1st Place Solutions for OpenImage2019 - Object Detection and Instance Segmentation

Yu Liu[1*] Guanglu Song[2*] Yuhang Zang[3] Yan Gao[4] Enze Xie[5] Junjie Yan[2]
Chen Change Loy[3] Xiaogang Wang[1]

[1]Multimedia Laboratory, The Chinese University of Hong Kong
[2]X-Lab, SenseTime Research
[3]Nanyang Technological University
[4]University of Chinese Academy of Sciences
[5]Hong Kong University

yuliu@ee.cuhk.edu.hk songguanglu@sensetime.com

Abstract

This article introduces the solutions of the two champion teams, 'MMfruit' for the detection track and 'MMfruitSeg' for the segmentation track, in OpenImage Challenge 2019.

It is commonly known that for an object detector, the shared feature at the end of the backbone is not appropriate for both classification and regression, which greatly limits the performance of both single stage detector and Faster RCNN [24] based detector. In this competition, we observe that even with a shared feature, different locations in one object has completely inconsistent performances for the two tasks. E.g. the features of salient locations are usually good for classification, while those around the object edge are good for regression. Inspired by this, we propose the Decoupling Head (DH) to disentangle the object classification and regression via the self-learned optimal feature extraction, which leads to a great improvement. Furthermore, we adjust the soft-NMS algorithm to adj-NMS to obtain stable performance improvement. Finally, a well-designed ensemble strategy via voting the bounding box location and confidence is proposed. We will also introduce several training/inferencing strategies and a bag of tricks that give minor improvement. Given those masses of details, we train and aggregate 28 global models with various backbones, heads and 3+2 expert models, and achieves the 1st place on the OpenImage 2019 Object Detection Challenge on the both public and private leadboards. Given such good instance bounding box, we further design a simple instance-level semantic segmentation pipeline and achieve the 1st place on the segmentation challenge.

1. Datasets

We used OpenImages Challenge 2019 Object Detection dataset [11] as the training data for most of cases. The dataset is a subset of OpenImages V5 dataset [14]. It contains 1.74M images, 14.6M bounding boxes, and 500 categories consisting of five different levels. Since the categories at different levels have a parent-children relationship, we expand the parent class for each bounding box in the inference stage. The whole OpenImageV5 with image-level label and segmentation label is used in weakly-supervised pretraining and label augmentation as mentioned in Sec. 5.9. We also use the COCO [19] and Object365 [1] to train some expert models for the overlapped categories.

2. Decoupling Head

Since the breakthrough of object detection performance has been achieved by seminal R-CNN families [10, 9, 24] and powerful FPN [18], the subsequent performance enhancement of this task seems to be hindered by some concealed bottlenecks. Even the advanced algorithms bolstered by AutoML [8, 28] have been delved, the performance gain is still limited to an easily accessible improvement range. As the most obvious distinction from the generic object classification task, the specialized sibling head for both classification and localization comes into focus and is widely used in most of advanced detectors including single stage family [20, 25, 12], two-stage family [6, 16, 30, 21, 17] and anchor free family [15]. Considering the two different tasks share almost the same parameters, a few works become conscious about the conflict between the two object functions in the sibling head and try to find a trade-off way.

[*] Equal Contribution

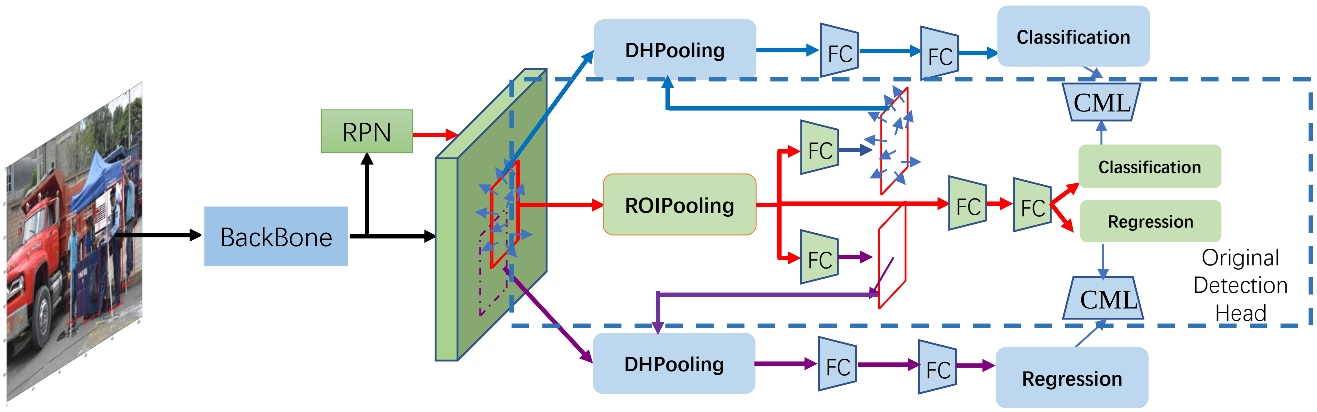

Figure 1. Pipeline of the Faster RCNN with DH. The whole pipeline are trained in an end-to-end manner.

IoU-Net [13] is the first to reveal this problem. They find the feature which generates a good classification score always predicts a coarse bounding box. To handle this problem, they first introduce an extra head to predict the IoU as the localization confidence, and then aggregate the localization confidence and the classification confidence together to be the final classification score. This approach does reduce the misalignment problem but in a compromise manner – the essential philosophy behind it is relatively raising the confidence score of a tight bounding box and reduce the score of a bad one. The misalignment still exists in each spatial point. Along with this direction, Double-Head R-CNN [27] is proposed to disentangle the sibling head into two specific branches for classification and localization, respectively. Despite of elaborate design of each branch, it can be deemed to disentangle the information by adding a new branch, essentially reduce the shared parameters of the two tasks. Although the satisfactory performance can be obtained by this detection head disentanglement, conflict between the two tasks still remain since the features fed into the two branches are produced by ROI Pooling from the same proposal.

In our work, we observation that the spatial misalignment between the two object functions in the sibling head can considerably hurt the training process, but this misalignment can be resolved by a very simple operator called Decoupling Head (DH), alternatively called TSD in [26]. Considering the classification and regression, DH decouples them from the spatial dimension by generating two disentangled proposals for them, which are estimated by the shared proposal. This is inspired by the natural insight that for one instance, the features in some salient area may have rich information for classification while these around the boundary may be good at bounding box regression. We give the brief description for the proposed Decoupling Head (DH) in this section. More details will be introduced in a detached paper TSD [26]. Extensive experiments demonstrate the advantages of DH compared with the original detection head in Faster RCNN.

2.1. Detail description

As shown in 1, different from the original detection head in Faster RCNN, in DH, we separate the classification and regression by auto-learned pixel-wised offset and global offset. The purpose of DH is to search the optimal feature extraction for classification and regression, respectively. Furthermore, to facilitate the learning of DH, we propose the Controllable Margin Loss (CML) to propel the whole learning.

Define the F as the output feature of the ROI pooling, the learned offsets for classification and regression are generated by:

$$C = \mathcal{F}_c(F; \theta_c) \tag{1}$$

$$R = \mathcal{F}_r(F; \theta_r) \tag{2}$$

where θ_c and θ_r are the parameters in fully connected layers $\mathcal{F}_c$ and $\mathcal{F}_r$. There are $C \in \mathbb{R}^{k\times k\times 2}$ and $R \in \mathbb{R}^{1\times 1\times 2}$ where k is the number of bins in ROI pooling.

Classification. For classification, the output feature of DHPooling is defined as:

$$\mathcal{C}(i,j) = \sum_{0\leq i,j\leq k} X(p_0 + p_{i,j} + C_{i,j,*})/n_{i,j} \tag{3}$$

where X is the input feature map and $n_{i,j}$ is the number of pixels in (i,j)-bin pre-defined in ROI pooling. The top-left corner is denoted as p_0.

Regression. For regression, the output feature of DHPooling is defined as:

$$\mathcal{R}(i,j) = \sum_{0\leq i,j\leq k} X(p_0 + p_{i,j} + R_{0,0,*})/n_{i,j} \tag{4}$$

To facilitate the training, we propose CML to optimize the learning. For classification, the CML is defined as:

$$L_c = |S_o - S + m_c|_+ \tag{5}$$

where S_o is the classification score in the original detection head and S is the classification score in DH. $|\cdot|_+$ is same as ReLU function. We use m_c to represent the pre-defined margin. Similarly, for regression the CML is written as:

$$L_r = |IoU_o - IoU + m_r|_+ \tag{6}$$

where IoU_o and IoU are the IoU of the refined proposal according to the predicted regression in original detection head and DH, respectively. m_o and m_r are set to 0.2 in our experiments.

More details and analysis will be presented on an independent article.

3. Adj-NMS

In the post-processing stage of object detection, NMS or soft-NMS is commonly used to filter the invalid bounding boxes. However, in our experiments, we find that a direct use of soft-NMS will degrade the performance. In order to better improve the performance, we adopt the Adj-NMS to incorporate the NMS and soft-NMS better. Given the detected bounding boxes, we preliminarily filter the boxes via the NMS operator with the threshold 0.5. And then, we adopt the soft-NMS operator to re-weight the scores of the other boxes by:

$$w = e^{-\frac{IoU^2}{\sigma}} \tag{7}$$

where w is the weight to multiply the classification score and σ is set to 0.5.

4. Model Ensemble

4.1. Naive Ensemble

For model ensemble, we adopt the solution in PFDet [2] and the commonly used voting strategy where the bounding box location and confidence are voted by the top k boxes. Given the bounding boxes $\mathcal{P}$ and the top k boxes P_i (i∈[1,k]) with higher IoU, we first using the method in PFDet to reweight the classification score for each model via the *map* in validation set. And then, the final classification score S of $\mathcal{P}$ is computed as:

$$C = S_{\mathcal{P}} + 0.05 * \sum_{i=1}^{k} S_{P_i} \tag{8}$$

The localization B is computed as:

$$B = 0.7 * B_{\mathcal{P}} + \frac{0.3}{k} * \sum_{i=1}^{k} B_{P_i} \tag{9}$$

where k is set to 4 in our experiments.

4.2. Auto Ensemble

We trained totally 28 models of different architectures, heads, data splits, class sampling strategies, augmentation strategies and supervisions. We first use the naive model ensemble mentioned above to aggregate detectors with similar settings, which reduces the detections from 28 to 11. Then we design and launch an auto ensemble method to merge them into 1.

Search space. Considering each detection as a leaf node and each ensemble operator as an parent node. The model ensemble can be formulated as a binary tree generation process. All the parent nodes are an aggregation of their children by a set of operations and the root will be the final detection. The search space includes the weight of detection score (a global scale factor for all the classes), box merging score, element dropout (only use the classification score or bounding box information of a model) and NMS type (naive NMS, soft-NMS and adj-NMS).

Search process. In the competition we adopt a two-stage searching process: first, we search the architecture of the binary tree with equal contribution for each child node; then we search the operators of parent nodes based on the fixed tree.

Result. Since such a large search space may lead to overfitting, we split the whole dataset (V5 train+val+test+challenge val) in to three parts, 80% for training and 2×10% as validation sets for tuning the ensemble strategy. The validation sets are elaborately mined to keep its distribution as similar to the whole dataset as possible. We only train the models ID 17-28 under this data setting. The autoEnsemble leads to 2.9%, 3.6% and 1.2%, 1.0% improvement on the two validation sets and ∼0.9% on the public lead-board compared to the Naive ensemble. We also observe an interesting result in the first stage: the detections with lower mAPs tend to locate at deeper leafs. We will provide an enhanced one-stage searching method and more details in an independent article.

5. Bag of Tricks for Detector

5.1. Sampling

OpenImages dataset [14] exhibits the long-tail distribution characteristics: the number of categories is not balanced, and some categories of data are scarce.

Data re-sampling, such as class-aware sampling mentioned in [22, 7] is a widely used technique to handle the class imbalance problem. For each category, the images are sampled such that the probability of having at least one category instance in 500 categories is equal. Table 1 shows the effectiveness of this sampling method. We use class-aware sampling in all the below-mentioned methods.

Note that before the ensemble, we first re-weight the box score of each class by the relative AP value as mentioned in Sec. 4.1

Method	Validation mAP
Baseline (X50 FPN)	58.88
+ Class Aware Sampling	64.64

Table 1. The effectiveness of the class aware sampling strategy.

5.2. Decoupling Backbone

For models ID 25-28, we decouple the classification and regression from the stride 8 in the backbone. One branch focuses on the classification task where regression is given a lower weight and the other branch is the opposite.

5.3. Elaborate Augmentation

For models trained with 512 accelerators, we design a 'full class batch' and a elaborate augmentation. For the 'full class batch', we guarantee that there are at least one sample for each class. For the elaborate augmentation, we first randomly select a class and obtain one image containing it. And then, we apply the random rotation on this image (larger rotated varience for class with severely unbalanced aspect ratio such as 'flashlight'). Furthermore, we randomly select a scale to crop the image covering the bounding box of this class. For the trick of selecting the scale, we first generate the maximum image area s_{mem_max} which is constrained by the memory of accelerator, and then, we randomly sample a scale from the minimum scale s_{stat_min} to $s_{max} = min(s_{mem\_max}, s_{stat\_max})$. The scale sampling obey the distribution of the ratio that longer side of a bbox divided by the long side of its image among the whole training set.

5.4. Expert Model

An expert model means that a detector trained on a subset of the dataset to predict a subset of categories. The motivation is that a general model is hard to perform well in all classes, so we need to select some categories for expert models to handle specifically.

There are two important factors to consider: the selection of positive and negative categories, and the ratio between the positive and negative categories. Previous papers [2] used predefined rules, such as selecting the least number or the worst-performing category in the validation set. The drawback of these predefined rules is that: it ignores the possibility of confusion between categories. E.g. ''Ski'' and ''Snowboard'' are an easy-to-confuse category pair. If we only choose ''Ski'' data to train an expert model, it is easy to treat the ''Snowboard'' in the validation set as ''Ski'', causing false-positive cases.

The definition of ''easy to confuse'' can be derived from three different perspectives:

a) **Hierarchy tag**: OpenImages dataset [14] has hierarchy-tag relationships between different categories. A straightforward method is to select sub-classes under the same parent node to train the expert model.

b) **Confusion matrix**: If the two categories are easily confused, they will cause many false-positives as reflected in the confusion matrix.

c) **Visual similarity**: The weight of the neural network can also be used to measure the distance between the two classes. [29] calculated the Euclidean distance of the features extracted by the last layer of ResNet-101 to define the visual similarity. We go further and consider the weights of the classification Fully Connected layer in the RCNN stage. The cosine angle between different categories are defined as:

$$\cos\Theta = \frac{v_1 \cdot v_2}{\|v_1\|\,\|v_2\|} \tag{10}$$

We verify that if the semantics of the two categories are similar, then the corresponding cosine angle is also close to 1.

We train our expert model as following three steps:

1) Select the initial category C_{pos}, such as the lowest ten categories of validation mAP. Add images containing C_{pos} to the positive data subset χ_{pos}.

2) Add the confused categories by using the cosine matrix. For each category c_i who satisfy the requirement that $dist(c_i, c_j) > thr, c_j \subseteq C_{pos}$, adding them to C_{neg}. thr equals 0.25 in our setting to ensure the ratio of positive and negative data is close to 1:3. Add images containing C_{neg} to the negative data subset χ_{neg}.

3) Train a detector with the $\chi_{pos+neg}$ to predict C_{pos} categories .

During the inference stage, each RoI will have a corresponding classification score with the shape of $(C_{pos} + 1)$. If the background classification score is larger than all other foreground scores, then this RoI will not be sent to the bounding box regression step. This modification can reduce a lot of unnecessary false-positive cases.

5.5. Anchor Selecting

We use k-means clustering to select the anchor for RPN[24], in our model, we have 18 anchor(ratio:0.1, 0.5, 1, 2, 4, 8. scale:8, 11, 14) per position for each FPN level.

5.6. Cascade RCNN

Cascade RCNN[4] is designed for high quality object detection and can improve AP at high IOU thresholds,eg AP0.75. However, in this competition, the evaluation criterion only considers AP0.5, so we modified the IOU threshold for each RCNN level in Cascade-RCNN and redesigned the weight of each stage for the final result. We set the IOU thresholds to 0.5,0.5,0.6,0.7, and set weight of each stage to 0.75,1,0.25,0.25. It offers an increase of 0.7 mAP compared to the standard Cascade RCNN.

5.7. Weakly Supervised Training

There is a serious class imbalance issue in the OpenImage object detection dataset. Some classes only have a few images, which cause the model to perform poorly on these classes. We add some images which only have image-level annotations to improve the classification ability of our model. Specifically, We combine data with bounding-box level annotations and image classification level annotations to build a semi-supervised dataset and integrate a fully-supervised detector(Faster-RCNN[24]) and a weakly-supervised detector(WSDDN[3]) in an end-to-end manner. When encountering bounding-box level data, we use it to train the fully-supervised detector and constrain the weakly supervisory detector. when encountering image classification level data, we use it to train weakly supervised detector, and mine pseudo ground-truth from weakly-supervised results to train the fully supervised detector.

5.8. Relationships Between Categories

There are some special relationships between categories in the OpenImage dataset. For example, some classes always appear along with other classes, like Person and Guitar. In the training set, Person appears in 90.7% of the images which have a guitar. So when detected a bounding box of guitar with high confidence and there is a bounding-box of person with a certain confidence, we can improve the confidence the bounding-box of person. We denote the number of objects of category i in the training set as C_i. The number of objects of category i co-occurring with category j as C_{ij}. We can get the conditional probability $p(i|j) = C_{ij}/C_i$. We assume that the max confidence over all proposals of category i in a image I should greater than the highest conditional probability, i.e. $\max_b(p(i|proposal_b, I)) \geq \max_j(p(j) * p(i|j))$.

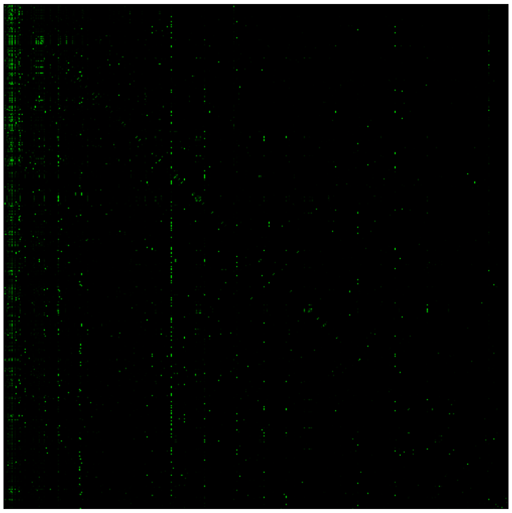

Figure 2. The co-occurrence conditional probability matrix.

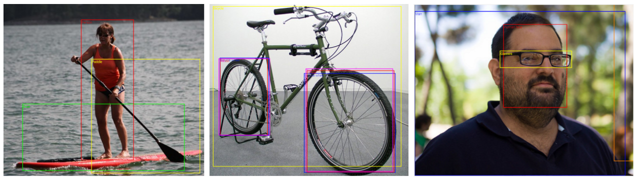

Figure 3. The paddle and the boat always appear at the same time(left). There are always wheels inside the boundingbox of the bicycle(middle), and the bounding box of the glasses is always surrounded by the bouding box of the person(right).

In addition to the co-occurrence relationship, there are two special relationships, surround relationship and being surrounded relationship, as shown in the Fig.3. Surround relationships mean that bounding boxes of certain categories always surround bounding box of certain other categories. Being surrounded relationships mean that certain categories always appear inside the bounding box of certain other categories.

These special relationships between categories can be evidence to improve or reduce the confidence of certain bounding boxes, thereby improving detection performance.

5.9. Data Understanding

Confusing classes. We find there are many confusing class definitions in OpenImage and some of them can be used to improve the accuracy. Such as 'torch has various semantic meanings in train and validation, which is out of algorithms ability. So we expand the training samples of these confusing classes by both mixing some similar classes and using extra images with only image-level label. Here are some more examples: 'torch' and 'flashlight', 'sword' and 'dagger', paper tower' and toilet paper, slow cooker and pressure cooker, kitchen knife and knife.

Insufficient label. We also find some classes like 'grape' has too many group boxes and few instance boxex, so we use the bounding box of its segmentation label to extend the detection label. For some other classes such as 'pressure cooker' and 'touch', we crawling the top-100 results from google image and directly feed them into the training pipeline without hand labelling. A good property of these 200 crawled images is their backgrounds are pure enough so we directly use [0,0,1,1] as their bounding boxes.

6. Implementation Details

The 28 final models are trained by PyTorch [23] and Tensorflow and all of the backbones are first pre-trained on ImageNet dataset. All of the models are trained under different settings: 13/26 epochs with batch size 2N @ N accelerators, where 'N's are in range of [32, 512] for different models based on the available number of accelerators. We warm up the learning rate from 0.001 to $0.004 \times N$ and

Model	DH	DCN	Validation Set	MT	PA	Public LB
ResNet50			64.64			49.79
ResNet50	✓		68.18			52.55
ResNet50	✓		68.18	✓		**55.88**
ResNext101		✓	68.7	✓	✓	55.046
ResNext101	✓	✓	71.71	✓	✓	**58.596**
SENet154		✓	71.13	✓	✓	57.771
SENet154	✓	✓	72.19	✓	✓	**60.5**

Table 2. Ablation studies on DH with different backbones. DCN and MT mean the deformable convnet [6] and multi-scale testing. PA indicates averaging the parameters of epoch [9,13].

Model	PFDet	Adj-NMS	Public LB
4 models			57.994
4 models	✓		59.4
4 models	✓	✓	60.351

Table 3. Ablation studies on PFDet and Adj-NMS. 4 models contain [ResNext101 with DCN, ResNext152 with DCN, SENet154 with DCN, SENet154 with DCN and Cascade RCNN] and all of the models adopt the basic configuration.

then decay it by 0.1 at epoch 9 and 11 (or 18 and 22 for the 2x setting). At the inference stage, for validation set, we straightforwardly generate the result and for challenge test, we adopt the multi-scale test with [600, 800, 1000, 1333, 1666, 2000] and the final parameters are generated by averaging the parameters of epoch [9,13] (or [19,26] for the 2x setting). The basic detection framework is FPN [18] with Faster RCNN and the class-aware sampling is used for them.

7. Results of Object Detection

7.1. Ablation Study on DH

We first study the effectiveness of DH on the validation set and challenge set with different backbones. Results are shown in Tab 2. For model ResNet50, we adopt the anchors with scale 8 and aspect ratio [$\frac{1}{2}$, 1, 2]. For model ResNext101 and SENet154, we adopt the anchors with scale [8,11,14] and aspect ratio [0.1, $\frac{1}{2}$, 1, 2, 4, 8]. Note that DH can always stably improve the performance by $3\sim4\%$.

7.2. Ablation Study on Adj-NMS and Voting Ensemble

Results are shown in Tab 3. Voting ensemble can obtain the $\sim$0.3 improvement. Note that the ensemble solution in PFDet cooperated with Adj-NMS can bring further improvement. The 4 models are trained with simple configuration without bells and whistles.

Model	Public LB
Single Model (ID 1-16)	[58.596 - 60.5]
Single Model (ID 17-28) + 1 expert	[N/A - 63.596]
Naive ensemble ID 1-16	61.917
Mix ensemble+voting ID1-28+3experts (V1)	67.2
V1+COCO+Object365	68.0
Final re-weighting	**68.174**

Table 4. Overview of the submissions.The final submision are generated with the models trained with full classes and the models trained with the specific models.

7.3. Final results

Given all the successful exploration, we train multiple backbones with the best setting and design as mentioned above, including: ResNet family, SENet family, ResNeXt family, NASNet, NAS-FPN and EfficientNet family. We conclude some of our recorded results and break down the final results we achieved on the public lead-board as in Tab. 4. $3experts$ mean the SEResNet154 trained with 150, 27, 40 classes with low AP on validation set. $COCO$ means that we find total 64 classes co-exists in COCO dataset and OpenImage dataset. And so, we straightforwardly adopt the Mask RCNN with ResNet152 and Cascade RCNN with ResNet50 as the 64-classes expert model which are strained on COCO dataset. $Object365$ means we trained the expert class model with embedding the same classes in Object365 and there are total 8 expert models for this. At the final re-weighting stage, we generate different weights for different models to ensemble.

8. Instance Segmentation

We observe that most state-of-the-art instance segmentation methods are based on Mask R-CNN in recent two years, in which the box regression and mask segmentation share almost the same features. We argue that this paradigm may not be the optimal solution. Instead, we revisit the traditional object detection and semantic segmentation and formulate instance segmentation problem as these two serial problems. In this way, the performance of segmentation can be further boost up due to more complex design of segmentation branch. In the bounding box detection step, we directly use the results from detection track. In the semantic segmentation step, we adopt state-of-the-art HRNet as our backbone to train network distinguish pixels from fore/background in one bounding box.

the original NASNet can not converge well in our experiments, here we use a modified version of it.

We modified some network parameters such as the depth multiplier to enable better convergency and training time-performance trade-off.

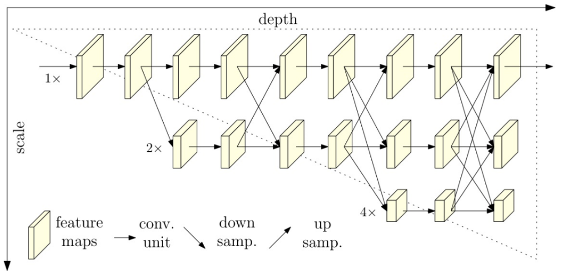

Figure 4. The basic architecture of HRNet.

8.1. Detection

Based on our design mentioned above, the detection result has linear impact to the segmentation. So we takes most of our time in improving the performance of detection.

Base detection. As mentioned in our detection report, we train 28 global models, 3 expert models for the 150, 27 and 40 classes with the lowest AP on validation set, 2 expert models on COCO and 8 expert models on Object365 for the overlapped classes. The backbones include the ResNet family, SENet family, ResNeXt family, NASNet, NAS-FPN and EfficientNet family. We design a decoupling head, decoupling backbone, adj-NMS, multiple ensemble strategies and bag of tricks (sampling, elaborate augmentation, expert model strategies, anchor selection, modified cascade RCNN, weakly supervised pre-training, categories relationship based re-scoring, etc.)

Refined detection. Since the evaluation metric of the detection track is AP0.5, our detection methods may overfit it and generate boxes not so tight. So we further train an cascade regressor head. Given the original detectors' results as the proposals, we only train an regressor based on the feature generated by ROI align.

8.2. Segmentation Module

8.2.1 HRNet

We give the detail description for the proposed Semantic Segmentation method. Due to the excellent and stable performance of the HRNet backbone on several tasks (e.g. image classification, object detection, semantic segmentation and keypoint), we choose HRNet as our backbone.

8.2.2 Architecture

The architecture is illustrated in Figure 4. There are four stages, and the 2nd, 3rd and 4th stages are formed by repeating modularized multi-resolution blocks. A multi-resolution block consists of a multi-resolution group convolution and a multi-resolution convolution. The multi-resolution group convolution is a simple extension of the group convolution, which divides the input channels into several subsets of channels and performs a regular convolution over each subset over different spatial resolutions separately.

Model	Training strategy	Test strategy	Public LB
HRNet[1]	[{200,300},600]	[250,550]	49.02
HRNet[2]	[112,112]	[112,112]	53.02
HRNet[2]	[256,600]	[256,600]	53.30
HRNet[2]	[256,600]	[256,600] with flip	54.90
Ensemble	-	-	55.39

Table 5. The results of different models and the ensemble submission. [1]Sampling 400k bboxes + ∼62 mAP Det, [2]All bboxes + ∼67 mAP Det

As shown in Fig. 5, in the final step, we rescale the low-resolution representations through bilinear upsampling to the high resolution, and concatenate the subsets of representations, resulting in the high-resolution representation, which we adopt for semantic segmentation.

8.2.3 Detail description

The network starts from a stem that consists of two strided 3×3 convolutions decreasing the resolution to 1/4. The 1st stage contains 4 residual units where each unit is formed by a bottleneck with the width 64, and is followed by one 3×3 convolution reducing the width of feature maps to C. The 2nd, 3rd, 4th stages contain 1, 4, 3 multi-resolution blocks, respectively. The widths (number of channels) of the convolutions of the four resolutions are C, 2C, 4C, and 8C, respectively. Each branch in the multiresolution group convolution contains 4 residual units and each unit contains two 3×3 convolutions in each resolution. In semantic segmentation , we mix the output representations from all the four resolutions through a 1×1 convolution, and produce a 15C-dimensional representation. Then, we pass the mixed representation at each position to a linear classifier with the softmax loss to predict the segmentation maps. Note that the segmentation maps are upsampled (4 times) to the input size by bilinear upsampling for both training and testing.

8.3. Implement details for segmentation track

We follow the similar training protocol [31, 32]. The data are cropped from original image by detected bounding box, then the short side are random scaled from 200 to 300, and random horizontal flipping. We use the similar rotation augmentation settings in our detection solution. We adopt the SGD optimizer with the base learning rate of 0.08, the momentum of 0.9 and the weight decay of 0.0005. The poly learning rate policy with the power of 0.9 is used for dropping the learning rate. All the models are trained for 200K iterations with the batch size of 6N, 8N and 16N on N GPUs for different input resolution. N is set to 8, 32 or 80 in different experiments. SyncBN is used.

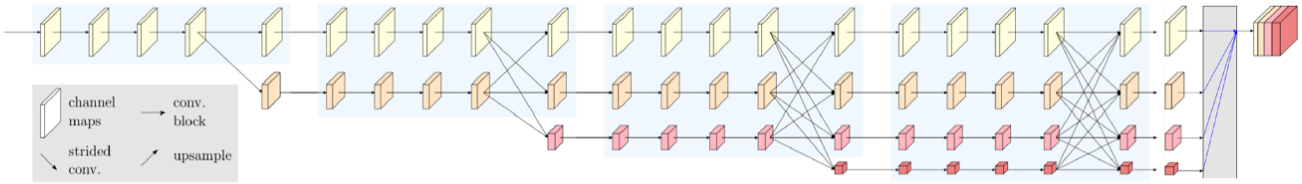

Figure 5. The whole pipeline. We rescale and align the 4 representations into 1 at the end as in HRNetv2.

8.4. Model Ensemble

For model ensemble, we apply different training strategies to obtain different models with backbone HRNet. There are 4 models used for ensemble, including: HRNet trained with random scale from 200 to 300, HRNet trained with fixed scale [256,600], HRNet trained with fixed scale [112,112], HRNet trained with [256,600] and test with flip mechanism. The bounding boxes are generated by the detection model trained on the 500 classes OpenImage dataset. The map on public LB is 68%. At the ensemble stage, given the mask map generated by different models, we directly adopt the voting mechanism for each pixel to obtain the final results.

8.5. Final Results of Segmentation

The results of different training strategy are show in Tab 5.

9. Acknowledgement

We appreciate the discussion with Kai Chen and Yi Zhang at the Multimedia Lab, CUHK. We also acknowledge the `mmdetection`[5] team for the wonderful codebase.

References

[1] *Object365*. https://www.objects365.org/overview.html. 1

[2] Takuya Akiba, Tommi Kerola, Yusuke Niitani, Toru Ogawa, Shotaro Sano, and Shuji Suzuki. Pfdet: 2nd place solution to open images challenge 2018 object detection track. *arXiv preprint arXiv:1809.00778*, 2018. 3, 4

[3] Hakan Bilen and Andrea Vedaldi. Weakly supervised deep detection networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2846–2854, 2016. 5

[4] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. 4

[5] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 8

[6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 1, 6

[7] Yuan Gao, Xingyuan Bu, Yang Hu, Hui Shen, Ti Bai, Xubin Li, and Shilei Wen. Solution for large-scale hierarchical object detection datasets with incomplete annotation and data imbalance. *arXiv preprint arXiv:1810.06208*, 2018. 3

[8] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2019. 1

[9] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1

[10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015. 1

[11] Google. Open images 2019 - object detection challenge, 2019. 1

[12] Zekun Hao, Yu Liu, Hongwei Qin, Junjie Yan, Xiu Li, and Xiaolin Hu. Scale-aware face detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6186–6195, 2017. 1

[13] Borui Jiang, Ruixuan Luo, Jiayuan Mao, Tete Xiao, and Yuning Jiang. Acquisition of localization confidence for accurate object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–799, 2018. 1

[14] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018. 1, 3, 4

[15] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 734–750, 2018. 1

[16] Buyu Li, Yu Liu, and Xiaogang Wang. Gradient harmonized single-stage detector. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8577–8584, 2019. 1

[17] Hongyang Li, Yu Liu, Wanli Ouyang, and Xiaogang Wang. Zoom out-and-in network with map attention decision for region proposal and object detection. *International Journal of Computer Vision*, 127(3):225–238, 2019. 1

[18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1, 6

[19] 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 *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 1

[21] Yu Liu, Hongyang Li, Junjie Yan, Fangyin Wei, Xiaogang Wang, and Xiaoou Tang. Recurrent scale approximation for object detection in cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 571–579, 2017. 1

[22] Wanli Ouyang, Xiaogang Wang, Cong Zhang, and Xiaokang Yang. Factors in finetuning deep model for object detection with long-tail distribution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 864–873, 2016. 3

[23] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 5

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 1, 4, 5

[25] Guanglu Song, Yu Liu, Ming Jiang, Yujie Wang, Junjie Yan, and Biao Leng. Beyond trade-off: Accelerate fcn-based face detector with higher accuracy. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7756–7764, 2018. 1

[26] Guanglu Song, Yu Liu, and Xiaogang Wang. Revisiting the sibling head in object detector. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2020. 2

[27] Yue Wu, Yinpeng Chen, Lu Yuan, Zicheng Liu, Lijuan Wang, Hongzhi Li, and Yun Fu. Rethinking classification and localization in r-cnn. *arXiv preprint arXiv:1904.06493*, 2019. 2

[28] Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019. 1

[29] Hao Yang, Hao Wu, and Hao Chen. Detecting 11k classes: Large scale object detection without fine-grained bounding boxes. *arXiv preprint arXiv:1908.05217*, 2019. 4

[30] Xingyu Zeng, Wanli Ouyang, Junjie Yan, Hongsheng Li, Tong Xiao, Kun Wang, Yu Liu, Yucong Zhou, Bin Yang, Zhe Wang, et al. Crafting gbd-net for object detection. *IEEE transactions on pattern analysis and machine intelligence*, 40(9):2109–2123, 2017. 1

[31] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 7

[32] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 267–283, 2018. 7