electronics

Article

A Light-Weight CNN for Object Detection with Sparse Model and Knowledge Distillation

Jing-Ming Guo *, Jr-Sheng Yang, Sankarasrinivasan Seshathiri and Hung-Wei Wu

Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei 106335, Taiwan; colayangtw@gmail.com (J.-S.Y.); D10507805@mail.ntust.edu.tw (S.S.); hungwei0901@gmail.com (H.-W.W.)
* Correspondence: jmguo@mail.ntust.edu.tw; Tel.: +886-2-29558168

Abstract: This study details the development of a lightweight and high performance model, targeting real-time object detection. Several designed features were integrated into the proposed framework to accomplish a light weight, rapid execution, and optimal performance in object detection. Foremost, a sparse and lightweight structure was chosen as the network's backbone, and feature fusion was performed using modified feature pyramid networks. Recent learning strategies in data augmentation, mixed precision training, and network sparsity were incorporated to substantially enhance the generalization for the lightweight model and boost the detection accuracy. Moreover, knowledge distillation was applied to tackle dropping issues, and a student–teacher learning mechanism was also integrated to ensure the best performance. The model was comprehensively tested using the MS-COCO 2017 dataset, and the experimental results clearly demonstrated that the proposed model could obtain a high detection performance in comparison to state-of-the-art methods, and required minimal computational resources, making it feasible for many real-time deployments.

Keywords: CNN; object detection; sparse model; knowledge distillation; student–teacher model

Citation: Guo, J.-M.; Yang, J.-S.; Seshathiri, S.; Wu, H.-W. A Light-Weight CNN for Object Detection with Sparse Model and Knowledge Distillation. *Electronics* **2022**, *11*, 575. https://doi.org/10.3390/electronics11040575

Academic Editors: Leonardo Galteri, Claudio Ferrari and Stefanos Kollias

Received: 29 December 2021
Accepted: 8 February 2022
Published: 14 February 2022

1. Introduction

In computer vision evolution, object detection [1] is ubiquitous in many real-time applications, but still has many challenges for deployment. The main purpose of object detection is to detect the target object in an image, and locate a plurality of target positions and object types in each picture. The traditional object detection methods have many limitations, due to the computing hardware and availability of data [2,3]. In addition, the majority of methods rely on human annotations to create a knowledge base for a particular task, and computers are only required to make the inference rules. However, in recent years, with the growth of artificial intelligence and computational resources, the whole process can be machine driven, without much human effort. The major difference is that, classical object detection methods involve obtaining features through human experience rules and expert opinions, whereas artificial intelligence makes use of a complex neural network architecture, which can be trained to automatically extract strong and discriminative features. The field of computer vision covers a very wide range of research fields such as scene recognition [4], image reconstruction [5], image generation [6], face recognition [7], traffic scenes [8], object detection [9], etc. The proposed method was developed by keeping the important practical constraints such as a good detection accuracy, higher frame rate processing, and minimal computational resources. The main application scope of the proposed method is in autonomous driving and smart video surveillance systems.

2. Related Work

Object detection is generally performed with images or videos, and the objective is to locate borders and also to indicate the range and location of the object. Subsequently, to classify the category of the object (such as people, cars, airplanes, horses, etc.), and give the classification probability. In comparison with simple image classification, this is a more challenging task, because the position of multiple objects needs to be detected from the image or video. Objects may vary greatly in size or be occluded by other objects (only part of the appearance can be seen); or there may be changes in environmental light and shadow (backlighting, shadows), differences in the same category (dogs of different breeds and colors), and differences in viewing angles (head-up view, aerial photography, depression angle, etc.). Additional challenges occur in practical scenarios, which demand multiple object detection algorithms, performing within limited time constraints and computation resources.

Object detection is mainly divided into two different types of method. In terms of architecture flow, they can be directly divided into one-stage object detection and two-stage object detection.

The earlier methods in object detection used low-level eigenvalues for preliminary estimation. To avoid deficiencies, a large number (thousands) of candidate regions were generally used for subsequent object judgements. This process consumes a lot of computing resources, and is incapable of listing all the large and small candidate areas. The early strategy was to refer to the lines and textures of the picture to estimate the possible object position. For example, the selective search method, which is still frequently used, is based on this strategy. In other approaches, the texture distribution in the area is used as the feature value, and then a separator, such as a support vector machine (SVM) [10], is used to make judgments. A common method is a histogram of gradients (HoG) [11], which uses the magnitude and direction of change of shading (the difference between adjacent image values) to describe the texture. As for the evaluation method of object detection, IoU (intersection over union) is most commonly used to calculate the overlap of the detected object's target and its real position. The main issues with the traditional methods are the robustness towards object size, lighting conditions, object categories, and that most are limited to single object detection. With the advancements in deep learning methods, recent works have predominantly been based on deep convolutional neural network architectures.

The former works on deep learning-based approaches involve two-stage object detection, such as object position detection and classification, which are performed separately, and, hence, there is a large limitation in performance. Some early representatives include region-based convolutional neural networks (R-CNN) [12], spatial pyramidal pose net (SPP-NET) [13], etc. To improve the above-mentioned methods, many streamlined one-stage object detection approaches have been proposed. The main difference is that the object location detection and classification are changed to one-stage object detection models, such as you only look once (YOLO) [14–16], single-shot detector (SSD) [17], fully convolutional one-stage object detector (FCOS) [18], EfficientDet [19], etc.

Recently, the YOLOV4 [16] method verified the influence of State-of-the-Art's Bag-of-Freebies and Bag-of-Specials target detection methods in the detector training process. These training techniques successfully improved the speed and accuracy. The backbone architecture uses an improved CSPDarknet53 based on the YOLOv3 [14]. By dividing the base layer into two parts, the originally separated two parts are recombined through the transition layer, including a 1×1 convolutional layer and a pooling layer. Thus, the network architecture can be made richer, and the gradient of fusion features reduces the amount of calculation. This strategy increases the overall learning ability of the CNN network and also makes it lighter. The part of neck replaces the original feature pyramid network (FPN) [20] structure with a path aggregation network (PA-Net) [21], which is based on FPN to improve the single path from the original upper layer to the lower layer, to two-way transmission, with the use of a spatial pyramid pooling (SPP) [22] method to change the original feature addition to feature merging. Although there is a slight increase in calculation, the detection efficiency is improved.

More recently, YOLOv4-tiny [23] was proposed, which has a better efficiency than YOLOv4, and had promising results with the MS-COCO [24] dataset. This method used a powerful model scaling method, and the FPN structure is inherited in its once-for-all structure. Moreover, another notable work, termed Nanodet [25], was proposed based on the assign guidance module (AGM) and the dynamic soft label assigner (DSLA), and was incorporated and implemented in mobile devices. The Nanodet model can present a higher FPS rate than YOLOv4-tiny and has a better accuracy. In this work, we considered the two latest lightweight object detection models as the baseline, and developed an even more efficient and lightweight model, which can perform better than the above methods in terms of the FPS and detection accuracy.

The critical limitations and contribution of the existing works are as follows:

The main improvements are in terms of the lightweight backbone, anchor-free detection, sparse modelling, data augmentation, and knowledge distillation. The integration has substantially improvised the training, inference speed, and detection accuracy.

In Figure 1, a comparison of the leading object detection models, with respect to the average precision (AP) and frames per second (FPS), is illustrated. In comparison to all of the former methods, it can be seen that the proposed model can achieve the highest FPS, as well as a better precision in comparison to YOLOv4-tiny and EfficientDet. Moreover, the proposed model has a lightweight backend that can be quickly trained and combined with anchor-free detection methods.

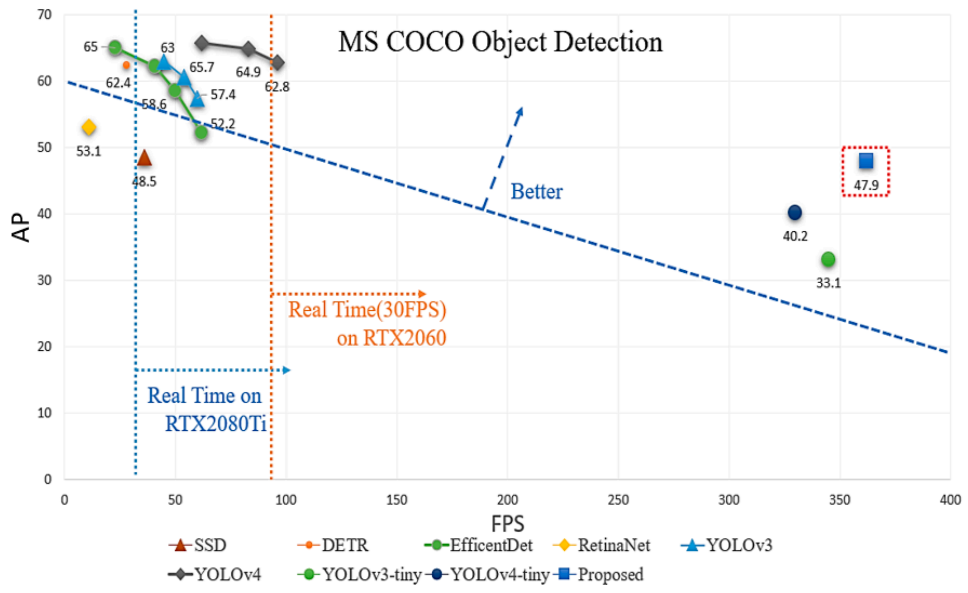


Figure 1. Comparison of average prediction (AP) accuracy with respect to the frame processing speed (FPS) against the state-of-the-art methods. SSD [17], EfficientDet [19], Retina Net [26], DETR [27], YOLO v3 [14], v3-tiny [15], v4 [16], v4-tiny [23].

It is concluded that most of the existing solutions are based on model search and iterative methods, which are computationally intensive and time-consuming. To overcome this problem, the proposed solution uses mixed precision [28] and sparse network generation [29] to improve the model training and inference speed.

In general, YOLOv4-tiny has a lower accuracy in place of its high FPS processing. In the proposed approach, the accuracy is improved through data augmentation and knowledge distillation [30], without compromising the processing speed. The proposed model has superior performance than YOLOv4-tiny, by around 8%; and this is achieved with a model size reduction of around 40%.

In a nutshell, the proposed method has advantages compared with other methods in terms of the processing efficiency and accuracy.

3. Proposed Work

The main network architecture of this study is shown in Figure 2 and is similar to one-stage object detection algorithms such as FCOS [18] and EfficientDet [19], and is mainly composed of a backbone, neck, and head.

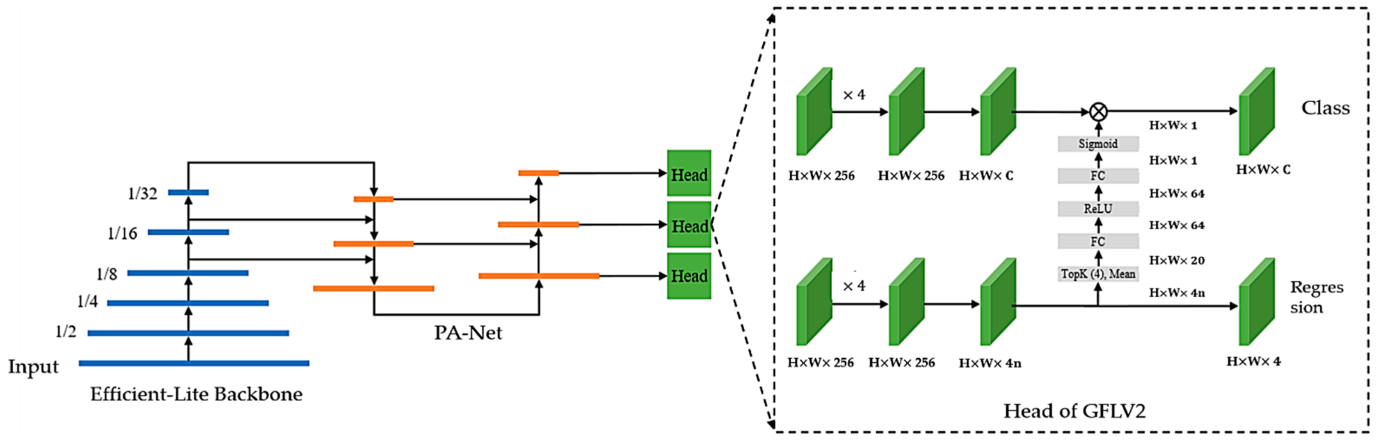


Figure 2. Proposed object detection architecture. The main components are the backbone architecture, which is based on the Efficient-Lite model [31], and a modified path aggregation network (PAN) [20] using Generalized focal loss v2 [32] as the detection head representation and loss function.

Normally, to localize and detect the category of the target from an image, we first extract some necessary feature data from the image, such as HOG features, and subsequently use these features to achieve positioning and classification. In the field of deep learning, the backbone of the network is responsible for extracting features from images. Among them, there are many well-known models to choose from, such as ResNet [33], MobileNet [34,35], ShuffleNet [36], etc. The reasons for choosing EfficientDet-lite [31] as the backbone for object detection in this study were as follows: EfficientDet based CNN model has achieved the state-of-the-art performance in object detection. It uses the AutoML MNAS [37] framework to execute a neural architecture search (NAS) to develop the base model and combine the features of compound scaling. It has a good accuracy and excellent model size, and has attracted widespread attention recently. EfficientNet-lite is a lightweight and improved version of EfficientNet, and the model removes the use of the squeeze-and-excite module, as this module is not optimized for mobile use, and the ReLU6 [38] activation function is replaced by a swish activation function. To make it easier to quantify, fixed stem and head modules are added to ensure the lightweight advantage of model scaling. Compared with the MobileNetV2 [35], ResNet-50 [33], and Inception-V4 [39], EfficientNet-lite has a better trade-off in performance between accuracy and size.

The classification of objects by this model does not have much room for improvement, due to the usage of a pre-trained backbone network. In addition, the accuracy of positioning has a great impact on the performance of the target detection algorithm. Among the various datasets or actual application scenarios, localization ambiguity is widespread. Specifically, it is difficult for lightweight object detection networks to detect fuzzy boundaries. Generalized focal loss [40] uses a general discrete distribution to express the uncertainty of the target frame, and the target frame is expressed as a probability distribution, without any prior knowledge restrictions. As opposed to the above methods, Zheng [28] introduced distillation learning into the localization branch of the target detection network, and proposed to use the localization distillation to improve the ability of the target frame, using a high-performance teacher network. This can solve the problems of location ambiguity and learning through distillation. Hence, the student network can solve the problem of location ambiguity similarly to the teacher network.

Conventional object detectors are usually trained offline, and, thus, researchers can make full use of this feature to find a better training method that improves accuracy without increasing inference overheads. In this approach, the training strategy is altered, referring to Bag of Freebies, which can improve the accuracy, for example, of some of the image pre-processing methods such as MixUp and CutMix. Furthermore, mosaic data augmentation, GIoU loss, label smoothing, knowledge distillation [30], etc., with mixed precision training [28] and sparsity model generation [29] are used.

The object detection and identification system proposed in this study is a one-stage object detection, inspired by Nanodet and based on a lightweight convolutional network, by modifying the path aggregation network (PAN) and using Generalized focal loss v2 as the detection head representation and loss function to achieve real-time object detection. In terms of the object detection, anchor-based models such as YOLO and SSD have always occupied a dominant position. The main purpose of this research was to develop an instant anchor-free detection model, which can provide a performance not inferior to YOLOv4-tiny, and that is also convenient for training and transplantation.

Some plug-in modules or post-processing methods can improve the detection accuracy, but have a small impact on the inference time, and these methods are called 'bag of specials'. Plug-in modules are usually used to enhance the specific attributes of the model, such as expanding the receptive field, introducing an attention mechanism, and strengthening the ability to combine features. Post-processing involves fine-tuning some mechanisms in accordance with the predicted results of the model. The post-processing comprises of fine-tuning some mechanisms based on the predicted results. For example, the structure of the FPN is adjusted along with the alternate activation function and Generalized Focal Loss v2 [32] to obtain superior classification accuracy.

Most of the previous studies have used FP32 (single precision) [23,25] for neural network training. In recent years, NVIDIA has proposed the use of FP32 + FP16 mixed format, namely mixed precision training (MPT). MPT [28] uses half-precision floating-point numbers to accelerate training, while minimizing precision loss. It uses FP16 to record weights and gradients. This has the effect of accelerating training, while reducing memory usage. The value range of FP16 is 5.96×10^{-8}~65,504, while FP32 is 1.4×10^{-45}~3.4×10^{-38}. It can be seen from the scope of FP16 that the biggest problem in comparison to the FP32 neural network is the reduction in accuracy. In addition, some points on the requirement of GPU computational resources for existing models and techniques for their efficient usage are presented.

Even though the introduction of GPUs has accelerated the training process; due to the rapid evolution in model complexity, the demand for multiple GPUs and tensor processing unit (TPUs) training keeps on increasing. For instance, the former architectures such as AlexNet used two GPUs to train ImageNet for 5–6 days in 2012. Similarly, training ResNeXt-101 in 2017 required eight GPU acceleration for more than 10 days. By 2019, Noisy-Student [41], a self-growth framework through knowledge distillation proposed by Google, required more than 1000 TPU training for 7 days. Some studies have begun to explore structured pruning and have achieved good results, but other studies have used unstructured sparsity pruning for accuracy; however, unstructured data (0-D) is difficult to use with modern vector and matrix math instructions, which has led to the development of more complex methods integrated with deep learning frameworks for acceleration.

As training models often uses the ReLU activation function, many weights are 0. In this study, it was observed that the gradient disappeared when training with ReLU6 after generating the sparsity model. This is due to the fact that 50% of the weights are not involved in the gradient update because of the mask, as shown in Figure 3. ReLU6 sets the negative values to 0, so that the overall network cannot continue to converge when very few weights can hold values. The unstructured sparse matrix caused by the ReLU6 cannot provide hardware acceleration through deep learning, and, thus, this study chose to use automatic sparsity [4] to generate 50% sparsity. This was the main reason for switching the activation function to the self-regularized non-monotonic activation function (Mish) [42] and sigmoid weighted linear units (SiLU) [43].

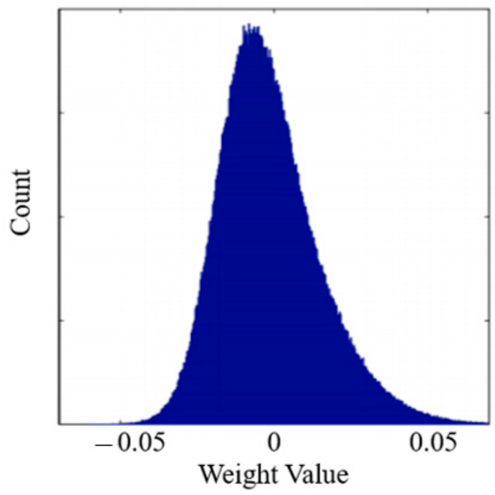


Figure 3. Model weight distribution before ReLU.

This study used the automatic sparsity module under the NVIDIA PyTorch Extension to generate sparse networks. As shown in Figure 4, in the weight matrix, two random masks for every four elements were called a 2:4 sparseness, and this can be supported in Tensor Core hardware level acceleration.

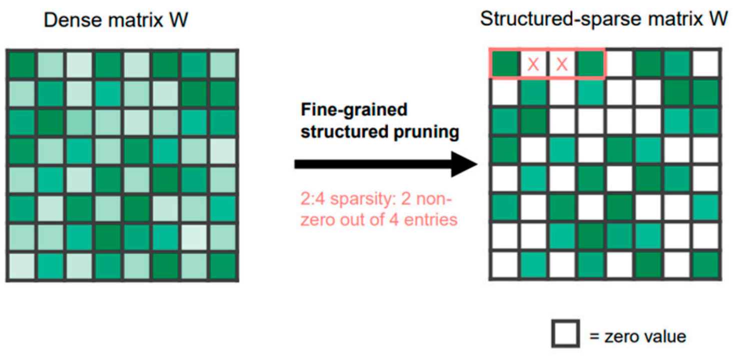

Dense matrix W
Structured-sparse matrix W
Fine-grained structured pruning
2:4 sparsity: 2 non-zero out of 4 entries
X
X
= zero value

Figure 4. 2:4 Sparsity Structure.

Using this approach to generate a sparse network, the original network could be trained to a certain stage, and subsequently the network sparseness could be improved. It can avoid the premature mask weight at the beginning, which causes the training convergence to be affected too much. As the convergence is stable, sparse network generation is used to improve the training speed and training accuracy.

4. Experiments and Results

This section covers the extensive experiments with the object detection technology to validate the performance of the proposed work. The description covers the dataset, evaluation metrics, and computational resources, and inference studies are provided. Furthermore, the implementation details of this study, including the model architecture for training and its parameters and the ablation studies are explained in detail. Finally, the public database MS COCO [24] was tested and compared with the former methods, as shown in Table 1. For model training and testing, 118 k and 5 k images were used, respectively. The results of each experiment were analyzed to verify the effectiveness of the proposed method. For evaluation, average precision (AP) was used, which refers to the area under the precision-recall curve. Generally, AP is calculated for all classes, and its average is defined as the mean average precision (mAP). Furthermore, the AP50 refers to the 50% region correctly detected in comparison to the ground truth, and for AP75 candidate images over 75% regions are counted.

Table 1. Comparison among the state-of-the-art detectors.

Model	Backbone	Model Size	FPS	mAP	AP 50	AP 75
SSD [17]	VGG	100 MB	36	28.8%	48.5%	30.3%
DETR [27]	R50+ Transformers	159 M	28	42.0%	62.4%	44.2%
EfficientDet [19]	EfficientNet-B0	15.5 MB	16	33.8%	52.2%	35.8%
RetinaNet [26]	ResNet101	329 MB	11	34.4%	53.1%	36.8%
YOLOv3 [14]	Darknet-53	246.5 MB	66	33.0%	57.9%	34.3%
YOLOv3 -Tiny [15]	Darknet-19	33.2 MB	345	15.3%	33.1%	12.4%
YOLOv4 -Tiny [23]	CSPNet-15	23.1 MB	330	22.0%	40.2%	20.7%
Proposed	EfficientNet -Lite	15.1 MB	362	30.7%	47.9%	32.4%

This study ensured the performance of the model for real-time applications with a good detection accuracy. To achieve a greater adaptability, EfficientNet-Lite with integrated scaling capabilities was used as the detection network (backbone). After fine-tuning, the SiLU was chosen as the activation function; compared to ReLU6, Mish, and LeakyReLU. The reason for this is that sparse pruning was used in the acceleration strategy. If ReLU is used, its characteristics cannot preserve negative values during the training process, as shown in Figure 5, which can easily lead to failures. Regarding the vanishing gradient issues, the model was tested with Mish and SiLU. SiLU was found to provide a better performance empirically, and, thus, was used as the activation function for the backbone model.

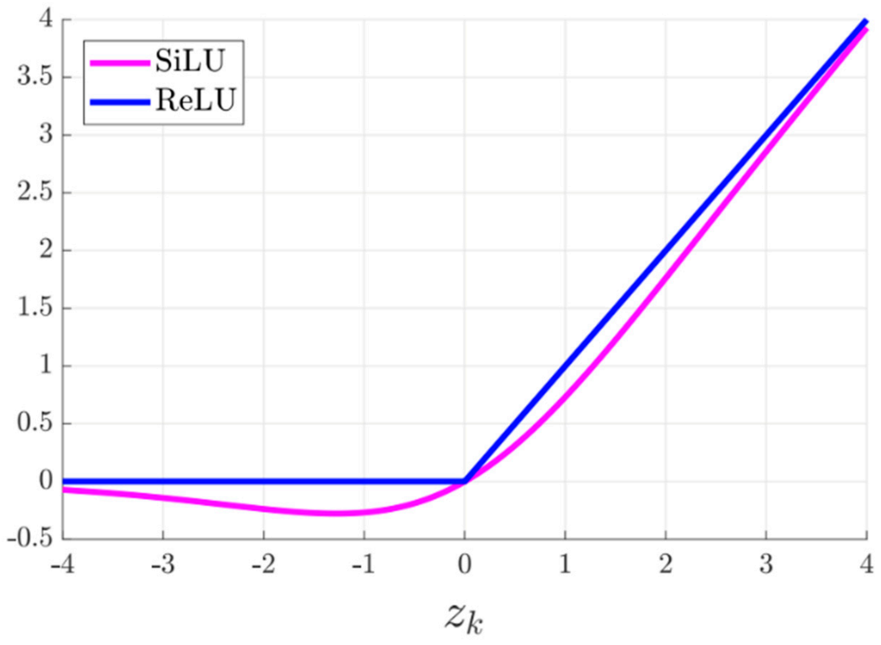


Figure 5. SiLU [43] and ReLU [38] activation function.

The learning rate was initialized from 0.0003 and increased up to 0.16 in 500 steps. The computer system comprised an Intel(R) Gold 5218 (2X) CPU with DDR4 2666 MHz ECC RDIMM 64 GB*8 RAM, and NVIDIA GeForce RTX 3090 GPU 24 GB*2. The batch size was set at 128. Mixed FP16 and FP32 precisions were adopted for training, the mixed parameters were set to NVIDIA Apex AMP O3, and the training time was around 4 days and 12 h.

During the inference stage, the model weights were saved and converted to ONNX [44] format. The converted model size was only 15.1 MB, compared to the 245 MB of YOLOv4 and 23.1 MB of YOLOv4-tiny. The lightweight model has great performance advantages for mobile or edge processing. Our proposed method uses a NVIDIA RTX 2080 Ti with NCNN [45] to give the model a more than 47 percent accuracy improvement over AP50, which also exceeds the detection performance of 300 FPS. We used the frames per second (FPS) and COCO2017 Val mAP as evaluation indicators to balance the evaluation of performance and accuracy. The FPS was tested using a single NVIDIA RTX 2080 Ti, to perform a batch size of 1 prediction time test, and convert it to per second. In total, there were 80 object categories in COCO, and mAP was calculated across the AP of each object category and taking the average value.

The ablation tests are provided in Table 2, in which EfficientNet-Lite0 and EfficientNet-Lite1 were adopted as the backbone of the network, to allow adaption in the experiment and achieve a good balance between performance and processing speed. EfficientNet was selected with experimental group 1 as the base, in which Lite0 as the backbone used 320 × 320 as the input resolution. Although the mAP of the control experimental group 1 was 5% lower than that of group 5, it had an advantage in terms of the speed, and it was still possible that the detection network could achieve a certain accuracy to reduce the probability of misjudgment or missed detection. Other experiments tested alternative activation functions and used different training strategies. If Generalized focal loss v2 was used, the model size could be slightly improved. Although the performance impact was small, it still had a positive impact on accuracy.

Table 2. Performance comparison of different EfficientNet versions.

Proposed	Backbone	Input Size	Activation Function	Head	Prune	Distillation	Mix Precision	mAP (0.5:0.95)	FPS
1	EfficientNet -Lite0	320	ReLU6	GFLv1				24.7%	405
2	EfficientNet -Lite1	416	Mish	GFLv1				27.35%	330
3	EfficientNet -Lite1	416	SiLU	GFLv1			●	27.61%	355
4	EfficientNet -Lite1	416	SiLU	GFLv2	●		●	29.72%	362
5	EfficientNet -Lite1	416	SiLU	GFLv2	●	●	●	30.7%	362

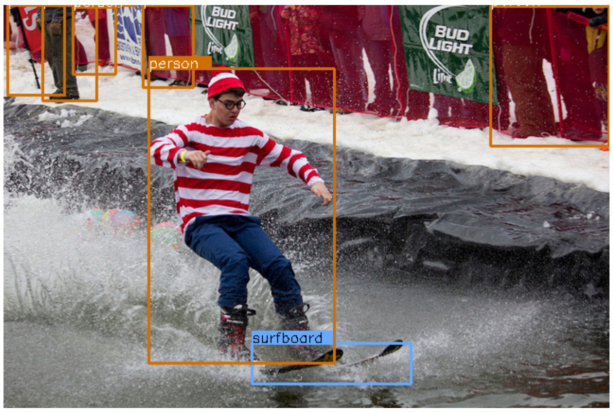
The MS-COCO [28] Val set was used to verify the performance of the proposed module. Experimental Group 2 used Mish as the activation function with reference to YOLOv4, and increased the input size to the same size 416 × 416 as YOLOv4-tiny, which was 2.6% higher than the baseline accuracy and around 5.3% higher than that of YOLOv4-tiny. Experiment Group 3 replaced the activation function with SiLU and added mixed precision training, which increased the performance by around 0.3% compared with experiment Group 2. Experimental Groups 4 and 5 both used the SiLU activation function and Generalized focal loss v2 as the detection head, and added mixed precision training and sparse network pruning, which could improve the performance in comparison with Group 3 by around 2.1%. If compared to the benchmark experimental Group 1, it was improved by around 5%, which is about 14.4% higher than YOLOv3-tiny and 7.7% than YOLOv4.

Experimental Group 5 added localization distillation on this basis to finally increase the accuracy to 30.7%. The overall parameters were only 4 M, and the final recorded model size was 15.1 MB.

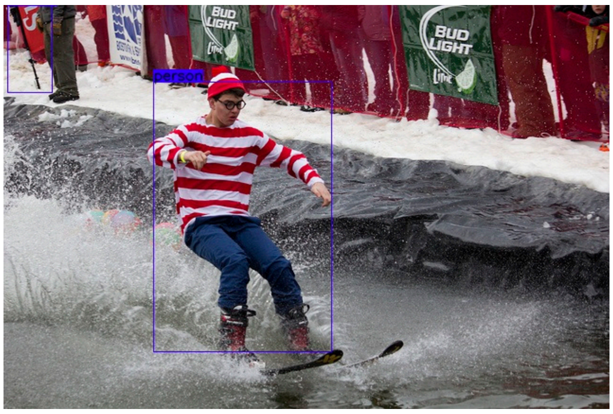
This study focused on balancing the execution speed and accuracy of the model. The one-stage anchor-free FCOS was used with ATSS [46] target sampling, and Generalized Focal Lossv2 [24] was used as the loss function to perform the target classification and regression calculation of the frame. This method removes the FCOS, and the centerness branch saves a lot of convolution operations. Furthermore, we added additional branches in order to use border regression statistics, to make up for the accuracy loss in classification; and the mixed precision training, sparse network pruning, model training parameter fine-tuning, mosaic stitching data enhancement, training strategies, such as data distillation, could not only ensure the speed of calculation, but also improved the accuracy. The evaluation value was superior to YOLOv4-tiny in terms of mAP by 7.7%.

Some practical images were adopted for comparison with Nanodet-320 and YOLOv4-tiny-416 and the proposed method with Group 5 configuration.

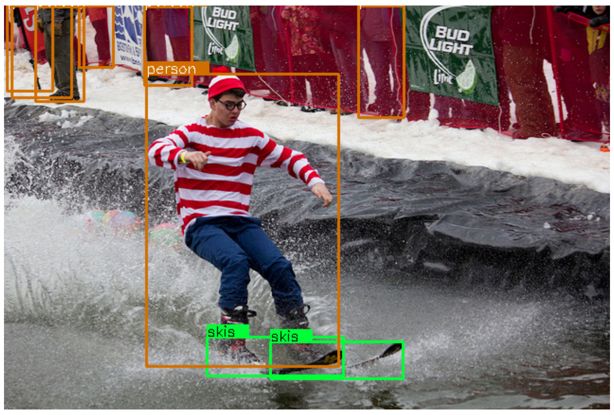
As shown in Figure 6, it can be seen that YOLOv4-tiny does not fit the boundary of the detected person, and the skis are not detected. In addition, the person at the top-left corner is not detected. On the other hand, Nanodet misclassifies skis as a surfboard. Conversely, the proposed method has more accurate results, which correctly classifies the skis, bounding boxes are precisely placed on the targets, and more individuals are detected in the top. Similar results are also found in the Figure 7, in which the proposed model detected more objects with good decision boundaries.



(**a**) Detected by Nanodet.

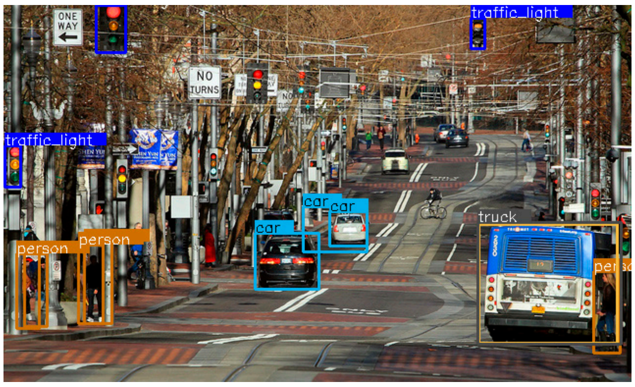


(**b**) Detected by Yolov4-tiny.

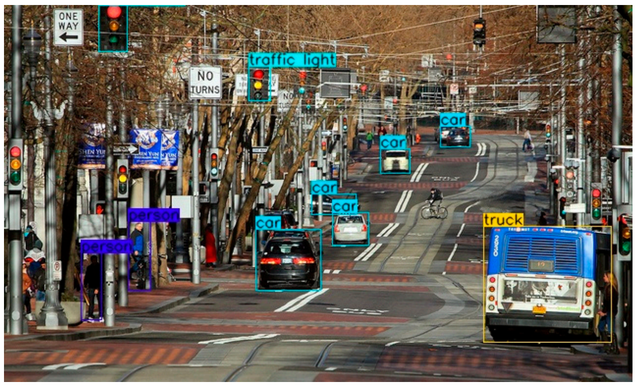


(**c**) Proposed.

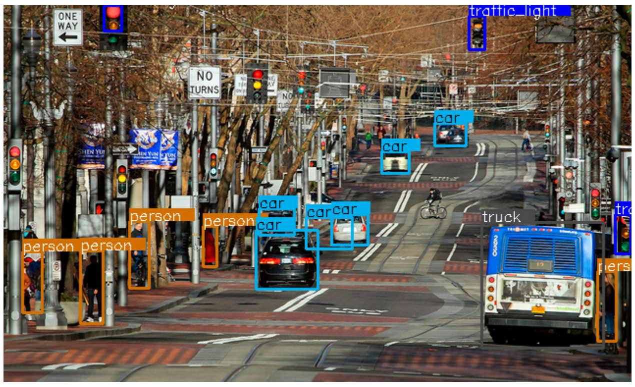
Figure 6. Performance comparison for water-skiing image.



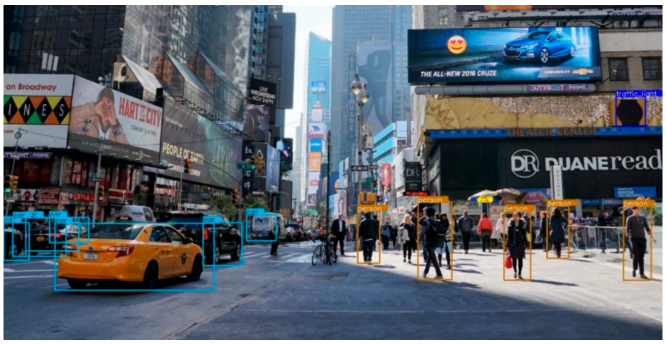
(**a**) Detected by Nanodet.



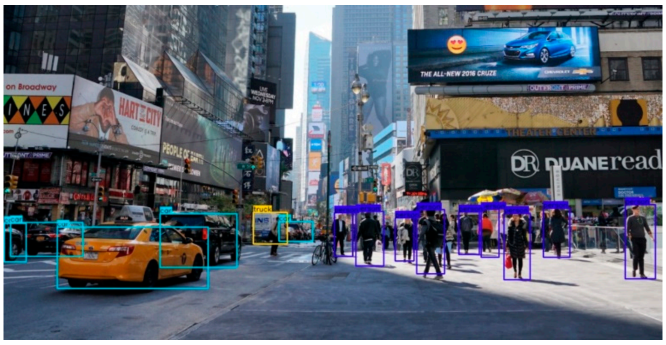
(**b**) Detected by Yolov4-tiny.



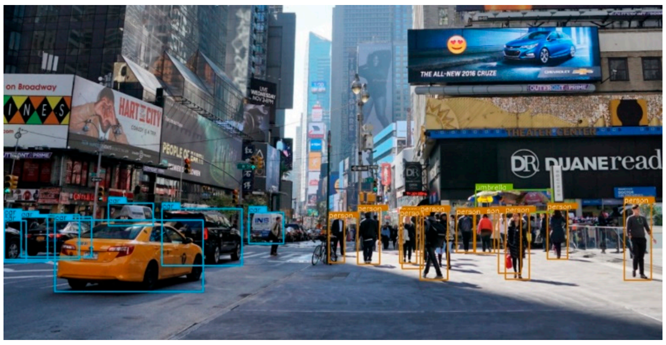
(**c**) Proposed.



(**d**) Detected by Nanodet.



(**e**) Detected by Yolov4-tiny.



(**f**) Proposed.

Figure 7. Performance comparison for detection of vehicles and people in city scenes.

From Figure 7a it can be seen that Nanodet failed to detect small vehicles, this is due to its smaller image input size to improve the detection speed, and, hence, it loses the ability to detect small objects. In Figure 7b, YOLOv4-tiny failed to detect some people standing near the vehicles, whereas from Figure 7c it can be inferred that the proposed method had a better multiple object detection capacity and also had a rapid execution. In addition, it can be seen that some traffic light signals and people cycling were not detected by the proposed method. From analysis, we found that an increase in input size can resolve some of these issues; however, this also affects the processing speed. Hence, there exists a clear trade-off between the prediction accuracy and detection speed, which will be addressed in the future. Figure 7d–f also illustrates the same phenomenon in city scenes, and the proposed model had a superior detection performance.

5. Conclusions

In this study, a lightweight CNN model was designed, and an accelerated training strategy was used to improve the inference speed, while maintaining a good detection accuracy. Robust object detection models normally require a deep architecture, which requires more parameters and is harder to train. In addition, many state-of-the-art objects models are trained with multiple GPUs or TPUs, which are not feasible for real-time applications and costlier for wide deployment. In this study, effective data enhancement was performed using the knowledge distillation method to improve the detection accuracy with the lightweight models. Moreover, the use of increased network sparsity, the reduction of excessive convolutional layer connections, the pre-selection mechanism of removing the box anchor, and the optimization of the loss function feedback mechanism of the detection head lead to a large-scale network using fewer model parameters. Experimental results on the public dataset MS-COCO 2017 showed that the proposed method could achieve a good detection accuracy, with rapid execution. The future scope of this work is to further improve the detection of multiple objects for high resolution images, without compromising the prediction speed. To summarize, the critical advantages of the proposed method are that the model can be easily trained and requires limited computational resources. Hence, the proposed method is feasible for many real-time applications and provides a more cost-effective solution.

Author Contributions: Conceptualization, J.-M.G.; methodology, J.-S.Y.; software, J.-S.Y.; validation, H.-W.W.; formal analysis, S.S.; investigation, J.-M.G.; resources, H.-W.W.; data curation, J.-S.Y.; writing—original draft preparation, S.S.; writing—review and editing, S.S.; visualization, H.-W.W.; supervision, J.-M.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Song, Z.; Chen, Q.; Huang, Z.; Hua, Y.; Yan, S. Contextualizing object detection and classification. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO, USA, 20–25 June 2011; pp. 1585–1592.
2. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26–30 June 2016; pp. 779–788.
3. Quang, T.N.; Lee, S.; Song, B.C. Object Detection Using Improved Bi-Directional Feature Pyramid Network. *Electronics* **2021**, *10*, 746. [CrossRef]
4. Herranz, L.; Jiang, S.; Li, X. Scene recognition with cnns: Objects, scales and dataset bias. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26–30 June 2016; pp. 571–579.
5. Kato, H.; Harada, T. Image reconstruction from bag-of-visual-words. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 955–962.
6. Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D.; Wierstra, D. Draw: A recurrent neural network for image generation. In Proceedings of the International Conference on Machine Learning, Bellevue, WA, USA, 28 June–2 July 2011; pp. 1462–1471.
7. Wang, H.; Wang, Y.; Zhou, Z.; Ji, X.; Gong, D.; Zhou, J.; Li, Z.; Liu, W. Cosface: Large margin cosine loss for deep face recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 5265–5274.

8. Wang, X.; Hua, X.; Xiao, F.; Li, Y.; Hu, X.; Sun, P. Multi-object detection in traffic scenes based on improved SSD. *Electronics* **2018**, *7*, 302. [CrossRef]
9. Zhao, L.; Li, S. Object detection algorithm based on improved YOLOv3. *Electronics* **2020**, *9*, 537. [CrossRef]
10. Wang, L. (Ed.) *Support Vector Machines: Theory and Applications*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2005.
11. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Diego, CA, USA, 20–26 June 2005; pp. 886–893.
12. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *38*, 142–158. [CrossRef] [PubMed]
13. Purkait, P.; Zhao, C.; Zach, C. SPP-Net: Deep absolute pose regression with synthetic views. *arXiv* **2017**, arXiv:1712.03452.
14. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
15. Adarsh, P.; Rathi, P.; Kumar, M. YOLO v3-Tiny: Object Detection and Recognition using one stage improved model. In Proceedings of the International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 6–7 March 2020; pp. 687–694.
16. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
17. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 11–14 October 2016; pp. 21–37.
18. Tian, Z.; Shen, C.; Chen, H.; He, T. Fcos. Fully convolutional one-stage object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 9627–9636.
19. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Seoul, Korea, 16–18 June 2020; pp. 10781–10790.
20. Lin, T.Y.; Dollar, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 22–25 July 2017; pp. 2117–2125.
21. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8759–8768.
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *37*, 1904–1916. [CrossRef] [PubMed]
23. Wang, C.Y.; Bochkovskiy, A.; Liao, H.Y.M. Scaled-yolov4: Scaling cross stage partial network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, WA, USA, 19–25 June 2021; pp. 13029–13038.
24. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft coco: Common objects in context. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 740–755.
25. NanoDet. Available online: https://github.com/RangiLyu/nanodet (accessed on 21 December 2021).
26. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal loss for dense object detection. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 2980–2988.
27. Zhu, X.; Su, W.; Lu, L.; Li, B.; Wang, X.; Dai, J. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv* **2020**, arXiv:2010.04159.
28. Micikevicius, P.; Narang, S.; Alben, J.; Diamos, G.; Elsen, E.; Garcia, D.; Ginsburg, B.; Houston, M.; Kuchaiev, O.; Venkatesh, G.; et al. Mixed precision training. *arXiv* **2017**, arXiv:1710.03740.
29. NVIDIA. Apex (A PyTorch Extension). Available online: https://github.com/NVIDIA/apex (accessed on 21 December 2021).
30. Zheng, Z.; Ye, R.; Wang, P.; Wang, J.; Ren, D.; Zuo, W. Localization Distillation for Object Detection. *arXiv* **2021**, arXiv:2102.12252.
31. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
32. Li, X.; Wang, W.; Hu, X.; Li, J.; Tang, J.; Yang, J. Generalized focal loss v2: Learning reliable localization quality estimation for dense object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 19–25 June 2021; pp. 11632–11641.
33. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 26–30 June 2016; pp. 770–778.
34. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
35. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 4510–4520.
36. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–22 June 2018; pp. 6848–6856.
37. Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; Le, Q.V. Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 2820–2828.

38. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the ICML, Haifa, Israel, 21–24 June 2010; pp. 807–814.
39. Szegedy, C.; Ioffe, S.; Vanhoucke, V.; Alemi, A.A. Inception-v4, Inception-Resnet and the impact of residual connections on learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017.
40. Li, X.; Wang, W.; Wu, L.; Chen, S.; Hu, X.; Li, J.; Tang, J.; Yang, J. Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection. *arXiv* **2020**, arXiv:2006.04388.
41. Xie, Q.; Luong, M.T.; Hovy, E.; Le, Q.V. Self-training with noisy student improves imagenet classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seoul, Korea, 16–18 June 2020; pp. 10687–10698.
42. Misra, D. Mish: A self-regularized non-monotonic activation function. *arXiv* **2019**, arXiv:1908.08681.
43. Elfwing, S.; Uchibe, E.; Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Netw.* **2018**, *107*, 3–11. [CrossRef] [PubMed]
44. Open Neural Network Exchange. Available online: https://onnx.ai/ (accessed on 21 December 2021).
45. Tencent NCNN. Available online: https://github.com/Tencent/ncnn (accessed on 21 December 2021).
46. Zhang, S.; Chi, C.; Yao, Y.; Lei, Z.; Li, S.Z. Bridging the gap between anchor-based and anchor-free detection via adaptive training sample selection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seoul, Korea, 16–18 June 2020; pp. 9759–9768.