Hindawi
Mathematical Problems in Engineering
Volume 2020, Article ID 9205373, 6 pages
https://doi.org/10.1155/2020/9205373

Research Article

Object Detection with the Addition of New Classes Based on the Method of RNOL

Haiquan Fang[1] **and Feijia Zhu**[2]

[1]*School of Public Administration, Zhejiang University of Technology, Hangzhou 310014, China*
[2]*Zhuhai Boda Creative Technology Company Limited, Zhuhai, Guangdong 519000, China*

Correspondence should be addressed to Haiquan Fang; fanghaiquan@zjut.edu.cn

Received 31 October 2019; Accepted 15 April 2020; Published 12 May 2020

Academic Editor: Thomas Hanne

Object detection plays an important role in many computer vision applications. Innovative object detection methods based on deep learning such as Faster R-CNN, YOLO, and SSD have achieved state-of the-art results in terms of detection accuracy. There have been few studies to date on object detection with the addition of new classes, however, though this problem is often encountered in the industry. Therefore, this issue has important research significance and practical value. On the premise that the old class samples are available, a method of reserving nodes in advance in the output layer (RNOL) was established in this study. Experiments show that RNOL can achieve high detection accuracy in both new and old classes over a short training time while outperforming the traditional fine-tuning method.

1. Introduction

Object detection involves the two distinct tasks of object recognition and location. It is not only necessary to identify the class of the object in the image but also able to locate the object within a rectangular area. In [1], only the object is recognized, but the object is not located in the rectangular area. Object detection is a common component of artificial intelligence and information technology systems including robot vision, unmanned aerial vehicle surveillance, automatic driving, intelligent video surveillance, and medical image analysis.

Many scholars have studied object detection. Most of the traditional methods are based on background subtraction [2–4]. Recently, many scholars have developed numerous object detection methods based on deep learning, such as Faster R-CNN [5], YOLO v3 [6], and SSD [7] and achieved state-of the-art results in regard to detection accuracy. When adding new classes, however, it is very time-consuming to train an object detection model from scratch on the premise that the old classes are available. How can the model training time be improved without sacrificing high detection accuracy in both new and old classes? This problem is often encountered in the industry; this issue has important research significance and practical value.

Fine-tuning [8] is the method most commonly used to solve the new-class addition problem at present. The fine-tuning method uses the weights of the old model except for the last output layer. Although this method can train the model in a short time, its detection accuracy is relatively low.

In this study, we developed the reserving nodes in advance in the output layer (RNOL) method to solve the object detection problem when adding new classes based on the Faster R-CNN and fine-tuning method. We conducted a series of experiments on the PASCAL VOC 2007 to validate the proposed method. The results show that, on the premise that the old classes are available, RNOL can train the model well and quickly when new classes are added. RNOL also demonstrated higher detection accuracy on both new and old classes than fine-tuning, discussed in detail as follows.

2. Related Work

Object detection is mainly based on a geometric principle first developed in the 1960s. With the emergence of neural

network and support vector machine techniques, object detection methodology has transformed from geometric to statistical. In recent years, advancements in computing and deep learning technology have brought about object detection frameworks based on deep learning such as R-CNN [8], Fast R-CNN [9], Faster R-CNN [5], YOLO [10], YOLO v2 [11], YOLO v3 [6], and SSD [7].

The new-class addition problem has a long history in the machine learning and artificial intelligence field [12–15]. The problem may be approached when old classes are not available [16, 17] or when old classes are available; there has been considerably less research centered on the latter scenario. Rebuffi et al. [18] researched the problem using a small number of old classes. Extant methods are not ideal as far as the detection accuracy of new or old classes, so it is difficult to meet industrial needs at present. In this paper, we discuss only scenarios wherein old classes are available.

3. Reserving Nodes in Advance in the Output Layer

For object detection problems considering the addition of new classes, the RNOL method primarily works by reserving the number of nodes in the output layer appropriately; the number of nodes in the output layer in this case exceeds the number of old classes. To operate RNOL, we first use a Faster R-CNN to build a model, then reserve the correct number of nodes in the output layer, and train the model on the old classes before saving the model and weight. Next, when new classes are added, the saved models and weights are loaded and the models are trained on both the new and old classes. Finally, we use the fully trained model to detect the test samples.

A diagram of the RNOL method is shown in Figure 1. Hollow dots in the output layer represent reserved nodes. The number of nodes in the output layer is larger than the number of the old classes, as mentioned above. The number of reserved nodes can be set artificially. This method is effective as long as the number of new classes is not greater than the number of reserved nodes. The *person* in Figure 1 belongs to the old class and the *horse* belongs to the new class. The proposed method resolves the problem of the coexistence of new and old classes. Compared to fine-tuning, the advantage of RNOL is that it can effectively utilize more weight information of the old class model, including the weight of the output layer.

The architectures of RNOL are consistent with those of Faster R-CNN, except the number of nodes in the output layer. The activation function of neurons in the output layer is softmax.

The old classes are marked as C_A. The model trained in the old classes is marked as $A(C_A)$, and the new classes are marked as C_B. The model trained in new and old classes is marked as $B(C_A \cup C_B)$.

4. Experiments

4.1. Datasets and Evaluation. We evaluated our method on the PASCAL VOC 2007 dataset, as mentioned above. VOC 2007 consists of $5K$ images in the trainval split and $5K$ images in the test split for 20 object classes. We used the standard mean average precision (mAP) at 0.5 IoU threshold as the evaluation metric; evaluation of the VOC 2007 experiments was conducted on the test split.

4.2. Implementation Details. We randomly initialized all new layers by drawing weights from a zero-mean Gaussian distribution with a standard deviation of 0.01. We used the stochastic gradient descent (SGD) with Nesterov momentum [19] to train the network in all experiments. We set the learning rate to 0.001, decay to 0.0001 after $50K$ iterations, and momentum to 0.9. In the second stage of training, i.e., learning the extended network with new classes, we used a learning rate of 0.001 and decay to 0.0001 after $10K$ iterations. The $A(C_A)$ network was trained for $70K$ iterations on PASCAL VOC 2007. The $B(C_B)$ network was trained for $20K$ iterations when only one class was added and 30K iterations when 10 classes were added simultaneously. For the Faster R-CNN, we took batches of two images each. All other layers (i.e., the shared convolutional layers) of the $A(C_A)$ network were initialized by pretraining a model for ImageNet classification [20]. We implemented this in Tensorflow [21].

4.3. Effects of RNOL. We sought to determine whether reserving nodes in advance in the output layer increases the computing time or affects the object detection accuracy compared to the traditional method.

We took 10 classes in alphabetical order from the VOC2007 dataset and ran two respective types of experiment. Experiment 1 involved reserving 10 nodes in the output layer (that is, the number of neurons in the output layer was 20). There was no reserved position in the output layer, in Experiment 2; that is, the number of neurons in the output layer was 10.

The detection accuracy of the two experiments is shown in Table 1, and the training times are shown in Table 2. We observed no significant difference in test results and training time between the two experiments, which suggests that RNOL does not increase the training time nor affect the detection accuracy.

4.4. Addition of One Class. In this experiment, we took 19 classes in alphabetical order from the VOC 2007 dataset as C_A and the remaining one as the only new class C_B. We then trained the $A(1-19)$ network on C_A and the $B(1-20)$ network on the VOC trainval containing the 20 classes. A summary of the evaluation of these networks on the VOC test set is shown in Table 3, and the full results are listed in Table 4.

As shown in Table 3, when applying the RNOL method on the basis of the old network $A(1-19)$, the new network $B(1-20)$ has 69.0% mAP after $20K$ iterations. When using the fine-tuning method on the basis of the old network $A(1-19)$ without RNOL, the new network $B(1-20)$ only has 68.1% mAP after $20K$ iterations. When applying the

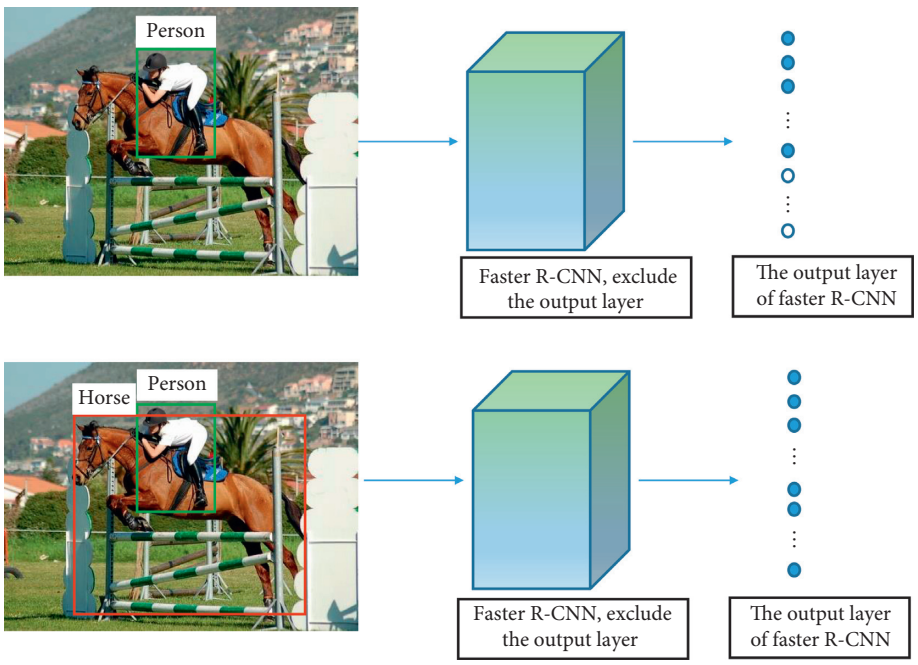

Figure 1: Diagram of RNOL.

Table 1: Detection results (mAP (%)).

Number of iterations (K)	Experiment 1	Experiment 2
10	59.1	56.6
20	65.1	65.3
30	67.5	67.6
40	68.5	68.3
50	68.0	68.9
60	70.2	70.1
70	70.3	69.9

Table 2: Training time (minutes).

Number of iterations (K)	Experiment 1	Experiment 2
10	93	94
20	181	179
30	265	265
40	350	350
50	436	438
60	522	521
70	608	607

Table 3: Test results when adding one class (mAP (%)).

Method	The number of iterations (K)	Old	New	All
A (1–19) RNOL	70	69.5	—	—
B (1–20) RNOL	20	69.1	68.6	69.0
A (1–19) without RNOL	70	69.2	—	—
B (1–20) fine-tuning	20	68.0	68.6	68.1
A (1–20) TFS	20	59.1	62.3	59.2
A (1–20) TFS	70	69.5	71.9	69.6

training from scratch (TFS) method, the network $A(1-20)$ only has 59.2% mAP after 20K iterations. The TFS method needs 70K training iterations to achieve the ideal accuracy. These results suggest that the RNOL method outperforms both the fine-tuning method and TFS method. When adding one class, the RNOL method yields higher accuracy in a shorter training time than fine-tuning or TFS.

We next compared the RNOL and fine-tuning methods in the new network $B(1-20)$ when adding one class. Each was trained 30K times, and the weights were saved every 5K iteration; each saved weight was loaded on the detection set. The test results are shown in Figure 2, where the RNOL method achieves its highest detection accuracy when training 20K iterations and then begins to decline. The fine-tuning method accuracy increases slowly over the experiment but does not readily exceed that of the RNOL method.

4.5. Addition of Multiple Classes. In this experiment, we took 10 classes in alphabetical order from the VOC 2007 dataset as C_A and the remaining 10 classes as C_B. We then trained the $A(1-10)$ network on C_A and the $B(1-20)$ network on the VOC trainval containing the 20 classes. A summary of the evaluation of these networks on the VOC test set is shown in Table 5, and the full results are listed in Table 6.

As shown in Table 5, on the basis of the old network $A(1-10)$ with RNOL, the new network $B(1-20)$ has 68.2% mAP after 30K iterations. On the basis of the old network $A(1-10)$ with fine-tuning alone and no RNOL, the new network $B(1-20)$ only has 67.3% mAP after 30K iterations. When using the TFS method, the network $A(1-20)$ only has 62.5% mAP after 30K iterations. The TFS method needs training 70K iterations to achieve the ideal accuracy. Once again, the RNOL method outperforms both fine-tuning and TFS methods. When adding 10 classes, the RNOL method

Table 4: Test results when adding one class (mAP (%)).

	A (1–19) RNOL	B (1–20) RNOL	A (1–19) without RNOL	B (1–20) fine-tuning	A (1–20) TFS	A (1–20) TFS
The number of iterations	70K	20K	70K	20K	20K	70K
Aero	69.1	68.8	70.6	69.2	61.6	75.2
Bike	77.5	78.1	78.3	77.8	67.9	78.9
Bird	67.5	66.2	68.3	65.9	55.6	67.7
Boat	58.5	56.2	55.8	52.2	38.4	55.8
Bottle	52.4	51.5	53.4	52.4	43.1	53.6
Bus	76.1	75.3	81.5	77.3	67.6	74.6
Car	79.7	80.2	80.1	80.0	74.1	79.8
Cat	84.3	83.9	78.4	78.7	73.9	79.2
Chair	50.3	48.3	49.5	49.4	40.1	52.9
Cow	74.4	73.3	71.2	71.0	57.8	73.6
Table	64.7	65.9	66.4	64.9	60.2	68.7
Dog	81.7	77.4	77.7	75.9	62.5	76.3
Horse	80.5	80.6	79.7	79.8	73.3	80.6
Mbike	75.9	76.1	73.4	74.2	65.1	76.6
Person	77.2	77.1	77.1	77.2	69.3	77.4
Plant	43.5	42.7	41.9	39.1	31.6	40.1
Sheep	66.2	68.3	67.1	66.1	52.3	67.2
Sofa	66.7	66.6	68.5	66.9	53.9	66.3
Train	74.1	74.7	75.8	74.9	73.8	75.6
Tv	—	68.6	—	68.6	62.3	71.9
mAP	69.5	69.0	69.2	68.1	59.2	69.6

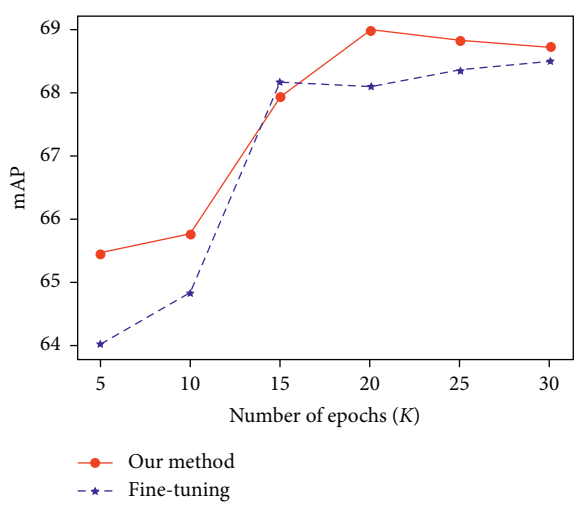

Figure 2: Comparison of two methods when adding one class.

Table 5: Test results when adding 10 classes (mAP (%)).

Method	The number of iterations (K)	Old	New	All
A (1–10) RNOL	70	70.3	—	—
B (1–20) RNOL	30	68.3	68.1	68.2
A (1–10) without RNOL	70	69.9	—	—
B (1–20) fine-tuning	30	67.5	67.1	67.3
A (1–20) TFS	30	61.9	63.2	62.5
A (1–20) TFS	70	69.1	70.1	69.6

achieves higher accuracy in a shorter training time than fine-tuning or TFS.

For adding 10 classes, we have compared RNOL and fine-tuning methods in the new network $B(1-20)$. Each was trained 30K times, the weights were saved every 5K iteration, and each saved weight was loaded on the detection set. The results are shown in Figure 3. The detection accuracy of RNOL is higher than that of the fine-tuning method when training 30K

Table 6: Test results when adding 10 classes (mAP (%)).

	A (1–10) RNOL	B (1–20) RNOL	A (1–10) without RNOL	B (1–20) fine-tuning	A (1–20) TFS	A (1–20) TFS
The number of iterations	70K	30K	70K	30K	30K	70K
Aero	70.1	69.3	69.4	68.7	67.9	75.2
Bike	78.9	78.4	78.7	77.2	73.7	78.9
Bird	67.6	66.5	68.9	66.6	59.6	67.7
Boat	62.4	57.3	58.8	56.7	48.4	55.8
Bottle	56.7	55.2	57.6	53.9	40.9	53.6
Bus	76.9	76.5	76.7	74.8	72.5	74.6
Car	80.2	80.1	80.1	79.7	76.9	79.8
Cat	79.1	79.6	79.6	78.9	74.6	79.2
Chair	53.8	48.7	53.1	48.6	40.2	52.9
Cow	76.8	71.6	75.9	69.7	64.2	73.6
Table	—	62.4	—	60.9	59.4	68.7
Dog	—	76.9	—	77.2	71.4	76.3
Horse	—	81.2	—	80.3	74.4	80.6
Mbike	—	71.6	—	71.9	70.1	76.6
Persn	—	74.9	—	75.6	66.2	77.4
Plant	—	40.1	—	37.8	34.1	40.1
Sheep	—	66.1	—	65.2	61.2	67.2
Sofa	—	63.5	—	59.7	56.7	66.3
Train	—	75.3	—	71.9	72.1	75.6
Tv	—	69.1	—	70.8	66.6	71.9
mAP	70.3	68.2	69.9	67.3	62.5	69.6

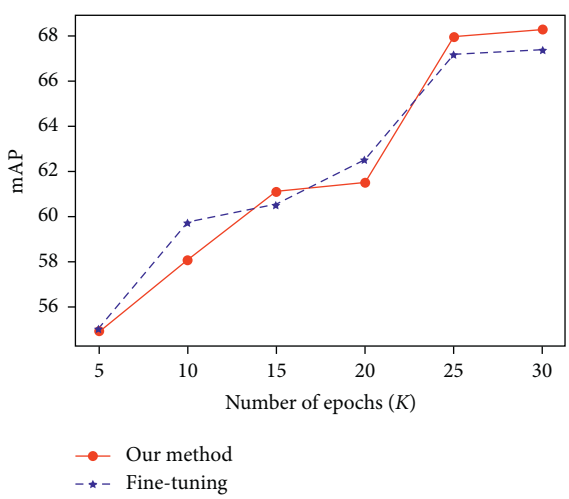

Figure 3: Comparison of two methods when adding 10 classes.

times. More iterations are needed to achieve higher detection accuracy as the number of added classes increases beyond one.

5. Conclusion

For object detection considering the addition of new classes when the old classes are available, we improved the Faster R-CNN model in this study by reserving nodes in advance in the output layer. Our experimental results show that RNOL can achieve high detection accuracy in both new and old classes in a short training time. Although the proposed method outperforms the fine-tuning method, its detection accuracy still has room for further improvement. One possible way to do this is to increase the number of training iterations, but it will increase the cost of training time.

Data Availability

We evaluated our method on the PASCAL VOC 2007 dataset.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] I. Lorencin, N. Lingua, N. Anđelić, V. Mrzljak, and Z. Car, "Marine objects recognition using convolutional neural networks," *Naše More*, vol. 66, no. 3, pp. 112–120, 2019.

[2] M. S. Kemouche, N. Aouf, and M. Richardson, "Object detection using Gaussian mixture-based optical flow modelling," *The Imaging Science Journal*, vol. 61, no. 1, pp. 22–34, 2013.

[3] W.-Y. Chiu and D.-M. Tsai, "Moving/motionless foreground object detection using fast statistical background updating," *The Imaging Science Journal*, vol. 61, no. 2, pp. 252–267, 2013.

[4] E. H. Zhang, Y. C. Li, and J. H. Duan, "Moving object detection based on confidence factor and CSLBP features," *The Imaging Science Journal*, vol. 64, no. 5, pp. 253–261, 2016.

[5] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[6] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, https://arxiv.org/abs/1804.02767.

[7] W. Liu, D. Anguelov, D. Erhan et al., "SSD: single shot multibox detector," https://arxiv.org/abs/1512.02325.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," 2014, https://arxiv.org/abs/1311.2524.

[9] R. Girshick, "Fast R-CNN," 2015.

[10] J. Redmon, S. Divvala, R. Girshick et al., "You only look once: unified, real-time object detection," 2015, https://arxiv.org/abs/1506.02640.

[11] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," 2016, https://arxiv.org/abs/1612.08242.

[12] J. C. Schlimmer and D. H. Fisher, "A case study of incremental concept induction," in *Proceedings of the 5th National Conference on Artificial Intelligence*, pp. 495–501, Philadelphia, PA, USA, August 1986.

[13] S. Thrun, "Is learning the n-th thing any easier than learning the first?" in *Proceedings of the NIPS'95 Proceedings of the 8th International Conference on Neural Information Processing Systems*, pp. 640–646, Amherst, MA, USA, November 1996.

[14] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Proceedings of the NIPS'00 Proceedings of the 13th International Conference on Neural Information Processing Systems*, pp. 388–394, Cambridge. MA, USA, January 2000.

[15] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: an incremental learning algorithm for supervised neural networks," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 31, no. 4, pp. 497–508, 2001.

[16] Z. Li and D. Hoiem, "Learning without forgetting," *Computer Vision—ECCV 2016*, vol. 9908, pp. 614–629, 2016.

[17] K. Shmelkov, C. Schmid, and K. Alahari, "Incremental learning of object detectors without catastrophic forgetting," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 3420–3429, Venice, Italy, October 2017.

[18] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "iCaRL: Incremental classifier and representation learning," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5533–5542, Honolulu, HI, USA, July 2017.

[19] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.

[20] O. Russakovsky, J. Deng, H. Su et al., "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[21] M. Abadi, P. Barham, and J. Chen, "Tensorflow: a system for large-scale machine learning," in *Proceeding of the OSDI'16 12th USENIX Conference on Operating Systems Design and Implementation (OSDI)*, pp. 265–283, Savannah, GA, USA, November 2016.