Multimedia Tools and Applications (2023) 82:7383–7400
https://doi.org/10.1007/s11042-022-13424-8

# Improving apparel detection with category grouping and multi-grained branches

Qing Tian[1,2] · Sampath Chanda[3] · Amit Kumar K C[3] · Douglas Gray[3]

Received: 17 January 2021 / Revised: 14 March 2022 / Accepted: 2 July 2022 /
Published online: 20 August 2022

## Abstract

Training an accurate object detector is expensive and time-consuming. One main reason lies in the laborious labeling process, i.e., annotating category and bounding box information for all instances in every image. In this paper, we examine ways to improve performance of deep object detectors without extra labeling. We first explore to group existing categories of high visual and semantic similarities together as one super category (or, a superclass). Then, we study how this knowledge of hierarchical categories can be exploited to better detect objects using multi-grained RCNN branches*. Experimental results on DeepFashion2 and OpenImagesV4-Clothing reveal that the proposed detection heads with multi-grained branches can boost the overall performance by as high as 2% with no additional time-consuming annotations. In addition, classes that have fewer training samples tend to benefit more from the proposed multi-grained heads with superclass grouping. In particular, we improve the mAP for the last 30% categories (in terms of training sample number) by 2.6% and 4.6% on DeepFashion2 and OpenImagesV4-Clothing, respectively.

**Keywords** Apparel detection · Deep neural networks · Multi-grained detection branches · Imbalanced data

Qing Tian - work partially done as an applied scientist intern at Amazon.

The term *multi-grained* is used to describe features of different abstractness and granularity levels, corresponding to *coarse-* and *fine-*grained categories. RCNN: Regional Convolutional Neural Networks [29].

✉ Qing Tian
qtian@bgsu.edu; qing.tian@mail.mcgill.ca

1 Department of Computer Science, Bowling Green State University, Hayes Hall, Bowling Green, OH 43403, USA

2 Department of Electrical & Computer Engineering, McGill University, 3480 University Street, Montreal, QC H3A 0E9, Canada

3 Visual Search & AR (A9), Amazon.com, Inc., 611 Cowper Street, Palo Alto, CA 94301, USA

## 1 Introduction

In the deep learning era, accurate object detection relies on a large amount of training data and annotations. However, instance-level annotations, such as bounding boxes, are expensive to collect compared to image-level labels. Therefore, it is important to make full use of currently available data and annotations. To this end, we propose to improve detection performance using multi-grained detection heads with superclass grouping in this paper. We integrate and utilize information from different abstraction levels during both training and inference. Such detectors that can extract and utilize hierarchical features are desirable in many real-world scenarios. For example, in case of fashion items, the features that separate different kinds of tee-shirts should be on different levels with those that distinguish tee-shirts from skirts. In practice, most items in catalogs of online retailers are organized hierarchically so that data and ground-truth labels are usually collected and used for training in a multi-grained fashion. However, most existing object detection algorithms such as SSD [24], YOLO and its variants [2, 26–28], RCNNs [10, 11, 29] are designed to exploit labels in a flat manner. Consequently, these models cannot benefit from hierarchical nature of the data.

In addition, most real-world datasets are imbalanced in that some categories have more samples than others. Strategies such as data resampling [4, 13] and cost-sensitive learning [19, 33, 44] can be adopted to alleviate this imbalance problem. Relatively speaking, fewer deep learning approaches [18, 35, 36, 43] have been proposed to address this issue for object detection. Detectors trained on imbalanced datasets can lead to low detection accuracy for small classes (i.e., classes with a limited number of samples). We hypothesize that closely-related fine-grained categories may utilize similar coarse-grained features. Utilizing multi-grained detection heads, we attempt to improve performance on small categories via learning shared coarse features with the help of data from related large categories. In this way, we make full use of available bounding box annotations.

In this paper, we apply our multi-grained detection heads to apparel detection because it is a challenging task that needs more attention and investigation. As discussed by many researchers in this field [12], apparel detection is challenging for several reasons: (1) deformation of clothing items, (2) various sizes (one-piece dress vs. earrings), (3) different styles in a certain clothing category. Accurate apparel detection can serve as a crucial first step of a wide range of e-commerce applications. For example, on online retailers like Amazon, people can upload their pictures of fashion items to retrieve similar or identical merchandise. If the detection is poor, totally irrelevant items may be returned.

## 2 Related works

In this section, we review some important deep-learning-based object detection approaches. One of the dominant paradigms in modern object detection is based on a two-stage approach [17]. In the first stage, a set of candidate proposals are generated using techniques like Selective Search [34] and EdgeBoxes [45]. The second stage refines the proposals into a final set of bounding boxes along with their corresponding categories. RCNN [11] and its variants, namely Fast RCNN [10] and Faster RCNN [29], belong to this category of detectors. Mask RCNN [15] is based on Faster RCNN with a separate mask branch added for instance segmentation. In an effort to produce more accurate bounding boxes, multi-step detectors have also been proposed that explore different ways of gradually refining the detections. Some examples are multi-region detector [8], CRAFT [40], AttractioNet [9], and

Cascade R-CNN [3]. Cascade R-CNN [3] consists of a sequence of detectors trained with increasing values of Intersection over Union (IoU). Recently, specialized RCNNs are shown to beat the baseline in particular use cases, such as Beta R-CNN for pedestrian detection [39] and Forest R-CNN [38] for large-vocabulary long-tailed scenarios.

Compared to the two-stage paradigm, one-stage object detectors have gained popularity mainly due to their efficiency. Unlike the final sparse prediction in RCNN, single-stage detectors perform dense prediction in the end (no intermediate ROI proposal process). OverFeat [30] is one of the first one-stage object detectors. More recently, SSD [24] and YOLOs [2, 26–28] have reduced the detection accuracy gap between one-stage and two-stage detectors while improving on the computation efficiency. To achieve better accuracy without much extra computation, Lin et al. [23] have proposed *focal loss* to assign higher weights to hard examples dynamically. One special variant of YOLO, called YOLO9000 [27], detects over 9000 object categories by taking advantage of both the abundance of fine-grained category labels in ImageNet [5] and the plentifulness of coarse-class bounding box annotations in the MS-COCO dataset [21]. Another work that uses coarse- and fine-grained category organization is [41]. It detects at the level of fine grained classes, while only requiring a small set of bounding box annotations at coarse-grained class level. Based on better backbones derived from extensive neural architecture search on hundreds of GPUs [31], Tan et al. [32] have developed an efficient and accurate family of object detectors, named EfficientDets.

In most modern detectors such as the above-mentioned Faster RCNN [29], the same level of (top) features is used to detect and classify different categories. However, representations discriminative for different categories may lie on different abstraction levels. Previous works, such as skip connection [14, 16] and layer aggregation [42], have been shown to be effective for integrating different-level information. As far as we know, few, if any, works have experimented with utilizing multi-level information in distinct detection heads (after ROI determination). Also, none has attempted to inject multi-level supervision information on such heads to explicitly learn features at different granularities.

Our work builds on the two-stage Faster RCNN backbone (one of the best-performing object detectors [28]) and we propose detection heads with multi-grained branches to exploit information from different abstractness levels. Unlike Mask RCNN [15] (another Faster-RCNN multiple-branch extension as mentioned previously), our multiple branches capture multi-grained information and tackle the same final task (fine-grained apparel detection in our case). To train these multi-grained branches, we first group the categories into hierarchical manner (i.e., superclass grouping).

## 3 Category grouping and multi-grained detection heads

In this section, we first discuss category grouping strategies and then present different detection head architectures to improve the overall detection performance as well as the performance for minority categories.

### 3.1 Superclass grouping for multi-grained detection heads

First, we need to build a hierarchical structure of the data. If the dataset contains hierarchical information, we respect the original hierarchy or superclass grouping. If not, the WordNet [25] hierarchy can be used as a guidance (as many works do [6, 27]). In this paper, we

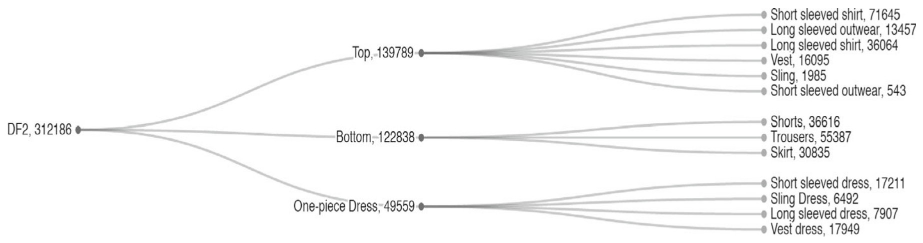


**Fig. 1** Category grouping and instance number in each category of DeepFashion2 training set

group fine-grained categories into coarse-grained super-categories based on visual and functional similarity. It is worth noting that the extra superclass grouping/labeling is cheap. It simply becomes a dictionary checkup procedure after a hierarchy containing supercategory-category pairs is defined. This cost can be negligible in most cases compared to expensive instance-level labeling. Specifically, we use two publicly available datasets, namely DeepFashion2 [7] and OpenImagesV4 [20] in this paper. The details of the two datasets and our category grouping strategies are as follows:

**DeepFashion2** contains 491K images of 13 clothing categories from both commercial shopping stores and consumers, out of which 312,186 training instances and 52,490 validation instances are publicly available. For our detection task, only category and bounding box information are utilized in our experiments. The 13 categories are grouped into 3 super categories, namely top, bottom, and one-piece dress, as shown in Fig. 1.

**OpenImagesV4** [20] contains 15.4 million bounding boxes for 600 categories on 1.9 million images, rendering it one of the largest existing public datasets with object location annotations (approximately 15 times more bounding boxes than ImageNet [5] and MS-COCO [21]). For our purpose, we select the subset of clothing instances, which we refer to as OpenImagesV4-Clothing dataset in the rest of the paper. There are 490,777 bounding box instances for training and 3,897 instances for validation in OpenImagesV4-Clothing. While grouping the categories, we respect the hierarchy provided by the dataset[1] with a few changes. In particular, we perform the following modifications to the original dataset (A $\rightarrow$ B means A's subclass B):

- We merge *helmet*, *hat*, and *fashion accessory* $\rightarrow$ *hat* as *headwear*.
- We merge *fashion accessory* $\rightarrow$ *necklace, tie*, and *clothing* $\rightarrow$ *scarf* as *neckwear*.
- We group *clothing* $\rightarrow$ *shorts*, *trousers* $\rightarrow$ *jeans*, and *skirt* $\rightarrow$ *miniskirt* together as bottom.

Figure 2 demonstrates the super category and category relationship with the number of training instances added to each category.

In the following section, we will present how we exploit the resulting hierarchical data structure through coarse- and fine-grained detection heads.

[1] https://storage.googleapis.com/openimages/2018_04/bbox_labels_600_hierarchy_visualizer/circle.html

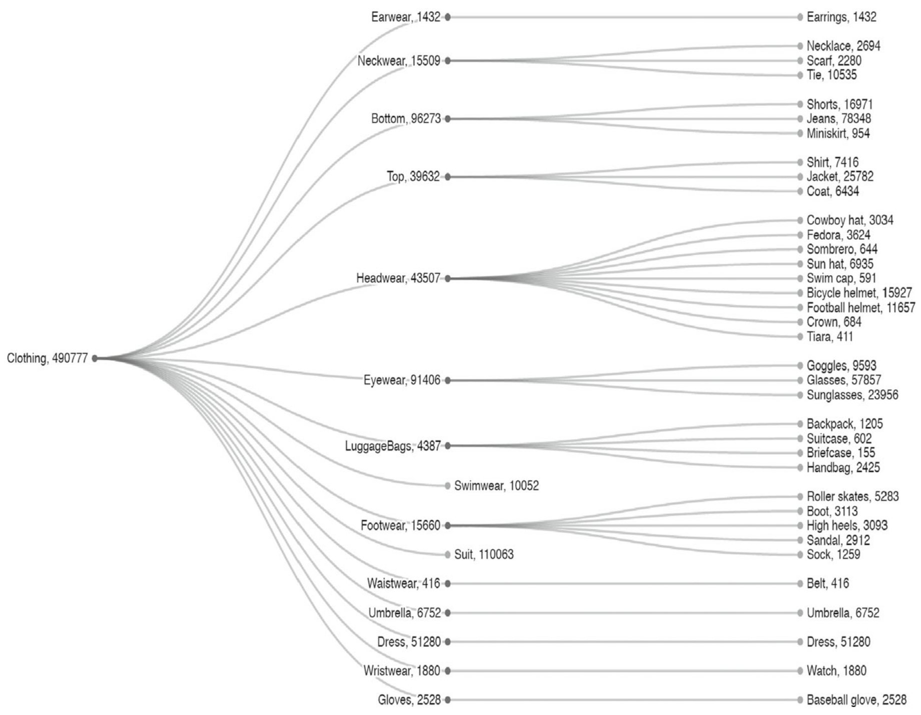


**Fig. 2** Category grouping and instance number in each category of OpenImagesV4-Clothing training set

### 3.2 Detection heads with multi-grained branches

In Faster RCNN, after backbone feature extraction and region-of-interest (ROI) proposal stages, a single branch with one set of bounding box regression and classification heads are plugged on top of the pooled/aligned ROI features. Consequently, it cannot integrate information from different abstract levels explicitly or utilize labels in a hierarchical manner. Instead, we propose to add several branches of different depths to the ROI features.

The entire pipeline is shown as Table 1.

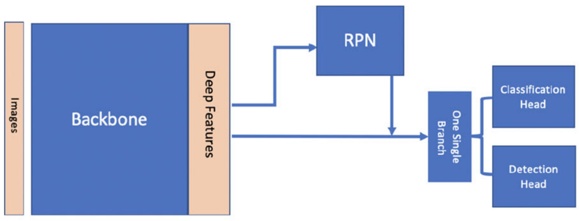
Different branch depths extract features of different granularities and abstractness levels, on which our localization and classification are based. We refer such head structures as

**Table 1** Object detector with multi-grained branches

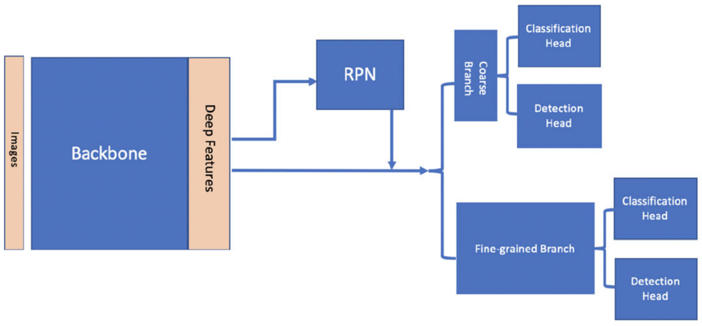
| | |
|---|---|
| Input: | Image |
| Backbone: | ResNet-101 (He et al. 2016) |
| Neck: | Feature Pyramid Networks or FPN (Lin et al. 2017) |
| Dense Proposal: | Region Proposal Network or RPN (Ren et al. 2016) |
| Sparse Multi-grained Detection Head: | |
| – One-Level prediction branch | |
| – Two-Level prediction branch | |
| – ... | |

*multi-grained* heads or detection heads with *multi-grained* branches. Each branch can be any type of network that helps increase feature abstractness, e.g., convolutional or fully connected. To be more specific, the multi-grained branch ideas/variants explored in this paper are demonstrated in Fig. 3. For our datasets, the labels have been organized in two levels. Therefore, we only study detection heads of two granularity levels here. They can be easily extended to cases with more granularity levels.

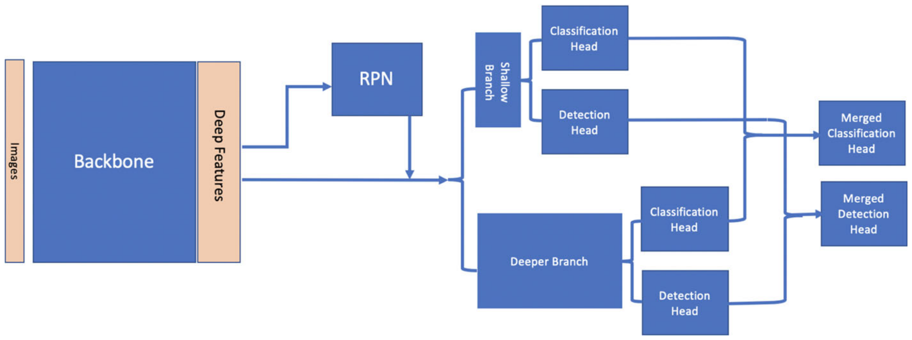
Figure 3a shows the original Faster RCNN structure [29]. We choose ROI alignment [15] instead of ROI pooling [29] for all our configurations.



(a) Single detection head in Faster RCNN



(b) Multi-grained detection heads with Faster RCNN backends



(c) Merged detection heads with Faster RCNN backends

**Fig. 3** Detection heads with single or multi-grained branches on Faster RCNN backends. RPN: region proposal network. Blue blocks represent network structures while light orange blocks indicate feature maps and input (only important feature maps are shown). In our experiments, more layers are added to the one single branch in (a) to match the size of others

In Fig. 3b, two detection branches of different depths are added to the ROI-aligned feature maps. With more depths, the longer branch is expected to learn more abstract and finer-grained features than the shallower branch. The same number of neurons are on top of the coarse- and fine-grained branches. Fine-grained supervision is applied on the fine-grained branch. We can inject either superclass (with grouping) or fine-grained class (w/o grouping) information onto the coarse branch. The latter is equivalent to grouping each category as a separate super category on its own. For the 'with grouping' option, the amount of work needed to perform category-level grouping is negligible compared to instance-level annotation. We use only the fine-grained bounding box regression head to evaluate localization performance.

In addition, based on Fig. 3b, we add extra merging layers that combine information from the coarse and fine-grained stages as shown in Fig. 3c. Compared to other parts of the network (backbone, RPN, and head branches), a merging layer is light. It only consists of one fully connected layer to integrate the outputs from different abstract levels and produce one final set of results. The lower layers to be merged are first concatenated before being fully connected with the merging layer. One difference of Fig. 3c from Fig. 3b is that apart from the supervision added at the end, no separate coarse-level supervision is applied.

**Loss function** For Fig. 3a and c, standard softmax cross entropy and smooth $L_1$ loss [10] are used as loss functions for box head classification and regression, respectively. For multi-grained branches without the additional merging head (Fig. 3b), we modify these losses to:

$$E_{\mathrm{ce}} = -\sum_{l=1}^{L}\sum_{c=1}^{C} w_k^{(l)} t_c^{(l)} \ln y_c^{(l)} \tag{1}$$

and

$$E_{\mathrm{loc}} = \sum_{l=1}^{L} \sum_{i\in(x,y,w,h)} w_r^{(l)} * smooth_{L_1}(o_i^u - v_i) \tag{2}$$

where $E_{ce}$ and $E_{loc}$ correspond to the cross entropy and bounding box regression losses, $l$ indicates level or stage, $L$ is the number of levels used ($L = 2$ in our case), $c$ represents category (including background), $y$ is the set of predicted results after softmax, $t$ is the set of ground-truth labels (one-hot encoded in our case), $w_k^{(l)}$ and $w_r^{(l)}$ are level $l$ weighting factors for classification and bounding box regression losses, respectively. $smooth_{L_1}$ stands for the smooth $L_1$ function as proposed in [10]. $o^u$ is predicted bounding-box offsets tuple for the true class $u$ and $v$ is ground-truth bounding-box regression target. In our experiments, $L$ equals 2. $l = 0$ and $l = 1$ represent the coarse-grained level and the fine-grained level, respectively. $w_k^{(l=0)} = 0.2$, $w_k^{(l=1)} = 1.0$, $w_r^{(l=1)} = 1.2$, and $w_r$ for the coarse branch is 0. We drop the loss component from the coarse branch to avoid the complex bounding box merging process from multiple branches. The total loss of the framework is the sum of the above two losses together with the binary classification and localization regression losses of the RPN subnetwork.

## 4 Experiments

In this section, we first describe the experimental setup (Section 4.1) and the evaluation metrics (Section 4.2) for our proposed solutions. Then, we present our quantitative and qualitative results and compare the proposed approaches and the baseline (Section 4.3).

**Table 2** Training time per iteration of the competing architectures

| | Baseline | Multi-branch | Merged |
|---|---|---|---|
| Training time | 0.4301s | 0.4311s | 0.4317s |

Baseline: Fig. 3a, Multi-branch: Fig. 3b, Merged: Fig. 3c. The results are reported on DeepFashion2 images with a batch size of 24 (8 Tesla V100 GPUs, 3 images on each GPU)

## 4.1 Experimental setup

Our implementations are based on Detectron2 [37]. Modifications are made to support detection heads with multi-grained branches and superclass grouping. Our frameworks are trained with a batch size of 24 for 200k iterations using a stochastic gradient descent (SGD) optimizer. The base learning rate is 0.005 and is decreased twice at 150K and 175K iterations by a factor of 10. We use a weight decay of 0.0001 and momentum of 0.9. For each RPN anchor, 5 scales and 3 aspect ratios are used to generate original anchor boxes. As in [22], an anchor box is considered positive if its Intersection over Union (IoU) with a groundtruth box exceeds 0.5.

All our frameworks studied adopt a ResNet101 backbone [14] with Feature Pyramid Network (FPN) [22] support. The shared backbone is pre-trained on the MS-COCO dataset [21]. The coarse-grained and fine-grained branches consist of one and two fully connected layers of 1024 neurons, respectively. More architectural details can be found in Appendix. Parameter size differences between multi-grained detection heads with and without merged heads are negligible. As for the baseline model, we add more layers to the single branch after ROI alignment in order to match the size of our multi-grained approach. In our experiments, all competing models have approximately 73M trainable parameters.

Tables 2 and 3 demonstrate the timing of the training and inference processes of the competing networks (DeepFashion2 images are used for this purpose). In Table 2, the training time of a network is per iteration where one batch contains 24 images (8 Tesla V100 GPUs, 3 images on each GPU).

The inference speed is shown in Table 3.

As we can see from the tables, the training/inference latency of the different nets are similar. The multi-branch network is slightly faster than the merged net and is slightly slower than the baseline network. Given that many hardware and environmental factors can influence the numbers, the differences are trivial.

## 4.2 Metrics

We adopt mean average precision (mAP) to evaluate our methods. In our context, the overall mAP score is calculated by taking the mean AP over all classes and over all IoU thresholds.

**Table 3** Inference time of the competing architectures (on DeepFashion2 images)

| | Baseline | Multi-branch | Merged |
|---|---|---|---|
| Inference time | 0.079064s | 0.080199s | 0.081445s |

Baseline: Fig. 3a, Multi-branch: Fig. 3b, Merged: Fig. 3c. The numbers reported are seconds/image per device, on 8 Tesla V100 devices

**Table 4** Results on the validation set of DeepFashion2

| | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP_s$ | $AP_m$ | $AP_l$ |
|---|---|---|---|---|---|---|
| YOLOv4 | 53.18 | 72.59 | 63.92 | 20.20 | 38.50 | 53.35 |
| Faster RCNN Baseline | 67.02 | 79.90 | 75.61 | 40.05 | 47.09 | 67.23 |
| Multi-grained W/O Group | 67.54 | 79.87 | 75.88 | 25.10 | 47.74 | 67.79 |
| Multi-grained Merged | 68.80 | 81.55 | 77.40 | 31.75 | 50.80 | 69.04 |
| Multi-grained with Group | 69.02 | 81.60 | 78.12 | 31.75 | 51.16 | 69.23 |

The Faster RCNN baseline is shown in Fig. 3a, 'W/O Group' indicates the naive multi-grained branches approach without grouping (Fig. 3b), 'with Group' refers to the multi-grained branches with superclass grouping (Fig. 3b), 'Merged' represents the merged heads approach (Fig. 3c). AP, without any postfix, represents AP[.5:.5:.95]. YOLOv4 results are reported just for reference (implementation based on [1])

Following the standard COCO metrics, for each framework on each dataset, we report AP (averaged over the IoU thresholds [0.5:0.5:0.95]), $AP_{50}$ (IoU threshold 0.5), $AP_{75}$ (IoU threshold 0.75), and $AP_s$ (small objects with area $< 32^2$), $AP_m$ (medium size objects), and $AP_l$ (large objects with area $> 96^2$).

## 4.3 Results

In this section, we first present our quantitative results. Our focus is on both overall mAP and per-category AP (especially for small categories). Then, qualitatively, we show some success and failure cases of our approaches.

### 4.3.1 Overall mAP

Tables 4 and 5 show the detection results on DeepFashion2 and OpenImagesV4-Clothing, respectively.

As we can see, it is beneficial to use information from different abstraction levels (in contrast to just the top one) to produce final detection results. The multi-grained branches with superclass grouping can clearly improve mAP scores by up to 2%. In general, the multi-grained branches utilizing superclass information is better than other multi-grained solutions on the two datasets. This shows that the injected superclass knowledge (at nearly no cost) helps with overall detection. It is worth mentioning that despite a lower overall

**Table 5** Results on the validation set of OpenImagesV4-Clothing

| | $AP$ | $AP_{50}$ | $AP_{75}$ | $AP_s$ | $AP_m$ | $AP_l$ |
|---|---|---|---|---|---|---|
| YOLOv4 | 30.61 | 49.52 | 33.43 | 7.20 | 22.29 | 32.80 |
| Faster RCNN Baseline | 47.88 | 67.19 | 52.45 | 8.21 | 29.05 | 52.21 |
| Multi-grained W/O Group | 48.40 | 67.79 | 53.53 | 8.96 | 31.31 | 52.85 |
| Multi-grained Merged | 49.48 | 68.59 | 54.98 | 8.79 | 29.23 | 54.12 |
| Multi-grained with Group | 49.50 | 68.34 | 55.12 | 9.77 | 29.19 | 53.91 |

The Faster RCNN baseline is shown in Fig. 3a, 'W/O Group' indicates the naive multi-grained branches approach without grouping (Fig. 3b), 'with Group' refers to the multi-grained branches with superclass grouping (Fig. 3b), 'Merged' represents the merged heads approach (Fig. 3c). AP, without any postfix, represents AP[.5:.5:.95]. YOLOv4 results are reported just for reference (implementation based on [1]) mAP score, the merged head actually achieves comparable mAP50 scores (slightly lower on DeepFashion2 and slightly higher on OpenImagesV4-Clothing). A possible contributing factor here is the relatively large amount of quality training data, especially in the larger OpenImagesV4-Clothing dataset. With enough training data, the network can possibly learn some hierarchical information itself. For this architecture to work well, much more data and annotations are needed. Comparing the 2nd and 3rd rows in Tables 4 and 5, we can see that the naive multi-grained branches without category grouping does not help much over the baseline. In some cases, the no-grouping solution even leads to worse performance than the baseline. One possible cause is the confusion that the W/O Group method introduces when injecting the same fine-grained information to both branches designed for learning different-level features.

### 4.3.2 Per-category AP

In addition to overall mAP, we report per-category AP in this section. The category columns are sorted in ascending order of the number of instances. It is evident that both datasets are imbalanced. In DeepFashion2, the largest category has 71,645 instances while the smallest

**Table 6** Per-category AP on DeepFashion2

| | Short sleeve outwear | Sling | Sling dress | Long sleeve dress | |
|---|---|---|---|---|---|
| Num | 543 | 1985 | 6492 | 7907 | |
| Baseline | 42.45 | 47.55 | 65.40 | 53.43 | |
| W/O Grp | 44.29 | 48.12 | 66.74 | 53.21 | |
| Merged | 46.11 | 49.63 | 68.41 | 56.28 | |
| W Group | 46.51 | 51.59 | 68.24 | 57.06 | |
| | Long sleeve outwear | Vest | Short sleeve dress | Vest dress | |
| Num | 13457 | 16095 | 17211 | 17949 | |
| Baseline | 72.57 | 66.93 | 72.31 | 72.69 | |
| W/O Grp | 73.80 | 67.47 | 72.24 | 72.95 | |
| Merged | 74.22 | 68.68 | 74.01 | 74.83 | |
| W Group | 73.97 | 68.28 | 73.73 | 74.81 | |
| | Skirt | Long sleeve shirt | Shorts | Trousers | s.sleeve shirt |
| Num | 30835 | 36064 | 36616 | 55387 | 71645 |
| Baseline | 74.20 | 73.26 | 71.08 | 75.32 | 81.12 |
| W/O Grp | 74.78 | 74.19 | 72.24 | 76.20 | 81.82 |
| Merged | 76.18 | 74.54 | 72.93 | 76.40 | 82.18 |
| W Group | 76.31 | 74.67 | 73.46 | 76.64 | 82.65 |

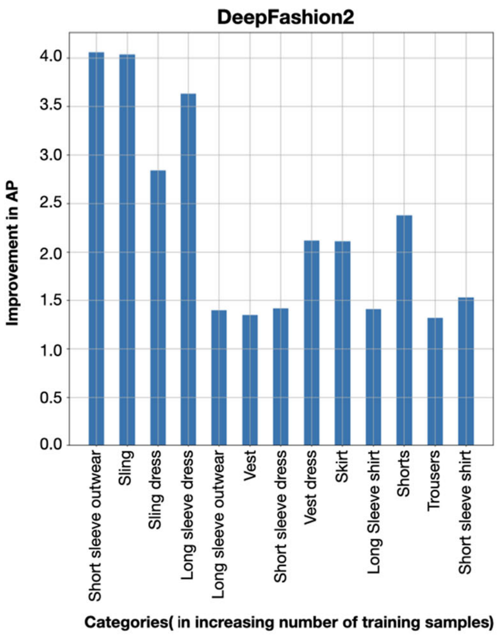
The columns are sorted in ascending order of the number of instances in that category. 's.sleeve shirt' stands for short sleeve shirt. 'Num' refers to the number of training bounding box instances in that category. 'Baseline': Fig. 3a, 'W/O Grp': Fig. 3b without grouping, 'W Group': Fig. 3b with grouping, 'Merged': Fig. 3c category has only 543 instances (130x smaller). In OpenImagesV4-Clothing, there are 500x more samples in the largest category than in the smallest category. A good overall detection score may be misleading if the small classes are what we care more about. Tables 6 and 7 demonstrate the detailed per-category AP scores for all four frameworks.

**Table 7** Per-category AP on OpenImagesV4-Clothing

| | Brief-case | Tiara | Swimcap | Suitcase | Sombrero |
|---|---|---|---|---|---|
| Num | 155 | 411 | 591 | 602 | 644 |
| Baseline | 4.81 | 51.20 | 49.16 | 33.90 | 38.04 |
| W/O Grp | 9.79 | 47.66 | 45.03 | 30.21 | 29.36 |
| Merged | 12.28 | 47.05 | 47.41 | 35.05 | 42.89 |
| W Group | 12.98 | 55.40 | 47.99 | 36.10 | 48.00 |
| | Crown | Miniskirt | Backpack | Sock | Scarf |
| Num | 684 | 954 | 1205 | 1259 | 2280 |
| Baseline | 47.55 | 51.19 | 43.90 | 57.82 | 37.74 |
| W/O Grp | 49.67 | 57.90 | 45.61 | 61.32 | 38.76 |
| Merged | 50.14 | 57.90 | 48.00 | 63.25 | 40.85 |
| W Group | 51.73 | 57.87 | 45.62 | 63.19 | 41.09 |
| | Hand-bag | Necklace | Sandal | Cowboy hat | High heels |
| Num | 2425 | 2694 | 2912 | 3034 | 3093 |
| Baseline | 76.53 | 60.50 | 40.28 | 38.61 | 42.52 |
| W/O Grp | 78.32 | 62.21 | 40.97 | 41.22 | 42.18 |
| Merged | 78.49 | 63.86 | 43.26 | 35.63 | 40.09 |
| W Group | 78.76 | 62.10 | 44.34 | 41.99 | 42.47 |
| | Boot | Fedora | Roller skates | Coat | Sunhat |
| Num | 3113 | 3624 | 5283 | 6434 | 6935 |
| Baseline | 57.86 | 58.55 | 44.18 | 43.77 | 50.14 |
| W/O Grp | 59.27 | 64.64 | 45.49 | 43.73 | 52.69 |
| Merged | 58.16 | 66.11 | 44.15 | 46.16 | 51.25 |
| W Group | 58.03 | 62.79 | 44.58 | 44.76 | 49.63 |
| | Shirt | Goggles | Tie | Football helmet | Bike helmet |
| Num | 7416 | 9593 | 10535 | 11657 | 15927 |
| Baseline | 59.49 | 37.36 | 72.11 | 52.12 | 41.31 |
| W/O Grp | 63.05 | 35.85 | 72.94 | 53.38 | 44.55 |
| Merged | 64.32 | 37.55 | 73.59 | 53.46 | 43.86 |
| W Group | 64.34 | 38.28 | 72.95 | 52.48 | 44.03 |
| | Shorts | Sunglasses | Jacket | Glasses | Jeans |
| Num | 16971 | 23956 | 25782 | 57857 | 78348 |
| Baseline | 41.74 | 31.51 | 41.92 | 44.05 | 56.33 |
| W/O Grp | 41.28 | 33.60 | 41.81 | 46.81 | 58.43 |
| Merged | 42.87 | 31.97 | 42.96 | 46.90 | 58.83 |
| W Group | 40.59 | 33.06 | 41.73 | 46.12 | 57.95 |

Only grouped categories are shown. The columns are sorted in ascending order of the number of instances. 'Num' refers to the number of training bounding box instances in that category. 'Baseline': Fig. 3a, 'W/O Grp': Fig. 3b without grouping, 'W Group': Fig. 3b with grouping, 'Merged': Fig. 3c

We can see from Tables 6 and 7 that, in general, injecting superclass grouping information to the coarse branch helps with small classes more than larger classes. The average mAP improvements (over the baseline) across all categories are 2.3% and 2.5% on the DeepFashion2 and OpenImagesV4-Clothing datasets, respectively. In comparison, we improve



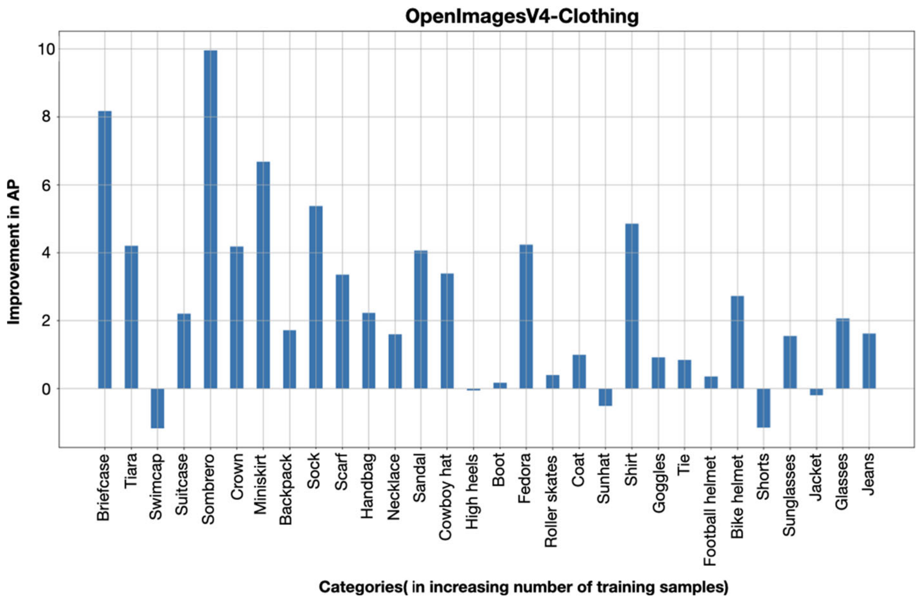


**Fig. 4** Improvement on AP for DeepFashion2 (top) and OpenImagesV4-Clothing (bottom) datasets. The categories are sorted in ascending order of number of training samples. We can see that the improvements are significant for categories with a small number of samples

the mAP for last 30% categories (in terms of number of training samples) by 2.55% and 4.59% for the two datasets, respectively. In particular, the mAP score is improved by as high as about 9.96% for the small *sombrero* category in OpenImages-Clothing that contains only 644 instances. To better illustrate the general trend that small classes benefit more, we present the improvement in AP for all categories and for both datasets in Fig. 4.

One possible reason for the trend is that similar classes can share many low-level features. Explicitly grouping similar classes together helps to learn such low-level features discriminatively, which is especially helpful for minority classes. In addition, category grouping may alleviate the over-fitting issue when attempting to train millions of parameters using only a few hundred instances for a class. Our approach of injecting superclass

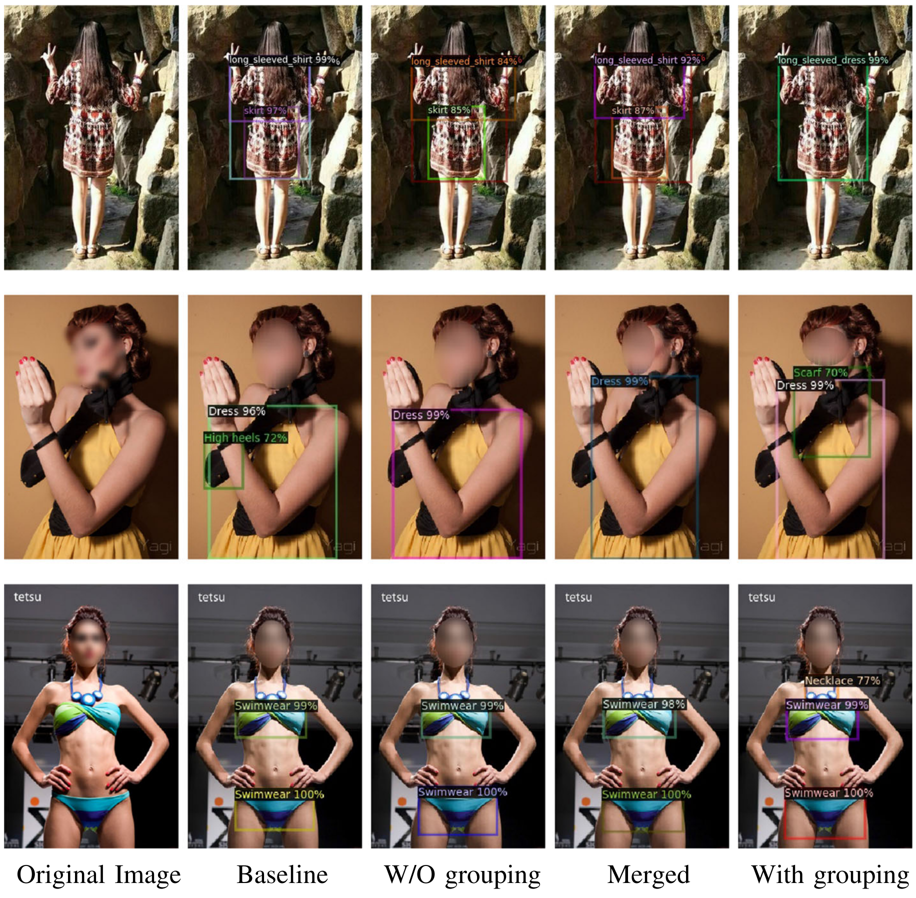


**Fig. 5** **Best viewed in color.** Qualitative results of different heads with single and multi-grained branches. ‘Baseline’: the Faster RCNN baseline (Fig. 3a), W/O grouping: multi-grained branches (Fig. 3b) without grouping, Merged: multi-depth branches with merged heads (Fig. 3c), W Grouping: multi-grained branches (Fig. 3b) with superclass grouping. The first image is from DeepFashion2 (cropped for clarity) and the other two are from OpenImagesV4-Clothing. We have blurred faces for privacy reasons

grouping information to the coarse branch is of great practical value. It can help address the 'cold start problem' when a new category, with a limited number of instances, is introduced or added.

That said, there are some small category exceptions where the performance improvement is small or even negative. This could be caused by the imperfection of superclass grouping. Also, different categories have different difficulties to be detected/recognized. Information from closely-related categories may not be enough to greatly improve the performance for some challenging categories.

#### 4.3.3 Qualitative results

Figure 5 demonstrates some successful qualitative examples on the two datasets. These results give an idea of how multi-grained branches with superclass grouping can beat the baseline. For example, our approach with superclass grouping correctly detects the long-sleeved dress in the first image and successfully recognizes the scarf and necklace in the second and third images. It is also worth mentioning that compared to the baseline, other multi-grained heads (i.e., W/O grouping and Merged) tend to have lower confidence in the false positives (e.g., the skirt and the long-sleeved shirt in the first image) and they do not misdetect the elbow region in the second image as high heels. One possible reason is that the multi-grained heads can pick up some useful grouping information themselves via learning, although the results are sub-optimal without explicit human guidance in the above-mentioned cases.

In Fig. 6, we demonstrate some of our typical failure cases. As we can see, our failure cases correspond mostly to false negatives.

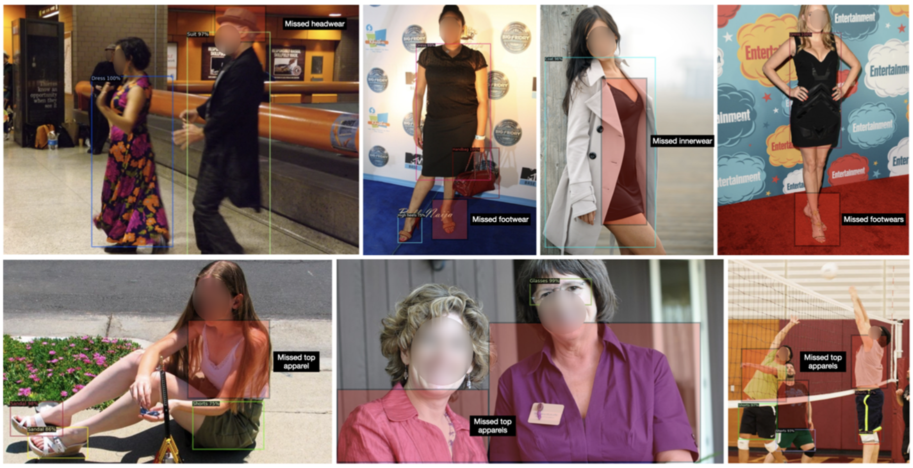


**Fig. 6 Best viewed in color.** Examples of failure cases of our proposed solution on OpenImagesV4-Clothing dataset. We highlight the missed detections with red boxes and also specify what category we missed to detect. We have blurred faces for privacy reasons

## 5 Discussion and future work

Although human hierarchy knowledge is useful (especially when labeled data is limited), it may not be well-aligned with the coarse-to-fine features that is deeply learned using a single branch with only fine-grained supervision on top. Unlike popular deep layer aggregation strategies [42], we have attempted to employ separate branches to capture coarse and fine features and to integrate multi-grained human expertise in a flexible and hierarchical manner (e.g., dog and golden retriever). Through shared lower layer features, the separate branches can interact with each other. Our multibranch solution (Fig. 3b with grouping) has shown to outperform the merged solution (Fig. 3c) that does not intake any human knowledge about the superclass.

Also, our frameworks with multi-grained branches may handle newer categories more easily. It is expected to alleviate the ‘cold start problem’ when a new category with a limited number of samples is first introduced. If the new category’s superclass already exists in our model, the retraining efforts can be focused more on the fine-grained branch than the coarse-grained branch and the lower layers. Such explorations are deferred to future work.

Apart from human hierarchy knowledge, other superclass grouping strategies can be explored or even learned if we have enough labeled data. Although both datasets we used are already very large publicly available datasets in this field, it would be of interest to experiment on even larger data and more annotations. It is possible that with unlimited data and annotations, merged head can successfully learn to integrate different abstractness level information to minimize the loss. That said, in most real-world cases with limited annotated data, superclass grouping as shown can be a simple yet effective way to improve detection performance.

Our multi-grained detection heads are designed for two-stage detectors and are built on top of Regions of Interest (ROI). It would be computationally expensive if we plug our multi-grained heads on all the regions at the end of a single detector (with no intermediate ROI proposal). If we limit the number of candidate regions, it becomes essentially a RCNN-like object detector (where the one-stage dense prediction head can be considered as the first stage in RCNN). That said, it is possible to add multiple dense prediction heads of different abstractness levels on top of some shared low-level features. Given the growing popularity of single-stage detectors, this would be another interesting future direction.

## 6 Conclusion

In this paper, we study several detection head solutions with multi-grained branches, which can make use of ROI information from different abstractness levels. According to our experiments on DeepFashion2 and OpenImagesV4-Clothing with Faster RCNN backends, detection heads with multi-grained branches can boost detection performance by up to 2% without requiring extra expensive annotations. Particularly, superclass grouping with human knowledge can greatly improve the performance for minority categories with fewer images. For example, in the sombrero case of OpenImagesV4-Clothing, its mAP is increased by as high as 10%.

## Appendix : detailed configuration of RPN and ROI heads

```
(proposal_generator): RPN(
  (anchor_generator): DefaultAnchorGenerator(
    (cell_anchors): BufferList()
  )
  (rpn_head): StandardRPNHead(
    (conv): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (objectness_logits): Conv2d(256, 3, kernel_size=(1, 1), stride=(1, 1))
    (anchor_deltas): Conv2d(256, 12, kernel_size=(1, 1), stride=(1, 1))
  )
)
(roi_heads): StandardROIHeads(
  (box_pooler): ROIPooler(
    (level_poolers): ModuleList(
      (0): ROIAlign(output_size=(7, 7), spatial_scale=0.25, sampling_ratio=0, aligned=
          ↪ True)
      (1): ROIAlign(output_size=(7, 7), spatial_scale=0.125, sampling_ratio=0, aligned
          ↪ =True)
      (2): ROIAlign(output_size=(7, 7), spatial_scale=0.0625, sampling_ratio=0,
          ↪ aligned=True)
      (3): ROIAlign(output_size=(7, 7), spatial_scale=0.03125, sampling_ratio=0,
          ↪ aligned=True)
    )
  )
  (box_head): FastRCNNConvFCHead(
    (fc1): Linear(in_features=12544, out_features=1024, bias=True)
    (fc2): Linear(in_features=1024, out_features=1024, bias=True)
  )
  (box_predictor): FastRCNNOutputLayers(
    (cls_score): Linear(in_features=1024, out_features=14, bias=True)
    (bbox_pred): Linear(in_features=1024, out_features=52, bias=True)
  )
  (super_box_head): SuperFastRCNNConvFCHead(
    (fc1): Linear(in_features=12544, out_features=1024, bias=True)
  )
  (super_box_predictor): FastRCNNOutputClsLayer(
    (cls_score): Linear(in_features=1024, out_features=4, bias=True)
  )
)
```

Note: ‘Super’ indicates the superclass or coarse branch. For clarity, ReLU layers are not shown.

**Acknowledgements** This work would not have been possible without the computing resources provided by Amazon AWS and the Ohio Supercomputer Center. This research was supported in part by the National Science Foundation (NSF) under Award No. 2153404.

## Declarations

**Competing interests** Competing interests: the authors have no competing interests to declare that are relevant to the content of this article.

## References

1. An implementation of YOLOv4 in PyTorch. https://https://github.com/bubbliiiing/yolov4-pytorch. Accessed 14 March 2022
2. Bochkovskiy A, Wang CY, Liao HYM (2020) Yolov4: Optimal speed and accuracy of object detection. arXiv: 2004.10934
3. Cai Z, Vasconcelos N (2018) Cascade r-cnn: delving into high quality object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6154–6162
4. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) Smote: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
5. Deng J, Dong W, Socher R, Li L.-J, Li K, Fei-fei L (2009) ImageNet: a large-scale hierarchical image database. In: CVPR09
6. Deng J, Krause J, Berg AC, Fei-Fei L (2012) Hedging your bets: optimizing accuracy-specificity trade-offs in large scale visual recognition. In: 2012 IEEE Conference on computer vision and pattern recognition, pp 3450–3457. IEEE
7. Ge Y, Zhang R, Wang X, Tang X, Luo P (2019) Deepfashion2: a versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5337–5345
8. Gidaris S, Komodakis N (2015) Object detection via a multi-region and semantic segmentation-aware cnn model. In: Proceedings of the IEEE international conference on computer vision, pp 1134–1142
9. Gidaris S, Komodakis N (2016) Attend refine repeat: active box proposal generation via in-out localization. arXiv:1606.04446
10. Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
11. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
12. Hara K, Jagadeesh V, Piramuthu R (2016) Fashion apparel detection: the role of deep convolutional neural network and pose-dependent priors. In: 2016 IEEE Winter conference on applications of computer vision (WACV). IEEE, pp 1–9
13. He H, Garcia EA (2009) Learning from imbalanced data. IEEE Trans Knowl Data Eng 21(9):1263–1284
14. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
15. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969
16. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
17. Jiao L, Zhang F, Liu F, Yang S, Li L, Feng Z, Qu R (2019) A survey of deep learning-based object detection. IEEE Access 7:128837–128868
18. Khan SH, Hayat M, Bennamoun M, Sohel FA, Togneri R (2017) Cost-sensitive learning of deep feature representations from imbalanced data. IEEE Trans Neural Netw Learn Syst 29( 8):3573–3587
19. Krawczyk B, Woźniak M, Schaefer G (2014) Cost-sensitive decision tree ensembles for effective imbalanced classification. Appl Soft Comput 14:554–562
20. Kuznetsova A, Rom H, Alldrin N, Uijlings J, Krasin I, Pont-Tuset J, Kamali S, Popov S, Malloci M, Duerig T et al (2018) The open images dataset v4: unified image classification, object detection, and visual relationship detection at scale. arXiv: 1811.00982
21. Lin T-Y, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco: common objects in context. In: European conference on computer vision, pp 740–755. Springer
22. Lin T-Y, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2117–2125

23. Lin T-Y, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988
24. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) Ssd: single shot multibox detector. In: European conference on computer vision. Springer, pp 21–37
25. Miller GA, Beckwith R, Fellbaum C, Gross D, Miller KJ (1990) Introduction to wordnet: an on-line lexical database. Int J Lexicogr 3(4):235–244
26. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
27. Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271
28. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv: 1804.02767
29. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. In: Advances in neural information processing systems, pp 91–99
30. Sermanet P, Eigen D, Zhang X, Mathieu M, Fergus R, LeCun Y (2013) Overfeat: integrated recognition, localization and detection using convolutional networks. arXiv: 1312.6229
31. Tan M, Le Q (2019) Efficientnet: rethinking model scaling for convolutional neural networks. In: International conference on machine learning, pp 6105–6114. PMLR
32. Tan M, Pang R, Le QV (2020) Efficientdet: scalable and efficient object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10781–10790
33. Ting KM (2000) A comparative study of cost-sensitive boosting algorithms. In: Proceedings of the 17th international conference on machine learning. Citeseer
34. Uijlings JR, Van De Sande KE, Gevers T, Smeulders AW (2013) Selective search for object recognition. Int J Comput Vis 104(2):154–171
35. Wang S, Liu W, Wu J, Cao L, Meng Q, Kennedy PJ (2016) Training deep neural networks on imbalanced data sets. In: 2016 International joint conference on neural networks (IJCNN). IEEE, pp 4368–4374
36. Wang Y-X, Ramanan D, Hebert M (2017) Learning to model the tail. In: Advances in neural information processing systems, pp 7029–7039. Accessed 14 March 2022
37. Wu Y, Kirillov A, Massa F, Lo W-Y, Girshick R (2019) Detectron2 https://github.com/facebookresearch/detectron2
38. Wu J, Song L, Wang T, Zhang Q, Yuan J (2020) Forest r-cnn: large-vocabulary long-tailed object detection and instance segmentation. In: Proceedings of the 28th ACM international conference on multimedia, pp 1570–1578
39. Xu Z, Li B, Yuan Y, Dang A (2020) Beta r-cnn: looking into pedestrian detection from another perspective. Advances in Neural Information Processing Systems
40. Yang B, Yan J, Lei Z, Li SZ (2016) Craft objects from images. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6043–6051
41. Yang H, Wu H, Chen H (2019) Detecting 11k classes: large scale object detection without fine-grained bounding boxes. In: Proceedings of the IEEE international conference on computer vision, pp 9805–9813
42. Yu F, Wang D, Shelhamer E, Darrell T (2018) Deep layer aggregation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2403–2412
43. Zhou Z-H, Liu X-Y (2005) Training cost-sensitive neural networks with methods addressing the class imbalance problem. IEEE Trans Knowl Data Eng 18( 1):63–77
44. Zhou Z-H, Liu X-Y (2010) On multi-class cost-sensitive learning. Comput Intell 26( 3):232–257
45. Zitnick CL, Dollár P (2014) Edge boxes: locating object proposals from edges. In: European conference on computer vision. Springer, pp 391–405