4th Place Solution to Open Images 2019 - Instance Segmentation

Takuya Ito
Universal Knowledge Inc.
tito@universal-knowledge.jp

Abstract

This article describes the model that achieved 4th place in the Open Images 2019 - Instance Segmentation Challenge on Kaggle.

1. Models

I used 18 models for ensemble. But I just made 3 models for this competition, and other 15 models are made for other competitions or open pre-trained models. I used not only segmentation model but also object detection model for ensemble, and 10 models out of 18 models were object detection model.

1.1. model1: Base Model

I made Hybrid Task Cascade[1] model with ResNeXt101-FPN backbone using mmdetection[2] as a base segmentation model. This is SOTA for instance segmentation. I used almost same configuration as original setting [3]. Only difference is total_epochs and img_scale. First I made a model with total_epochs was 12 and img_scale was (1024, 768). Then using this model as initial weights, I trained another 12 epochs with img_scale [(1600, 400), (1600, 1400)]. I used same method as Miras Amir's winning solution of iMaterialist (Fashion) 2019 [4] for TTA and ensemble for these 2 models.

Private and public LB scores of this model was 0.4832, 0.5264 respectively.

1.2. model2: Additional Segmentation Models

I made 2 additional segmentation models for ensemble.

1.2.1 model2-1: Hrnet

I made cascade mask rcnn hrnet[5] uinsg mmdetection[2] for ensemble. Configuration of this model is almost same as original setting [6]. Only difference is img_scale was (1024, 768).

Private and public LB scores of this model were 0.4345 and 0.4663 respectively.

1.2.2 model2-2: Expert Model

This is a expert model for 150 classes which have fewer GT images. Architecture and configuration are almost same as model1. Only differences are total_epochs=12 and img_scale=(1024, 768) and num_classes=150.

1.3. model3: OID Models

I made 2 models (yolo and retinanet model) for Open Images 2019 - Object Detection Track (OID). I used these models for ensemble.

1.4. model4: VRD Models

I made many experiments for Open Images 2019 - Visual Relationship Track (VRD). I used these results for ensemble too. Eventually, model3 and model4 turned out not to be usefull for ensemble.

1.5. model5: External Models

I used 5 external pre-trained models. Two of them are traindend for COCO and three are trained for openimages V4.

For COCO models, I made 58 class mapping between coco and open images, and used there inference results for ensemble. model5-1 is from X-101-64x4d-FPN [7], and model5-2 is from faster_rcnn_inception_resnet_v2_atrous_coco [8].

For models trained for openimages V4, I could use these inference results directory. model5-3 is from tfhub [9], and model5-4, model5-5 are from ZFTurbo's github [10].

2. Sampling and Data Balancing

It takes very long time to train segmentation model. So I tried to reduce images for training. In order to sampling images, I set threshold(N) for image count of each class. If image count for the class is smaller than N, I kept all images for the class. And if image count is larger than N, then I sampled N image for the class. N was 1K - 5K depending on models. This sampling reduced 850K total training images to 200K images and makes training time faster about 4 times for N=5K sampling.

Table 1. Models

model name	source	architecture	backbone	task	No. of class
model1	trained for this competition	HTC	resnext 101	segmentation	300
model2-1	trained for this competition	cascade rcnn	hrnet	segmentation	300
model2-2	trained for this competition	HTC	resnext 101	segmentation	150
model3-1	trained for OID	yolo	darknet	object detection	300
model3-2	trained for OID	retina	resnext 101	object detection	300
model4-1	trained for VRD	cascade rcnn	resnext 101	object detection	4
model4-2	trained for VRD	HTC	resnext 101	segmentation	7
model4-3	trained for VRD	HTC	resnext 101	segmentation	33
model4-4	trained for VRD	cascade rcnn	resnext 101	object detection	57
model4-5	trained for VRD	cascade rcnn	resnext 101	object detection	57
model4-6	trained for VRD	cascade rcnn	resnext 101	object detection	57
model4-7	trained for VRD	cascade rcnn	hrnet	object detection	57
model4-8	trained for VRD	yolo	darknet	object detection	57
model5-1	external model	HTC	resnext 101	segmentation	58
model5-2	external model	faster_rcnn	nas	segmentation	58
model5-3	external model	faster_rcnn	inception_resnet	segmentation	300
model5-4	external model	retina	ResNet101	object detection	300
model5-5	external model	retina	ResNet152	object detection	300

3. Ensemble

I visually checked some False Positive data, and found that there was no problem with segmentation but classification has issues in most cases.

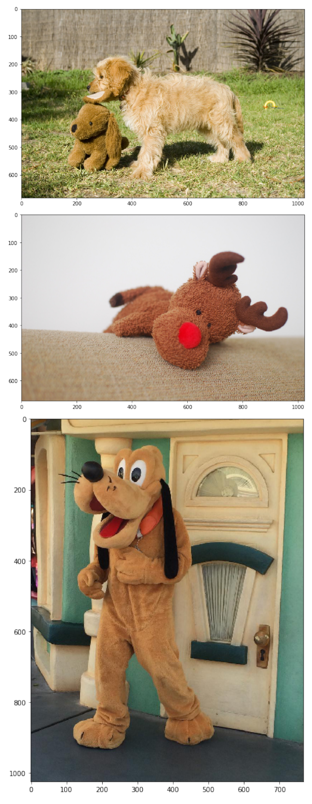
Fig. 1. original images

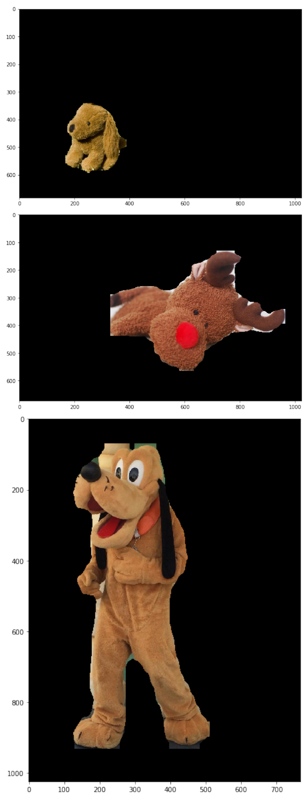
Fig. 2. cropped images

Fig.1 and Fig.2, are some examples of this kind of False Positive. Thinking of IoU threshold of metrics is 0.5, the segmentation is almost perfect. On the other hand, the model predicts them as a teddy bear with very high probability. So I mainly focused on classification probability improvement for ensemble. The procedure is as follows.

ensemble procedure 0: predicting bounding box
Even for segmentation model, bounding box is predicted at the same time.

ensemble procedure 1: grouping predictions
I used Bounding Box to group the predictions in order to use object detection model for ensemble. I grouped predictions with 50% IoU threshold of Bounding Box.

ensemble procedure 2: extracting score and IoU
Scores and IoUs of prediction from each models are extracted for each groups. Where IoU is the calculated with best segmentation model in the group.

ensemble procedure 3-1: weighted average
This is simple weighted average of all 18 models.

$$Score = \sum_{i=1}^{18} Score_i * IoU_i * Weight_i \qquad (1)$$

ensemble procedure 3-2: xgboost
I tried to use xgboost to predict Score. Scores and IoUs, LabelID were used as features, and objective was rank:pairwise.

Table 2 is the results of weighted average and xgboost.

Table 2. Ensemble Results

model	private	public
weighted average	0.5113	0.5477
xgboost	0.5098	0.5500

Xgboost model looked to be improved when I looked at public LB Score, but private LB dropped.

4. Experiments

In order to see the importance of each models, I combined model1 and other model group one by one and submited the prediction to know the private and public LB Scores. External models (model5) had biggest impact and models for VRD (model4) had small impact to LB Scores.

Table 3. Private and Public LB Scores

model	private	public
model1	0.4832	0.5264
model1 + model2	0.4917	0.5335
model1 + model3	0.4929	0.5339
model1 + model4	0.4840	0.5286
model1 + model5	0.5075	0.5450

Table 4 is the results of ablation studies, removing each model group one by one. External models (model5) had biggest impact again. Models for VRD (model4) and OID (model3) had little impact to LB.

Table 4. Ablation Study: Private and Public LB Scores

model	private	public
full	0.5113	0.5477
full - model2	0.5091	0.5455
full - model3	0.5112	0.5476
full - model4	0.5127	0.5493
full - model5	0.4978	0.5412

5. Data and Pre-Trained Networks Used

I did not use external dataset.

I used pre-trained weights for initialization of model1 and model2, model3, model4. Some of these weights are pre-trained on COCO and ImageNet datasets.

And I used pre-trained models for model5. These models are pre-trained on COCO and OpenImagesV4 datasets.

6. Hardware

I used local 1080ti x 2 and titanRTX. And in the very ending of this competition, I used V100 x 8 instance on GCP.

These resources are shared by 3 open image competitions.

References

[1] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, Dahua Lin. Hybrid Task Cascade for Instance Segmentation. arXiv:1901.07518v2

[2] mmdetection
https://github.com/open-mmlab/mmdetection/

[3] mmdetection configuration source code of HTC with ResNeXt101-FPN backbone
https://github.com/open-mmlab/mmdetection/blob/master/configs/htc/htc_dconv_c3-c5_mstrain_400_1400_x101_64x4d_fpn_20e.py

[4] Miras Amir's winning solution of iMaterialist (Fashion) 2019
https://github.com/amirassov/kaggle-imaterialist

[5] Ke Sun, Yang Zhao, Borui Jiang, Tianheng Cheng, Bin Xiao, Dong Liu, Yadong Mu, Xinggang Wang, Wenyu Liu, Jingdong Wang. High-Resolution Representations for Labeling Pixels and Regions. arXiv:1904.04514v1

[6] mmdetection configuration source code of cascade mask rcnn with hrnet backbone
https://github.com/open-mmlab/mmdetection/blob/master/configs/hrnet/cascade_mask_rcnn_hrnetv2p_w32_20e.py

[7] pre trained model: mmdetection: X-101-64x4d-FPN
https://github.com/open-mmlab/mmdetection/blob/master/configs/htc/

[8] pre trained model: tensorflow model zoo: faster rcnn inception resnet v2 atrous coco
https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

[9] pre trained model: tfhub: faster rcn inception resnet v2
https://tfhub.dev/google/faster_rcnn/openimages_v4/inception_resnet_v2/1

[10] pre trained model: ZFTurbo's github
https://github.com/ZFTurbo/Keras-RetinaNet-for-Open-Images-Challenge-2018