425@gmail.com, kaihua001@e.ntu.edu.sg
jingyuanchen91@gmail.com , ascymiao@ntu.edu.sg, hanwangzhang@ntu.edu.sg

Abstract

Real-world visual recognition requires handling the extreme sample imbalance in large-scale long-tailed data. We propose a "divide&conquer" strategy for the challenging LVIS task: divide the whole data into balanced parts and then apply incremental learning to conquer each one. This derives a novel learning paradigm: ***class-incremental few-shot learning****, which is especially effective for the challenge evolving over time: 1) the class imbalance among the old-class knowledge review and 2) the few-shot data in new-class learning. We call our approach* ***Learning to Segment the Tail*** *(LST). In particular, we design an instance-level balanced replay scheme, which is a memory-efficient approximation to balance the instance-level samples from the old-class images. We also propose to use a meta-module for new-class learning, where the module parameters are shared across incremental phases, gaining the learning-to-learn knowledge incrementally, from the data-rich head to the data-poor tail. We empirically show that: at the expense of a little sacrifice of head-class forgetting, we can gain a significant 8.3% AP improvement for the tail classes with less than 10 instances, achieving an overall 2.0% AP boost for the whole 1,230 classes*[1].

1. Introduction

The long-tail distribution inherently exists in our visual world, where a few head classes occupy most of the instances [48, 1, 37, 44]. This is inevitable when we are interested in modeling large-scale datasets, because the class observational probability in nature follows Zipf's law [31]. Therefore, it is prohibitively expensive to counter the nature and collect a balanced sample-rich large-scale dataset, catering for training a robust visual recognition system using the prevailing models [13, 9, 34, 4].

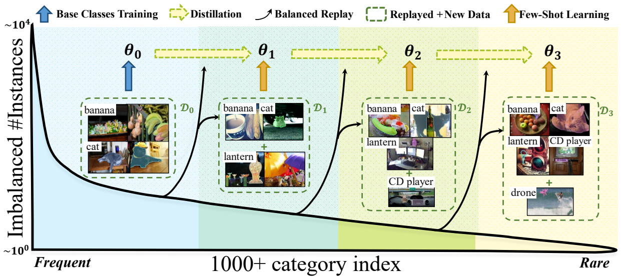


Figure 1. **Overview of the proposed *Learning to Segment the Tail* (LST) method for LVIS [10].** To tackle the severe imbalance, we divide the overall dataset into balanced sub-parts $\mathcal{D}_i$ and train the instance segmentation model θ_i phase-by-phase incrementally. We use knowledge distillation and the proposed *balanced replay* to confront the catastrophic forgetting [28] in the fewer- and fewer-shot learning over time. Here, θ_3 is the resultant model.

In this paper, we study a practical large-scale visual recognition task on the challenging real-world dataset: Large Vocabulary Instance Segmentation (LVIS) [10]. As shown in Figure 1, across the 1k+ instance object classes, the number of training instances per class drops from thousands in the head to only a few in the tail (*i.e.*, 26k+ "banana" *vs*. only 1 "drone"). Empirical studies show that the models trained using such a long-tailed dataset tend to please the common classes but neglect the rare ones [10]. The reasons are two-fold: 1) class imbalance causes the head classes trained thousand times more than the tail classes, and 2) the few-shot samples in the long tail render the generalization a great challenge (*i.e.*, around 300 classes with less than 10 samples). Therefore, the key solution for LVIS is to well address not only the **imbalance** but also the **few-shot learning** at a large scale.

Unfortunately, conventional works on either "imbalance" or "few-shot" are fundamentally not scalable to LVIS. On the one hand, it is well-known that works on data re-sampling [12, 3, 11] — up-sampling the rare tail classes or down-sampling the frequent head classes — can prevent the training from being dominated by the head. Nonetheless, as they do not introduce any new diversity, they struggle in the trade-off between the tail over-fit — the heavy

[1] Code is available at https://github.com/JoyHuYY1412/LST_LVIS

repetitions of the few-shot samples, and the head under-fit — the significant abandon of the many-shot samples. On the other hand, conventional few-shot learning that transfers the model from a data-rich "base set" to a data-poor "novel set" [39, 43], however, is not yet practical in LVIS, as any base or novel split will be eventually imbalanced due to the scale, undermining the generalization ability that is already challenging in few-shot learning [42]. Besides, the scale also raises major memory issues in the episodic training [41] adopted by recent meta-learning based methods [20, 29].

An intuitive strategy to address the scale is to *divide* the large "body" into "parts", *conquer* each of them, and then *merge* them incrementally. As illustrated in Figure 1, each subset is more balanced and easier to handle. Essentially, the "divide&conquer" strategy for LVIS poses a novel learning paradigm: **class-incremental few-shot learning**. However, the merge to stitch the parts back to a whole is no longer a trivial adoption of any off-the-shelf class-incremental learning method [33, 36]. The reason is that different from traditional class-incremental learning scenarios, our incremental phases over time, will face 1) more imbalanced data of the old classes and 2) fewer data of the new classes. This leaves the network more vulnerable to "catastrophic forgetting" [28] in learning the new classes, not to mention the fact that they are fewer- and fewer-shot.

To implement the novel paradigm for the LVIS task, we propose the *balanced replay* scheme for knowledge review and the meta-learning based *weight-generator* module for fast few-shot adaptation. We call our approach: Learning to Segment the Tail (LST). In a nutshell, LST can be summarized in Algorithm 1. After training the first phase that comes with the abundant labeled data as the bootstrap, we start the incremental learning in T phases (*e.g.*, T equals 3 in Figure 1). Given the relatively balanced subset $\mathcal{D}_t$ in t-th phase using data replay (`BalancedReplay` in Section 3.3), new classes can be learned and old classes can be fine-tuned simultaneously (`UpdateModel` in Section 3.2). To transfer the knowledge step by step from the "easy" many-shot head to the "difficult" few-shot tail, we furthur adopt a *meta weight generator* [8] (`MWG` in Section 3.4).

We validate the proposed LST on the large-scale long-tailed benchmark LVIS, which contains 1,230 entry-level instance categories. Experimental results show that our LST improves the instance segmentation results over the baseline by 7.0∼8.0% AP on the tail classes while gaining a 2.2% overall improvement for the whole classes. The results illuminate us a promising direction for tackling the severe class imbalance in long-tailed data: class-incremental few-shot learning.

Our contributions can be summarized as follows:

- We are among the first to study the task of large vocabulary instance segmentation, which is of high practical value by focusing on the severe class imbalance and few-shot learning in the field of instance segmentation.
- We develop a novel learning paradigm for LVIS: class-incremental few-shot learning.
- The proposed Learning to Segment the Tail (LST) for the above paradigm outperforms baseline methods, especially over the tail classes, where the model can adapt to unseen classes instantly without training.

Algorithm 1 Learning to Segment the Tail (T+*1* phases)

```
Input: $\{\mathcal{G}_i\}_{i=0,1,...T}$                ▷ Dataset pre-processing
Output: $\theta_T$                ▷ The final phase model parameters
$\theta_0 \leftarrow \arg\min_{\theta_0} L_{inst}(\mathcal{G}_0;\ \theta_0)$        ▷ Base classes training
for $t = 1 \rightarrow T$ do
    $\mathcal{D}_t \leftarrow$ BALANCEDREPLAY($\{\mathcal{G}_i\}_{i=0,1,...t}$);
    $\theta_t \leftarrow$ UPDATEMODEL($\mathcal{D}_t,\ \theta_{t-1}$)
end for

function UPDATEMODEL($\mathcal{D}_t,\ \theta_{t-1}$)
    $\theta_t \leftarrow \theta_{t-1}$                ▷ Model initialization
    repeat
        when *use meta-module* then
            $\mathcal{G}_t^{sup} \leftarrow \mathcal{G}_t$            ▷ Sample support set
            $\theta_t \leftarrow$ MWG($\mathcal{G}_t^{sup}, \theta_t$)
        end when            ▷ Few-shot weight generation
        $\theta_t \leftarrow \arg\min_{\theta_t}[L_{inst}(\mathcal{D}_t;\theta_t)+L_{kd}(\mathcal{G}_t,\theta_{t-1};\theta_t)]$
    until converge        ▷ Old & new classes fine-tuning
end function
```

2. Related Work

Instance segmentation. Our instance segmentation backbone is based on the popular region-based frameworks [22, 13, 5, 25], in particular, Mask R-CNN [13] and its semi-supervised extension Mask^X R-CNN [17], which can transfer mask predictor from merely box annotation. However, they cannot scale up for the large-scale long-tailed dataset such as LVIS [10], which is the focus of our work.

Imbalanced classification. Re-sampling and re-weighting are the two major efforts to tackle the class imbalance. The former aims to re-balance the training samples across classes [16, 3, 11, 6]; while the latter focuses on assigning different weights to adjust the loss function [18, 40, 47, 7]. Some works on generalized few-shot learning [46, 21] also deal with an extremely imbalanced dataset, extending the test label space of few-shot learning to both base and novel rare classes. We propose a novel re-sampling strategy. Different from previous works that perform on image-level re-sampling, we address the imbalance of dataset on instance-level.

Learning without forgetting & learning to learn. Existing works mainly focus on how to learn new knowledge with less forgetting, and how to generalize from the learning process, *i.e.*, learning to learn. To cope with the ever-evolving data, class-incremental learning methods [36, 16, 38, 2] adapt the original model trained on old classes to new classes, where knowledge distillation [15, 23] and old data replay [33, 26] are applied to minimize the forgetting. For few-shot learning, meta-learning based works transfer the learning-to-learn knowledge through feature representation [29, 32, 20], classifier weights [46, 8], and the regression of model parameters [42, 43] from the data-rich base classes, to obtain a good model initialization for the data-poor new classes. We propose a class-incremental few-shot learning paradigm that can be seen as a non-trivial combination of these two fields.

3. Learning to Segment the Tail

LVIS is a **L**arge **V**ocabulary **I**nstance **S**egmentation dataset, which contains 1,230 instance classes [10]. The number of images per class in LVIS has a natural long-tail distribution, with 700+ classes containing less than 100 training samples. To tackle the challenging dataset in the proposed LST using the "divide&conquer" strategy, we first present the division method in Section 3.1, and discuss our class-incremental learning pipeline in Section 3.2. In Section 3.3 and Section 3.4, we detail how to use `BalancedReplay` and `MWG` for knowledge review and few-shot adaptation.

3.1. Dataset Pre-processing

Our guideline for the division is to alleviate the intra-phase imbalance of the dataset, where each of division is relatively balanced. We first sort the classes by the number of instance-level samples in a descending order, obtaining a sorted class set $\mathcal{C}$. Then we divide the sorted categories into mutually exclusive groups $\{\mathcal{C}_i\}$. Correspondingly, we have a sub-dataset $\mathcal{G}_i$ with images and annotations for each $\mathcal{C}_i$.

Specifically, after grouping the sorted top b classes as the bootstrap group $\mathcal{C}_0$, and splitting the remaining classes into T evenly spaced bins $\{\mathcal{C}_i\}_{i=1,2,...T}$, we obtain the sorted class sets with groups $\mathcal{C} = \mathcal{C}_0 \cup \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_T$. By assigning data to the corresponding groups, we convert the whole dataset into $\{\mathcal{G}_i\}_{i=0,1,...T}$, as shown in line 1 of Algorithm 1, where each $\mathcal{G}_i$ is composed of all the annotated images containing any instance of $\mathcal{C}_i$. Following this setting, the data is fed to the network step-wisely, so that our model is trained in a class-incremental learning style.

3.2. Class-Incremental Instance Segmentation

Class-incremental learning aims to learn a unified model that can recognize classes of both previous and current phase [33]. In our scenario, we aim to train our network on $\{\mathcal{G}_i\}_{i=0,1,...T}$, obtaining models from θ_0 to θ_T, and finally deliver θ_T as our resultant model that can detect all instance classes in LVIS. Here, we adopt the popular definition inherited from works in incremental learning and few-shot learning [8, 36]: classes in $\mathcal{C}_0$ are termed as **base** classes; for phase $t = 1, 2, \cdots, T$, classes in $\{\mathcal{C}_i\}_{i=0,1,...t-1}$ are called **old** classes and classes in current $\mathcal{C}_t$ are called **new** classes. For training and evaluation in each phase t , we will not handle anything in the future classes $\{\mathcal{C}_i\}_{i=t+1,...,T}$. As phases t goes by, the data in $\mathcal{G}_t$ for new classes becomes fewer and fewer, and the data for old classes become more and more imbalanced. To tackle the inter-phase imbalance, we propose a novel sampling scheme for the old data, which will be discussed in Section 3.3.

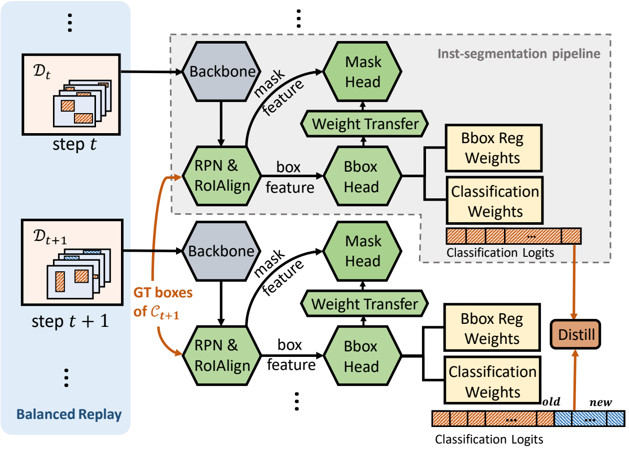


Figure 2. Overview of our framework for learning the instance segmentation model incrementally. It is based on the two-stage instance segmentation architecture, training the overall imbalanced dataset in incremental phases with sampled data for both old and new classes. In incremental phases, the weights of backbone are frozen, and the distillation is calculated using ground truth box annotations between the classification logits of the current and previous networks to avoid forgetting.

Our overall architecture is shown in Figure 2. We build our class-incremental learning framework based on Mask^X R-CNN [17], which is a modified version of Mask R-CNN [13]. Mask^X R-CNN is an instance segmentation model that can be used in partially supervised domain by obtaining a category's mask parameters from its bounding box parameters. We adopted this weight transfer module so that the class-agnostic transfer function weights can be shared between incremental phases, which can 1) alleviate the training burden for massive mask layers of 1,230 classes and 2) avoid the unstable knowledge distillation of the 28 × 28 mask logits across classes (*i.e.*, 28 × 28 times more compared to the class logits distillation in Eq. (2)). Besides, we replaced the last classifier layer in the detection branch with scaled cosine similarity operator, because it has been shown effective in eliminating the bias caused by the significant variance in magnitudes [33, 8, 16]. Formally, given the feature vector $\boldsymbol{x}$, the output logits vector $\boldsymbol{y}$ of cosine similarity classifier with weights $\boldsymbol{w}$ is:

$$\boldsymbol{y} = \overline{\boldsymbol{w}}^T \overline{\boldsymbol{x}} \tag{1}$$

where $\overline{\boldsymbol{w}} = \boldsymbol{w} / \|\boldsymbol{w}\|$ and $\overline{\boldsymbol{x}} = \boldsymbol{x} / \|\boldsymbol{x}\|$ are the L2-Normalized vectors. Then the class-specific mask weights in the mask branch are generated from $\boldsymbol{w}$ using the class-agnostic weight prediction function in Mask^X R-CNN [17].

The overall class-incremental learning pipeline is shown in Algorithm 1, and it is composed of two stages:

Stage 1. Base classes training. This training phase ($t = 0$) delivers the model θ_0 for base classes, where the backbone and RoI heads are jointly trained. The trained classification weight vectors for top b classes are denoted as $\boldsymbol{W}^{\mathcal{B}} = [\boldsymbol{w}_1, \boldsymbol{w}_2, ...\boldsymbol{w}_b]$. We assume that if the data in base classes are sufficiently abundant and relatively balanced, the training of θ_0 can work effectively as the bootstrap for the whole system.

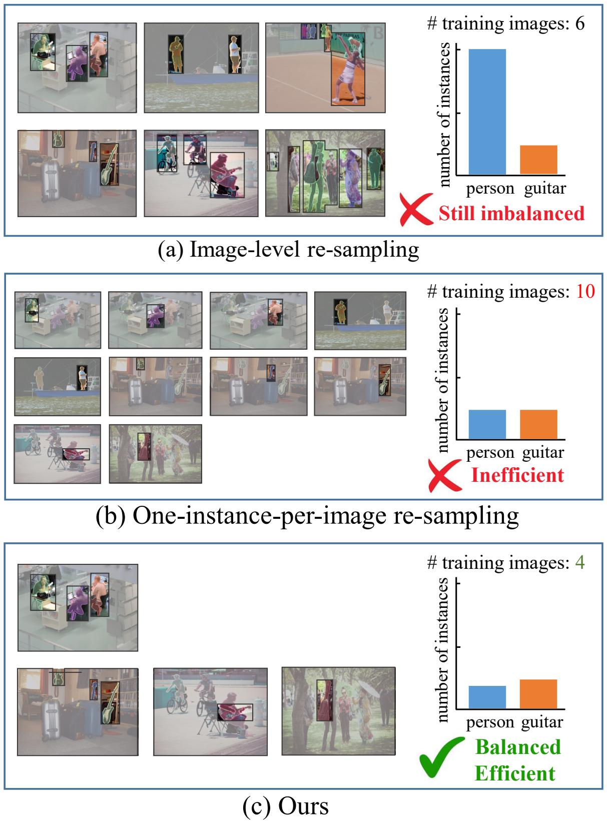
We calculate the instance segmentation loss $L_{inst} = L_{RPN} + L_{cls} + L_{box} + L_{mask}$. The RPN loss L_{RPN}, classification loss L_{cls}, bounding-box loss L_{box} and mask loss L_{mask} are identical as those defined in Fast R-CNN [9] and Mask R-CNN [13].

Stage 2. Class-incremental learning. In each phase t (from 1 to T), the number of classifiers is expanded, which leads to the following adjustments to the training procedure in Stage 1:

Network Expansion. After initialized from the last phase's model θ_{t-1}, the current model needs to grow for recruiting new class-specific layers, *i.e.*, the bounding-box, classification and regression layers and the mask prediction layer for new classes. Recall our modifications of the backbone, the weights of mask layers can be transferred from the weights of box layers, so the expansion of the network is only implemented on the box head.

Freezing* and *knowledge distillation. As discussed in class-incremental learning works [33, 16], these two strategies are broadly used to address catastrophic forgetting, the significant performance drop on previous data when adapting a model to new data. Data rehearsal [33] is another strategy to prevent forgetting by reviewing old data, which is discussed in Section 3.3. In our scenario, 1) by *freezing* the weights in the backbone, a strong constraint on the previous representation is imposed, 2) by *knowledge distillation*, the discriminative representation learned previously is not shifted severely during the new learning step. Our distillation loss is defined as:

$$L_{kd} = \|\boldsymbol{y}_{t-1} - \boldsymbol{y}'_t\| \tag{2}$$



(a) Image-level re-sampling

(b) One-instance-per-image re-sampling

(c) Ours

Figure 3. A running example of different re-sampling strategies. Given images of "person" and "guitar" from different phases, we show the observable instances for each image in training ROI heads using different re-sampling strategies. As shown in (c), compared to (a) and (b), by omitting the annotations of "person" in images except for the ones we sampled, our instance-level balanced replay can construct a relatively balanced dataset with much less computation overhead.

where $\boldsymbol{y}_{t-1}$ and $\boldsymbol{y}'_t$ are the output logits vectors for classes in $\{\mathcal{C}_i\}_{i=0,1,...t-1}$ using both old model θ_{t-1} trained in phase $t-1$ and current θ_t, respectively. Note that the output $\boldsymbol{y}_t$ in phase t also incorporates new categories in $\mathcal{C}_t$, we use $\boldsymbol{y}'_t$ to indicate the sliced logits only corresponding to previous classes $\{\mathcal{C}_i\}_{i=0,1,...t-1}$. $\|\cdot\|$ is the L2-distance to measure the difference between logits. We choose L2-distance here in avoid of the grid search of *temperature* as in conventional distillation loss [23], thanks to the already normalized logits (*i.e.*, logits lie in the same range $[-1, 1]$) using cosine.

The purpose of Eq. (2) is to let the new model mimic the the old model's behavior (*i.e.*, generate similar output logits), so that the knowledge learned from old network can be preserved. It is worth noting that distillation requires the same input sample going through old and new networks separately. Different from the classification task, in instance segmentation, proposals are dynamically predicted. To this end, we use the ground truth bounding boxes of novel classes as samples in each step for distillation. Over-

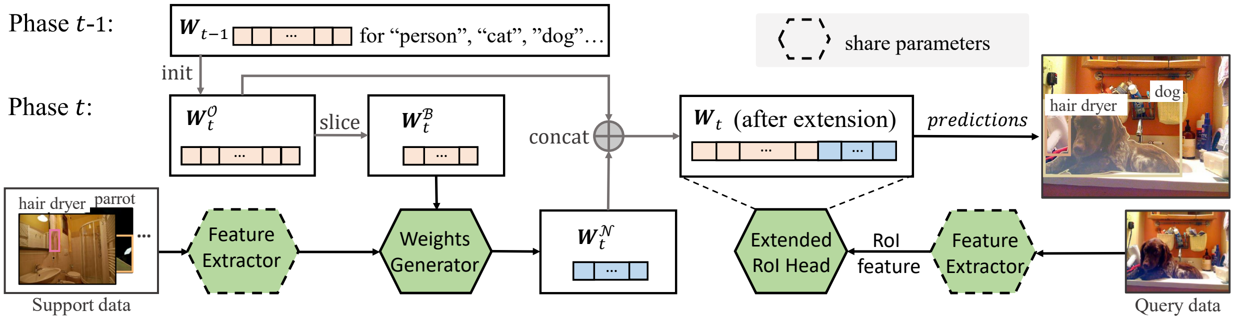


Figure 4. Architecture of our framework combined with weight generator. At the start of each step, classifier weights for old classes are copied from the previous network. Based on the features of new classes samples and base classifier weights, weights for new classes are predicted by our weight generator and concatenated. After obtaining the whole classifier, our weight generator is jointly trained with the network using input image on both old and new categories.

all, for each incremental phase t, knowledge distillation loss is added to the final loss as $L = L_{inst} + L_{kd}$.

3.3. Instance-level Data Balanced Replay

As shown in Figure 1, within each incremental phase, the variance of instance number is narrowed. However, the inter-phase imbalance (*i.e.*, the gap in the number of samples among phases) exists, leading to a dilemma: if we replay all the previous data, it will definitely break the balance, introducing the imbalance back to our network; if we discard replay, catastrophic forgetting will happen [33].

Moreover, previous re-sampling strategies [10, 35] can not be applied gracefully in the instance-level vision tasks. For image-level re-sampling that regularizes the number of images per category, the inherent class co-occurrence may hinder its effectiveness. For example, in Figure 3 (a), as "guitar" usually co-occur with "person"[2], the adjustment on the number of "guitar" instances will always unnecessarily adjust the number of "person" instances at the same time. An alternative one-instance-for-one-image strategy in Figure 3 (b) can assure the absolute balance, however, the additional computational cost for feed-forwarding the same image multiple times is tremendous. Based on those observations, we proposed the *Instance-level Data Balanced Replay* strategy. For phase t, it works as follows:

1) calculate $\bar{n}_{\mathcal{C}}$: the average number of instances **over all categories** in set $\mathcal{C}_t$;
2) calculate $\{\bar{n}_k\}$: the average number of instances **over all images** containing annotations from the corresponding old category $k \in \{\mathcal{C}_i\}_{i=0,1,\dots t-1}$;
3) construct the replay set $\mathcal{R}_t$: for each category k, randomly sample $\lceil \bar{n}_{\mathcal{C}} / \bar{n}_k \rceil$ images from images in $\{\mathcal{G}_i\}_{i=0,\dots t-1}$ containing category k, where only those annotations belonging to category k are considered valid in the training.

As illustrated in Figure 3 (c), by replaying the balanced set $\mathcal{R}_t$ of old data using the above strategy, we dynamically collects a relatively balanced dataset $\mathcal{D}_t = \mathcal{R}_t \cup \mathcal{G}_t$ in each phase t.

3.4. Meta Weight Generator

So far, the proposed class-incremental pipeline is able to tackle the intra-&inter-phase imbalance while preserves the performance of classes from the previous phases. However, the challenge of few-shot learning becomes severe as we approach to the tail classes. Therefore, we adopt a Meta Weight Generator (MWG) module [46] as shown in Figure 4, which utilizes the base knowledge learned and inherited from the previous phases to dynamically generate the weight matrix of the current phase. The motivation is: given robust feature backbone and classifiers learned for the base classes (*i.e.*, Stage 1 in Section 3.2), it is possible learning to directly "write" new classifiers for the new classes, based on the new sample feature itself and its similarities to the base classifiers [8]. For an intuitive example, we can customize a "drone" classifier by using a "drone" sample feature and how the sample looks like the base classes, *e.g.*, 50% "airplane", 30% "fan", and 20% "frisbee".

Formally, in the t-th incremental phase, we decompose the classifier weight matrix $\boldsymbol{W}_t$ into two parts: $\boldsymbol{W}_t^{\mathcal{O}}$, $\boldsymbol{W}_t^{\mathcal{N}}$ for the old and the new classes, respectively. Following the Gidaris&Komodakis's work [8], $\boldsymbol{W}_t^{\mathcal{N}}$ is dynamically generated. In particular, we retrieve the base classifier weights $\boldsymbol{W}_t^{\mathcal{B}}$ from $\boldsymbol{W}_t^{\mathcal{O}}$, then learn how to compose $\boldsymbol{W}_t^{\mathcal{N}}$. Take an image containing RoIs of a new category c as an example, for each RoI feature vector $\boldsymbol{x}$, 1) the feature vector $\boldsymbol{x}$ is fed to an attention kernel function to get the coefficients $\boldsymbol{m}$ as: $\boldsymbol{m} = Att(\boldsymbol{K}, \boldsymbol{V}\boldsymbol{x})$, where $\boldsymbol{m}$ are the weight coefficients used to attend b base classifiers weights $\boldsymbol{W}_t^{\mathcal{B}}$, $\boldsymbol{V}$ is a learnable matrix that transforms $\boldsymbol{x}$ to the query vector, and $\boldsymbol{K}$ is a set of learnable keys (one per base category); 2) the classification weight $\boldsymbol{w}_c$ is first generated for each RoI feature $\boldsymbol{x}$ independently and then averaged over all RoIs of category

[2] We use "person" to replace "baby" used to represent a set of synonymous labels: "child", "boy", "girl", "man", "woman" and "human" in LVIS for readability.

c in this image as the final predicted weight vector of category c. For each RoI feature $\boldsymbol{x}$, the corresponding classifier weight is calculated as:

$$\boldsymbol{w} = \mathbf{a} \odot \mathbf{x} + \mathbf{b} \odot (\mathbf{W}_t^{\mathcal{B}} \mathbf{m}), \tag{3}$$

where $\odot$ denotes element-wise multiplication, $\mathbf{a}$ and $\mathbf{b}$ are learnable weight vectors.

For the initialization of the t-th phase, $\mathbf{W}_t^{\mathcal{O}}$ is copied from the previous phase $t-1$. For the episodic training [41], each episode is composed of a support set and a query set sampled from $\mathcal{D}_t$. The support set is for applying `MWG` to generate $\mathbf{W}_t^{\mathcal{N}}$ (Eq. (3)), and the query set is for collecting loss from the predictions using the full model θ_t: the concatenated classifiers $[\mathbf{W}_t^{\mathcal{O}}, \mathbf{W}_t^{\mathcal{N}}]$ as well as other network parameters, and then update θ_t. This joint training assures that the classifier weights and the meta-learner are synchronized in the t-th phase. After the episodic training, we set the weights for a novel category c by averaging the predicted weights of all the instances of class c in $\mathcal{D}_t$. Then, the meta-module can be completely detached, and we are ready to deliver the model θ_t.

4. Experiments

We conducted experiments on LVIS [10] using the standard metrics for instance segmentation. AP was calculated across IoU threshold from 0.5 to 0.95 over all categories. AP50 (or AP75) means using an IoU threshold 0.5 (or 0.75) to identify whether a prediction is positive. To better display the results from the head to the tail, $\text{AP}_{(0,1]}$, $\text{AP}_{(0,5)}$, $\text{AP}_{(0,10)}$, $\text{AP}_{[10,100)}$, $\text{AP}_{[100,1000)}$, $\text{AP}_{[1000,-)}$ were evaluated for the sets of categories which containing only 1, <5, <10, $10 \sim 100$, $100 \sim 1{,}000$ and $\geq 1{,}000$ training object instances. AP for object detection was reported as AP^{bb}.

4.1. Implementation Details

We implemented our architectures and other baselines (*e.g.*, Mask^X R-CNN [17]) on the Mask R-CNN [13] code base `maskrcnn_benchmark`[3]. For Section 3.2, we implemented as follows: 1) mask weights were generated by a class-agnostic MLP mask branch together with the weights transferred from the classifiers of the box head following Hu *et al*. [17]; 2) cosine normalization was applied to both of the feature vectors and the classifier weights, to obtain the classification logits. Note that the ReLU non-linearity in the final layer was removed to allow the feature vectors to take both positive and negative values.

We initialized the scaling factor of cosine similarity as 10. All the models were initialized using the released model pre-trained on COCO [24], and trained by using SGD with 1e-4 weight decay and 0.9 momentum. Each minibatch had 8 training images, and the images were resized to that its shorter edge is 800-pixel. No other augmentation was used except for horizontal flipping. Models were evaluated using the 5k `val` images. Following Gupta *et al*. [10], we increased the number of detections per image up to top 300 (*vs*. top 100 for COCO) and reduced the minimum score threshold from the default of 0.05 to 0.0.

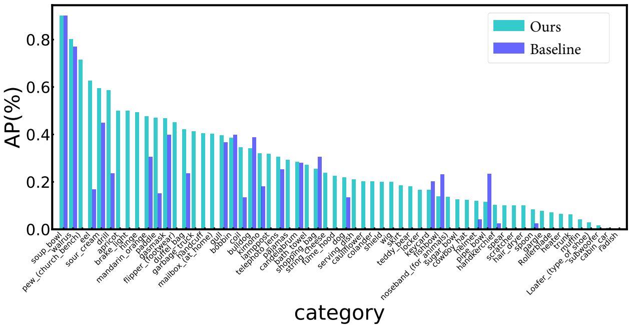


Figure 5. Performance comparison on a subset of tail classes between our LST and the joint training baseline (Mask^X R-CNN). We observe that the baseline APs for many few-shot categories is zero due to the extreme imbalance.

For Section 3, in Stage 1, we chose $b = 270$, where each of the top b classes has 400+ instances. 512 RoIs were selected per image, and the positive-negative ratio is $1:3$. For training the top b classes, the learning rate was set to 0.01 and decayed to 0.001 and 0.0001 after 6 epochs and 8 epochs (10 epochs in total). In Stage 2, we split the rest classes into 6 groups. For each incremental phase, we only sampled 100 proposals per image as the number of valid annotations per image shrinks when adopting our balanced replay strategy. Recall the freezing operation in Section 3.2, we froze the top 3 layers of ResNet [14] in the backbone in each incremental learning phase. The learning rate was from 0.002 and divided by 10 after 6 epochs (10 epochs in total). More experiments on the choice of b and the number of phases are presented in Section 4.3.

4.2. Results and Analyses on LVIS

Results. As shown in Table 1, our method evaluated at the last phase, *i.e*., the whole dataset, outperforms the baselines in the tail classes ($\text{AP}_{(0,10)}$ and $\text{AP}_{[10,100)}$) by a large margin. The overall AP for both object detection and instance segmentation improves. Especially, as shown in Figure 5, we randomly sampled 60 classes from the tail classes, whose number of instances in the training set is smaller than 100, and reported the result with and without using our LST which is class-incremental. We observe that our approach obtains remarkable improvement in most tail categories. We also compared our method with other re-sampling methods proposed to tackle the imbalanced data, where *repeat-factor sampling* [10] essentially up-samples the images containing annotations from tail classes, and *class-aware sampling* [35] is an alternate oversampling method. The results

[3] https://github.com/facebookresearch/maskrcnn-benchmark

Model	$AP_{(0,1]}$	$AP_{(0,5)}$	$AP_{(0,10)}$	$AP_{[10,100)}$	$AP_{[100,1000)}$	$AP_{[1000,-)}$	AP	AP50	AP75	AP^{bb}
Baseline [13]	0.0	0.0	0.0	12.8	20.9	**28.3**	17.9	28.9	18.8	17.9
Modified backbone	0.0	0.0	0.0	13.9	19.9	27.6	17.8	28.2	18.8	17.7
Class-aware Sampling [35]	0.0	0.0	0.0	20.0	20.2	24.5	19.5	31.6	20.5	19.3
Repeat-factor Sampling [10]	4.0	0.0	2.9	19.9	21.4	27.8	20.8	33.3	22.0	20.6
LST w/o MWG (Ours)	12.0	9.3	**11.7**	**27.1**	21.3	22.3	22.8	36.4	24.1	22.3
LST w MWG (Ours)	**13.6**	**10.7**	11.2	26.8	**21.7**	23.0	**23.0**	**36.7**	**24.8**	**22.6**

Table 1. Results of our LST and the comparison with other methods on LVIS val set. All experiments are performed based on ResNet-50-FPN Mask R-CNN.

Model	$AP_{(0,10)}$	$AP_{[10,100)}$	$AP_{[100,1000)}$	$AP_{[1000,-)}$	AP
Baseline	3.5	20.1	**25.1**	**31.5**	23.0
Ours	**14.4**	**30.0**	25.0	26.9	**26.3**

Table 2. Results of our LST and baseline implemented on ResNeXt-101-32x8d-FPN Mask R-CNN.

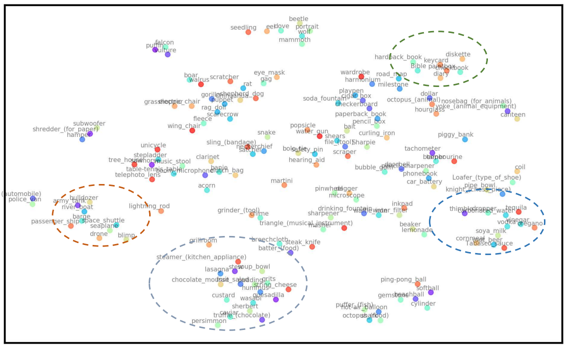


Figure 6. t-SNE [27] embeddings of the coefficients for few-shot categories in the last phase. As noted, semantically and visually similar classes are close (*i.e.*, “diary” and “diskette”, “custard” and “wasabi”).

show that our method surpasses all the other image-level re-sampling approaches on the tail classes, bringing an improvement in overall AP as well. In Figure 6, we visualized the predicted coefficients vectors $\boldsymbol{m}$ of our weight generator for samples in the last phase. The coefficient vectors of visually or semantically similar classes tend to be close, which shows our weight generator's effectiveness in relating the learning process for data-rich and data-poor classes. Due to limited resources, all the above models were implemented on ResNet-50-FPN. We further report the result applying our method to ResNeXt-101-32x8d-FPN [45] in Table 2 (b = 270, 3 phases), which also shows significant improvement. With more powerful computing resource available, we would like to follow the settings of Tan *et al.*'s work [19] to further improve our performances. We believe that our findings are regardless of visual backbones and data augmentation tricks.

Analyses. Oksuz *et al.* [30] pointed out that the imbalance among different foreground categories, owing to the dataset itself, undermines the performance of popular recognition models. The results of our baseline models in Table 1 validate this opinion, showing the tendency that the recognition on rare categories performs much worse than the frequent ones (0.0% *vs.* 28.3%) in LVIS. By re-balancing the dataset, previous re-sampling works like Gupta *et al.* [10] or Shen *et al.* [35] somewhat improve the performance for the tail classes. However, we show that they are less effective than our LST. The reason is that they struggle in the trade-off between the tail over-fit and the head under-fit. Furthermore, recall Figure 3, our method is more suitable for instance-based tasks as we essentially tackle the overall imbalance over **instances**. What is more, for Gupta *et al.*'s work [10], the threshold used for guiding the re-sampling of the whole dataset is sensitive to the data distribution and thus needs to be carefully tuned. As a result, the method is not flexible when new observations are added to the current dataset, bringing about an expansion of the tail. In contrast, the experiments in Section 4.3 show that our method is robust to the distribution inside each incremental phase, revealing the potential of our work to be applied to open classes with rarer data.

b	$phase\_size$	$\#phases$	AP
110	160	7	22.4
270	160	6	22.8
270	320	3	22.9
270	480	2	22.9
270	960	1	21.8
590	160	4	21.2
590	320	2	21.4

Table 3. Ablation study for different size of base classes b and the number of incremental phases.

4.3. Ablation Study

Choose of b and the size of phase. The influence of different b and the number of phases is shown in Table 3. We empirically show that, on the one hand, the final performance is sensitive to the choice of b, as the training on the more imbalanced base dataset (*i.e.*, b = 590) undermines the reliability of θ_0 and further influences the following phases. On the other hand, the results are relatively robust to the size of each incremental phase, as the balanced replay can always provide a relatively balanced dataset when $phase\_size$ locates in a moderate range.

Knowledge distillation. We split the rest 960 classes into 6 phases, and examined the influence of using knowledge distillation in each phase by comparing the performances on

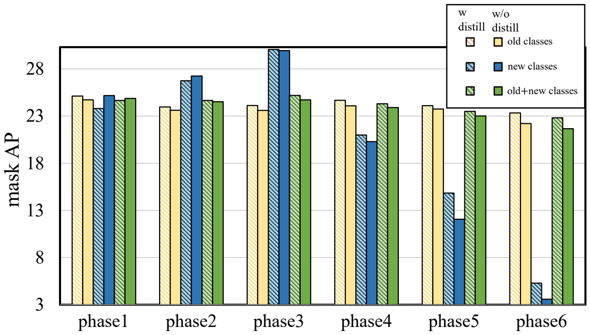
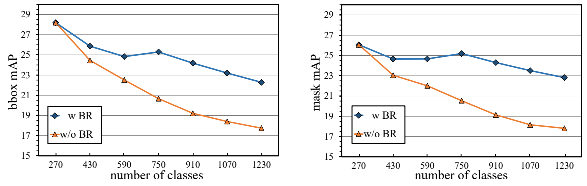


Figure 7. Comparison between networks with using knowledge distillation (shadow fill bars) and without using it (solid fill bars). Results are reported for old classes (yellow bars), new classes (blue bars) and old&new classes (green bars).



(a) LVIS-bboxAP (6 phases)

(b) LVIS-maskAP (6 phases)

Figure 8. Performance comparison for the models trained with and without our balanced replay. For every incremental phase, the detection and instance segmentation performance evaluated on new&old classes are reported.

new classes, old classes, new&old classes, respectively. As shown in Figure 7, models trained without distilling classification logits of two adjacent phases perform consistently worse than the model using the distillation on new&old classes. In the first few phases, the performance of new classes without distillation is higher, because it is trivial that when the new-class data is abundant, "forgetting" all the old classes are beneficial to focus on the performances of new classes. But, when the number of instances for each category become fewer and fewer, the distillation becomes more important for both new and old classes. The final instance segmentation AP for the whole dataset with and without knowledge distillation is 22.8% *vs*. 21.6%, demonstrating the effectiveness of the distillation.

Balanced replay. Figure 8 shows the effect of our Balanced Replay (BR) compared to the baseline that uses all the data from old&new classes in each phase. It is worth noting that although more data is used for training, the severe imbalance causes the gradually worse performance than our method's. Besides, our method needs far less storage memory consumption and training iterations to converge.

Meta weight generator. We examined the performance of our system with and without using meta weight generator. As shown in Table 1, both of them offer a very significant boost on few-shot recognition, while the meta-module based method does better on extreme few-shot classes (*i.e*., $AP_{(0,1]}$, $AP_{(0,5)}$). More specifically, we evaluated the models at each phase for all classes and report the performance of new classes (Figure 9). It is easy to see that among the two, the meta-module based solution exhibits better few-shot recognition behavior, especially for the <5-shot classes in the last phase (5.3% *vs.* 8.0%), without affecting the recognition performance of all classes. However, compared to the conventional training, the episodic training for meta-module is memory-inefficient. In our implementation, 160 is the maximum phase size for network armed with `MWG`, so we only report the results using 6 incremental phases. We would like to explore a better combination of meta-learning and fine-tuning in future work.

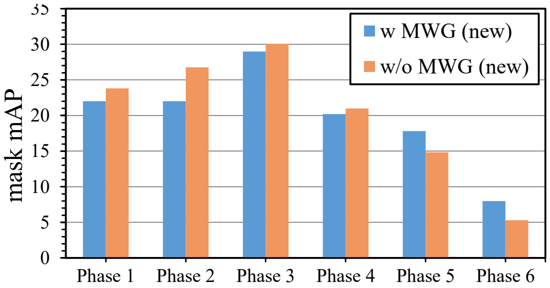


Figure 9. Performance comparison for the models trained with and without our meta weight generator. For every incremental phase, the instance segmentation performance was evaluated on the whole new&old classes, and we only report the results on the new classes to highlight the few-shot learning performances.

5. Conclusions

We addressed the problem of large-scale long-tailed instance segmentation by formulating a novel paradigm: class-incremental few-shot learning, where any large dataset can be divided into groups and incrementally learned from the head to the tail. This paradigm introduces two new challenges over time: 1) for countering the catastrophic forgetting, the old classes are more and more imbalanced, 2) the new classes are more and more few-shot. To this end, we develop the Learning to Segment the Tail (LST) method, equipped with a novel instance-level balanced replay technique and a meta-weight generator for few-shot classes adaptation. Experimental results on the LVIS dataset [10] demonstrated that LST could gain a significant improvement for the tail classes and achieve an overall boost for the whole 1,230 classes. LST offers a novel and practical solution for learning from large-scale long-tailed data: we can use only one downside — head-class forgetting, to trade off the two challenges — the large vocabulary and few-shot learning.

Acknowledgements. We thank all the reviewers for their constructive comments. This work was supported by Alibaba-NTU JRI, and partly supported by Major Scientific Research Project of Zhejiang Lab (No. 2019DB0ZX01).

References

[1] Bryan C Russell, Antonio Torralba, Kevin P Murphy, and William T Freeman. Labelme: a database and web-based tool for image annotation. In *IJCV*, 2008. 1

[2] Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-End Incremental Learning. In *ECCV*, 2018. 3

[3] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. In *Journal of artificial intelligence research*, 2002. 1, 2

[4] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid Task Cascade for Instance Segmentation. In *CVPR*, 2019. 1

[5] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. MaskLab: Instance Segmentation by Refining Object Detection With Semantic and Direction Features. In *CVPR*, 2018. 2

[6] 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

[7] Qi Dong, Shaogang Gong, and Xiatian Zhu. Class rectification hard mining for imbalanced deep learning. In *ICCV*, 2017. 2

[8] Spyros Gidaris and Nikos Komodakis. Dynamic Few-Shot Visual Learning Without Forgetting. In *CVPR*, 2018. 2, 3, 4, 5

[9] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 4

[10] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A Dataset for Large Vocabulary Instance Segmentation. In *CVPR*, 2019. 1, 2, 3, 5, 6, 7, 8

[11] Haibo He, Yang Bai, E. A. Garcia, and Shutao Li. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *IEEE International Joint Conference on Neural Networks*, 2008. 1, 2

[12] Haibo He and Edwardo A Garcia. Learning from imbalanced data. In *IEEE Transactions on Knowledge & Data Engineering*, 2008. 1

[13] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2, 3, 4, 6, 7

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 6

[15] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In *NeurIPS*, 2014. 3

[16] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a Unified Classifier Incrementally via Rebalancing. In *CVPR*, 2019. 2, 3, 4

[17] Ronghang Hu, Piotr Dollár, Kaiming He, Trevor Darrell, and Ross Girshick. Learning to Segment Every Thing. In *CVPR*, 2018. 2, 3, 4, 6

[18] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning Deep Representation for Imbalanced Classification. In *CVPR*, 2016. 2

[19] Buyu Li Quanquan Li Wanli Ouyang Changqing Yin Junjie Yan Jingru Tan, Changbao Wang. Equalization loss for long-tailed object recognition. *ArXiv:2003.05176*, 2020. 7

[20] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-Shot Object Detection via Feature Reweighting. In *ICCV*, 2019. 2, 3

[21] Aoxue Li, Tiange Luo, Tao Xiang, Weiran Huang, and Liwei Wang. Few-Shot Learning With Global Class Representations. In *ICCV*, 2019. 2

[22] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully Convolutional Instance-Aware Semantic Segmentation. In *CVPR*, 2017. 2

[23] Zhizhong Li and Derek Hoiem. Learning Without Forgetting. In *ECCV*, 2016. 3, 4

[24] 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. 6

[25] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path Aggregation Network for Instance Segmentation. In *CVPR*, 2018. 2

[26] Yaoyao Liu, Yuting Su, An-An Liu, Bernt Schiele, and Qianru Sun. Mnemonics training: Multi-class incremental learning without forgetting. In *CVPR*, June 2020. 3

[27] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. In *JMLR*, 2008. 7

[28] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, 24:109–165, 1989. 1, 2

[29] Ethan Fetaya Richard S. Zemel Mengye Ren, Renjie Liao. Incremental few-shot learning with attention attractor networks. In *NeurIPS*, 2019. 2, 3

[30] Kemal Oksuz, Baris Can Cam, Sinan Kalkan, and Emre Akbas. Imbalance Problems in Object Detection: A Review. *arXiv preprint arxiv:1909.00169*, 2019. 7

[31] David M W Powers. Applications and explanations of zipf's law. *Association for Computational Linguistics*, page 151–160, 1998. 1

[32] Hang Qi, Matthew Brown, and David G. Lowe. Low-Shot Learning With Imprinted Weights. In *CVPR*, 2018. 3

[33] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. In *CVPR*, 2017. 2, 3, 4, 5

[34] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1

[35] Li Shen, Zhouchen Lin, and Qingming Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *ECCV*, 2016. 5, 6, 7

[36] Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. Incremental Learning of Object Detectors Without Catastrophic Forgetting. In *ICCV*, 2017. 2, 3

[37] Merrielle Spain and Pietro Perona. Measuring and predicting importance of objects in our visual world. In *Technical Report CNS- TR-2007-002*, 2007. 1

[38] Gan Sun, Yang Cong, and Xiaowei Xu. Active Lifelong Learning With ”Watchdog”. In *AAAI*, 2018. 3

[39] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *CVPR*, 2019. 2

[40] Kai Ming Ting. A comparative study of cost-sensitive boosting algorithms. In *ICML*, 2000. 2

[41] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *NeurIPS*, 2016. 2, 6

[42] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Learning to model the tail. In *NeurIPS*, 2017. 2, 3

[43] Yu-Xiong Wang, Deva Ramanan, and Martial Hebert. Meta-learning to detect rare objects. In *ICCV*, 2019. 2, 3

[44] 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

[45] Saining Xie, Ross Girshick, Piotr Dollar, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, July 2017. 7

[46] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Learning Classifier Synthesis for Generalized Few-Shot Learning. *arXiv preprint arxiv:1906.02944*, 2019. 2, 3, 5

[47] Zhi-Hua Zhou and Xu-Ying Liu. On Multi-Class Cost-Sensitive Learning. In *AAAI*, 2006. 2

[48] George Kingsley Zipf. The psycho-biology of language: An introduction to dynamic philology. In *Routledge*, 2013. 1