Team PFDet's Methods for Open Images Challenge 2019

Yusuke Niitani* Toru Ogawa* Shuji Suzuki*
Takuya Akiba Tommi Kerola Kohei Ozaki Shotaro Sano
Preferred Networks, Inc.
{niitani,ogawa,ssuzuki,akiba,tommi,ozaki,sano}@preferred.jp

Abstract

We present the instance segmentation and object detection methods used by team PFDet for Open Images Challenge 2019. Massive dataset size, huge class imbalance and sparse annotations are handled. Using this method, team PFDet achieved 3rd and 4th place in the instance segmentation and object detection track, respectively.

1. Introduction

Open Images Detection Dataset V5 (OID) [9] is currently the largest publicly available object detection dataset, including 1.7M annotated images with 12M bounding boxes. The diversity of images in training datasets is the driving force of the generalizability of machine learning models. Successfully trained models on OID would push the frontier of object detectors with the help of data.

Since the number of images in OID is extremely large, the speed of training is critical. For faster training, we use Fast R-CNN [5] instead of the more commonly used Faster R-CNN [16]. Fast R-CNN omits time-consuming online RoI (Region of Interest) generation during training by pre-computing RoIs. We find that the selection of pre-computed RoIs plays an important role in achieving good accuracy. For instance, when the number of pre-computed RoIs is small during training, the network overfits to those RoIs. Faster R-CNN by default achieves a high variation of RoIs during training, so the aforementioned problem is unique to Fast R-CNN.

OID is a federated object detection dataset [6, 9]. This means that for each image, only a subset of categories is annotated. This is in contrast to exhaustively annotated datasets such as COCO[11]. Federated annotation is a realistic approach to expand the number of categories covered by the dataset, since without sparsifying the number of annotated classes, the number of annotations required may explode as the total number of categories increases.

*The starred authors are contributed equally and ordered alphabetically.

When training a detector on an exhaustively annotated dataset like COCO, the loss makes an assumption that no objects is inside an unannotated region. However, for a federated object detection dataset, such assumption may be violated because some regions contain an object of an unverified category. We handled this problem by ignoring loss for unverified categories.

In addition to the previously mentioned uniqueness of OID, the dataset poses an unprecedented class imbalance for an object detection dataset. The instances of the rarest class *Pressure Cooker* are annotated in only 13 images, but the instances of the most common class *Person* are annotated in more than 800k images. The ratio of the occurrence of the most common and the least common class is 183 times larger than in COCO [11]. Typically, this class imbalance can be tackled by over-sampling images containing instances of rare classes. However, this technique may suffer from degraded performance for common classes.

As a practical method to solve class imbalance, we train models exclusively on rare classes and ensemble them with the rest of the models. We find this technique beneficial especially for the first 250 rarest classes, sorted by their occurrence count.

To summarize our major contributions:

- **Fast R-CNN**: We present the effectiveness of Fast R-CNN, and proposed methods to alleviate performance penalty introduced by per-computed RoIs.
- **Using Only Verified Categories**: We found that it is helpful to ignore unverified categories during training.
- **Expert Models**: We present the effectiveness of using expert models, especially for classes that rarely appear in the dataset.

2. Methods

2.1. Model architecture

Two stage object detectors such as Faster R-CNN [16] are known to achieve excellent accuracy, but their GPU us-

age efficiency during training is sub-optimal. This is because RoIs used to train heads are determined between feature extraction and loss calculation. Thus, GPUs are forced to wait for the ground-truth assignment of RoIs. To make efficient use of GPUs, we use Fast R-CNN, which precomputes RoIs and assigns ground-truths to RoIs in parallel to feature extraction.

For the instance segmentation track, we add a mask head [7] that predicts a segmentation of an object given a region around it and its category. The 300 categories for instance segmentation is a subset of the 500 categories for detection. We use all the 500 detection categories even for training the instance segmentation model, where only 300 of these have instance masks. This worked better compared to only using the 300 instance segmentation categories in preliminary experiments.

2.2. Learning with a federated dataset

In federated object detection datasets [9, 6], for each image, categories are grouped into positively verified, negatively verified and unverified. For positively verified categories, annotations are exhaustively made for all objects of those categories. For negatively verified categories, the annotators have made sure that the image contains no objects of those categories. For unverified categories, the objects of those categories may or may not exist in the image.

During training, each RoI is assigned to one of the ground truth boxes if there is any that has sufficiently large enough intersection. This assignment is used to calculate classification loss and localization loss [16]. In this work, the classification loss is calculated as the sum of sigmoid cross entropy loss for each proposal and each category as:

$$\begin{aligned} \mathcal{L}_{cls} &= -\sum_i \sum_c l_{ic} \log p_{ic} \\ l_{ic} &\in \{-1, 0, 1\} \ , \end{aligned} \tag{1}$$

where $l_{ic} = 1$ and $l_{ic} = -1$ when the i-th RoI is assigned or not assigned to category c, respectively. Also, l_{ic} can be set to 0, which means that the classification loss for category c is ignored for the i-th RoI.

When the RoI i is not assigned to any of the ground truth boxes, we set $l_{ic} = -1$ for categories c that are positively and negatively verified and $l_{ic} = 0$ for categories c that are unverified. When the RoI i is assigned to a ground truth bounding box with the category c', we set $l_{ic'} = 1$ and $l_{ic} = -1$ for all negatively verified and positively categories that are not c'. For the unverified categories, $l_{ic} = 0$. In practice, verified categories are expanded based on the category hierarchy.

2.3. Expert models

In OID, there is an extreme class imbalance, which makes it difficult for a model to learn rare classes. For instance, there are 238 classes that are annotated in less than 1000 images, but the most common class *Person* is annotated in 807k images. We use expert models fine-tuned from a model trained with the entire dataset as done in our previous year's submission [2].

We select a subset of categories to which an expert model is trained based on one of the following criteria:

- Occurrence ranking in the detection subset of the dataset. We group categories that are in a neighboring ranking so that sampling imbalance does not occur among the categories in a subset.
- Occurrence ranking in the instance segmentation subset of the dataset.
- Semantic similarity of categories. We cluster categories based on their similarity of embeddings by an imagenet pretrained feature extractor [14].

When training an expert model, the annotation of categories that the expert is not responsible for is dropped. Images that do not contain an annotation of the targeting categories are discarded. Also, the bias term of the classification layer is reinitialized so that the network only outputs targeting categories.

2.4. Ensembling

When aggregating predictions from multiple models, we apply non-maximum suppression to predictions from each model and apply suppression once again to concatenation of the predictions. In the second suppression step, we group predictions that have a large enough intersection and is assigned to the same category. We compute a representative prediction from each group, and the collection of them is the final prediction. Given a group, the bounding box assigned to the group is the bounding box of the most confident prediction in the group. The segmentation is calculated as the average segmentation from predictions in the group weighted by confidence and spatial proximity.

2.5. Pre-computed RoIs

Faster R-CNN computes different set of RoIs depending on the model weights for given image. Thus, high variation of RoIs could be achieved without any effort. This variation is lost when we use Fast R-CNN models with the same set of pre-computed RoIs. This comes at the cost of degraded performance. For training, RoIs that are used to compute head losses are sampled from a pool of RoIs. When training Fast R-CNN, the pool of RoIs is pre-computed, and we find that the number of RoIs in the pool needs to be very high in order for the network not to overfit to the RoIs. In many published works [7], the number of RoIs in the pool for each image is up to 2000. However, we find that this is not large enough, so we prepare up to 16000 RoIs per image.

Selection of pre-computed RoIs is also important when ensembling models. The set of RoIs used by each model should be different when ensembling. This is because predictions made with different set of RoIs complement each other better.

2.6. Post-processing

As stated in the dataset description, the size of an annotated object is larger than 40×80 or 80×40 [1] Thus, any predictions with a small segmentation are unlikely to be counted as true positives during evaluation. We omitted predictions whose area of the segmentation is less than 1600 pixels.

The competition submission file size is limited to 5GB. Our submission file sometimes exceeded this limit, especially after ensembling. To combat the effect of this limit, we found that predictions for some categories occur much more frequently than the rest. We drop predictions of frequently predicted categories to shrink the file size in order to meet the limit.

3. Experiments

We used COCO [11], LVIS [6] and Objects365 as the external data. We use Feature Pyramid Networks [10] for our experiments. The feature extractor is SENet [8]. The initial bias for the final classification layer is set to a large negative number to prevent unstable training in the beginning. We set the initial weight of the base extractor by the weights of an image classification network trained on the ImageNet classification task [4]. We use stochastic gradient descent with the momentum set to 0.9 for optimization. The base learning rate is set to $0.00125 \times$ batchsize. We used up to 120 GPUs. The best single model is trained with the batch size set to 240. We used multi-node batch normalization [15] to make training stable. We trained for 16 epochs. The learning rate is scheduled by a cosine function $\eta = \eta_0 \frac{\cos{(\%\text{ of progress} \times \pi)} + 1}{2}$, where η and η_0 are the learning rate and the initial learning rate. We scale images during training so that the length of the smaller edge is between $[650, 1056]$. Also, we randomly flip images horizontally to augment training data. In addition to that, for training expert models, we used an augmentation policy searched by AutoAugment [3]. During inference, we did not do any test-time augmentation. We used non-maximum suppression with threshold for intersection over union set to 0.5. We use Chainer [17, 1, 13] as our deep learning framework.

3.1. Pre-computed RoIs

To study the best ensembling strategy of Fast R-CNN models, we experimented ensembles of predictions from two models with two sets of RoIs. The result is shown in Table 1. When using the same set of RoIs for the prediction of two models, the performance did not improve from the single model. By using different sets of RoIs, the ensemble outperformed the single model.

Table 1: Comparison of ensembling results with different set of RoIs. The top three rows show the scores with single models with different weights and RoIs. The bottom two rows show the scores of ensembling predictions from two models. The last row shows the result when different set of RoIs are used to make prediction for each of the two models.

Model	RoI	Segmentation val mAP
A	1	70.62
B	1	71.03
B	2	71.02
A, B	1, 1	71.02
A, B	1, 2	**71.78**

Table 2: Ablative study on the number of categories assigned to expert models. The mean average precision of the baseline model is 65.39.

# of categories per expert	# of experts	detection val mAP
50	1	69.19
25	2	70.45
10	5	71.37

3.2. Expert models

Table 2 shows an ablative study of expert models with different number of categories assigned. This is the result when training expert models for the 50th to 99th rarest categories. When training multiple expert models for these categories, the categories are split into disjoint sets. These splits are made based on how frequent the categories appear in OID. For instance, when training two expert models, the first expert model is responsible for the 50th to 74th rarest categories and the second model is responsible for the 75th to 99th rarest categories. As seen in the table, when an expert is responsible for smaller number of categories, the performance for each category improves on average. Since the computational budget is limited, it is difficult to make the number of categories assigned to expert models small. Thus, the numbers of categories responsible by expert models in our final submission vary.

3.3. Competition results

Our final submission consists of the predictions from the following models:

- Two Fast R-CNN models trained on 500 detection categories. One of them is trained for 16 epochs and another is trained for 24 epochs.

[1] https://storage.googleapis.com/openimages/web/factsfigures.html

Table 3: Mean average precision on the instance segmentation track. We did not set a file size limit for the validation set, so the post processing of removing small masks was not evaluated on the validation set. Some of the predictions of Faster R-CNN models are from last year's competition, so we could not evaluate them on the validation set.

	val	public test	private test
Full (16 epochs)	70.62	51.33	46.33
Full (24 epochs)	71.02	51.80	47.17
Ensemble of above two	71.61	52.67	47.32
+ Expert Models	75.74	54.55	50.55
+ Remove small masks		54.83	50.76
+ Faster R-CNN models		**55.33**	**51.10**

Table 4: Mean average precision on the detection track. Some of the predictions of Faster R-CNN models are from last year's competition, so we did not have a way to evaluate them on the validation set.

	val	public test	private test
Full (16 epochs)	68.05	59.03	55.81
Full (24 epochs)	68.44	59.32	56.32
Ensemble of above two	69.02	60.31	57.14
+ Expert Models	73.10	64.54	61.26
+ Faster R-CNN models		**65.45**	**62.22**

- 47 Fast R-CNN expert models. On average each expert predicts 43 categories.
- Faster R-CNN models. We used predictions from Faster R-CNN models trained in the preliminary experiments. Some of them are from the last year's submission [12, 2].

The results for instance segmentation and object detection are shown in Table 3 and Table 4. We ranked 3rd and 4th place in the instance segmentation and object detection tracks, respectively.

For instance segmentation, all ensemble results exceeded the file size limit of 5GB. Thus, we needed to drop some predictions for the frequently predicted categories. Therefore, by adding more models, the instance segmentation test results did not improve as much as the validation scores and the object detection results.

4. Conclusion

In this paper, we described the instance segmentation and object detection submissions to Open Images Challenge 2019 by team PFDet. Thanks to the fast research cycle enabled by an efficient usage of large GPU clusters, we developed several techniques that led to 3rd and 4th place in the the instance segmentation and object detection track, respectively.

Acknowledgments We thank K. Uenishi, R. Arai, T. Shiota and S. Omura for helping with our experiments.

References

[1] Takuya Akiba, Keisuke Fukuda, and Shuji Suzuki. ChainerMN: Scalable Distributed Deep Learning Framework. In *LearningSys workshop in NIPS*, 2017.

[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. In *ECCV Workshop*, 2018.

[3] Golnaz Ghiasi Tsung-Yi Lin Jonathon Shlens Quoc V. Le Barret Zoph, Ekin D. Cubuk. Learning data augmentation strategies for object detection. In *arxiv*, 2019.

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[5] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.

[6] Agrim Gupta, Piotr Dollár, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019.

[7] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017.

[8] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *CVPR*, 2018.

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

[10] Tsung-Yi Lin, Piotr Dollár, Ross B Girshick, Kaiming He, Bharath Hariharan, and Serge J Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context. *ECCV*, 2014.

[12] Yusuke Niitani, Takuya Akiba, Tommi Kerola, Toru Ogawa, Shotaro Sano, and Shuji Suzuki. Sampling techniques for large-scale object detection from sparsely annotated objects. In *CVPR*, 2019.

[13] Yusuke Niitani, Toru Ogawa, Shunta Saito, and Masaki Saito. Chainercv: a library for deep learning in computer vision. In *ACM MM*, 2017.

[14] Wanli Ouyang, Xiaogang Wang, Cong Zhang, and Xiaokang Yang. Factors in finetuning deep model for object detection. In *CVPR*, 2016.

[15] Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *CVPR*, 2018.

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

[17] Seiya Tokui, Ryosuke Okuta, Takuya Akiba, Yusuke Niitani, Toru Ogawa, Shunta Saito, Shuji Suzuki, Kota Uenishi, Brian Vogel, and Hiroyuki Yamazaki Vincent. Chainer: A deep learning framework for accelerating the research cycle. In *KDD*, 2019.