MegDet: A Large Mini-Batch Object Detector

Chao Peng* Tete Xiao[1]* Zeming Li[2]* Yuning Jiang Xiangyu Zhang Kai Jia Gang Yu Jian Sun
[1]School of Electronics Engineering and Computer Science, Peking University, jasonhsiao97@pku.edu.cn
[2]School of Software, Tsinghua University, lizm14@tsinghua.edu.cn
Megvii Inc. (Face++), {pengchao, jyn, zhangxiangyu, jiakai, yugang, sunjian}@megvii.com

Abstract

*The improvements in recent CNN-based object detection works, from R-CNN [11] and Fast/Faster R-CNN [10, 29] to recent Mask R-CNN [14] and RetinaNet [22], mainly come from new network, or framework, or loss design. But mini-batch size, a key factor in the training, has not been well studied. In this paper, we propose a Large Mini-Batch Object Detector (MegDet) to enable the training with much larger mini-batch size than before (e.g. from 16 to 256), so that we can effectively utilize multiple GPUs (up to 128 in our experiments) to significantly shorten the training time. Technically, we suggest a learning rate policy and Cross-GPU Batch Normalization, which together allow us to successfully train a large mini-batch detector in much less time (e.g., from 33 hours to 4 hours), and achieve even better accuracy. The MegDet is the backbone of our submission (mmAP 52.5%) to COCO 2017 Challenge, where we won the **1st** place of Detection task.*

1. Introduction

Tremendous progresses have been made on CNN-based object detection, since seminal work of R-CNN [11], Fast/Faster R-CNN series [10, 29], recent state-of-the-art detectors like Mask RCNN [14] and RetinaNet [22]. Taking COCO [23] dataset as an example, its performance has been boosted from 19.7 AP in Fast R-CNN [10] to 39.1 AP in RetinaNet [22], in just two years. The improvements are mainly due to better backbone network [16], new detection framework [29], novel losses [22], improved pooling method [5, 14], and so on.

A recent trend on CNN-based image classification uses very large min-batch size to significantly speed up the training. For example, the training of ResNet-50 can be accomplished in an hour [13] or even in 31 minutes [37] , using mini-batch size 8,192 or 16,000, with little or small sacrifice on the accuracy. In contract, the mini-batch size remains very small (e.g., 2-16) in object detection literitures. Therefore in this paper, we study the problem of mini-batch size in object detection and present a technical solution to successfully train a large mini-batch size object detector.

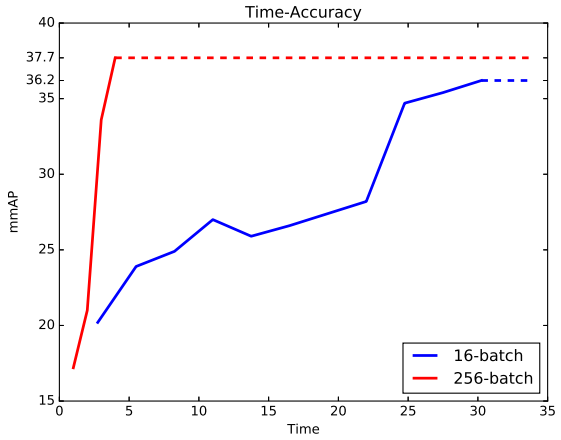

Figure 1: Validation accuracy of the same FPN object detector trained on COCO dataset, with mini-batch size 16 (on 8 GPUs) and mini-batch size 256 (on 128 GPUs). The large mini-batch detector is more accurate and its training is nearly an order-of-magnitude faster.

What is wrong with the small mini-batch size? Originating from the object detector R-CNN [11] series, a mini-batch involving only 2 images is widely adopted in popular detectors like Faster R-CNN and R-FCN. Though in state-of-the-art detectors like RetinaNet and Mask R-CNN the mini-batch size is increased to 16, which is still quite small compared with the batch size (e.g., 256) used in current image classification. There are several potential drawbacks associated with small mini-batch size. First, the training time is notoriously lengthy. For example, the training of ResNet-152 on COCO takes 3 days, even using the mini-bath size 16 on a machine with 8 Titian XP GPUs. Second, training with

*Equal contribution. This work is done when Zeming Li and Tete Xiao are interns at Megvii Research.

small mini-batch size fails to provide accurate statistics for batch normalization [18] (BN). Usually, the mini-batch size for ImageNet classification network is set to 256 in order to obtain a good batch normalization statistics, which is significantly larger than the mini-batch size used in current object detector setting. Last, the numbers of positive and negative training examples within a small mini-batch are more likely imbalanced, which might hurt the final accuracy. Figure 2 gives some examples with imbalanced positive and negative proposals. And Table 1 compares the statistics of two detectors with different mini-batch sizes, at different training epochs on COCO dataset.

Why do not simply increase the min-batch size? As in the image classification problem, the main dilemma we are facing is: the large min-batch size usually requires a large learning rate to maintain the accuracy, according to "equivalent learning rate rule" [13, 19]. But a large learning rate could lead to the failure of convergece; if we use a smaller learning rate to ensure the convergence, we often get inferior results.

We propose a solution to address the above dilemma. First, we present a new explanation of linear scaling rule and borrow the "warmup" learning rate policy [13] to gradually increase the learning rate at the very early stage. This increases the chance of convergence. Second, to address the accuracy and convergence issues, we introduce Cross-GPU Batch Normalization (CGBN), to exploit the fact that we have large number of samples in a min-batch. CGBN not only improves the accuracy but also makes the training much more stable. More importantly, we pleasely observed that we can keep the final accuracy while increasing the batch size, with the aid of the CGBN. This is significant because we are able to safely enjoy the rapidly increased computational power from industry.

Our MegDet (ResNet-50 as backbone) can finish COCO training in 4 hours on 128 GPUs, reaching even higher accuracy. In contrast, the small mini-batch counterpart takes 33 hours, achieving lower accuracy. This means that we can speed up the innovation cycle by nearly an order-of-magnitude, as shown in Figure 1. Based on MegDet, we secured **1st** place of COCO 2017 Detection Challenge.

Our technical contributions can be summarized as:

- We give a new interpretation of linear scaling rule, in the context of object detection, based on an assumption of maintaining equivalent variances.
- We are the first to perform BN in the object detection framework. We demonstrated that our Cross-GPU BN not only benefits the accuracy, but also makes the training easy to converge, especially for the large mini-batch size.
- We are the first to finish the COCO training (based on ResNet-50) in 4 hours, using 128 GPUs, and achieving improved accuracy.

Epoch	Batch Size	Ratio(%)
1	16	5.58
	256	**9.82**
6	16	11.77
	256	**16.11**
12	16	16.59
	256	**16.91**

Table 1: Ratio of positive and negative samples in the training (at epoch 1, 6, 12). The larger batch size makes the ratio more balanced, especially at the early stage.

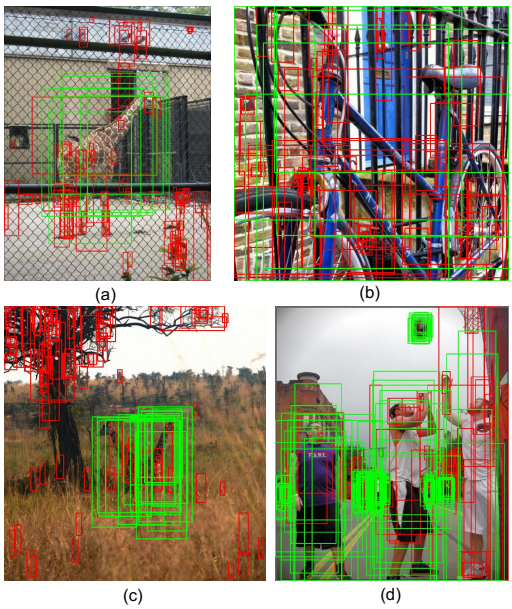

Figure 2: Example images with positive and negative proposals. (a-b) two examples with imbalanced ratio, (c-d) two examples with moderate balanced ratio. Note that we subsampled the negative proposals for visualization.

- Our MegDet leads to the winning of COCO 2017 Detection Challenge.

2. Related Work

CNN-based detectors have been the mainstream in current academia and industry. We can roughly divide existing CNN-based detectors into two categories: one-stage detectors like SSD [24], YOLO [27, 28] and recent RetinaNet [22], and two-stage detectors [31, 1] like Faster R-CNN [29], R-FCN [6] and Mask-RCNN [14].

For two-stage detectors, let us start from the R-CNN family. R-CNN [11] was first introduced in 2014. It employs Selective Searchh [35] to generate a set of region proposals and then classifies the warped patches through a

CNN recognition model. As the computation of the warp process is intensive, SPPNet [15] improves the R-CNN by performing classification on the pooled feature maps based on a spatial pyramid pooling rather than classifying on the resized raw images. Fast-RCNN [10] simplifies the Spatial Pyramid Pooling (SPP) to ROIPooling. Although reasonable performance has been obtained based on Fast-RCNN, it still replies on traditional methods like selective search to generate proposals. Faster-RCNN [29] replaces the traditional region proposal method with the Region Proposal Network (RPN), and proposes an end-to-end detection framework. The computational cost of Faster-RCNN will increase dramatically if the number of proposals is large. In R-FCN, position-sensitive pooling is introduced to obtain a speed-accuracy trade-off. Recent works are more focusing on improving detection performance. Deformable ConvNets [7] uses the learned offsets to convolve different locations of feature maps, and forces the networks to focus on the objects. FPN [21] introduces the feature pyramid technique and makes significant progress on small object detection. As FPN provides a good trade-off between accuracy and implementation, we use FPN as the default detection framework. To address the alignment issue, Mask RCNN [14] introduce the ROIAlign and achieve state-of-the-art results for both object detection and instance segmentation.

Different from two-stage detectors, which involve a proposal and refining step, one-stage detectors usually run faster. In YOLO [27, 28], a convolutional network is followed with a fully connected layer to obtain classification and regression results based on a 7×7 grid. SSD [24] presents a fully convolutional network with different feature layers targeting different anchor scales. Recently, RetinaNet is introduced in [22] based on the focal loss which can significantly reduce false positives.

Large mini-batch training has been an active research topic in image clasisifcation. In [13], imagenet training based on Resnet50 can be finished in one hour. [37] presents a training setting which can finish the Resnet50 training in 31 miniutes without lossing classification accuracy. However, the topic of large mini-batch training for object detection is rarely discussed in past research.

3. Approach

In this section, we present our Large Mini-Batch Detector (MegDet), to finish the training in less time while achieving higher accuracy.

3.1. Problems with Small Mini-Batch Size

The early generation of CNN-based detectors use very small mini-batch sizes like 2 in Faster-RCNN and R-FCN. Even with the state-of-the-art detectors like RetinaNet and Mask RCNN, the batch size is set as 16. There exist a few problems when training with a small mim-batch size. First, we have to pay much longer training time if a small mini-batch size is utilized for training. As shown in Figure 1, the training of a ResNet-50 detector based on a mini-batch size of 16 takes more than 30 hours. If the batch size is set to 2, the training time could be more than one week. Second, in the training of detector, we usually fix the statistics of Batch Normalization calculated on pre-trained ImageNet dataset. If the batch size is small, we are not able to compute accurate statistics of BN. Using the BN statistics from ImageNet is a sub-optimal choice. Last, the ratio between positive and negative samples could be very imbalanced. In Table 1, we provide the statistics for the ratio between positive and negative training examples. We can see that a small mini-batch size leads to more imbalanced training examples, especially at the initial stage. This imbalance may affect the overall detection performance.

As we discussed in the introduction, simply increasing the mini-batch size has to deal with the tradeoff between convergence and accuracy. To address this issue, we first discuss the learning rate policy for the large mini-batch.

3.2. Learning Rate for Large Mini-Batch

The learning rate policy is strongly related to the SGD algorithm. Therefore, we start the discussion by first reviewing the structure of loss for object detection network,

$$\begin{aligned} L(x, w) &= \frac{1}{N}\sum_{i=1}^{N} l(x_i, w) + \frac{\lambda}{2}||w||_2^2 \\ &= l(x, w) + l(w), \end{aligned} \tag{1}$$

where N is the min-batch size, $l(x, w)$ is the task specific loss and $l(w)$ is the regularization loss. For Faster R-CNN [29] framework and its variants [6, 21, 14], $l(x_i, w)$ consists of RPN prediction loss, RPN bounding-box regression loss, prediction loss, and bounding box regression loss.

According to the definition of mini-batch SGD, the training system needs to compute the gradients with respect to weight w, and update them after every iteration. When the size of mini-batch changes, such as $\hat{N} \leftarrow k \cdot N$, we expect the learning rate r should also be adapted to maintain the efficiency of training. As the best practice, previous works [19, 13, 37] use *Linear Scaling Rule*, that changes the new learning rate to $\hat{r} \leftarrow k \cdot r$. Since one step in large mini-batch $\hat{N}$ should match the effectiveness of k accumulative steps in small mini-batch N, the learning rate r shall be also multiplied by the same ratio k to counteract the scaling factor N in loss. This is based on a *gradient equivalence* assumption [13] in the SGD updates. This rule of thumb has been well-verified in image classification, and we find it is still applicable for object detection. However, in the following, we give a different interpretation under a weaker assumption.

In image classification, every image has only one annotation and $l(x, w)$ is a simple form of cross-entropy. As for object detection, every image has different number of box annotations, thus resulting in different ground-truth distribution. Considering the different annotation type on two tasks, the assumption of gradient equivalence between different mini-batch sizes might be less likely to be hold in object detection. Here, we introduce another explanation based on the following variance analysis.

Variance Equivalence. Different from the gradient equivalence assumption, we assume that the variance of gradient remain the same during k steps. Given the mini-batch size N, if the gradient of each sample $\nabla l(x_i, w)$ obeying i.i.d., the variance of gradient on $l(x, w)$ is:

$$\begin{aligned} \mathrm{Var}(\nabla l(x, w_t)) &= \frac{1}{N^2} \sum_{i=1}^{N} \mathrm{Var}(\frac{\partial l(x_i, w_t)}{\partial w_t}) \\ &= \frac{1}{N^2} \times \left(N \cdot \sigma_l^2\right) \\ &= \frac{1}{N} \sigma_l^2. \end{aligned} \tag{2}$$

Similarly, for the large mini-batch $\hat{N} = k \cdot N$, we can get the following expression:

$$\mathrm{Var}(\nabla l_{\hat{N}}(x, w_t)) = \frac{1}{kN} \sigma_l^2. \tag{3}$$

Instead of expecting equivalence on weight update, here we want to *maintain the variance of one update in large mini-batch $\hat{N}$ equal to k accumulative steps in small mini-batch N*. To achieve this, we have:

$$\begin{aligned} \mathrm{Var}(r \cdot \sum_{t=1}^{k} (\nabla l_N^t(x, w))) &= r^2 \cdot k \cdot \mathrm{Var}(\nabla l_N(x, w)) \\ &\approx \hat{r}^2 \mathrm{Var}(\nabla l_{\hat{N}}(x, w)) \end{aligned} \tag{4}$$

Within Equation (2) and (3), the above equality holds if and only if $\hat{r} = k \cdot r$, which gives the same linear scaling rule for $\hat{r}$.

Although the final scaling rule is the same, our *variance equivalence* assumption Equation (4) is weaker because we just expect that the large mini-batch training can maintain equivalent statistics on the gradients. We hope the variance analysis presented here can shed light on deeper understanding of learning rate in wider applications.

Warmup Strategy. As discussed in [13], the linear scaling rule may not be applicable at the initial stage of the training because the weights changing are dramatic. To address this practical issue, we borrow *Linear Gradual Warmup*, also introduced in [13]. That is, we set up the learning rate small enough at the beginning, such as r. Then, we increase the learning rate with a constant speed after every iteration, until to $\hat{r}$.

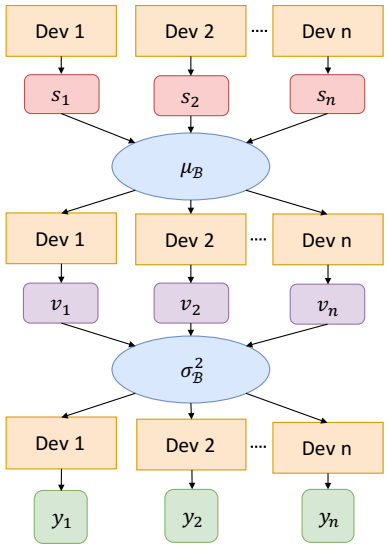

Figure 3: Implementation of Cross-GPU Batch Normalization. The gray ellipse depicts the synchronization over devices, while the rounded boxes represents paralleled computation of multiple devices.

The warmup strategy can help the convergence. But as we demonstrated in the experiments later, itself is not able to enable us to use a large mini-batch size, e.g., 64 or 128 or 256. Next, we introduce a Cross-GPU Batch Normalization, which is the main workhorse of large mini-batch training.

3.3. Cross-GPU Batch Normalization

Batch Normalization [18] is an important technique for training a very deep convolutional neural network. Without batch normalization, training will consume much more time or even fail to converge. However, previous object detection frameworks, such as FPN [21], initialize model with an ImageNet pre-trained model, after which the batch normalization layer is fixed during the whole fine-tuning procedure. In this work, we make an attempt to perform batch normalization for object detection.

It is worth noting that the input image of classification network is 224×224 or 299×299, and a single NVIDIA TITAN Xp GPU with 12 Gigabytes memory is enough for 32 or more images. In this way, batch normalization can be computed on each device alone. However, for object detection, a detector needs to handle objects of various scales, thus higher resolution images are needed as its input. In [21], input of size 800×800 is used, significantly limiting the number of possible samples on one device. Thus, we have to perform batch normalization crossing multiple GPUs to collect sufficient statistics from more samples.

To implement batch normalization across GPUs, we need to compute the aggregated mean/variance statistics over all devices. Most existing deep learning frameworks utilize the BN implementation in cuDNN [4] that only provides a high-level API without permitting modification of internal statistics. Therefore we need to implement BN in terms of preliminary mathematical expressions and use an "AllReduce" operation to aggregate the statistics. These fine-grained expressions usually cause significant runtime overhead and the AllReduce operation is missing in most frameworks.

Our implementation of Cross-GPU Batch Normalization is skeched in Figure 3. Given n GPU devices in total, sum value s_k is first computed based on the training examples assigned to the device k. By averaging the sum values from all devices, we obtain the mean value $\mu_{\mathcal{B}}$ for current mini-batch. This step requires an AllReduce operation. Then we calculate the variance for each device and get $\sigma^2_{\mathcal{B}}$. After broadcasting $\sigma^2_{\mathcal{B}}$ to each device, we can perform the standard normalization by $y = \gamma \frac{x-\mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}}+\epsilon}} + \beta$. Algorithm 1 gives the detailed flow. In our implementation, we use NVIDIA Collective Communication Library (NCCL) to efficiently perform AllReduce operation for receiving and broadcasting described in Algorithm 1.

Note that we only perform BN across GPUs on the same machine. So, we can calculate BN statistics on 16 images if each GPU can hold 2 images. To perform BN on 32 or 64 images, we apply *sub-linear memory* [3] to save the GPU memory consumption by slightly compromising the training speed.

In next section, our experimental results will demonstrate the great impacts of CGBN is on both accuracy and convergence.

Input: Values of input x on multiple devices
in a minibatch: $\mathcal{B} = \bigcup_{i=1}^{n} \mathcal{B}_i$, $\mathcal{B}_i = \{x_{i_1 \ldots i_n}\}$
BN parameters: γ, β
Output: $y = \text{CGBN}(x)$
1: **for** $i = 1, \ldots, n$ **do**
2: compute the device sum s_i over set B_i
3: **end for**
4: reduce the set $s_{1,\ldots,n}$ to minibatch mean $\mu_{\mathcal{B}}$
5: broadcast $\mu_{\mathcal{B}}$ to each device
6: **for** $i = 1, \ldots, n$ **do**
7: compute the device variance sum v_i over set B_i
8: **end for**
9: reduce the set $v_{1,\ldots,n}$ to minibatch variance $\sigma^2_{\mathcal{B}}$
10: broadcast $\sigma^2_{\mathcal{B}}$ to each device
11: compute the output: $y = \gamma \frac{x-\mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}}+\epsilon}} + \beta$ over devices

Algorithm 1: Cross-GPU Batch Normalization over a mini-batch $\mathcal{B}$.

4. Experiments

We conduct experiments on COCO Detection Dataset [23], which is split into train, validation, and test, containing 80 categories and over $250,000$ images. We use ResNet-50 [16] pre-trained on ImageNet [8] as the backbone network and Feature Pyramid Network (FPN) [21] as the detection framework. We train the detectors over 118,000 training images and evaluate on 5000 validation images. We use the SGD optimizer with momentum 0.9, and adopts the weight decay 0.0001. The base learning rate for batch size 16 is 0.02. For other batch size settings, the linear scaling rule described in Section 3.2 is applied. As for large mini-batch, we use the sublinear memory [3] and distributed training to remedy the GPU memory constraints.

We have two training policies in all experiments: 1) *normal*, decreasing the learning rate at epoch 8 and 10 by multiplying scale 0.1, and ending at epoch 11; 2) *long*, decreasing the learning rate at epoch 11 and 14 by multiplying scale 0.1, halving the learning rate at epoch 17, and ending at epoch 18. Unless otherwise specified, we use the normal policy.

4.1. Large mini-batch size, no BN

We start our study through the different batch size settings, without batch normalization. We conduct the experiments with mini-batch size 16, 32, 64, and 128. For mini-batch sizes 32, we observed that the training has some chances to fail, even we use the warmup strategy. For mini-batch size 64, we are not able to manage the training to converge even with the warmup. We have to low the learning rate by half to make the training to converge. For mini-batch size 128, the training failed with both warmup and half learning rate. The results on COCO validation set are shown in Table 2. We can observe that: 1) mini-batch size 32 achieved a nearly linear acceleration, without loss of accuracy, compared with the baseline using mini-batch size 16; 2) lower learning rate (in mini-batch size 64) results in noticeable accuracy loss; 3) the training is harder or even impossible when the mini-batch size and learning rate are larger, even with the warmup strategy.

Mini-Batch size	mmAP	Time (h)
16	36.2	33.2
32	**36.4**	**15.1**
64	failed	–
64 (half learning rate)	36.0	7.5
128 (half learning rate)	failed	–

Table 2: Comparisons of different mini-batch sizes, without BN.

4.2. Large mini-batch size, with CGBN

This part of experiments is trained with batch normalization. Our first, key observation is that *all trainings easily converge*, no matter of the mini-batch size, when we combine the warmup strategy and CGBN. This is remarkable because we do not have to worry about the possible loss of accuracy caused by using smaller learning rate.

Batch size	BN size	# of GPUs	mmAP	Time(h)
16-base	0	8	36.2	33.2
2	2	2	31.5	131.2
4	4	4	34.9	91.4
8	8	8	35.9	71.5
16	2	8	31.0	45.6
16	16	8	37.0	39.5
32	32	8	37.3	45.5
64	64	8	35.3	40.9
64	32	16	37.1	19.6
64	16	32	37.1	11.2
128	32	32	37.1	11.3
128	16	64	37.0	6.5
256	32	64	37.1	7.2
256	16	128	37.1	**4.1**
16 (long)	16	8	37.7	65.2
32 (long)	32	8	**37.8**	60.3
64 (long)	32	16	37.6	30.1
128 (long)	32	32	37.6	15.8
256 (long)	32	64	37.7	9.4
256 (long)	16	128	37.7	5.4

Table 3: Comparisons of training with different mini-batch sizes, BN sizes (the number of images used for calculating statistics), GPU numbers, and training policies. "long" means that we apply the long training policy. When the BN size ≥ 32, the sublinear memory is applied and thus slightly reduces training speed. Overall, the large mini-batch size with BN not only speeds up the training, but also improves the accuracy.

The main results are summarized in Table 3. We have the following observations. First, within the growth of mini-batch size, the accuracy almost remain the same level, which is consistently better that the baseline (16-base). In the meanwhile, a larger mini-batch size always leads to a shorter training cycle. For instance, the 256 mini-batch experiment with 128 GPUs finishes the COCO training only 4.1 hours, which means a $8x$ acceleration compared to the 33.2 hours baseline.

Second, the best BN size (number of images for BN statistics) is 32. With too less images, e.g. 2, 4, or 8, the BN statistics are very inaccurate, thus resulting a worse performance. However, when we increase the size to 64, the accuracy drops. This demonstrates the mismatch between image classification and object detection tasks.

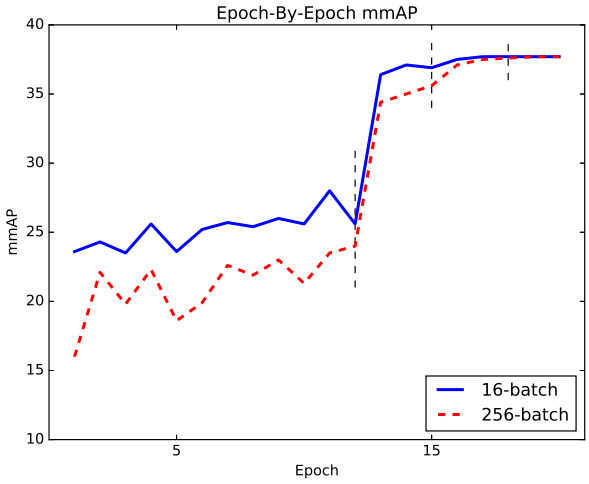

Figure 4: Validation accuracy of 16 (long) and 256 (long) detectors, using the long training policy. The BN sizes are the same in two detectors. The vertical dashed lines indicate the moments of learning rate decay.

Third, in the last part of Table 3, we investigate the long training policy. Longer training time slightly boots the accuracy. For example, "32 (long)" is better that its counterpart (37.8 v.s. 37.3). When the mini-batch size is larger than 16, the final results are very consist, which indicates the true convergence.

Last, we draw epoch-by-epoch mmAP curves of 16 (long) and 256 (long) in Figure 4. 256 (long) is worse at early epochs but catches up 16 (long) at the last stage (after second learning rate decay). This observation is different from those in image classification [13, 37], where both the accuracy curves and convergent scores are very close between different batch size settings. We leave the understanding of this phenomenon as the future work.

name	mmAP	mmAR
DANet	45.7	62.7
Trimps-Soushen+QINIU	48.0	65.4
bharat_umd	48.1	64.8
FAIR Mask R-CNN [14]	50.3	66.1
MSRA	50.4	69.0
UCenter	51.0	67.9
MegDet (Ensemble)	**52.5**	**69.0**

Table 4: Result of (enhanced) MegDet on test-dev of COCO dataset.

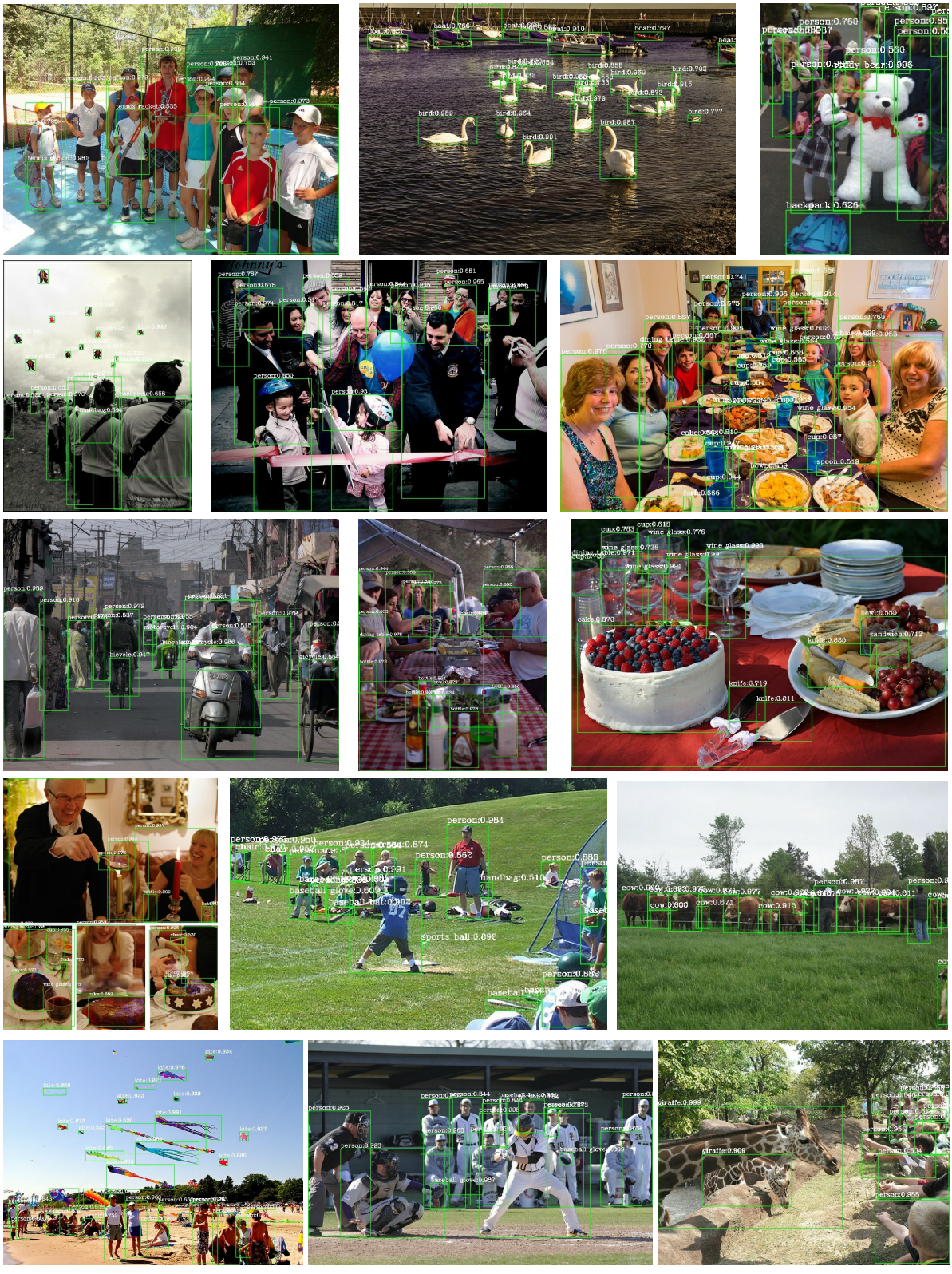

Figure 5: Illustrative examples for our MegDet on COCO dataset.

5. Concluding Remarks

We have presented a large mini-batch size detector, which achieved better accuracy in much shorter time. This is remarkable because our research cycle has been greatly accelerated. As a result, we have obtained 1st place of COCO 2017 detection challenge. The details are in Appendix.

Appendix

Based on our MegDet, we integrate the techniques including OHEM [33], atrous convolution [38, 2], stronger base models [36, 17], large kernel [26], segmentation supervision [25, 32], diverse network structure [12, 30, 34], contextual modules [20, 9], ROIAlign [14] and multi-scale training and testing for COCO 2017 Object Detection Challenge. We obtained **50.5** mmAP on validation set, and **50.6** mmAP on the test-dev. The ensemble of four detectors finally achieved **52.5**. Table 4 summarizes the entries from the leaderboard of COCO 2017 Challenge. Figure 5 gives some exemplar results.

References

[1] S. Bell, C. Lawrence Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2874–2883, 2016. 2

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv preprint arXiv:1606.00915*, 2016. 8

[3] T. Chen, B. Xu, C. Zhang, and C. Guestrin. Training deep nets with sublinear memory cost. *arXiv preprint arXiv:1604.06174*, 2016. 5

[4] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. *arXiv preprint arXiv:1410.0759*, 2014. 5

[5] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3150–3158, 2016. 1

[6] J. Dai, Y. Li, K. He, and J. Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387, 2016. 2, 3

[7] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. *arXiv preprint arXiv:1703.06211*, 2017. 3

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 5

[9] S. Gidaris and N. Komodakis. Object detection via a multi-region and semantic segmentation-aware cnn model. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1134–1142, 2015. 8

[10] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 1, 3

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 1, 2

[12] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio. Maxout networks. In *International Conference on Machine Learning*, pages 1319–1327, 2013. 8

[13] 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. 1, 2, 3, 4, 6

[14] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 2, 3, 6, 8

[15] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014. 3

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 5

[17] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. *arXiv preprint arXiv:1709.01507*, 2017. 8

[18] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015. 2, 4

[19] A. Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014. 2, 3

[20] J. Li, Y. Wei, X. Liang, J. Dong, T. Xu, J. Feng, and S. Yan. Attentive contexts for object detection. *IEEE Transactions on Multimedia*, 19(5):944–954, 2017. 8

[21] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 3, 4, 5

[22] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 2, 3

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1, 5

[24] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016. 2, 3

[25] J. Mao, T. Xiao, Y. Jiang, and Z. Cao. What can help pedestrian detection? In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 8

[26] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters – improve semantic segmentation by global convolutional network. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 8

[27] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016. 2, 3

[28] J. Redmon and A. Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016. 2, 3

[29] S. Ren, K. He, R. Girshick, and J. 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, 2, 3

[30] S. Ren, K. He, R. Girshick, X. Zhang, and J. Sun. Object detection networks on convolutional feature maps. *IEEE transactions on pattern analysis and machine intelligence*, 39(7):1476–1481, 2017. 8

[31] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *ICLR*, 2014. 2

[32] A. Shrivastava and A. Gupta. Contextual priming and feedback for faster r-cnn. In *European Conference on Computer Vision*, pages 330–348. Springer, 2016. 8

[33] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–769, 2016. 8

[34] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017. 8

[35] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013. 2

[36] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 8

[37] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer. Imagenet training in minutes. *arXiv preprint arXiv:1709.05011*, 2017. 1, 3, 6

[38] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 8