PFDet: 2nd Place Solution to Open Images Challenge 2018 Object Detection Track

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

Abstract

We present a large-scale object detection system by team PFDet. Our system enables training with huge datasets using 512 GPUs, handles sparsely verified classes, and massive class imbalance. Using our method, we achieved 2nd place in the Google AI Open Images Object Detection Track 2018 on Kaggle. [1]

1. Introduction

Open Images Detection Dataset V4 (OID) [6] 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.

Training a deep learning model on OID with low parallelization would lead to prohibitively long training times, as is the case for training with other large-scale datasets [2]. We follow the work of MegDet [11] and use multi-node batch normalization to stably train an object detector with batch size of 512. Using ChainerMN [1], a distributed deep learning library, we demonstrate highly scalable parallelization over 512 GPUs.

OID is different from its predecessors, such as MS COCO [8], not merely in terms of the sheer number of images, but also regarding the annotation style. In the predecessors, instances of all classes covered by the dataset are always exhaustively annotated, whereas in OID, for each image, instances of classes not verified to exist in the image are not annotated. This is a realistic approach to expanding the number of classes covered by the dataset, because without sparsifying the annotated classes, the number of annotations required may explode as the total number of classes increases.

The problem with sparsifying the annotated classes is that most of the CNN-based object detectors learn by assuming that all regions outside of the ground truth boxes belong to the background. Thus, in OID, these learning methods would falsely treat a bounding box as the background when an unverified instance is inside the box. We find that the sparse annotation often leads to invalid labels, especially for classes that are parts of the other classes, which we call *part classes* and *subject classes*, respectively. For instance, a human arm usually appears inside the bounding box of a person. Based on this finding, we propose *co-occurrence loss*. For bounding box proposals that are spatially close to the ground truth boxes with a subject class annotation, co-occurrence loss ignores all learning signals for classifying the part classes of the subject class. This reduces noise in the training signal, and we found this leads to a significant performance improvement for part classes.

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 MS COCO [8]. 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 the number of images with these classes decreases within the same number of training epochs.

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.

Our final model integrates solutions to the three noteworthy challenges of the OID dataset: a large number of images, sparsely verified classes, and massive class imbalance. We use Feature Pyramid Network (FPN) [7] with SE-ResNeXt-101 and SENet-154 [4] as backbones trained with

*The authors contributed equally and they are ordered alphabetically.

[1] https://www.kaggle.com/c/google-ai-open-images-object-detection-track

sigmoid loss and cosine annealing as a learning rate scheduler [9].

To summarize our major contributions:

- **Training at Scale**: We present the feasibility to train object detectors on a batch size of 512 using ChainerMN [1] and 512 GPUs.
- **Co-occurrence Loss**: We present *co-occurrence loss* to ignore instances that are falsely labeled as negative for classes that are unverified using class-wise relationships constructed in advance.
- **Expert Models**: We present the effectiveness of using expert models, especially for classes that rarely appear in the dataset.

2. Method

In this section we present our object detection system that allows fast large-scale training with high accuracy.

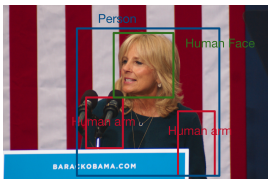
2.1. Basic Architecture

We use a two-stage Faster R-CNN style object detection framework [12] and leverage an SE-ResNeXt or SENet [4] model as the backbone feature extractor. To increase the global context information in the extracted features, we add an FPN and a pyramid spatial pooling (PSP) [15] module to the backbone. Additionally, we increase the context information in the head network by concatenating features from twice the area around each RoI to the head before the fully-connected layers [17]. We increase the number of scales of features extracted by the feature extractor to five from four, which is used in the original work of FPN [7]. This modification allows the network to gather even greater global context information.

Batch normalization (BN) is used ubiquitously to speed up convergence of training [5]. We use multi-node batch normalization [11] to share batch statistics of images across computing nodes so that the number of images used for collecting statistics is sufficiently large to stably compute the statistics. To maximize the effectiveness of BN, we add BN layers to the top-down path of FPN in addition to the BN layers included in the base feature extractor, and the head network.

We train the CNN by first expanding ground truth labels to include all ancestor classes using the semantic hierarchy prepared in OID. We formulate the learning problem as in a multi-label setting, and we use a sigmoid cross entropy loss for each class separately [14]. In the case when the ground truth class is not a leaf of the semantic hierarchy, we do not compute sigmoid cross entropy loss for descendants of the ground truth classes.

We use non-maximum weighted (NMW) [16] suppression during test time to reduce duplicate detections. It was found that this works better than standard non-maximum suppression (NMS). NMS was used in the RPN while training.



(a) An image with annotations for some human parts (*Human face* and *Human arm*).

(b) An image with no verification of human part classes.

Figure 1: Images and annotations from the OID dataset. In the right image (1b), even though *Human face* clearly exists, there is no annotation because the class is not verified.

2.2. Co-occurrence Loss

In the OID dataset, for each image, only the instances of classes that are verified are annotated. In Figure 1, images from OID with different coverage of verified labels are shown. In Figure 1a, relatively more human part classes are labeled compared to Figure 1b, which has no annotation of human part classes such as *Human face*. With sparse annotations, conventional losses for training CNN-based object detectors [7] would falsely label regions around instances as negatives.

To reduce false training signals, we introduce co-occurrence loss. The main idea behind this loss is that for some classes, the relationship with other classes is informative enough that we can safely ignore false training samples. For instance, given a ground truth bounding box of a person, it is highly likely that a human face exists inside the box even if human faces are not verified to exist in the image by the human annotator. To implement the loss, we gather pairs of classes that satisfy the relationship "if a proposal is inside a ground truth box of class X, it is safe to ignore treating the proposal as a negative sample of class Y". We use a pair when class Y is a part of class X or instances of class Y is usually a possession of an instance of class X. For instance, tires are parts of cars and jeans are usually possessed by a person.

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. Each expert model is fine-tuned on a very small subset of the full category space,

which we find to perform well for rare classes.

2.4. Ensembling

For the final submission, we use an ensemble of models trained on all 500 classes and the expert models. We do not apply duplicate suppression for individual models, but instead apply suppression once on the concatenation of the outputs of all models.

Since the distribution of the performance over classes is different among models, we prioritize outputs of models that are expected to perform better based on validation scores. For each model m, we compute a weight w_c^m for class c, which is multiplied to the confidence scores of the outputs of the model for this class. Suppose that the mean of the validation scores of all models for class c to be μ_c, we set the weight w_c^m to α if model m performs lower than the average μ_c. Otherwise, we compute the weight by simply interpolating between α and 1 linearly. The computation is done as $w_c^m = \frac{s_c^m - \mu_c}{t_c - \mu_c} + \alpha \frac{t_c - s_c^m}{t_c - \mu_c}$, where s_c^m is the validation score of the model m for class c and t_c is the highest validation score for class c.

3. Experiments

We used the split of the OID dataset for the ECCV2018 competition. The recommended train and validation splits were used. We never used the validation split to train networks. The weights of the base feature extractor are pretrained on ImageNet. In addition to the OID dataset, we used MS COCO [8] to train expert models for classes that are in the intersection of the label spaces of OID and MS COCO.

We use SGD with corrected momentum [3] and a linear learning rate scaling rule with respect to the number of GPUs. The initial learning rate is set to 0.01 for batch size of 8. The training starts with a warm-up phase. Cosine annealing is used to attenuate the learning rate over time.

We scaled images during training so that the length of the smaller edge is between $[640, 1056]$. Also, we randomly flipped images horizontally to augment training data. For the final submission, we augmented outputs at the test-time by concatenating outputs from inputs of multiple scales with and without horizontal flip.

3.1. Software and Hardware Systems

We use Chainer [13] as our deep learning framework, ChainerMN [1] to scale up training and ChainerCV [10] for quick prototyping. For training, we used MN-1b, an in-house cluster owned by Preferred Networks, Inc. It consists of 64 nodes, where each node has two Intel Xeon Gold 6154 CPUs (3.0 GHz, 18 cores), 384 GB memory and eight NVIDIA Tesla V100 (32 GB memory). The nodes are interconnected by two Mellanox Infiniband EDR.

Table 1: Performance of a single model with single scale testing on the validation split.

	validation mAP
Baseline (FPN with SE-ResNeXt-101)	60.0
+ multi-scale training	60.3 (+0.3)
+ PSP and add BN to head	60.4 (+0.1)
+ Cosine Annealing	63.4 (+3.1)
+ Add FPN scale	64.5 (+1.1)
+ Co-occurrence loss	65.2 (+0.7)
+ 16 epochs	65.8 (+0.6)
+ Context head	66.0 (+0.2)
+ SENet-154 and additional anchors	67.5 (+1.5)

Table 2: Ensemble of models with test-time augmentation.

	val mAP	Public LB	Private LB
Single best model	69.95	55.81	53.43
+ class20 experts	71.73	59.34	55.87
+ class10 experts	72.33	60.19	56.61
+ All the others except COCO	73.98	61.83	57.97
+ COCO	74.07	62.34	58.48
+ class-weight ensemble		62.88	58.63
Competition winner		61.71	58.66

3.2. Results

We first study the effectiveness of different techniques on the validation set. Our baseline is FPN [7] with SE-ResNeXt-101 [4] as the backbone trained with sigmoid loss for 12 epochs. The learning rate is multiplied by $\frac{1}{10}$ at epoch 8 and 11. Mean average precision (mAP) of the validation split over 500 classes is shown in Table 1. For the final model, we increased the variation of anchors by using very tall and very wide ones. The ratios of the width and height of the anchors are $[\frac{1}{3}, \frac{1}{2}, 1, 2, 3]$. Using 512 GPUs, the computing time of training the final model was 33 hours for 16 epochs. The scaling efficiency when using 512 GPUs was 83% in comparison to the single-node (i.e. 8 GPUs) baseline.

We also show the results after ensembling models in Table 2. Our final model outperformed the competition winner in the public leaderboard by 1.18 mAP and falls behind them by only 0.02 mAP on the private leader board.

In Table 3, we show a comparison of a model trained with co-occurrence model and a model with identical setup except for the co-occurrence loss. For 47 classes affected by co-occurrence loss, we see 9.2 AP improvement on average.

In Table 4, the ablative study of expert models is shown. For the rarest 250 classes, we see improvement with expert models. However, for more common classes, the fine-tuning has a negative effect. Also, we consistently see better results if we reduce the size of the classes to which expert models are fine-tuned.

Table 3: Ablative study of co-occurrence loss on classes that can be ignored by the loss. The scores are AP calculated on the validation set of the dataset.

	Arm	Ear	Nose	Mouth	Hair	Eye	Beard	Face	Head	Foot	Leg	Hand	Glove	Hat	Dress	Fedora
Baseline	40.9	17.5	34.7	21.4	63.8	27.3	55.5	82.7	55.1	50.7	41.6	32.3	**63.4**	64.9	70.6	67.0
Co-occurrence	**55.2**	**62.6**	**69.6**	**55.2**	**74.7**	**64.0**	**76.8**	**91.4**	**78.9**	**59.5**	**54.4**	**53.6**	60.8	**69.0**	**73.9**	**70.3**

	Footwe.	Sandal	Boot	Sports.	Coat	Sock	Glasse.	Belt	Helmet	Jeans	High h.	Scarf	Swimwe.	Earrin.	Bicycl.	Shorts
Baseline	61.9	53.6	**61.6**	52.9	58.0	**70.6**	74.9	**66.8**	80.2	62.7	76.6	71.6	**63.4**	82.0	75.1	69.7
Co-occurrence	**68.5**	**58.9**	57.9	**61.2**	**73.3**	67.1	**85.4**	61.9	**82.4**	**77.6**	**78.8**	**75.8**	**63.4**	**86.1**	**75.8**	**75.4**

	Baseba.	Minisk.	Cowboy.	Goggles	Jacket	Shirt	Sun ha.	Suit	Trouse.	Brassi.	Tie	Licens.	Wheel	Tire	Handle	Average
Baseline	**67.2**	**62.5**	65.0	79.3	69.5	70.9	61.3	83.7	62.5	**82.6**	84.7	72.1	48.3	49.4	41.1	61.1
Co-occurrence	62.2	58.7	**73.3**	**86.7**	**74.3**	**81.6**	**66.4**	**87.0**	**69.8**	74.5	**91.5**	**74.6**	**66.4**	**69.6**	**46.2**	**70.3**

Table 4: Ablative study of expert models. Column "Index 11-100" lists the mean validation scores for the 11th class to the 100th class ordered by the occurrence in the dataset. Other columns similarly select the classes. The row "Full" lists scores of a model without fine-tuning. A row "ClassX experts" lists scores of expert models fine-tuned on class subsets of length X.

	Index 11-100	Index 101-250	Index 251-350
Full	51.9	70.5	**70.9**
Class10 experts	**65.6**	**73.1**	66.3
Class40 experts	61.0	66.3	50.9

4. Conclusion

In this paper, we presented a large-scale object detector by team PFDet, that allows scalable, fast object detection training on a large dataset using 512 GPUs. The resulting fast research cycle allowed us to leverage several techniques that led to 2nd place in the Google AI Open Images Object Detection Track 2018 on Kaggle.

Acknowledgments We thank K. Fukuda, K. Uenishi, R. Arai, S. Omura, R. Okuta, and T. Abe for help with the experiments, and R. Calland for helping to improve the manuscript.

References

[1] T. Akiba, K. Fukuda, and S. Suzuki. ChainerMN: Scalable Distributed Deep Learning Framework. In *LearningSys workshop in NIPS*, 2017.

[2] T. Akiba, S. Suzuki, and K. Fukuda. Extremely large minibatch SGD: Training ResNet-50 on ImageNet in 15 minutes. In *Deep Learning at Supercomputer Scale Workshop in NIPS*, 2017.

[3] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[4] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *CVPR*, 2018.

[5] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.

[6] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, S. Kamali, M. Malloci, J. Pont-Tuset, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, and K. Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

[7] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[8] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollr. Microsoft coco: Common objects in context. *ECCV*, 2014.

[9] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *ICLR*, 2017.

[10] Y. Niitani, T. Ogawa, S. Saito, and M. Saito. Chainercv: a library for deep learning in computer vision. In *ACM MM*, 2017.

[11] C. Peng, T. Xiao, Z. Li, Y. Jiang, X. Zhang, K. Jia, G. Yu, and J. Sun. Megdet: A large mini-batch object detector. In *CVPR*, 2018.

[12] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[13] S. Tokui, K. Oono, S. Hido, and J. Clayton. Chainer: a next-generation open source framework for deep learning. In *LearningSys workshop in NIPS*, 2015.

[14] J. Uijlings, S. Popov, and V. Ferrari. Revisiting knowledge transfer for training object class detectors. In *CVPR*, 2018.

[15] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[16] H. Zhou, Z. Li, C. Ning, and J. Tang. Cad: Scale invariant framework for real-time object detection. In *ICCV Workshops*, 2017.

[17] Y. Zhu, C. Zhao, J. Wang, X. Zhao, Y. Wu, H. Lu, et al. Couplenet: Coupling global structure with local parts for object detection. In *ICCV*, 2017.