Cook warm, then make it cool - 2nd Place Solution to Open Images 2019, Instance Segmentation track

Artur Kuzin
ODS.ai
X5 Retail Group
kuzin.artur@gmail.com

Abstract

This article describes the solution of the 2nd place of the Open Images 2019 - Instance Segmentation Challenge on Kaggle. Key features of the solution: 1) Filtering data and gradually adding more noise annotation. 2) No random initialization of weights. 3) Training with partial freezing of layers. 4) Sampling of images and proposals.

1. Data

I used the original dataset without additional external data. As a validation, I selected 3520 samples from validation set. The rest of the validation data I added to the train part. I believe that left too few samples in validation. Since not all improvements in validation correlated with the leaderboard. In addition, the metric is very dependent on the size of the dataset.

1.1. Filtering

One of the key features of the dataset is a large number of omissions in annotation. To speed up convergence, I used the following heuristic. We leave all samples where:

1. The number of marked objects above count-threshold
2. There is at least one class object that occurs in the dataset below appearance-threshold.

I started training with $count - threshold$ = 8 and $appearance - threshold$ = 500. And finished with $count - threshold$ = 3, $appearance - threshold$ = 1000. If I took a subset with $count - threshold$ = 2, $appearance - threshold$ = 1200, then this worsened the score of validation.

1.2. Class selection

I used 275 leaf classes as target. And trained one models for all classes.The remaining classes was created heuristics from hierarchy.

1.3. Sampling

Each class has the same contribution to the overall speed, so rare classes need to be predicted as well as frequent ones. To implement this, I used two samplers:

1. Image sampler. I implemented the class aware sampler described in the solution of PFDen team for last year Object Detection track [3]
2. The sampler of the proposals. Described in Libra R-CNN paper [4] and implemented in mmdetection framework.

1.4. Augmentation

I used Albumentation [1] library for augmentations. I used for the main part of the training following set:

- RandomBrightnessContrast, p = 0.5
- CLAHE, p = 0.5
- ToGray, p = 0.2
- Cutout, p = 0.9
- JpegCompression, p = 0.4, qualitylower = 70, qualityupper = 99
- RandomRotate90, p = 0.1
- RandomFlip, p = 0.5

During exploration data analyses I noticed several images rotated upside down. Some of the pictures were black and white. Also test set have more JPEG compression than train.

2. Models

I used Hybrid Task Cascade models[2]. For final submission I used COCO pretrained X-101-64x4d-FPN (c3-c5) and X-101-32x4d-FPN models. Both models were

trained on COCO dataset using backbone pretrained on imagenet1k. But my best single model was WSL resnext 101 32x8. To build this model I took backbone convolution layers for WSL imagenet pretrain, and the rest part (neck, head, mask) from X-101-64x4d-FPN (c3-c5) trained on Open Images. For COCO pretrained weights all output layers were rebuild for new 275 classes through concatenations tensors of old weights. So all layers were initialized from pretrained weighs.

3. Train

To speed up the whole training process, I trained the models defrosting parts sequentially. At first I trained with a frozen backbone until complete convergence. Then sequentially unfreeze groups of convolutions of backbone. Along with this, at each stage I added more samples with noisy annotation, gradually changing $count - threshold$ and $appearance - threshold$. After fully training the model, I turned off augmentations that add bias in the data, froze the backbone and trained only FPN, RPN, and head at maximum batch size (on 32Gb V100). At all stages, an input size of 1024x1024 was used. Along with SGD optimizer with default parameters.

4. Predict

I chose the following values for nms: soft-nms, nms-threshold = 0.75, iou-threshold = 0.5, min-score = 0.0001, score-threshold = 0.0001 through grid search of parameters on validation.

Then I found the issue in the mmdet github `https://github.com/open-mmlab/mmdetection/issues/300`, where it was recommended to increase the number of bboxes. I set it like this: nms-pre = 12000, nms-post = 2000, max-num = 2000.

For ensemble I used the implementation of Miras Amir implemetation of HTC ensemble from Kaggle imaterialist competition `https://github.com/amirassov/imaterialist/blob/master/mmdetection/mmdet/models/detectors/ensemble_htc.py`. I took two of the best checkpoints of two model (X-101-64x4d-FPN (c3-c5) and X-101-32x4d-FPN), MirrorTTA and 2 scalesTTA. The final submission had a size of 7+ GB. Therefore, I threw out all the predicts with confidence below 0.001.

5. What I tried, but did not get an improvement on LB.

By validation, I found classes that had lower validation scores than the same classes in the final solution from the Object Detection track. Of these, I selected objects that have a round shape (Grape, Radish, Cabbage, Tennis ball). So I replaced all the segmentation predictions with ovals obtained from the boxes.

Re-calibrate the probabilities based on the probabilities from Object Detection for classes that have a higher validation score.

I tried an additional set of augmentations, but they did not improve the score of validation

- ShiftScaleRotate, p = 0.5, shift-limit = 0.1, scale-limit = 0.1, rotate-limit = 5, border-mode = 0
- HueSaturationValue, p = 0.3
- ChannelShuffle, p = 0.3
- RGBShift, p = 0.3

I believe I had an bugs in the implementation of the rotations augmentation. Color augmentations did not work, since many classes are strongly tied to color (fruits and vegetables).

GCNet, Se-bloc worsened score on validation.

References

[1] Alexander Buslaev, Alex Parinov, Eugene Khvedchenya, Vladimir I. Iglovikov, and Alexandr A. Kalinin. Albumentations: fast and flexible image augmentations. *arXiv e-prints*, page arXiv:1809.06839, Sep 2018.

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv e-prints*, page arXiv:1906.07155, Jun 2019.

[3] 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 e-prints*, page arXiv:1810.06208, Oct 2018.

[4] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra R-CNN: Towards Balanced Learning for Object Detection. *arXiv e-prints*, page arXiv:1904.02701, Apr 2019.