LIGHTDET: A LIGHTWEIGHT AND ACCURATE OBJECT DETECTION NETWORK

Qiankun Tang[1, 2] *Jie Li*[1, 2] *Zhiping Shi* [3] *Yu Hu*[1, 2*]

[1]Research Center for Intelligent Computing Systems, State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
[2] University of Chinese Academy of Sciences
[3] Beijing Advanced Innovation Center for Imaging Theory and Technology, Capital Normal University

ABSTRACT

The extensive computational burden limits the usage of accurate but complex object detectors in resource-bounded scenarios. In this paper, we present a lightweight object detector, named LightDet, to address this dilemma. We design a lightweight backbone that is able to capture rich low-level features by the proposed Detail-Preserving Module. To effectively aggregate bottom and top-down features, we introduce an efficient Feature-Preserving and Refinement Module. A lightweight prediction head is employed to further reduce the entire network complexity. Experimental results show that our LightDet achieves 75.5% mAP on PASCAL VOC 2007 at the speed of 250 FPS and 24.0% mAP on MS COCO dataset.

***Index Terms*—** Lightweight, object detection, real-time

1. INTRODUCTION

Object detection has been improved largely thanks to the development of convolutional neural network (CNN). Modern CNN-based object detection frameworks [1, 2] usually adopt complex architectures [3, 4] and large images to achieve top performance. However, these accurate networks having enormous computation and parameter amount are unsuitable for resource-bounded mobile platforms (*e.g.*, robot, smartphone), which contain restrictive memory access and computation.

To meet the need of mobile scenarios, several researches have been conducted to design lightweight object detectors. Some works prefer to apply small backbone networks [5, 6, 7] to accurate detection frameworks. For instance, MobileNet-SSD [5] changed the backbone of SSD [8] to MobileNet [5], MobileNetV2-SSDLite [6] combined the MobileNetV2 [6] with a lite version of SSD detection head. However, the designed backbones for classification are not fully suitable for object detection since detection needs rich low-level information which is less crucial for classification. Thus, PeleeNet [9] proposed a variant of DenseNet [10] with some efficient modules. ThunderNet [11] replaced all 3×3 depth-wise convolutions in ShuffleNetV2 [7] with 5×5 depth-wise convolutions to enlarge the receptive field, and involved more low-level information by increasing the channels of first few layers. There are some works focus on decomposing the standard convolution of accurate detection networks into group convolution or depth-wise separable convolution. For example, Tiny-DSOD [12] introduced depth-wise feature pyramid network and depth-wise dense block into DSOD [13]. We find that directly applying the effective designs of full-size detection framework to lightweight detectors brings high computational cost with limited accuracy improvement.

In this paper, we introduce the LightDet, a lightweight one-stage object detector. We propose a Detail-Preserving Module (DPM) to extract and preserve more low-level features. LightDet takes DPM as a stem block and further stacks several ShuffleV2 blocks [7] to obtain semantic features. To better combine the bottom features with the top-down semantic features for detection, we investigate the drawbacks in feature pyramid network and design a Feature-Preserving and Refinement Module (FPRM). FPRM involves more low-level information and refines the semantic features before fusion. At last, we apply a lightweight prediction head with small size convolutional kernel to replace the heavy detection head widely adopted in current state-of-the-art one-stage detectors.

Extensive experiments on the public datasets (PASCAL VOC and MS COCO) indicate that LightDet achieves much better accuracy with less computation than the state-of-the-art lightweight object detectors. Meanwhile, LightDet is able to run at more than 200 FPS on a single NVIDIA GTX 1080 Ti.

2. RELATED WORKS

Object Detection CNN-based object detection frameworks can be roughly divided into two groups: two-stage methods and one-stage methods. Two-stage methods [1, 2] first output a set of object candidates by the region proposal methods [2], then these candidates are fed into the detection network to predict the specific categories and locations. These

*Corresponding author: Yu Hu, huyu@ict.ac.cn. This work is supported in part by the National Key R&D Program of China under grant No. 2018AAA0102701, in part by the Science and Technology on Space Intelligent Control Laboratory under grant No. HTKJ2019KL502003, and in part by the Innovation Project of Institute of Computing Technology, Chinese Academy of Sciences under grant No. 20186090.

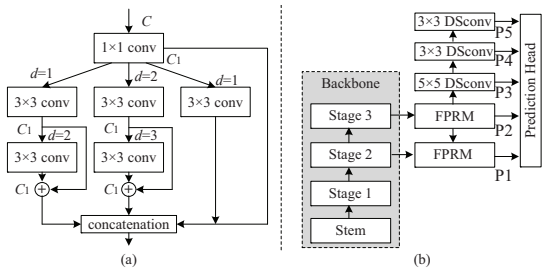

Fig. 1. (a) Framework of the proposed DPM. d represents the dilation rate. (b) Architecture of the proposed LightDet. DSconv means the depth-wise separable convolution.

methods can produce accurate results but are computationally expensive with low inference speed. While one-stage methods [8, 14] directly evolve predefined anchor boxes (anchor-based) or pixels in feature map (anchor-free) [15] into final bounding boxes with specific categories. These methods have compact pipeline and relatively high speed. However, the above-mentioned methods usually have hundreds of convolutional layers and extensive channels, which require massive resources and computation. In this paper, we follow the anchor-based one-stage pipeline and propose a lightweight object detector dedicating for the resource-bounded platform.

Lightweight Object Detectors The one-stage pipeline is the mainstream to design lightweight object detectors due to its compactness and high efficiency. Therefore, some works explored to combine the lightweight classification backbone [5, 6, 7] with the detection head of one-stage methods or applied the convolution factorization principle into current full-size one-stage methods [13]. Even so, we find they still involve redundant computation. On the other side, ThunderNet [11] adopted a two-stage pipeline by introducing a modified ShuffleNetV2 into the Light-Head R-CNN [16] with a compressed RPN and two feature enhancement modules.

3. METHOD

3.1. Detail-Preserving Module

Current state-of-the-art object detectors usually take the network designed for classification as backbone. However, the classification networks [3, 4, 5, 6, 7] rapidly down-sample the input to 4-8 times by a few first layers. This operation loses much low-level features which are beneficial for localization in object detection. A naive way to improve this is stacking more convolutional layers in the beginning [13]. However, this improvement requires additional computational overhead and is unsuitable for lightweight networks.

We propose an efficient Detail-Preserving Module (DPM) to extract and preserve more low-level features, as depicted in Fig. 1(a). We first utilize a 3×3 convolutional layer and output channel of C=16. To reduce the computational complexity of DPM, a point-wise convolutional layer is applied to lower the dimension of extracted features to C_1 channels. The reduced features are re-sampled by K=C/C_1 branches including an identity mapping. One branch adopts a 3×3 convolutional layer and the rest branches consist of two 3×3 convolutional layers with dilation of d=k and d=k+1, $k\in[1, K\text{-}2]$. Because large dilation has a large receptive field, we take the second convolutional layer with larger dilation as the residual block in each branch to avoid involving more background information. The outputs from K branches are merged by concatenation to preserve more low-level features.

Stage	Component	Output	Stride
Stem	Convolution (3×3)	$16 \times 320 \times 320$	1
	DPM	$16 \times 320 \times 320$	1
	MaxPooling (3×3)	$16 \times 160 \times 160$	2
	DPM	$32 \times 80 \times 80$	2
Stage 1	ShuffleV2 block	$116 \times 40 \times 40$	2
	3×ShuffleV2 block	$116 \times 40 \times 40$	1
Stage 2	ShuffleV2 block	$232 \times 20 \times 20$	2
	7×ShuffleV2 block	$232 \times 20 \times 20$	1
Stage 3	ShuffleV2 block	$464 \times 10 \times 10$	2
	3×ShuffleV2 block	$464 \times 10 \times 10$	1
	ShuffleV2 block	$512 \times 10 \times 10$	1

Table 1. The backbone architecture of LightDet.

We take DPM as a stem block and further stack several ShuffleV2 blocks to abstract more semantic features, therefore, the designed backbone network has the capability of capturing rich low-level and semantic features. **Table 1** gives the detailed architecture of our backbone, named LightNet.

3.2. Feature-Preserving and Refinement Module

FPN [17] is widely applied in object detection framework to combine low-resolution information with high-resolution ones, as shown in Fig. 2(a). However, the FPN structure has some drawbacks when being utilized in a lightweight network. First, FPN uses a 1×1 convolutional layer as lateral connection to reduce channel numbers and transfer bottom features. Deeper and larger networks have strong capability to extract and save features with enormous channels, so a 1×1 convolutional layer is enough. While lightweight networks have relatively weak modeling capability and less channels, this operation will lose much information. Second, FPN directly up-samples lower-resolution features and uses element-wise addition to aggregate features. However, addition only mixes the features of different abstraction levels, but cannot preserve both of them and will weaken the representative of lightweight network. Third, we find that combining the FPN with lightweight backbone is computationally inefficient.

The above observations motivate us to propose a Feature-Preserving and Refinement Module (FPRM), which consists of a Feature-Preserving Module (FPM) and a Feature Refinement Module (FRM), as shown in Fig. 2(b). FPRM tackles bottom features and top-down features in different strategies. For FPM, we set the output dimension of lateral features to C_2 channels. We split the lateral input with C_l channels into

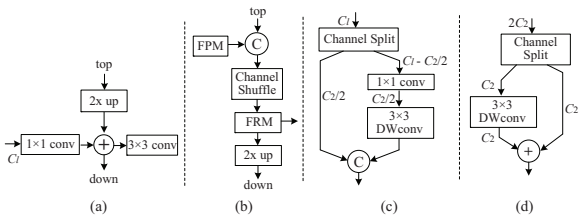

Fig. 2. (a) FPN. (b) Feature-Preserving and Refinement Module. (c) Feature-Preserving Module. (d) Feature Refinement Module. DWconv means the depth-wise convolution.

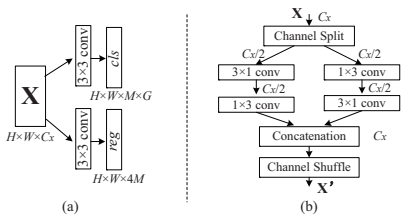

Fig. 3. (a) Original prediction head. (b) Proposed lightweight prediction head. It is followed by two 1×1 convolutional layers for classification (*cls*) and regression (*reg*).

two parts with C_2/2 and C_l-C_2/2 channels. For the part with C_l-C_2/2 channels, we use a point-wise convolutional layer to reduce its channel to C_2/2, then a 3×3 depth-wise convolutional layer to incorporate more context features. These two parts are merged by concatenation. Therefore, FPM is able to preserve more bottom features and context information, as shown in Fig. 2(c). We up-sample the top-down features by a bilinear interpolation layer, then aggregate the up-sampled features with the lateral features by concatenation instead of addition to avoid information loss. However, the concatenated features have higher dimensions. We present the FRM to reduce their dimensions instead of the commonly used 1×1 convolutional layer, which imposes more computational cost and parameters. The aggregated features are averagely divided into two parts. One part is processed by a 3×3 depth-wise convolutional layer. Then these two parts are merged by addition. Therefore, we refine the preserved part by transferring the information from other parts in a residual way and obtain the lower-dimensional features, as shown in Fig. 2(d).

3.3. Lightweight Prediction Head

One-stage methods use 3×3 convolutional layer as prediction head for classification and localization. As shown in Fig. 3(a), given a input feature $\mathbf{X} \in \mathbb{R}^{H \times W \times C_x}$ with height H, width W and channel C_x, we need predict G categories for M anchor boxes at a location. The computation of this prediction head is $C_x \times M \times (G+4) \times 3 \times 3 \times H \times W$. This prediction head becomes a computational bottleneck.

PeleeNet solved this issue by using a residual prediction block followed by small kernel size prediction layers. However, its solution is computationally expensive. We propose a more lightweight one. The input $\mathbf{X}$ is split into two lower-dimensional branches, where each one has half channels of $\mathbf{X}$. Each branch is transformed by a set of 1×3, 3×1 filters. The outputs of two branches are merged by concatenation so that the number of channels keeps the same as input. We further apply channel shuffle [7] to enable information communication between two split branches, as shown in Fig. 3(b). This head allows us to use 1×1 convolutional layer for classification and localization. Its computation is $[(C_x/2 \times C_x/2 \times 3 \times 1) \times 4 + C_x \times M \times G + C_x \times M \times 4] \times H \times W$, which is only 40.7% of original prediction head in our settings.

3.4. LightDet Architecture

We take the image with size of 320×320 as input and use 5 scales of feature maps: 20×20, 10×10, 5×5, 3×3, 1×1 (called $P_1 \sim P_5$) for prediction. As shown in Fig. 1(b), $P_1 \sim P_2$ are the outputs of FPRM and $P_3 \sim P_5$ are generated from P_2 by 3 cascaded 3×3 depth-wise separable convolutional layers with stride 2. In our preliminary experiments, we also use the feature map of 40×40 for prediction. However, we find that including this scale of feature map only improves 0.4% mAP but brings 90 MFLOPs. We also observe that $P_3 \sim P_5$ generate many false positives. One hypothesis is that the 3×3 depth-wise separable convolutional layer cannot extract enough discriminative information, so we change the kernel of P_3 to 5×5 and the stride of P_5 to 1 (P_5 scale becomes 3×3).

4. EXPERIMENTS

We adopt the same anchor boxes setting strategy as SSD, *i.e.*, two scales of anchor boxes for P_1 and one scale for the others and each scale has 6 aspect ratios (*i.e.,* 1, 1, $1/2$, $1/3$, 2, 3), thus M=6. We set K=4, C_2=C_x=128. Moreover, we adopt the same data augmentation and loss functions as SSD. The convolutional layers are initialized by "MSRA" [18] method.
Implementation Details. We train LightDet on a single GPU with batch size of 64. The network is optimized by SGD with a weight decay of 0.0005 and a momentum of 0.9 for 64.5k iterations. The initial learning rate is set to 0.008 and decays by a factor of 0.1 at 38.7k and 51.6k iterations. The training data is the union set of PASCAL VOC 2007 *trainval* and VOC 2012 *trainval*, then we report results on VOC 2007 *test* set.
Baseline. We replace the stem block in Table 1 with a 3×3 convolutional layer with stride 2 and 32 channels followed by a MaxPooling with stride 2, which is a common practice in lightweight classification network. We append the FPN with original heavy prediction head to this network and 6 scales of feature maps with an additional 40×40 for prediction. It gets 66.8% mAP with 719 MFLOPs.

4.1. Ablation Study

We gradually insert our proposed modules and other improvements into the baseline to evaluate their effect on computation and performance, as shown in **Table 2**.

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
Baseline	✓							
DPM		✓	✓	✓	✓	✓	✓	✓
FPM			✓	✓	✓	✓	✓	✓
Concat. in FPN				✓				
FRM					✓	✓	✓	✓
Prediction head						✓	✓	✓
Modify $P_3 \sim P_5$							✓	✓
Drop 40×40 feature								✓
MFLOPs	719	842	814	879	779	555	555.3	465
VOC 2007 mAP	66.8	71.0	71.9	72.6	73.1	73.5	74.4	74.0

Table 2. The effectiveness of our proposed modules.

Method	Input	Backbone	FPS	MFLOPs	mAP(%)
YOLO v2 [19]	416×416	Darknet-19	67	17500	76.8
SSD [8]	300×300	VGG-16	46	31750	77.2
Tiny-YOLO [14]	416×416	Tiny-Darknet	207	6970	57.1
MobileNet-SSD [5]	300×300	MobileNet	59.3	1150	68.0
Pelee [9]	304×304	PeleeNet	-	1210	70.9
Tiny-DSOD [13]	300×300	DDB-Net	105	1060	72.1
ThunderNet (*ms-train*) [11]	320×320	SNet146	248	**461**	75.1
LightDet	320×320	LightNet	**250**	**465**	74.0
LightDet (*ms-train*)	320×320	LightNet	**250**	**465**	**75.5**

Table 3. Comparison with other lightweight object detectors. *ms-train* means multi-scale training.

Effectiveness of DPM. Results in column *b* of **Table 2** show that though DPM involves high computational cost, it leads to a gain of 4.2% mAP to 71.0%. This indicates that DPM is able to capture more informative features from low-levels.

Effectiveness of FPM. We replace the lateral connection of 1×1 convolutional layer in FPN with our FPM. FPM brings 0.9% mAP improvement while lowering 28 MFLOPs.

Concatenation in FPN. If we replace the element-wise addition in FPN with concatenation and adopt a 1×1 convolutional layer for dimensionality-reduction, we can get additional 0.7% mAP which reveals that preserving bottom and top-down features is beneficial for lightweight object detection. However, this operation incurs extra computational cost.

Effectiveness of FRM. Results in column *d* and *e* of **Table 2** show that adopting the proposed FRM to lower dimension while refining top-down features is more effective than a 1×1 convolutional layer. Comparing *b* and *e* in **Table 2**, our FPRM is not only more effective but also more efficient than FPN.

Lightweight Prediction Head. Our proposed lightweight prediction head not only greatly reduces the computational cost but also slightly promotes accuracy (column *f*).

Modification of $P_3 \sim P_5$. A slight modification of $P_3 \sim P_5$ brings a noticeable accuracy gain (column *g* in **Table 2**) with negligible computation.

Drop 40×40 Feature Map. Comparing the results of *h* and *g* in **Table 2**, we find that dropping the 40×40 feature map only sacrifices 0.4% mAP but enjoys significant computational savings.

4.2. Results on PASCAL VOC 2007

We also compare our LightDet with state-of-the-art lightweight object detectors, as listed in **Table 3**. Our LightDet has only 465 MFLOPs, which is less computational cost than state-of-the-art lightweight one-stage methods. Our thoughtful design allows LightDet to achieve much better accuracy than other methods. We note that ThunderNet achieves its final result by adopting multi-scale training and Soft-NMS [20]. Therefore, for fairness, we adopt the same settings and achieve 75.5% mAP. Our LightDet has slightly more FLOPs than ThunderNet in that LightDet needs to predict the category for each anchor box of each level, and the number of categories and anchor boxes for prediction affect a lot on the FLOPs. While ThunderNet is a two-stage method, it adopts a very light detection head and sparse proposals (*e.g.*, 200) for prediction.

Runtime Analysis. In the *4th* column of **Table 3**, we compare the speed of LightDet with other methods. Our speed is measured on a single NVIDIA GTX 1080 Ti. We merge the parameters of Batch Normalization into its preceding convolutional layer, as is a common practice. LightDet runs much faster than other lightweight detectors.

Method	Input	Backbone	MFLOPs	AP	AP_{50}	AP_{75}
YOLO v2 [19]	416×416	Darknet-19	17500	21.6	44.0	19.2
SSD [8]	300×300	VGG-16	31750	25.1	43.1	25.8
Light-Head R-CNN [16]	800×1200	ShuffleNetV2 [7]	5650	23.7	-	-
MobileNet-SSD [5]	300×300	MobileNet	1200	19.3	-	-
MobileNet-SSDLite [6]	320×320	MobileNet	1300	22.2	-	-
MobileNetV2-SSDLite [6]	320×320	MobileNetV2	800	22.1	-	-
Pelee [9]	304×304	PeleeNet	1290	22.4	38.3	22.9
Tiny-DSOD [13]	300×300	DDB-Net	1120	23.2	40.4	22.8
ThunderNet (*ms-train*) [11]	320×320	SNet146	473	23.6	40.2	24.5
LightDet	320×320	LightNet	503	**24.0**	**42.7**	**24.5**

Table 4. Comparison with other lightweight object detectors on MS COCO *test-dev*.

4.3. Results on MS COCO

We further conduct experiments on the MS COCO dataset. We train LightDet on *trainval 35k* set and report final accuracy on *test-dev* set. The initial learning rate is 0.004 and divided by factor of 10 after 210k and 280k iterations. The total number of iteration is 320k with a batch size of 64.

The results are listed in **Table 4**. The increased categories bring intractable computation, but LightDet still achieves a higher performance of 24.0% mAP than other lightweight object detectors. We think the narrowed ROI features in ThunderNet for prediction are inferior for challenging dataset, while the preserved features of LightDet contain more beneficial information for prediction and thus higher accuracy.

5. CONCLUSION

We present a lightweight object detector LightDet designed for running in resource-bounded environment. We introduce a Detail-Preserving Module as stem block to capture and preserve more low-level information for detection. The Feature-Preserving and Refinement Module is proposed to aggregate bottom and top-down features more effective than FPN. We further investigate the drawbacks of prediction head in prior one-stage methods and present a lightweight one. The experimental results on PASCAL VOC 2007 verify the effectiveness of our proposed modules in both computation and accuracy. Our LightDet achieves better performance than other lightweight object detectors.

6. REFERENCES

[1] R. Girshick, "Fast R-CNN," in *ICCV*, 2015, pp. 1440–1448.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *TPAMI*, pp. 1137–1149, 2017.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[4] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He, "Aggregated residual transformations for deep neural networks," *arXiv preprint arXiv:1611.05431*, 2016.

[5] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[6] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.

[7] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun, "Shufflenet v2: Practical guidelines for efficient cnn architecture design," in *ECCV*, 2018, pp. 116–131.

[8] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg, "SSD: Single shot multibox detector," in *ECCV*, 2016, pp. 21–37.

[9] Robert J Wang, Xiang Li, and Charles X Ling, "Pelee: A real-time object detection system on mobile devices," in *NIPS*, 2018, pp. 1967–1976.

[10] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017, pp. 4700–4708.

[11] Zheng Qin, Zeming Li, Zhaoning Zhang, Yiping Bao, Gang Yu, Yuxing Peng, and Jian Sun, "Thundernet: Towards real-time generic object detection," in *ICCV*, 2019, pp. 6718–6727.

[12] Yuxi Li, Jianguo Li, Jiuwei Li, and Weiyao Lin, "Tiny-DSOD: Lightweight object detection for resource-restricted usage," in *BMVC*, 2018.

[13] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue, "DSOD: Learning deeply supervised object detectors from scratch," in *CVPR*, 2017, pp. 1919–1927.

[14] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," in *CVPR*, 2016, pp. 779–788.

[15] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He, "FCOS: Fully convolutional one-stage object detection," in *ICCV*, 2019, pp. 9627–9636.

[16] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun, "Light-head R-CNN: In defense of two-stage object detector," *arXiv preprint arXiv:1711.07264*, 2017.

[17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature pyramid networks for object detection," in *CVPR*, 2017, pp. 2117–2125.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, 2015, pp. 1026–1034.

[19] Joseph Redmon and Ali Farhadi, "YOLO9000: better, faster, stronger," in *CVPR*, 2017, pp. 7263–7271.

[20] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis, "Soft-nms–improving object detection with one line of code," in *ICCV*, 2017, pp. 5561–5569.