Journal of Machine Learning Research 24 (2023) 1-14 Submitted 6/23; Published

CoolDet: Cooling-down in Training Adaptive Rate Large Batch for Object Detection

Ilham Syahid Syamsudin 23522033@STD.STEI.ITB.AC.ID
School of Electrical Engineering and Informatics
Institut Teknologi Bandung
Bandung, West Java, Indonesia

Achmad Imam Kistijantoro IMAM@ITB.AC.ID
School of Electrical Engineering and Informatics
University Center of Excellence on Artificial Intelligence for Vision, Natural Language Processing & Big Data Analytics (U-CoE AI-VLB)
Institut Teknologi Bandung
Bandung, West Java, Indonesia

Editor: -

Abstract

LARS, an earlier study on large batch optimization settings, proposed an adaptive learning rate and successfully train in large batch settings without significant accuracy loss on image classification models (e.g., ResNet-50 and AlexNet). However, this optimization has not been well-studied for other task like object detection. In this paper, we propose a new large batch optimization framework for object detection using LARS, named CoolDet. We have also discovered that local learning rates (LR) by LARS in object detection increase as the step grows. Specifically, we suggest using warm-up, SyncBN, and cosine annealing. Cosine annealing is used to decrease the global LR in order to stabilize training due to local LR growth. By combining these techniques with warm-up and SyncBN, we have been able to train the COCO dataset using Mask-RCNN and obtained an mAP of 38.68, which exceeds previous studies with the same settings.

Keywords: large batch optimization, LARS, distributed training, cosine annealing, object detection

1 Introduction

In recent years, the progress made in deep neural networks has been truly impressive, particularly in the field of computer vision. This advancement has allowed machines to achieve an unprecedented level of accuracy in comprehending and interpreting visual data. Object detection, a fundamental component, has played a significant role in supporting a wide range of applications such as autonomous driving, surveillance systems, and augmented reality (Pathak et al., 2018). One of supporting aspects is huge amounts of data that helps convolutional neural networks (CNN) performs well.

In other side, training of large network with amounts of data takes a lot of time. As the complexity and scale of visual data continue to grow exponentially, traditional machine learning approaches often face limitations in processing power and scalability. To address

these challenges, data parallelism has emerged as a powerful paradigm, enabling the training models across a network of interconnected computing nodes and data (Wang, 2022).

The trivia way to speed up CNN training is to add more computational unit (e.g. more GPU nodes) and train network using data-parallel. It is important to ensure that the chunk size assigned to each worker is sufficiently large to utilize of computational resources, otherwise, it would be useless. Consequently, as the number of workers is scaled up, the batch size increases. However, as observed by Krizhevsky (2014), it is worth nothing that using a large batch size can have a detrimental effect on the model's accuracy.

Linear Scaling Rule (LSR) proposed by Krizhevsky (2014) which when the batch size is increased by a factor of K, the Learning Rate (LR) should also be scaled by the same factor K. Applying using larger LR makes more difficult and sometimes diverge especially on the initial phase. To solve this problem, Goyal et al. (2017) suggested using warm-up strategy that starting LR with small number and gradually increased to base LR.

Furthermore, You et al. (2017a) proposed Layer-wise Adaptive Rate Scaling (LARS) algorithm that introduced local LR for adapting each layer's learning rate based on the norm of its weights and the norm of its gradients. This ratio varies a lot between different layers, which makes it necessary to use a separate LR for each layer and leads to better stability. Similar algorithm with same philosophy and good performance in the large batch training is LAMB, the algorithm that proposed by You et al. (2020) for training BERT (Devlin et al., 2019).

Despite the successfully results for training large batch in image classification task, there are still lack of works that focus on training large batch for object detection. Some of works proposed using LSR, and other optimization such as warm-up and Sync Batch Normalization (SyncBN) done by MegDet (Peng et al., 2018) and LargeDet (Wang et al., 2020). MegDet use SGD and enables training up to batch size 256 using 128 GPU with warm-up and SyncBN. LargeDet introduced PMD-LAMB, LAMB-based algorithm, that also successfully trains network up to batch size 1056 using 160 GPU. As comparison, the original from the object detector R-CNN series involving only 2 images per batch. However, this paper will use LARS an SGD-based optimizer that shares the same philosophy as LAMB.

One of the gradient-based optimization to improve rate of convergence are using warm restarts. Loshchilov and Hutter (2017) proposed Stochastic Gradient Descent with Warm Restarts (SGDR), a simple warm restart technique that utilize cosine annealing for SGD and leads to decreased test error on CIFAR-10 and CIFAR-100 dataset using single model or ensemble model (Huang et al., 2017). We will use warm restarts and cosine annealing to improve our convergence rate training.

Table 1: Comparisons of best mAP

	GPU	mAP	Time
LargeDet (Wang et al., 2020)	160	37.61	17m
MegDet (Peng et al., 2018)	8	37.8	60h 18m
CoolDet (ours)	8	38.68	4h 21m

In this paper, we analyze local LR that produced by LARS and found that local LR will be increased as the model has more layers. If global LR increased as epoch this could be lead to divergence. On the other hand, if global LR still remains same this would instability. We will be using SGDR to tackle the issue. To be specific, we proposed CoolDet that first using warm-up strategy to prevent divergence on the initial phase. Then using LARS to improve model performance under larger batch size setting. We also combined with SyncBN to increase generalization and robustness of the network in large batch size training. Finally we are using SGDR with cosine annealing-based algorithm to improve stability and convergence rate in model gradient as well as improving model performance. We named CoolDet after it gradually decrease LR just like cooling-down phase. As stated in Table 1, CoolDet successfully exceeds previous methods got 38.68 with 4 hours on 8 GPUs. In contrast, other method with same amount of GPUs received 37.8 mAP for 60 hour training. Since we have some limitation, we only can use 8 GPUs for training.

In summary, our contributions can be stated as:

1. We are the first that train object detector using LARS and found most of local LR is increasing as epoch grows, while this could leads to instability or even diverge training.

2. We present combined framework to train on object detection and improve convergence rate in training large batch, named CoolDet, and successfully achieving higher accuracy. We demonstrate use cooling-down cosine annealing to improve stability and performance model.

3. We effectively utilize 8 GPU to finish COCO in 4 hours and scale up to 128 batch size without signification accuracy drop. We provide the source code in here: `https://github.com/ilhamsyahids/cooldet`

2 Related Work

2.1 CNN-based Detectors

The present deep learning based methodologies for object detection may be divided into two major categories, which are single-stage and two-stage strategies. In the event of classical two-stage algorithms, like Faster R-CNN (Ren et al., 2016), R-FCN (Dai et al., 2016), Mask R-CNN (He et al., 2017), among others, generating a significant number of proposals in the first stage, although with relatively low accuracy, these are subsequently refined in the second stage.

Unlike other approaches, YOLO (Redmon et al., 2016), SSD (Liu et al., 2016), RetinaNet (Lin et al., 2018), and others perform efficiently with a more straightforward method. These make predictions directly on the entire feature map without the requirement of proposal generation, thereby delivering superior speed. In the present era, certain single-stage detectors have been capable of yielding comparable results to their two-stage counterparts.

2.2 Large Batch Optimization

Linear Scaling Rule and Warmup. In recent years, the training of image classification models using large batch sizes has garnered considerable attention from researchers. The

Linear Scaling Rule (LSR) has been widely adopted as a standard practice for training CNNs, as it ensures that the accuracy of models trained using different batch sizes remains relatively constant. The LSR has been mathematically explained in Goyal et al. (2017); Peng et al. (2018) and is commonly used in daily practice.

Training with large mini-batch size usually requires large learning rate as well as mentioned in LSR. This method could leads divergence in early epoch. Warm-up strategy has been extensively demonstrated as a solution as mention by Goyal et al. (2017); You et al. (2017a); Peng et al. (2018); Wang et al. (2020). Warm-up train using small LR and gradually increase to target LR. By adopting the warm-up strategy in conjunction with LSR, Goyal et al. (2017) was able to scale the batch size of ResNet-50 to 8192. Although it has been demonstrated in image classification task, but the use of large learning rate in object detection most likely leading to failure due to divergence.

A noteworthy study conducted by You et al. (2017a) introduced the Layer-wise Adaptive Rate Scaling (LARS) algorithm, which successfully scaled the batch size of ResNet-50 to 32768. As successor of the LARS algorithm, You et al. (2020) proposed LAMB. Both LARS and LAMB leverage same philosophy which is the norms of weights and gradients to adjust the learning rate of each layer. The primary difference between the two algorithms is that LARS originates from the commonly used Stochastic Gradient Descent (SGD) algorithm, while LAMB is a variant of the ADAM algorithm (Kingma and Ba, 2017).

Sync Batch Normalization. Batch Normalization (Ioffe and Szegedy, 2015) is one of important technique for normalizing layer inputs and also acts as a regularizer. Originally this allows us to handle internal covariate shift. In distributed context, batch data in one node might not be relevant for normalizing. Additionally for object detection, the network detector needs to handle objects various scales and resolution. To handle this, we have to collect statistics from other node and synchronize it. MegDet (Peng et al., 2018) have been analyzed and successfully using Cross-GPU Batch Normalization (SyncBN) to perform batch normalization across multiple GPU to collect sufficient statistics from more samples, making it possible to train a detector with a large batch size and received higher accuracy.

3 Methods

In this section, we first analyze the problems with LARS on relatively large layer detectors. Then we combined warm-up, SyncBN, and cosine annealing to handle that problems. Last, we present CoolDet as object detection framework that finish training in relatively less time and achieving higher accuracy.

3.1 Problems with LARS

LARS (You et al., 2017a) has been successfully train ResNet-50 and AlexNet in large batch size settings without losing accuracy significantly. It also adopted warm-up and batch norm to further improve the network as it increase performance. However, LARS are not well explored with scaling more nodes and still use benchmarking model which relatively small in layers. Problem with scaling has been further explored in You et al. (2017b); Mikami et al. (2018) which scale into x2048 KNL and x3456 Tesla V100 respectively and achieve notable results. But still using ResNet-50 as base model. While LAMB, the successor of

LARS, been using in Wang et al. (2020) as base algorithm, but in contrast utilizing other models in LARS have remains very small literature.

As Figure 1, we argue that in two-stage object detector, local LR mean will be increased. This changes is different with original local LR LARS and may lead to instability. Based on LARS algorithm:

$$\Delta w_t^l = \gamma * \lambda^l * \nabla L(w_t^l) \tag{1}$$

$$\lambda^l = \eta \times \frac{\|w^l\|}{\|\nabla L(w_t^l)\|} \tag{2}$$

As λ^l increasing, network will be raised as well, it would leads to training unstable. Lowering γ could be the solution to stabilize the training. Therefore, we will be using LARS and Mask-RCNN to know further about local LR updates and tackle the problem using some of our methods. This also allows us to better understanding for further adaptive training as LARS become first adaptive LR in this area.

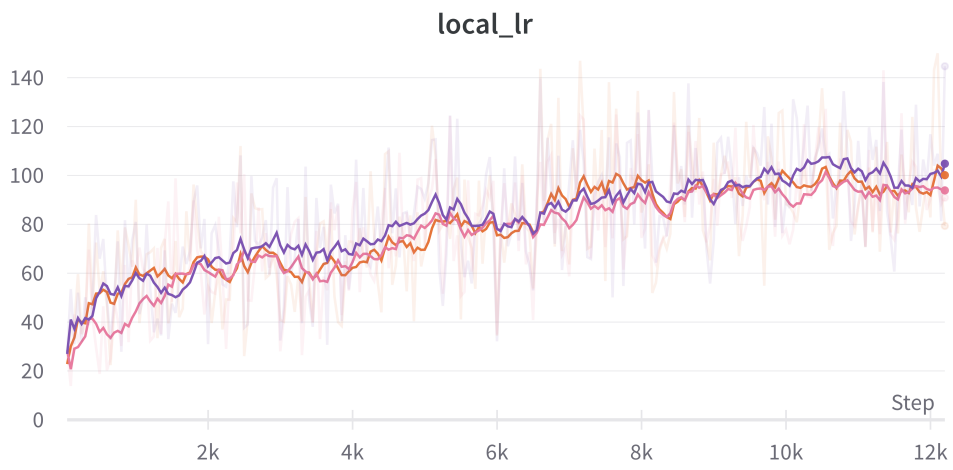

Figure 1: Average of local LR on network detectors

3.2 Warm Restart

As argued in 3.1 and based on Wang et al. (2020), the network that composed by many layers with various feature channel numbers, feature map sizes and weight distributions may endows each layer diverse tolerance to the learning rate may leads divergence. One of the idea to tackle this is by using LR scheduler. To be specific, we borrowed warm restart technique proposed by Loshchilov and Hutter (2017).

Warm restart technique is using cosine annealing as base algorithm:

$$\eta_t = \eta_{min}^i + \frac{1}{2}(\eta_{max}^i - \eta_{min}^i)(1 + cos(\frac{T_{cur}}{T_i}\pi)) \tag{3}$$

LR will in range of η^i_{max} and η^i_{min}, T_{cur} represented as step or epoch since last restart and T_i is step or epoch that will be restart. The idea is opposite to warm-up which is initially LR will have max value (η_{max}) and will gradually decreased to min LR and restart to max value at that epoch. Noted that if T_i is max epoch that means we do not use restart technique instead only decreasing the LR till the minimum at the end of training.

This technique has been used in Wide Residual Network (Zagoruyko and Komodakis, 2017) which is ResNet-based and successfully lowering test error in CIFAR-10 and CIFAR-100 dataset from 6.43% and 25.16% till 3.14% and 16.21% respectively.

3.3 CoolDet Framework

We proposed new framework for training large batch combining LARS with warm-up, SyncBN and cosine annealing named CoolDet. This combination is based on previous research and give the remarkable results. Warm-up used for prevent divergence in early training and SyncBN to collect statistics data during training. Cosine annealing to allows us decrease LR as we analyzed that local LR increase in object detection task.

In summary, CoolDet addresses coverage of using LARS in object detection. We found that local LR is rapidly raise and leads to unstable and diverge training. To tackle this, we introduce using warm restart, cosine annealing based algorithm.

4 Experiments

In this section, we conduct experiments using COCO 2017 dataset (Lin et al., 2014) to validate our framework. We are training over 118K train data and evaluate 5K validation data scattered into 91 categories. For evaluation, we use standard COCO metric mAP@0.5:0.95, which averages mAP over IoUs from 0.5 to 0.95. For model, we follow Peng et al. (2018); Wang et al. (2020) settings to fairly comparisons which we use ResNet-50 (He et al., 2015) pre-trained on ImageNet (Deng et al., 2009) as the backbone network and Feature Pyramid Network (FPN) (Lin et al., 2017) as the detection framework. All of the experiments are maximum to 10 epoch and conducted on NVIDIA DGX-1 Server unless mention other.

4.1 Experiments Baseline

We first training with baseline settings to reference other experiments. We use the SGD optimizer with momentum 0.9, and adopts the weight decay 0.0001. The base learning rate for mini-batch size 16 is 0.02. In this experiments, we explored warm-up epoch, LR scheduler and we take note on mAP and time. LR scheduler that we use are multistep, cosine annealing and no scheduler (no changes on LR). The results are summarized in Table 2.

We can observe that:

1. No changes or increasing LR as conducted in no scheduler and multistep with gamma 2 have higher mAP. But still not even comparable to previous research (33.92 vs 36.7 (Wang et al., 2020) vs 36.2 (Peng et al., 2018)). Unexpectedly, we traing 90% faster than MegDet (3h vs 33h).

Table 2: Baseline evaluation results

Scheduler	warmup	gamma	step	mAP_{bbox}	mAP_{segm}	time
No Scheduler	1	-	-	**33.92**	30.52	3h 23m
	3	-	-	33.65	30.32	3h 23m
Multistep	1	0.1	5, 8	30.45	27.24	3h 24m
	1	2	5, 8	33.70	**30.65**	3h 23m
	3	2	5, 8	33.39	30.47	3h 23m
Cosine Annealing	1	-	-	30.83	27.52	3h 35m
	-	-	-	28.87	26.35	3h 26m

2. Warm-up with epoch 1 have better mAP than warm-up until epoch 3. Thus, we will be use warm-up only in epoch 1.

4.2 Experiments LARS

We conduct this experiments as baseline settings except using LARS as optimizer and the LR. When using LR 0.02, we are not able to manage to make training converge even with warm-up. We also perform cosine annealing restart with cycle on every 2 epoch. We summary the results in Table 3 and Figure 2 denoted as LR changes over step.

Table 3: LARS evaluation results

Scheduler	LR	warmup	gamma	step	mAP_{bbox}	mAP_{segm}	time
No Scheduler	0.001	-	-	-	-	-	failed
	0.0003	-	-	-	32.20	30.76	3h 33m
	0.0003	1	-	-	34.54	32.26	3h 29m
Multistep	0.001	1	0.5	3,5,8	36.32	33.35	3h 29m
	0.001	1	0.75	3,5,8	35.66	32.93	3h 29m
	0.001	1	0.75	5,8	35.58	32.87	3h 31m
	0.001	1	2	5,8	18.84	18.48	3h 30m
	0.001	1	0.1	1	29.36	28.73	3h 32m
Cosine Annealing	0.001	-	-	-	-	-	failed
	0.0003	-	-	-	31.96	30.64	3h 43m
	0.002	1	-	-	-	-	failed
	0.001	1	-	-	**36.93**	**33.82**	3h 33m
With Restart	0.001	1	2	-	-	-	failed
	0.001	1	0.5	-	36.41	33.06	3h 33m

From the experiments results, we have the following observations. First, decreasing LR have higher mAP than no changes or increasing LR and higher mAP is in using cosine annealing. Multistep with gamma 0.5 with step on 3, 5, 8 epoch along with cosine annealing are higher than others and with slightly different mAP. We found that these 2 settings have same LR changes as in Figure 2. This observations is different from previous baseline experiments and we argue due to increasing local LR. Second, warm-up still proven to be

good in handling divergence and improve performance. Third, cosine annealing exceeds baseline MegDet and LargeDet (36.93 vs 36.2 vs 36.7). Thus, we will be using cosine annealing with warm-up for next experiments.

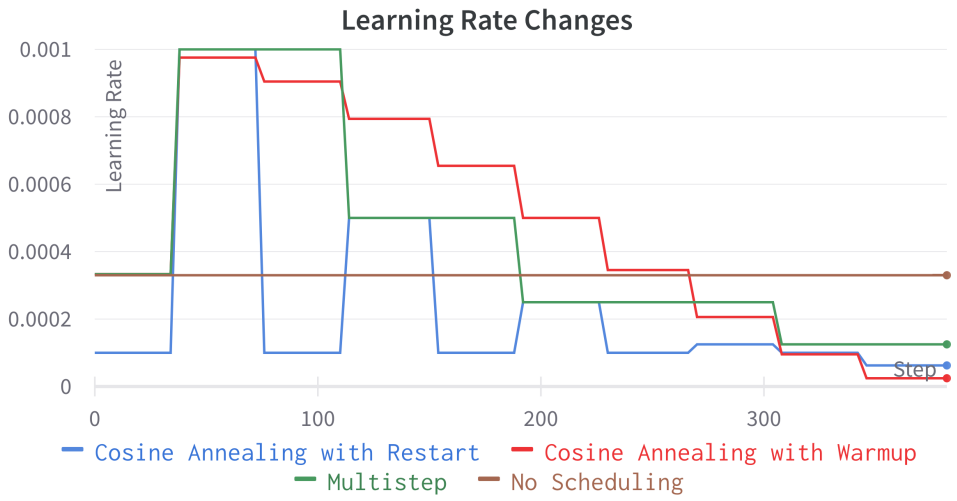

Figure 2: LR changes using LARS

LR Hyperparameters Cosine Annealing Previous experiments conduct in cosine annealing are not further explore with LR. In this experiments, we try to compare some of LR which is and 0.00033, 0.0007, 0.0008, 0.0009, 0.001, 0.0015, and 0.0016 and use them for next experiments. The results are summarized in Figure 3. We can observe that LR with 0.0008 surprisingly exceeding other mAP in 38.02.

4.3 Experiments Local LR

This experiments mainly to observe local LR changes on detectors. We collect all the local LR in each layer and average it. There are around 225 layers with 183 Backbone, 34 RoI Head, and 8 RPN. To reduce bias, we perform three times experiments. The results in stated in Figure 1.

We found that for every run, there is increasing trend of local LR average. Local LR average start with 20 and gradually increase to 100 in the end of training. We observed that all the layers in RPN (Figure 4) and half of RoI Head (Figure 5) layers are raising. For LARS, local LR is important for updating weight and involving in evaluation metrics. Empirically, as in Equation 1, if γ and λ^l higher than normally, this could lead training unstable as observed in 4.2. Therefore, strategy to gradually lower LR could improve stability and performance.

4.4 Experiments Batch Size and SyncBN

After we found that lowering LR gradually helps stability of the network, we conduct experiments of large batch size and using SyncBN. For this experiments, we change our

Figure 3: Cosine Annealing LR hyperparameters

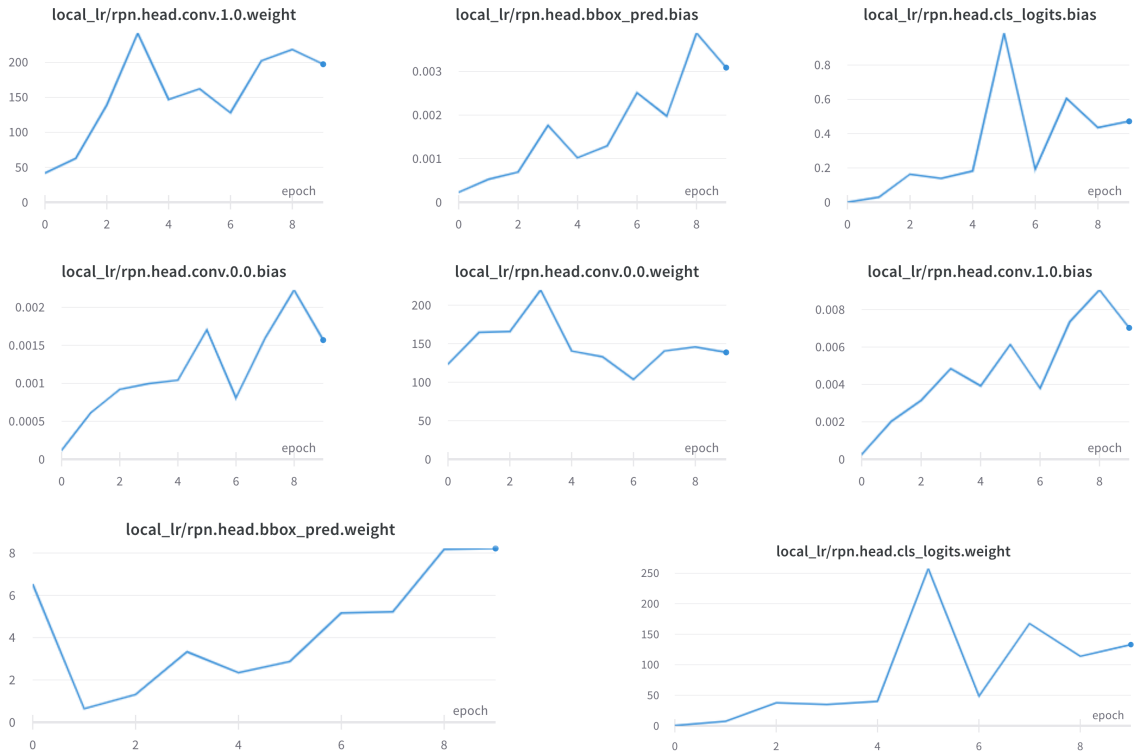

Figure 4: Local LR of RPN Layer. All of the layers observed to be increasing

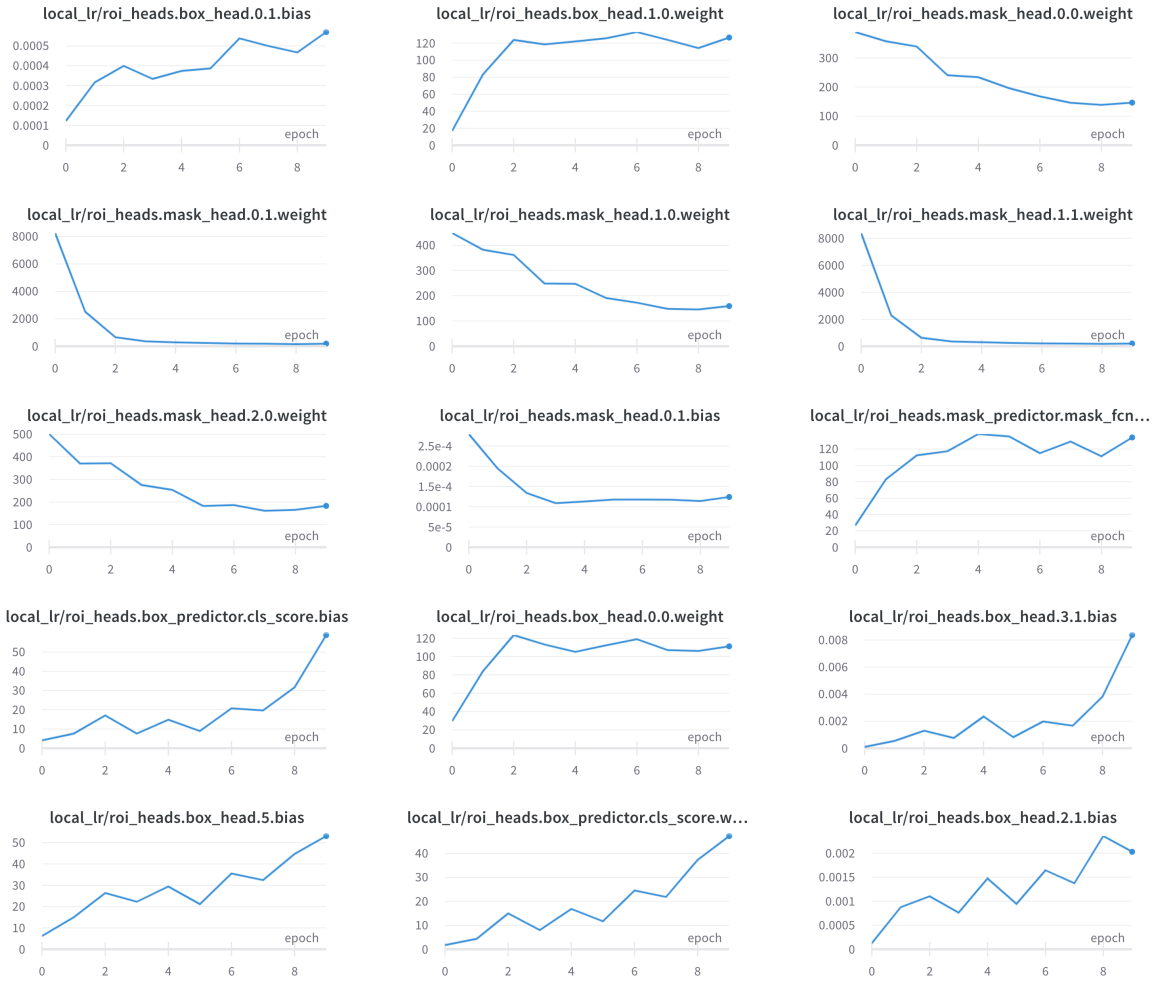

Figure 5: Sample local LR of RoI Head. We observed only 13 out of 34 that decreasing, remains are increasing

infrastructure to use AWS p3.16xlarge due to our limitation of previous server. We are using two machine with each has 8 GPU Tesla V100 16GB. Table 4 denoting as experiments without using SyncBN and Table 5 experiments that using SyncBN.

Table 4: Batch size experiments without using SyncBN

Batch Size	# of Hosts	# of GPUs	mAP BBox	mAP Segm	Time
64	1	8	38.37	34.71	4h 5m
128	2	16	36.83	33.67	2h 10m

Table 5: Batch size experiments using SyncBN

Batch Size	# of Hosts	# of GPUs	mAP BBox	mAP Segm	Time
64	1	8	38.68	34.96	4h 21m
128	2	16	37.02	33.98	2h 41m

We achieve remarkable results, in batch size 64 the evaluation receive higher than previous research. In compare with MegDet (Peng et al., 2018) and LargeDet (Wang et al., 2020) that have same settings, we got higher mAP (38.68 vs 37.8 vs 37.61). We also observe than when batch size grows, the network fails to maintain mAP. We argue that we do not follow LSR and still using same parameter as batch size 64. Surely, large batch size leads to faster training. We obtained almost 2 times in training time.

For experiment without SyncBN, detector achieve 38.37 while with SyncBN is 38.68. We conjecture that due to insufficient data in normalization and this may leads to higher loss, so when SyncBN applied it improve the mAP. The loss could be seen in this Table 6.

5 Conclusion

We presented a new large batch optimization framework for object detection named CoolDet, which we combined LARS, warm-up, warm restart, and SyncBN. We successfully finish training COCO 2017 dataset using Mask-RCNN and achieve remarkable mAP 38.68 which is higher than previous research. Additionally, we perform experiments with multi-GPU

Table 6: Comparisons between batch size and SyncBN. Less is better

Loss	64		128	
	w Sync-BN	wo Sync-BN	w Sync-BN	wo Sync-BN
loss_objectness	**0.01777**	0.02107	**0.02663**	0.03037
loss_classifier	**0.1258**	0.1411	**0.1953**	0.2184
loss_mask	**0.2302**	0.2393	**0.2444**	0.2547
loss_rpn_box_reg	0.0416	**0.04107**	**0.05913**	0.06054
loss_box_reg	**0.1778**	0.183	**0.252**	0.2621
training_step_loss	**0.5932**	0.6255	**0.7773**	0.826
training_loss	**0.6571**	0.6917	**0.6899**	0.7292

along with multi-host to demonstrate. Hopefully, our research could open for other future findings of large batch optimization.

Acknowledgments and Disclosure of Funding

This work was supported for infrastructure resources by University Center of Excellence on Artificial Intelligence for Vision, Natural Language Processing & Big Data Analytics (U-CoE AI-VLB), ITB and Haraj[1].

References

Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://aclanthology.org/N19-1423`.

Priya Goyal, Piotr Dollár, Ross B. Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training imagenet in 1 hour, 2017. URL `http://arxiv.org/abs/1706.02677`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017. URL `https://arxiv.org/abs/1703.06870`.

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get m for free, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks, 2014. URL `http://arxiv.org/abs/1404.5997`.

1. `https://haraj.com.sa`

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017. doi: 10.1109/CVPR.2017.106.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016. doi: 10.1007/978-3-319-46448-0_2. URL https://doi.org/10.1007%2F978-3-319-46448-0_2.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

Hiroaki Mikami, Hisahiro Suganuma, Pongsakorn U-chupala, Yoshiki Tanaka, and Yuichi Kageyama. Massively distributed sgd: Imagenet/resnet-50 training in a flash, 2018. URL https://arxiv.org/abs/1811.05233.

Ajeet Ram Pathak, Manjusha Pandey, and Siddharth Rautaray. Application of deep learning for object detection. *Procedia Computer Science*, 132:1706–1717, 2018. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2018.05.144. URL https://www.sciencedirect.com/science/article/pii/S1877050918308767. International Conference on Computational Intelligence and Data Science.

Chao Peng, Tete Xiao, Zeming Li, Yuning Jiang, Xiangyu Zhang, Kai Jia, Gang Yu, and Jian Sun. Megdet: A large mini-batch object detector. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6181–6189, 2018. doi: 10.1109/CVPR.2018.00647. URL https://openaccess.thecvf.com/content_cvpr_2018/papers/Peng_MegDet_A_Large_CVPR_2018_paper.pdf.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

Guanhua Wang. *Distributed Machine Learning with Python.* Packt, Birmingham, 2022.

Tong Wang, Yousong Zhu, Chaoyang Zhao, Wei Zeng, Yaowei Wang, Jinqiao Wang, and Ming Tang. Large batch optimization for object detection: Training coco in 12 minutes. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer*

Vision – ECCV 2020, pages 481–496, Cham, 2020. Springer International Publishing. ISBN 978-3-030-58589-1.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks, 2017a. URL https://arxiv.org/abs/1708.03888.

Yang You, Zhao Zhang, Cho-Jui Hsieh, James Demmel, and Kurt Keutzer. Imagenet training in minutes, 2017b. URL https://arxiv.org/abs/1709.05011.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes, 2020.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks, 2017.