# Large Batch Optimization for Object Detection: Training COCO in 12 Minutes

Tong Wang[1,2], Yousong Zhu[1,3], Chaoyang Zhao[1], Wei Zeng[4,5], Yaowei Wang[5], Jinqiao Wang[1,2,6], and Ming Tang[1]

[1] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2] School of Articial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3] ObjectEye Inc., Beijing, China
[4] Peking University, Beijing, China
[5] Peng Cheng Laboratory, Shenzhen, China
[6] NEXWISE Co., Ltd, Guangzhou, China
{tong.wang,yousong.zhu,chaoyang.zhao,jqwang,tangm}@nlpr.ia.ac.cn, weizeng@pku.edu.cn, wangyw@pcl.ac.cn

**Abstract.** Most of existing object detectors usually adopt a small training batch size (*e.g.* 16), which severely hinders the whole community from exploring large-scale datasets due to the extremely long training procedure. In this paper, we propose a versatile large batch optimization framework for object detection, named *LargeDet*, which successfully scales the batch size to larger than 1K for the first time. Specifically, we present a novel Periodical Moments Decay LAMB (PMD-LAMB) algorithm to effectively reduce the negative effects of the lagging historical gradients. Additionally, the Synchronized Batch Normalization (SyncBN) is utilized to help fast convergence. With LargeDet, we can not only prominently shorten the training period, but also significantly improve the detection accuracy of sparsely annotated large-scale datasets. For instance, we can finish the training of ResNet50 FPN detector on COCO within 12 minutes. Moreover, we achieve 12.2% mAP@0.5 absolute improvement for ResNet50 FPN on Open Images by training with batch size 640.

**Keywords:** Object detection, Large batch optimization, Periodical moments decay

## 1 Introduction

With the advent of Neural Networks, researchers have made great progress in various domains, including computer vision [11], speech [1], natural language processing [21] and so on. Among the many factors contributing to their success, the available of massive data is definitely one of the most important one. It is widely accepted that large-scale dataset helps the convolutional neural networks

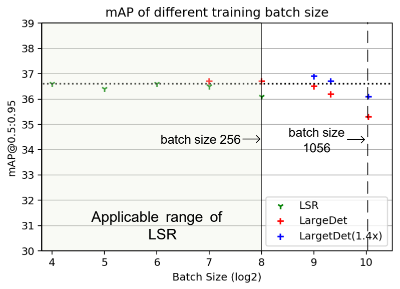

**Fig. 1.** The performance of ResNet50 FPN detector on COCO dataset when using different training batch sizes. LSR means the Linear Scaling Rule (LSR). LargeDet is the proposed framework. 1.4x indicates the training iterations is extended to 1.4 times. The horizontal black dotted line is the performance of baseline with batch size 16

(CNN) generalize well. Consequently, there is a tendency that the size of dataset is growing larger and larger.

Object detection, as a fundamental computer vision task, also benefits from large-scale datasets. From PASCAL VOC [5] to MS COCO [14], the volume of the dataset becomes larger in terms of the number of images and the diversity of categories. Nowadays, datasets like Open Images [12] and Objects365 [20] come in an unprecedented scale. However, previous classical object detectors such as Faster R-CNN [19], Mask R-CNN [7] and RetinaNet [13] *etc.* usually adopt a relatively small mini-batch (16, for example) to train. Such small batch size restrains researchers from making full use of rapidly increased computational power. And the unbearable long training procedure of large-scale datasets severely impedes its development both in research and industry.

Nevertheless, it is not trivial to directly increase the batch size as large batch size training always suffers from performance degradation [8]. A rule of thumb for training neural network is the Linear Scaling Rule (LSR) [10], which suggests that when the batch size becomes K times, the learning rate should also be multiplied by K. However, since the LSR requests the learning rate to grow proportional to the batch size, it has divergence issue when the batch size increases to a certain value, *e.g.* 256. To tackle this issue, [23] proposed Layer-wise Adaptive Rate Scaling (LARS) algorithm to adjust each layer's learning rate based on the norm of its weights and the norm of its gradients. Another similar algorithm is LAMB which is first proposed in [24] for the fast training of BERT [4]. Both LARS and LAMB have the same design philosophy and yield good performance in the large batch training for image classification.

Although researchers have achieved promising results on large batch training for image classification, there are fewer works that concentrate on large batch training for object detection. Since the emergence of those large-scale datasets like Open Images and Objects365, the need to explore large batch training for object detection has become more urgent than ever before. MegDet [17] successfully trains a detector with batch size 256 by using LSR and some other measures like Warm-up and Cross-GPU Batch Normalization, but the exploration to larger batch size still remains blank. *What happens if the batch size exceeds 256? Is there any chance to train a detector with an even larger batch size?*

In this paper, we systematically analyze the large batch optimization and propose a large batch training framework called LargeDet for object detection. To be specific, we first theoretically analyze the problems of LSR and give its applicable range of training batch size. Then, motivated by LAMB [24], we propose a Periodical Moments Decay LAMB algorithm (PMD-LAMB), which can further improve the performance under larger batch size setting. Additionally, we combine the PMD-LAMB with SyncBN to improve the stability and generalization of the network especially when the batch size is extremely large. As showed in Figure 1, with the aid of LargeDet, we successfully scale the batch size from 256 to 1056 with ResNet50 FPN detector on COCO dataset without much accuracy loss, which is the first work to train an object detector using a batch size larger than 1K. The experimental results demonstrate that the proposed LargeDet not only speeds up the training cycle, but also achieves higher accuracy on large-scale datasets, such as Open Images.

In summary, our contributions are three-fold:

1. We propose PMD-LAMB algorithm to effectively eliminate the optimization difficulty of large batch size training, which significantly expands the applicable range of LSR.
2. We present a simple yet effective large batch training framework, named LargeDet, with which we can successfully scale the batch size up to 1056 without much accuracy drop.
3. LargeDet helps us to shrink the training time of ResNet50 FPN on COCO to 12 minutes. Besides, it also helps us achieve 12.2% and 10.8% absolute improvement with ResNet50 and ResNet101 FPN detector respectively on Open Images V5 with much less training time.

## 2 Related Work

### 2.1 CNN-based Detectors

For object detection, current deep learning based approaches can be roughly divided into two categories, single-stage and two-stage approaches. Classical two-stage algorithms, including Faster R-CNN [19], R-FCN [2], Mask R-CNN [7], Deformable Convolutional Networks [3], *etc.* all generate numerous but less accurate proposals in the first stage, and then refine them in the second stage. Single-stage approaches such as YOLO [18], SSD [15], RetinaNet [13], *etc.* work in a more concise manner. They make predictions on the whole feature map directly without proposal generating process, thus enjoying higher speed. Nowadays, some single-stage detectors, like ExtremeNet [25] and FCOS [22] can also achieve competitive results as two-stage detectors.

### 2.2 Large Batch Optimization

In the past few years, large batch size training for image classification has attracted a lot of researchers' attention. Linear Scaling Rule (LSR) has been considered as a rule of thumb for training CNNs, which guarantees roughly equivalent accuracy when using different batch sizes to train. It is mathematically explained in [17, 6] and widely applied in daily practice. By utilizing LSR and Warm-up strategy, [6] is able to scale the batch size to 8192 for ResNet50. [23] proposes Layer-wise Adaptive Rate Scaling (LARS) algorithm to successfully scale the batch size for ResNet50 to 32768. The successor of LARS algorithm is LAMB, which is first proposed in [24]. Both of LARS and LAMB all leverage the norm of weights and the norm of gradients to adjust each layer's learning rate. The only difference lies in that LARS originates from the most commonly used SGD algorithm, while LAMB is a variant of ADAM [9].

Different from the flourish in the research of large batch training for image classification, there are few works which focus on the large batch training for object detection. [17] analyzes the effect of Batch Normalization in object detection and provides a solution for large batch training. Specifically, they implement Cross-GPU Batch Normalization to enable collecting sufficient statistics from more samples, making it possible to train a detector with a large batch size. By combining LSR, Cross-GPU Batch Normalization and Warm-up strategy, they are able to train the Faster R-CNN detector with batch size 256. So far, the object detection with larger batch size is still the terra incognita which deserves to be further researched.

## 3 Method

In this section, we first analyze the problems of Linear Scaling Rule (LSR) for detectors with large batch size. Then we introduce the proposed PMD-LAMB algorithm. Last, we present the object detection framework LargeDet designed for large batch training and propose three practical guidelines.

### 3.1 Problems of Linear Scaling Rule

MegDet [17] is the first successful attempt to train a detector using a large batch size of 256. It mainly adopts the Linear Scaling Rule (LSR) [17, 6] to obtain roughly equivalent accuracy when different batch sizes are adopted. In addition, the Warm-up strategy and Cross-GPU Batch Normalization [17] are further used to guarantee the early convergence of the model and make the training more stable. However, the empirical limits of LSR guideline are not well explored in [17] since the learning rate can not be scaled infinitely. Consequently, it is a problem whether LSR is still suitable for a larger batch size.

Figure 2 shows the experimental results when LSR is applied to different batch sizes. When the training batch size is less than or equal to 256, LSR can maintain almost the same level of accuracy as the 16 mini-batch baseline (with a

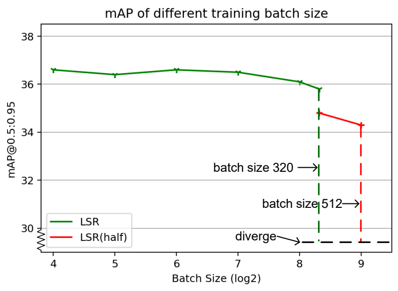

**Fig. 2.** The performance of ResNet50 FPN detector on COCO dataset when we use LSR. Half means we lower the learning rate to a half. The black dotted line represents the network diverges and the accuracy is 0

learning rate of 0.02). Nevertheless, if the training batch size is larger than 256, increasing the learning rate proportionally will lead to performance degradation or even network divergence. The LSR no longer holds. When we compromise to use a lower learning rate, *e.g.* 0.32 for batch size 512, the network will not diverge, but the final accuracy drops dramatically (34.3 *vs.* 36.7). This is mainly because a small learning rate leads to insufficient training, and the model can not converge to an optimal value. Therefore, how to tackle the optimization difficulty is the main issue with the larger batch size training.

## 3.2 Periodical Moments Decay LAMB

As the discussion in 3.1, the main issue which hinders us from using a larger batch size is that the network diverges with a large learning rate. Previous optimization algorithms usually set a global learning rate for the network which is shared by all layers. In fact, the neural network is composed of a great many layers with various feature channel numbers, feature map sizes and weight distributions. This diversity endows each layer diverse tolerance to the learning rate. To make it more intuitive, we calculate the ratio between each layer's weight norm and gradient norm of a ResNet50 FPN detector at iteration 10000. We observe that there are significant differences in the ratios among different layers from Figure 3. When the learning rate is too large, the layers which have less tolerance to the big learning rate collapse at first, and then the whole network diverges. To tackle this issue, [24] proposed LAMB to adaptively adjust each layer's learning rate based on the ratio of its weights' norm and gradients' norm. This algorithm endows each layer a proper learning rate, thus making it possible to train a network with a larger batch size. For LAMB, each update of the network parameters is determined by the exponential moving averages of the gradient ($m_t$) and the squared gradient ($v_t$). $m_t$ and $v_t$ are the estimate of the first and second order moments of the gradients, respectively, as in Equation (1) and (2).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{1}$$

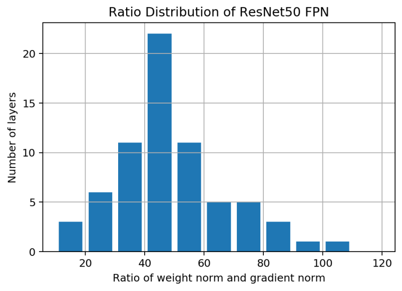

**Fig. 3.** The ratio distribution of ResNet50 FPN detector at iteration 10000

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2}$$

For large batch optimization, what we concern most is the fast convergence of the algorithm. As the batch size grows larger, the training iterations reduce proportionally. How to achieve expected accuracy in limited iterations is the core. For LAMB, the exponential moving average incorporates the knowledge of previously observed data, making the training process more stable. Nevertheless, overindulging in previous gradients may be harmful for fast convergence. With the iterative training, the network becomes more and more intelligent. Thus, the generated gradients become more precise to reflect the correct optimizing direction. While the heavily dependence on previous knowledge limits the network's update towards the more accurate direction. Historical gradients will dominate the training process and the network is prone to get trapped in saddle points. To help the network jump out of the saddle points, we argue to appropriately reduce the strong dependence of current update step on previously acquired information.

**Algorithm 1** Periodical Moments Decay LAMB

```
Require: network parameters $x_1 \in \mathcal{R}^d$, classification loss function $L_{cls}$, box regression
    loss function $L_{reg}$, weight decay $\lambda$, learning rate $\eta$, hyper-parameters $0 < \beta_1, \beta_2 <$
    $1$, scaling factor $\alpha$, $\epsilon > 0$, cycle length $T$
 1: Set $m_0 = 0, v_0 = 0$
 2: for $t = 1$ to $N$ do
 3:     Draw b samples $S_t$ from $\mathcal{P}$.
 4:     Compute $g_t = \frac{1}{|S_t|} \sum_{s_t \in S_t} (\nabla L_{cls}(x_t, s_t) + \nabla L_{reg}(x_t, s_t))$.
 5:     $\xi = \cos(\frac{\pi}{2} \cdot \frac{t\%T}{T})$
 6:     $m_t = \xi \beta_1 m_{t-1} + (1 - \beta_1) g_t$
 7:     $v_t = \xi \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
 8:     Compute update step $r_t = \frac{m_t}{\sqrt{v_t + \epsilon}}$
 9:     $x_{t+1} = x_t - \eta \frac{\alpha ||x_t||}{||r_t + \lambda x_t||} (r_t + \lambda x_t)$
10: end for
```

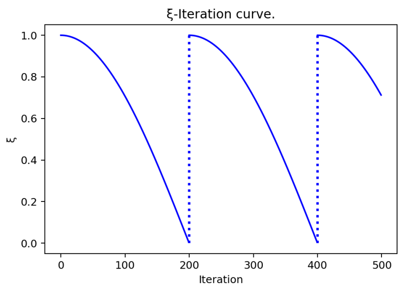

**Fig. 4.** An example of $\xi$-Iteration curve (T=200)

Therefore, we propose a novel periodical moments decay LAMB to reduce the network's heavy dependence on previous acquired knowledge. To be specific, coefficient $\xi$ is introduced to control the dependence on the accumulated moments. As formulated in Equation (4) and (5), $\xi$ is multiplied to the accumulated first- and second-order moments. When $\xi$ equals to one, previously acquired knowledge is fully utilized to participate in current network parameter update. A smaller $\xi$ indicates that the accumulated moments influence less to current update step. We perform such moments decay in a periodical way. The whole training process is divided into several consecutive time windows with cycle length T iterations. In each cycle, the coefficient $\xi$ decays from 1 to 0, as illustrated in Figure 4. Equation (3) formulates the computation of $\xi$, where $t$ indicates the current iteration. T is an introduced hyper-parameter of the cycle length. The pseudo-code of PMD-LAMB is showed in Algorithm 1. To explore the effect of PMD-LAMB, we train two ResNet50 FPN detectors with LAMB and PMD-LAMB, respectively. They are both trained on COCO dataset with batch size 1056. The loss curves are visualized in Figure 5. We can see that these two loss curves tangle with each other at the initial training stage. While the loss curve of PMD-LAMB has lower loss values when the training process levels off. This phenomenon indicates that PMD-LAMB helps the network converge faster and optimize to a better point with the same number of iterations, thus yielding higher accuracy.

$$\xi = \cos(\frac{\pi}{2} \cdot \frac{t\%T}{T}) \tag{3}$$

$$m_t = \xi \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{4}$$

$$v_t = \xi \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{5}$$

There are two vital hyper-parameters in PMD-LAMB. Hyper-parameter T controls the cycle length. If T is set to a small value, the relaxation of current update step on previous moments is performed at a high frequency. Despite our intuition is to reduce the dependence on previous gradients, directly abandoning

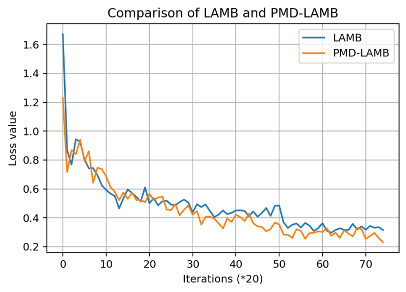

**Fig. 5.** The loss curves of ResNet50 FPN trained with LAMB and PMD-LAMB, respectively

the historical knowledge is inadvisable. A too small cycle length T leads to fluctuated and unstable training process, thus should be avoided. Another is the scale factor $\alpha$. Since the ratio of network layer's weight and gradient is usually large, we need the scale factor $\alpha$ to rescale the actual learning rate for each layer to a proper magnitude. Otherwise, the network will diverge.

## 3.3 LargeDet Framework and Guidelines

With the core of PMD-LAMB, we propose LargeDet for large batch training by combining LSR and SyncBN. In object detection, it is a common practice to adopt frozen BN which inherits the batch statistics from ImageNet pre-trained models and doesn't update the parameters during training. Frozen BN forces the network's convolutional parameters to adapt to the batch statistics inherited from ImageNet. Such adaption usually requires a long training procedure, thus is harmful for fast convergence. Out of this reason, we use SyncBN to allow the network to collect its own data statistics during training. With the aid of LargeDet, we successfully scale the batch size from 256 to 1056. We summary our findings into the following three guidelines.

*G1) LSR works well at a batch size smaller than 256, but is not suitable for a larger batch size.* An intrinsic problem of LSR lies in that the network will diverge under a large learning rate. As LSR suggests, a large learning rate is essential to keep accuracy when a large batch size is applied. While the divergence of network is inevitable when the learning rate is higher than certain threshold. By carefully tunning the warm-up hyper-parameters, one can reduce the risk of divergence and make it work at a larger batch size. However, as the learning rate continues to grow, the network diverges no matter how to change the configurations of warm-up.

*G2) SyncBN is beneficial for large batch optimization.* As analyzed before, frozen BN requires a long training procedure to force the network's convolutional parameters to adapt to the batch statistics inherited from ImageNet. For a larger batch size, the training iterations is reduced proportionally, which gives the network less time to perform this adaption. Insufficient adaption leads to inferior performance. SyncBN enables the network to collect accurate batch statistics from the target dataset. Thus it is beneficial to achieve satisfactory performance in less iterations.

*G3) PMD-LAMB guarantees the convergence of detectors under large learning rate and paves the way for large batch optimization.* PMD-LAMB dynamically adjusts each layer's learning rate based on the ratio of the weights' norm and the gradients' norm. Such layer-wise learning rate successfully addresses the divergence issue of LSR when large batch size is adopted. In addition to this, PMD-LAMB reduces the strong dependence on previous gradients by cyclically decaying the weight of accumulated first and second moments to reduce the negative effect of historical gradients. PMD-LAMB helps the network generalize well and achieves better performance compared to LAMB. In general, these two features make it the key component of our framework.

In summary, LargeDet addresses the divergence issue of LSR and makes it feasible to train detectors with a larger batch size. Consequently, it endows us the ability to fully utilize the increasing computational power to reduce the training time and provides a practical way to tackle with large-scale datasets.

## 4 Experiments

In this section, we conduct comprehensive experiments on MS COCO [14] to validate the effectiveness of our proposed framework. With the aid of LargeDet, we are able to shrink the training time of ResNet50 FPN detector on COCO dataset to 12 minutes and finish the training of Mask R-CNN with ResNet50 backbone in 17 minutes with 160 NVIDIA Tesla V100 GPUs. Besides, we perform experiments on large-scale dataset Open Images [12] as well. The results show that LargeDet is well suited for such large-scale datasets and it outperforms traditional small batch training both in training speed and accuracy.

All our experiments are conducted on the NVIDIA DGX2 server. In terms of software, we use the NVIDIA-optimized maskrcnn-benchmark[7] [16] to train the detectors.

### 4.1 Experiments on COCO

We first perform experiments on the challenging MS COCO2017 benchmark. We use the ∼118K training images to train the detector and the final performance is reported on the 5000 validation images. For evaluation, we use standard COCO metric mAP@0.5:0.95, which averages mAP over IoUs from 0.5 to 0.95. We use the representative ResNet50 pre-trained on the ImageNet as the backbone and adopt the Feature Pyramid Network (FPN) as the detection framework.

For batch size 16, we use a base learning rate 0.02, with the total training iteration of 90000. We multiply the learning rate by factor 0.1 at iteration 60000,

[7] `https://github.com/mlperf/training_results_v0.6/tree/master/NVIDIA/benchmarks/maskrcnn/implementations/pytorch`

and again at 80000. For larger batch size, the global learning rate is set according to the LSR and the training iterations is reduced proportionally to keep the total training epochs constant. For SGD, the hyper-parameters, like weight decay and momentum, follow the default settings in [16]. For LAMB or PMD-LAMB algorithm, we adopt the weight decay 0.0005, $\beta_1$ 0.9, $\beta_2$ 0.999 by default.

**Large Batch Size Training with LSR** We first explore the applicable range of LSR. Specifically, we conduct experiments using different batch sizes with and without SyncBN. When SyncBN is applied, we set the BN size to 32 as suggested in [17]. The results are summarized in Table 4.1.

**Table 1.** The results of LSR with different batch sizes. Half means we lower the learning rate to a half

| Batch Size | w.o. SyncBN | w. SyncBN |
|---|---|---|
| 16 | 36.7 | 36.6 |
| 32 | 36.6 | 36.4 |
| 64 | 36.6 | 36.6 |
| 128 | 36.5 | 36.5 |
| 256 | failed | 36.1 |
| 256 (half) | 35.6 | 35.0 |
| 512 | failed | failed |
| 512 (half) | failed | 34.3 |

For experiments without BN, we observe that when the batch size is smaller than 128, the detector can achieve almost the same accuracy as small batch size (16) training. When the batch size becomes larger, the network fails to converge. SyncBN helps it converge at batch size 256, but still fails at batch size 512. Even if we can use a lower learning rate to make the network converge, it yields severe performance drops. Although we follow the settings as [17], the results we obtained is slightly different. We conjecture that the difference may be caused by different implementation details of SyncBN. It is worth noticing that when the batch size is larger than 64, the warm-up hyper-parameters need to be carefully tuned to avoid divergence whether there is SyncBN or not. Above experiments demonstrate that LSR can not handle a larger batch size. Thus it is of great significance to explore larger training batch size.

**Larger Batch Size Training with LargeDet** As mentioned above, when the batch size is not that large (below 256), the LSR still holds with the help of SyncBN. However, it fails when the batch size becomes larger. To tackle this issue, we conduct experiments using LargeDet with different batch sizes. Since the performance with large batch size is our concern, we start our experiments from batch size 128.

**Table 2.** Comparisons of training with different batch sizes and methods

| Batch Size | GPUs | Method | mAP |
|---|---|---|---|
| 16 | 8 | SGD | 36.7 |
| 128 | 64 | LAMB | 36.0 |
| | | PMD-LAMB | 36.7 |
| 256 | 64 | LAMB | 36.2 |
| | | PMD-LAMB | 36.7 |
| 512 | 64 | LAMB | 35.5 |
| | | PMD-LAMB | 36.5 |
| | | PMD-LAMB (1.4x) | 36.9 |
| 640 | 64 | LAMB | 35.6 |
| | | PMD-LAMB | 36.2 |
| | | PMD-LAMB (1.4x) | 36.7 |
| 1056 | 96 | LAMB | 34.8 |
| | | PMD-LAMB | 35.3 |
| | | PMD-LAMB (1.4x) | 36.1 |

We summary the main results in Table 4.1. The result of batch size 16 is the baseline. From the experimental results, we have the following observations. First, our proposed PMD-LAMB algorithm effectively solves the divergence issue when large batch size is applied and successfully expands the applicable range of LSR. As showed in Table 4.1, we successfully scale the batch size up to 1056. Besides, our proposed PMD-LAMB algorithm outperforms LAMB on different batch sizes. For batch size 128 and 256, the detector can recover the baseline performance by using PMD-LAMB without extending the training iterations. For larger batch size, PMD-LAMB can still provide consistent performance gain, 1% improvement at batch size 512 compared to LAMB, which shows the proposed PMD-LAMB is an effective method to further improve the performance.

Second, by slightly extending the training time, the performance can be further improved. For a relatively large batch size, it is hard to recover the performance in the same training epochs as the small batch size training baseline. For batch size 512 and 640, it still suffers little accuracy drop. And for batch size 1056, the accuracy drop is more, 1.4% lower than the baseline. We argue that this is mainly due to the inadequate statistics of BN with less iterations. By slightly increasing the training iterations, 1.4 times in our experiments, the performance can be further improved. For batch size 512, 1.4x training iterations yield 36.9 mAP, even higher than the small batch size training accuracy. In summary, our proposed framework is an effective approach to train an object detector when using a large batch size.

**The Effect of Hyper-parameters** In PMD-LAMB, there are two hyper-parameters introduced. One is the scale factor $\alpha$, the other is the cycle length T. We perform ablation experiments to explore how they affect the performance. In the ablation study, we use LargeDet to train the detector with batch size 256.

**Table 3.** The effect of hyper-parameters in PMD-LAMB

| $\alpha$ | 0.012 | | | | | 0.008 | 0.01 | 0.012 | 0.015 | 0.018 |
|---|---|---|---|---|---|---|---|---|---|---|
| T | 10 | 200 | 500 | 1000 | $\infty$ | 200 | | | | |
| mAP | 35.6 | 36.7 | 36.7 | 36.6 | 36.2 | 36.4 | 36.5 | 36.7 | 36.2 | 35.2 |

The results are summarized in Table 4.1. We can see that our method works well in a quite wide range for T. But a very small value will lead to obvious accuracy loss. For scale factor $\alpha$, it should be set to a proper value. A small $\alpha$ leads to insufficient training and results in inferior performance. While a large $\alpha$ causes a fluctuated training process, which deteriorates the final performance as well. From the results in Table 4.1, we can know that setting $\alpha$ to 0.012, T to 200 is a good choice.

## 4.2 Training COCO in 12 Minutes

With LargeDet, we are able to challenge the limit of fast training of COCO. We choose Faster R-CNN with FPN and Mask R-CNN detectors and explore how fast can we train a relatively strong detector. We adopt the ResNet50 as the backbone. For Faster R-CNN, when the detection mAP@0.5:0.95 reaches 36.6, we stop the training. For Mask R-CNN, the threshold is 37.6 detection mAP@0.5:0.95 and 33.9 segmentation mAP@0.5:0.95. We conduct experiments on 10 NVIDIA DGX-2 servers, 160 GPUs in total.

**Table 4.** The results of fast training experiments on COCO. Faster means Faster R-CNN with FPN. Mask represents the Mask R-CNN detector. Steps indicates when we lower the learning rate by factor 0.1

| | Steps | Stop iter | Stop mAP | Time |
|---|---|---|---|---|
| Faster | 2400,3200 | 3360 | box:36.65 | 11m53s |
| Mask | 3000,4000 | 4120 | box:37.61 seg:34.98 | 16m55s |

The total batch size is set to 320 for both Faster R-CNN and Mask R-CNN detectors. We set the global learning rate to 0.4 according to the LSR. Table 4.2 shows the results of fast training experiments. We are able to finish the training of Faster R-CNN in 12 minutes, and Mask R-CNN in 17 minutes, which will substantially accelerate the experimental cycle and facilitate the exploration of more effective strategies.

## 4.3 Experiments on Open Images

Open Images V5 is the largest existing dataset with object location annotations which contains a total of 16M bounding boxes for 600 object classes on 1.9M images. It is split into three subsets, train, validation and test subset. We use the train subset which contains 1.7M images to train and report the accuracy on validation subset. For evaluation, we use the official evaluation code for Open Images which evaluates only at IoU threshold 0.5.

In this section, we conduct experiments on Open Images V5 with batch size 16, 320 and 640, and compare the training time and accuracy. We adopt the FPN detector with ResNet50 and ResNet101 backbone. For experiment with batch size 16, the learning rate is set to 0.02 and other settings follows the default configurations for COCO. For large batch size 320 and 640, the learning rate is set to 0.4 and 0.8, respectively. We train the detector for 12 epochs, and multiply the learning rate by 0.1 and 0.01 at epoch 8 and 11, respectively. To make a fair comparison, we do not modify other settings such as anchor ratio and anchor number, *etc.* And we adopt multi-scale training for all experiments.

**Table 5.** Experiments on Open Images V5 with different batch sizes and backbones

| Backbone | BatchSize | GPUs | mAP@0.5 | Time |
|---|---|---|---|---|
| Res50 | 16 | 8 | 42.9 | 87h |
| | 320 | 32 | 53.1 | 23h |
| | 640 | 64 | *55.1* | 12h |
| Res101 | 16 | 8 | 45.5 | 108h |
| | 320 | 32 | 54.0 | 27h |
| | 640 | 64 | *56.3* | 14h |

The results are summarized in Table 4.3. We achieve a nearly linear acceleration by utilizing more GPUs. With LargeDet, we reduce the training time of ResNet50 from 87 hours to 12 hours, approximately 8× acceleration. The same acceleration can be observed for ResNet101. Except for reducing the training time, we also obtain a considerable performance improvement when training with a larger batch size. For ResNet50 backbone, by using a batch size of 640, we achieve 12.2% mAP@0.5 absolute improvement, compared to the baseline trained with batch size 16. For ResNet101, we can achieve 10.8% improvement as well. We also observe that the improvements brought by LargeDet increase as the batch size grows.

We conjecture the performance gain may come from the following two aspects. On the one hand, large batch size increases the category richness in a mini-batch. Open Images V5 has object location annotations for 600 classes. According to official statistics, there are 8.4 boxes objects per image on average. If the batch size is 16, there are 135 objects in a mini-batch on average. Even if all these boxes are from different categories, the number of the total categories

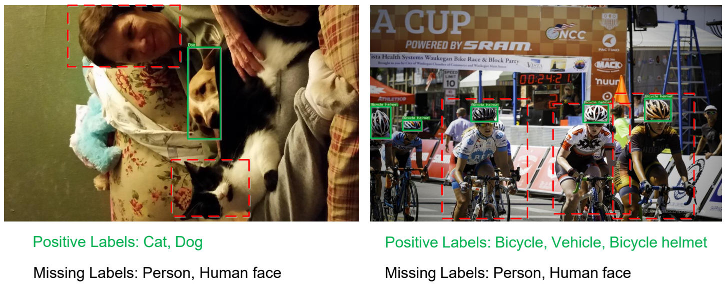
Positive Labels: Cat, Dog
Missing Labels: Person, Human face
Positive Labels: Bicycle, Vehicle, Bicycle helmet
Missing Labels: Person, Human face

**Fig. 6.** Example annotations in Open Images V5. Green boxes indicate the annotated ground truth. Red boxes are some missing instances which are not annotated in Open Images. Positive labels indicate some object classes are present in current image. Missing labels are those classes which are present in the image but no corresponding instances are annotated

in a mini-batch is merely 135, which is far lower than the total category 600. By increasing the batch size, the category richness can be greatly improved and the detector can achieve better performance.

On the other hand, large batch optimization is an effective strategy to fight against the label noise which is widespread in large-scale datasets. For instance, missing annotations frequently occur in Open Images, as illustrated in Figure 6. During training, such missing instances are prone to be mistaken for negative samples, which will cause incorrect training signals and confuse the detector. By increasing batch size, these wrong signals can be suppressed to a great extent by numerous correctly annotated instances.

## 5 Conclusions

We present a large batch optimization framework for object detection called LargeDet, with which we successfully scale the batch size to 1056 with a ResNet50 backbone. Besides, LargeDet supports us to finish the training of ResNet50 FPN detector on COCO in 12 minutes. Moreover, large batch training provides a novel means to tackle the widespread label noise in large scale datasets. By training ResNet50 FPN with a batch size of 640, we achieve a remarkable 12.2% absolute improvement in terms of mAP@0.5 on Open Images V5. And the improvement for ResNet101 FPN is 10.8%. We hope that our research can serve for the future exploration of the large-scale datasets.

## Acknowledgement

This work was supported by the Research and Development Projects in the Key Areas of Guangdong Province (No.2019B010153001) and National Natural Science Foundation of China under Grants No.61772527, No.61976210, No.61806200, No.61702510, No.61876086 and No.61633002.

## References

1. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Process. Mag. **29**(6), 82–97 (2012). https://doi.org/10.1109/MSP.2012.2205597, `https://doi.org/10.1109/MSP.2012.2205597`
2. Dai, J., Li, Y., He, K., Sun, J.: R-FCN: object detection via region-based fully convolutional networks. In: Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain. pp. 379–387 (2016), `http://papers.nips.cc/paper/6465-r-fcn-object-detection-via-region-based-fully-convolutional-networks`
3. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017. pp. 764–773 (2017). https://doi.org/10.1109/ICCV.2017.89, `https://doi.org/10.1109/ICCV.2017.89`
4. Devlin, J., Chang, M., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). pp. 4171–4186 (2019), `https://www.aclweb.org/anthology/N19-1423/`
5. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J.M., Zisserman, A.: The pascal visual object classes (VOC) challenge. International Journal of Computer Vision **88**(2), 303–338 (2010). https://doi.org/10.1007/s11263-009-0275-4, `https://doi.org/10.1007/s11263-009-0275-4`
6. Goyal, P., Dollár, P., Girshick, R.B., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., He, K.: Accurate, large minibatch SGD: training imagenet in 1 hour. CoRR **abs/1706.02677** (2017), `http://arxiv.org/abs/1706.02677`
7. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017. pp. 2980–2988 (2017). https://doi.org/10.1109/ICCV.2017.322, `https://doi.org/10.1109/ICCV.2017.322`
8. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings (2017), `https://openreview.net/forum?id=H1oyRlYgg`
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), `http://arxiv.org/abs/1412.6980`
10. Krizhevsky, A.: One weird trick for parallelizing convolutional neural networks. CoRR **abs/1404.5997** (2014), `http://arxiv.org/abs/1404.5997`
11. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States. pp. 1106–1114 (2012), `http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks`

12. Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J.R.R., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Duerig, T., Ferrari, V.: The open images dataset V4: unified image classification, object detection, and visual relationship detection at scale. CoRR **abs/1811.00982** (2018), http://arxiv.org/abs/1811.00982
13. Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P.: Focal loss for dense object detection. In: IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017. pp. 2999–3007 (2017). https://doi.org/10.1109/ICCV.2017.324, https://doi.org/10.1109/ICCV.2017.324
14. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V. pp. 740–755 (2014). https://doi.org/10.1007/978-3-319-10602-1_48, https://doi.org/10.1007/978-3-319-10602-1_48
15. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., Berg, A.C.: SSD: single shot multibox detector. In: Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part I. pp. 21–37 (2016). https://doi.org/10.1007/978-3-319-46448-0_2, https://doi.org/10.1007/978-3-319-46448-0_2
16. Massa, F., Girshick, R.: maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark (2018)
17. Peng, C., Xiao, T., Li, Z., Jiang, Y., Zhang, X., Jia, K., Yu, G., Sun, J.: Megdet: A large mini-batch object detector. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 6181–6189 (2018). https://doi.org/10.1109/CVPR.2018.00647, http://openaccess.thecvf.com/content_cvpr_2018/html/Peng_MegDet_A_Large_CVPR_2018_paper.html
18. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016. pp. 779–788 (2016). https://doi.org/10.1109/CVPR.2016.91, https://doi.org/10.1109/CVPR.2016.91
19. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. IEEE Trans. Pattern Anal. Mach. Intell. **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031, https://doi.org/10.1109/TPAMI.2016.2577031
20. Shao, S., Li, Z., Zhang, T., Peng, C., Yu, G., Zhang, X., Li, J., Sun, J.: Objects365: A large-scale, high-quality dataset for object detection. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
21. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada. pp. 3104–3112 (2014), http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks
22. Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: The IEEE International Conference on Computer Vision (ICCV) (October 2019)
23. You, Y., Gitman, I., Ginsburg, B.: Scaling SGD batch size to 32k for imagenet training. CoRR **abs/1708.03888** (2017), http://arxiv.org/abs/1708.03888

24. You, Y., Li, J., Hseu, J., Song, X., Demmel, J., Hsieh, C.: Reducing BERT pre-training time from 3 days to 76 minutes. CoRR **abs/1904.00962** (2019), http://arxiv.org/abs/1904.00962
25. Zhou, X., Zhuo, J., Krahenbuhl, P.: Bottom-up object detection by grouping extreme and center points. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)