IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 13, No. 1, March 2024, pp. 1129~1138
ISSN: 2252-8938, DOI: 10.11591/ijai.v13.i1.pp1129-1138

Backbone search for object detection for applications in intrusion warning systems

Nguyen Duc Thuan, Nguyen Thi Lan Huong, Hoang Si Hong
School of Electrical and Electronic Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam

Article Info

Article history:

Received Apr 8, 2023
Revised Oct 15, 2023
Accepted Nov 9, 2023

Keywords:

Architecture search
Dataset
Neural network
Object detection
Thermal image

ABSTRACT

In this work, we propose a novel backbone search method for object detection for applications in intrusion warning systems. The goal is to find a compact model for use in embedded thermal imaging cameras widely used in intrusion warning systems. The proposed method is based on faster region-based convolutional neural network (Faster R-CNN) because it can detect small objects. Inspired by EfficientNet, the sought-after backbone architecture is obtained by finding the most suitable width scale for the base backbone (ResNet50). The evaluation metrics are mean average precision (mAP), number of parameters, and number of multiply–accumulate operations (MACs). The experimental results showed that the proposed method is effective in building a lightweight neural network for the task of object detection. The obtained model can keep the predefined mAP while minimizing the number of parameters and computational resources. All experiments are executed elaborately on the person detection in intrusion warning systems (PDIWS) dataset.

Corresponding Author:

Hoang Si Hong
School of Electrical and Electronic Engineering, Hanoi University of Science and Technology
No. 1 Dai Co Viet Road, Hanoi, Vietnam
Email: hong.hoangsy@hust.edu.vn

1. INTRODUCTION

In recent years, object detection algorithms using deep learning have achieved remarkable results. They have been widely used in the implementation of practical systems such as intrusion warning systems [1], [2]. In these systems, object detection algorithms are used to detect human intrusions early by processing images from surveillance cameras [3]. With large systems, processing locally is essential because it is timely and reduces the load on the central server [4]. However, implementing deep learning algorithms on such device surveillance cameras is generally difficult because of hardware constraints [5]. Therefore, scientists have proposed methods to compact deep learning models, to efficiently implement them in practice.

In general, there are two approaches to deep learning model compaction: network architecture search (NAS) and model compression [6]. For the first approach, an optimal network architecture will be searched in a large space of potential architectures. This method is proven to be effective in cases where the designer has almost no knowledge of the data used [7]. The popular methods for object detection problems can be mentioned as: Detection NAS (DetNAS) [8], method in [9], feature pyramid network NAS (FPN-NAS) [10], auto feature pyramid network (Auto-FPN) [11], structural-to-modular NAS (SM-NAS) [12]. The common property of these methods is that they give a relatively good performance because the preset criteria are achieved during searching. In return, the computational cost for them including training time, power consumption is often very expensive and often have to run on powerful hardware.

For the second approach, the design of the neural network requires the programmer to have data knowledge such as the variety of data based on previous studies [13]. After that, a regular neural network model called the base model will be established without regard to the compaction criteria. To optimize the network, one will take advantage of model compression methods including pruning and quantization. These methods observe the base model to remove unnecessary parameters or approximate parameters with fewer byte representations. Methods of this type are usually not specific to any task, but rather general in machine learning. Well-known methods can be mentioned as: Filter pruning via geometric median (FPGM) [14], method in [15], method in [16], quantization after training (QAT) [17] and binarized neural networks (BNN) [18]. Compared with neural network architecture search, model compression methods significantly reduce the computational cost while maintaining the optimal criteria.

In this work, to save computational resources, we propose a simple neural network architecture search model for object detection. We refer to previous literature to design an object detection model based on faster region-based convolutional neural network (Faster R-CNN) [19] and ResNet50 [20]. We were inspired by EfficientNet [21] to find the width scale for the base model ResNet50 so that the scaled model meets the criteria for mAP, the number of parameters and the number of multiply-accumulate operation (MAC) operations. The reason we only looked for the width scale is that we found that many efficient object detection models use the feature pyramid network (FPN) structure [22], so scaling to the width should not affect it. Our main motivation is to suggest a simple, low-cost method that still guarantees reservation requirements.

2. RELATED WORKS

2.1. Faster R-CNN

The premier model within the R-CNN family is Faster R-CNN, introduced in 2015 [19]. Subsequent versions of R-CNN family networks have undergone enhancements primarily focused on computational efficiency, incorporating diverse training phases, reducing inference time, and boosting overall performance measured by mean average precision (mAP). These networks typically consist of: a) an algorithm to find "bounding boxes" or possible object positions in the image; b) the stage of extracting the features of the object, usually using a convolution neural network (CNN) network; c) a classification network to predict the class of object; and d) a regression layer to make the coordinates of the bounding boxes more accurate.

Faster R-CNN combines 2 modules shown in Figure 1. The first module is to use a deep neural network (DNN) to propose the regions, called region proposal network (RPN) and the second module is the Fast R-CNN model that uses the proposed regions. Fast R-CNN will take the suggested regions from the RPN to determine the object corresponding to the anchor. Faster R-CNN also has a CNN backbone network, a region of interest (ROI) pooling layer and a full connectivity layer, followed by two sub-branches to perform the two tasks of classifying objects and finding the best bounding box based on the regression.

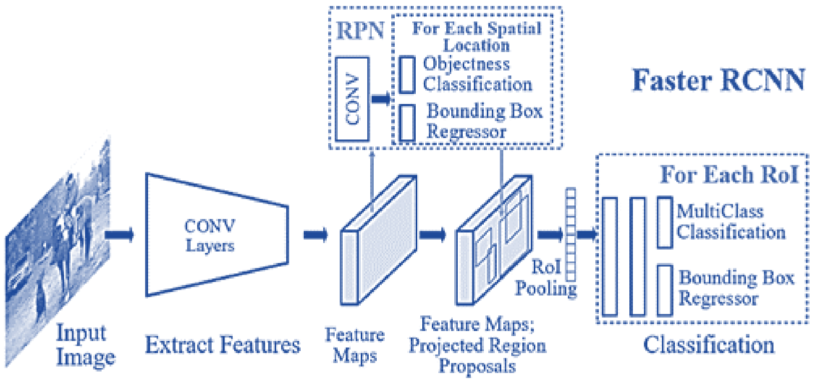

Figure 1. Faster R-CNN architecture [23]

2.1.1. Region proposal network (RPN)

Region proposal network RPN is the main innovation that makes faster R-CNN the best in the R-CNN family. RPN solves problems by training the neural network to substitute for selective search in previous versions, which is very slow. A region proposal network takes as input an image of any size and outputs a region proposal (set of locations of rectangles that can contain objects), along with the corresponding rectangle's probability of containing the object.

RPN has 2 main steps:

- Feed forward images via DNN to get convolutional features (We can use the available backbone networks such as VGG-16, ResNet50).
- Using the convolutional features slide up window: to create region proposals, we use a convolutional layer with sliding windows called anchors. The output of this layer is the input of two fully-connected layers that predict the location of the regions (Box-regression layer), as well as the probability of containing the object (Box-classification) of that box as shown in Figure 2.

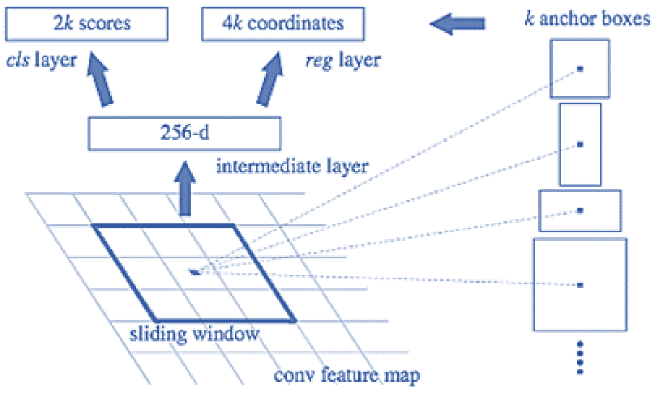

Figure 2. RPN architecture with anchor boxes [19]

2.1.2. Loss function

For each module in Faster R-CNN, we define a different loss function. With the Fast R-CNN module, the loss used is traditional cross-entropy. Whereas for the RPN module, the loss function is defined as follows,

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \frac{\lambda}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*) \quad (1)$$

where i is the index of the anchor in the mini-batch and p_i is the predicted probability of anchor i being an object. The ground-truth label value p_i^* is 1 if the anchor is positive, and 0 when the anchor is negative. t_i is a 4-dimensional vector representing the predicted bounding box coordinates. t_i^* is a 4-dimensional vector representing the coordinate value of the ground-truth box corresponding to the positive anchor. N is the number of anchor boxes taken out for consideration, λ is the equilibrium coefficient (=1). L_{cls} is the loss cross-entropy of 2 classes (Object and non-object), L_{reg} uses SmoothL1Loss.

2.2. PDIWS dataset

PDIWS [2] is a rare thermal image dataset for human detection in intrusion warning systems. The special feature of this dataset is that it is artificial data, synthesized by the method of image editing in the gradient domain. Each image in the dataset is made up of a human subject with an intrusive pose and a background. The detailed specification of the dataset is described in Table 1. Accordingly, the data set includes 2,000 training images and 500 test images in a ratio of 80:20. Note that the subjects in these two subsets are completely distinct, so it ensures randomness when testing. In addition, the subjects are also scaled in different proportions before being glued to the background, so it describes the various distances from the camera to the subject. Figure 3(a) shows an example of a creeping subject, Figure 3(b) shows an example of crawling subject, Figure 3(c) shows an example of stooping subject, Figure 3(d) shows an example of a climbing subject, and Figure 3(e) show an example of other subject in the dataset. In this paper, we divide the training set of the PDIWS dataset into two parts: training (1,750 images) and validation (250 images) at a ratio of 7:1, evenly distributed across classes. The purpose of this is to observe and evaluate the training process to decide to stop at the right time to avoid overfitting.

Table 1. PDIWS dataset description

Dataset specification	
Total number of training images	2,000
Total number of test images	500
Number of classes	5
Number of subjects	1,000
Number of backgrounds	50
Image resolution	320x240
Synthesis method	Poisson editing
Camera type	Testo 875-2i
Field of view	32°x23°

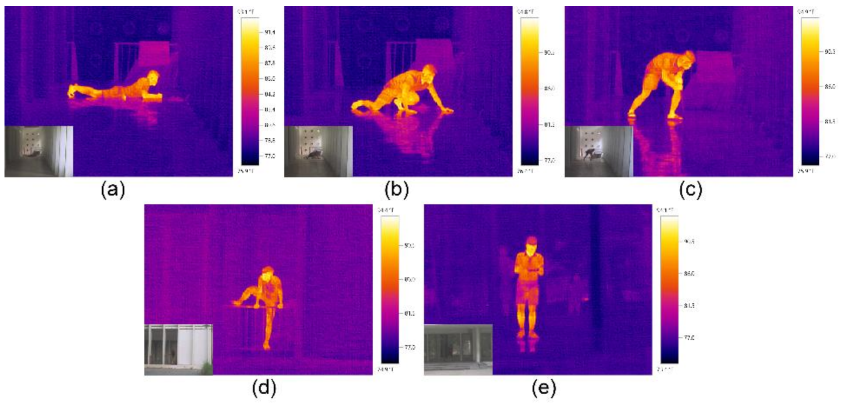

Figure 3. Example subject images in PDIWS dataset. (a) Creeping, (b) Crawling, (c) Stooping, (d) Climbing and (e) Other

3. METHOD

In this section, we describe in detail the architecture of the proposed model, the backbone width search algorithm, and the criteria for method/model evaluation. An overview of the proposed method is shown in Figure 4. The core part of the object detection algorithm is Faster R-CNN with the feature extractor based on ResNet50 highlighted in orange. The neural network architecture search algorithm named random width search is represented by curved orange arrows.

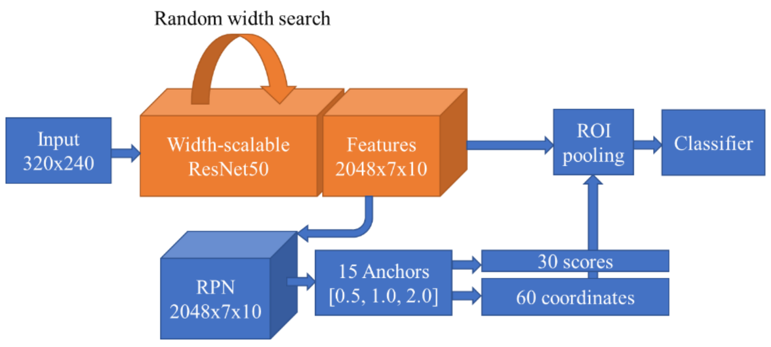

Figure 4. The architecture of the proposed backbone width search for object detection is based on Faster R-CNN and ResNet50

3.1. Object detection architecture

The proposed object detection architecture is based on Faster R-CNN. We divide this architecture into three parts for the convenience of description: feature extractor, region proposal network and output classifier (including region of interest (ROI) pooling and classifier). First, the input image is passed through a feature extractor to discover the features of the image. In the proposed architecture, the feature extractor is chosen as ResNet50 because it is popular in the object detection task along with visual geometry group network (VGG). We add a factor called width scale (w) to modify ResNet50 architecture as shown in Table 2 with the note that the number of output channels is the smallest upper bound divisible by 8. The output of the feature extractor is 2,048 feature maps with the size of each map reduced by 32 times in each dimension compared to the input. This 2048 feature map will go in two directions, one goes into the region proposal network and the other goes into the ROI pooling layer.

Table 2. Proposed ResNet50 architecture with width scale

Layer name	Kernel size	Output channels	Number of blocks
conv1	7×7	64×w	1
	1×1	64×w	
conv2_x	3×3	64×w	3
	1×1	256×w	
	1×1	128×w	
conv3_x	3×3	128×w	4
	1×1	512×w	
	1×1	256×w	
conv4_x	3×3	256×w	6
	1×1	1,024×w	
	1×1	512×w	
conv5_x	3×3	512×w	3
	1×1	2,048×w	
fully-connected	-	1,000×w	1
fully-connected	-	5	1

At the region proposal network, at each pixel in the feature map, it is further fed into the convolution layer to generate feature vectors of length 256. The number of feature vectors is equal to the number of predefined anchor boxes. Then, a fully-connected layer is used to predict the object's presence and bounding box according to each anchor box. These predictions are then fed into ROI pooling to remove overlapping regions of the same object. An important point in the RPN network is the anchor box. They are usually preselected according to the size and proportions of the objects contained in the training data. In the proposed method, the recommended anchor boxes are 15 types with aspect ratios of 0.5, 1.0, and 2.0.

Finally, the output classifier with ROI pooling and classifier is activated. The predicted bounding boxes then capture corresponding regions in the 2,048 feature maps. The non-maximum suppression (NMS) algorithm is then utilized to remove boxes containing the same object. Then the filtered boxes are reshaped to 7×7 by the ROI pooling layer before passing through the classifier. The classifier is a fully-connected layer that projects the class of the object. Loss functions for the proposed method are the same as the original faster R-CNN described in section 2.1.

3.2. Backbone width search algorithm

This is the most important part of the proposed method. As argued in the Introduction section, scaling the neural network in breadth is advantageous for inserting feature pyramid networks, which have recently been widely advertised. Therefore, we propose a simple solution to scale the width with the essence of changing the number of output channels of each hidden layer in the neural network. Our algorithm is described in Algorithm 1 as follows,

Algorithm 1: Backbone width search

```
input: width_scale ~ U{num_of_trial, 0;
1}, num_of_epoch,
model=Faster_RCNN, backbone_hist=[ ]
for interation = 1 to num_of_trial do:
backbone.layer.out_channel *=
width_scale[interation]
if backbone in backbone_hist then:
continue
end if
backbone_hist.add(backbone)
for epoch = 1 to num_of_epoch do:
loss_dict = model.train()
mAP, num_of_param, MACs = model.eval()
model.save()
log.save(loss_dict, mAP, num_of_param,
MACs)
end for
end for
output: model, log
```

Accordingly, we have to predetermine the number of trials (num_of_trial) and select width scales following the uniform distribution U(0;1). At each value of the width scale, we rebuild the backbone by multiplying the number of output channels of each hidden layer with the width scale. The resulting network architecture is saved to check for duplicates in subsequent attempts. Then the model is trained from scratch with the new architecture and then the model along with the loss dictionary, mAP, number of parameters, and number of MACs values will be saved. In this paper, we choose 10 trials for simplification. The configuration parameters for the training process will be presented in the results section.

3.3. Evaluation criteria

3.3.1. Mean average precision (mAP)

In binary classification, the model assigns prediction scores to samples. Categorization into classes depends on a set threshold. If the score meets or exceeds the threshold, the sample is in one class; otherwise, it's in the other. Positive classification occurs when the score is at or above the threshold; negative classification applies when the score falls below it. We use the terms precision and recall to evaluate the performance of the binary classification for each class as shown in Figure 5,

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \tag{2}$$

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \tag{3}$$

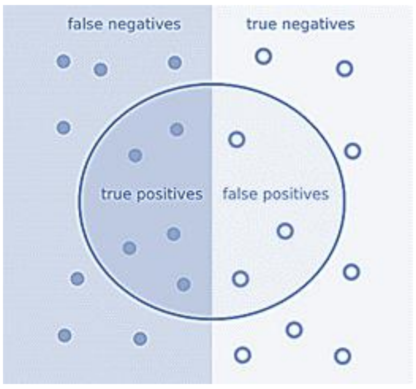

Figure 5. Illustration of binary classification predictions

Of course, we want a model to have high precision and high recall. To take into account both metrics, we use the precision-recall curve, which is a plot of the precision (y-axis) and the recall (x-axis) for different probability thresholds. Then we compute average precision (AP) by averaging the precision values on the precision-recall curve where the recall is in the range [0,1],

$$AP = \sum_{i=0}^{N} p\left(\frac{i}{N}\right) \tag{4}$$

where N denotes the number of thresholds and p(r) denotes the value of the precision when the recall is equal to r. The mAP is calculated by the mean of AP values over all classes.

3.3.2. Number of parameters and multiply-accumulate operations (MACs)

For a fully-connected layer, given the number of input neurons is a and the number of output is b. In this case, the number of parameters and MACs is determined by,

$$Parameters_{FC} = a \times b \tag{5}$$

$$MACs_{FC} = a \times a \times b \tag{6}$$

For a 2D convolutional layer, given the input is a×a×n, output of b×b×m and kernel size is k×k. Then the two terms parameters and MACs are defined by,

$$Parameters_{CNN} = k \times k \times n \times m \quad (7)$$

$$MACs_{CNN} = k \times k \times n \times b \times b \times m \quad (8)$$

4. RESULTS AND DISCUSSION

Our experiments are performed on the PDIWS dataset with the following training parameters: number of epochs (100), batch size (2), learning rate (0.005), momentum (0.9), optimizer (Adam). They are executed using Pytorch framework on a computer with Intel i7 12700F CPU, 16 GB RAM, and 24GB Nvidia GeForce GTX 3090 GPU. We first present the results of the backbone architecture search Table 3 with the criteria mAP_50 is mAP at IoU=0.5, mAP_75 is mAP at IoU=0.75 and mAP_coco is the average mAP with IoU=[0.5, 0.95] with step 0.05. As expected, architectures with small-width scales perform worse than larger ones. This can be explained by the small size of the model, the number of neurons is not enough to learn the necessary features. Meanwhile, network architectures larger than a certain threshold will not improve performance. Specifically, width scale values greater than 0.5 results in mAP_50 not being too different. The mAP value increases gradually as the width scale increases but does not exceed 47.55%, the increased amplitude is only approximately 2%. This trend is also present in mAP_75 and mAP_coco. Meanwhile, with a width scale of less than 0.5, the survey mAP values decrease exponentially as the width scale decreases from 0.5 to 0.1. On a scale of 0.1, the value of mAP_50 is only 18.17%, mAP_75 is 16.05%, mAP_coco is 13.54%, all extremely bad. In addition, for the number of parameters and MACs, the architectures searched had the number of parameters in million (M) exponential to the width scale (0.55M at 0.1 and 55M at 0.1). The number of MAC operations follows a similar trend, and varies in the form of an exponential function.

Table 3. Performance comparison between different searched model width

Model width	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
mAP_50 (%)	18.17	30.24	39.29	42.71	45.23	45.42	45.78	46.07	46.28	47.55
mAP_75 (%)	16.05	28.4	37.85	41.01	43.32	43.61	44.00	43.98	44.78	46.24
mAP_coco (%)	13.54	24.87	34.76	38.33	40.51	40.36	40.97	40.86	41.85	42.61
Parameters (M)	0.55	2.12	4.99	8.89	13.79	19.76	26.99	35.35	44.82	55.15
MACs (B)	0.08	0.28	0.62	1.13	1.89	2.89	4.09	5.54	7.31	9.48

Visualization of the table results can be found in Figure 6. Here, we can easily observe the trend of the criteria when the width scale changes. In order to choose a suitable scale, some threshold values must be predefined. In this paper, we think that the mAP_50 value must be above 45% for the model to be called good because this is the near-convergence value of the model. Therefore, from Figure 6, we choose a width scale value of 0.5 as the best value with mAP_50 reaching 45.42%, mAP_75 reaching 43.32%, mAP_coco reaching 40.51%, the number of parameters is 13.79M and the number of MAC operations is 1.89 billion (B). In addition, we can also choose another threshold value such as model size if the hardware is constrained or inference time if it requires real-time calculation.

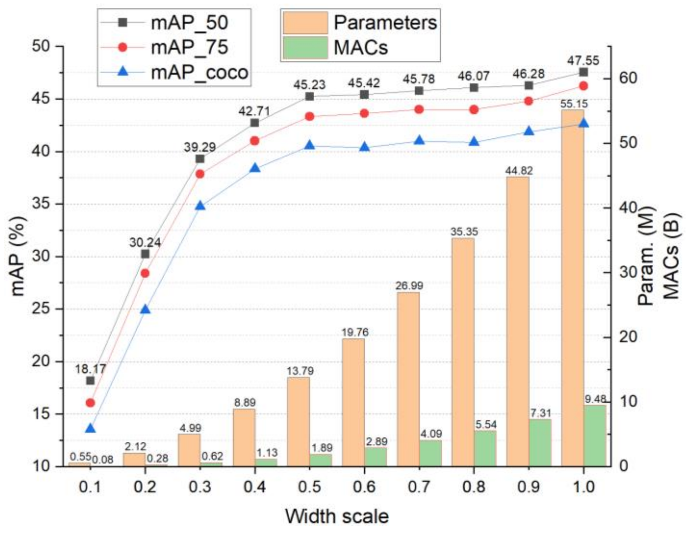

Figure 6. Visualization of search results

To increase the credibility of the obtained model as well as of the proposed method, comparative experiments were conducted. Table 4 summarizes the mAP values (at IoU=0.5), the number of parameters, MACs and inference time of various methods/models with different backbones. Specifically, the methods/models selected for the survey are well-known names such as single shot detection (SSD) [24], fully convolutional one-stage (FCOS) [25], you only live once version 3 (YOLOv3) [26]. Following them are popular backbones such as VGG-16 [27], ResNet50 [20], and MobileNet-V2 [28]. The results show that the criteria have trade-offs, so we should have a metric to compare the methods and choose the best one. Figure 7 shows some example results on test set with labeled bounding boxes over all classes.

Table 4. Comparison between the obtained model with other object detection methods

Method	Backbone	mAP (%)	Parameters (M)	MACs (B)	Inference (s)
Faster R-CNN	VGG-16	45.76	23.92	41.45	0.647
Faster R-CNN	MobileNet-V2	37.48	4.35	1.64	0.036
Faster R-CNN	ResNet50	47.55	55.15	9.48	0.216
SSD	VGG-16	40.48	23.84	41.12	0.374
SSD	MobileNet-V2	34.73	4.21	1.45	0.027
SSD	ResNet50	43.39	54.84	9.22	0.127
FCOS	VGG-16	41.73	23.81	41.06	0.329
FCOS	MobileNet-V2	33.94	4.18	1.42	0.018
FCOS	ResNet50	43.43	54.85	9.13	0.105
YOLOv3	VGG-16	44.17	23.72	40.85	0.246
YOLOv3	MobileNet-V2	36.76	4.12	1.32	0.012
YOLOv3	ResNet50	47.15	54.62	8.94	0.062
Proposed	0.5 ResNet50	45.23	13.79	1.89	0.068
	0.6 ResNet50	45.42	19.76	2.89	0.071

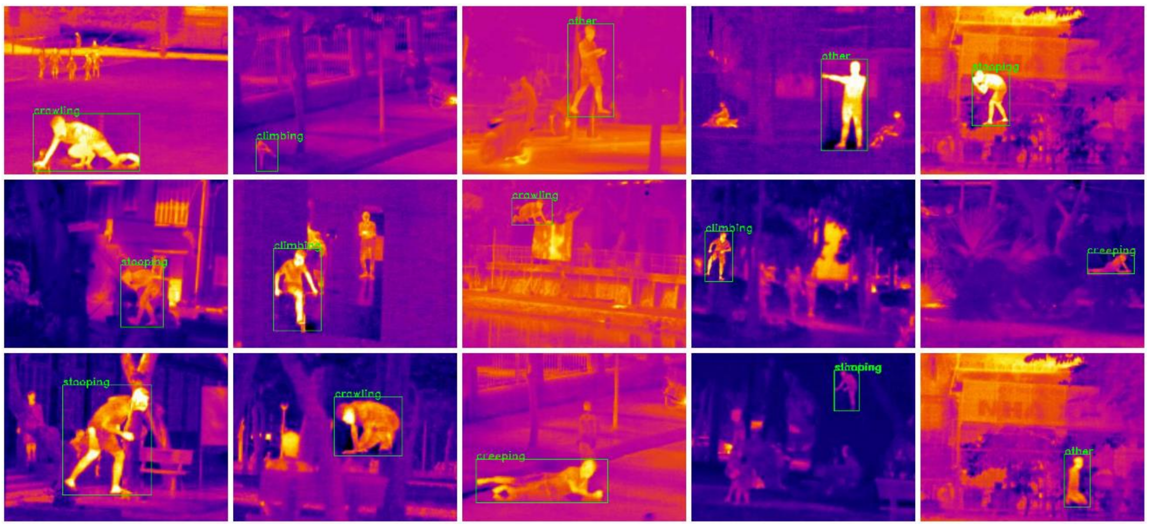

Figure 7. Examples of predicted bounding boxes over five classes in a test set

5. CONCLUSION

We have proposed a new and simple backbone architecture search method for object detection based on Faster R-CNN and ResNet50. The proposed method is performed on the PDIWS dataset so that it can be deployed on intrusion warning systems. The obtained model exhibits object detection with over 45% mAP with only 13.79M parameters and 1.89B MAC operations, outperforming other model scales. Comparative experiments with classical methods are also performed to evaluate the value of the obtained model. In summary, this work has contributed to supplementing theory and experimental results for the trend of object detection model optimization.

ACKNOWLEDGEMENTS

Nguyen Duc Thuan was funded by the Master, PhD Scholarship Programme of Vingroup Innovation Foundation (VINIF), code VINIF.2022.ThS.086.

REFERENCES

[1] N. D. Thuan, "Backbone-width-search-Faster-RCNN," [Online]. Available: https://github.com/thuan-researcher/Backbone-width-search-Faster-RCNN.

[2] N. D. Thuan, L. H. Anh, and H. S. Hong, "PDIWS: Thermal imaging dataset for person detection in intrusion warning systems," *IEEE Workshop on Statistical Signal Processing Proceedings*, vol. 2023-July, pp. 71–75, 2023, doi: 10.1109/SSP53291.2023.10208055.

[3] N. U. Huda, B. D. Hansen, R. Gade, and T. B. Moeslund, "The effect of a diverse dataset for transfer learning in thermal person detection," *Sensors (Switzerland)*, vol. 20, no. 7, 2020, doi: 10.3390/s20071982.

[4] G. Wang *et al.*, "BED: A real-time object detection system for edge devices," *International Conference on Information and Knowledge Management, Proceedings*, pp. 4994–4998, 2022, doi: 10.1145/3511808.3557168.

[5] N. D. Thuan, T. P. Dong, B. Q. Manh, H. A. Thai, T. Q. Trung, and H. S. Hong, "Edge-focus thermal image super-resolution using generative adversarial network," *2022 International Conference on Multimedia Analysis and Pattern Recognition, MAPR 2022 - Proceedings*, 2022, doi: 10.1109/MAPR56351.2022.9924742.

[6] X. He, K. Zhao, and X. Chu, "AutoML: A survey of the state-of-the-art," *Knowledge-Based Systems*, vol. 212, 2021, doi: 10.1016/j.knosys.2020.106622.

[7] P. Ren *et al.*, "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Computing Surveys*, vol. 54, no. 4, 2021, doi: 10.1145/3447582.

[8] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "DetNAS: Backbone search for object detection," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[9] X. Du *et al.*, "SpineNet: Learning scale-permuted backbone for recognition and localization," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 11589–11598, 2020, doi: 10.1109/CVPR42600.2020.01161.

[10] G. Ghiasi, T. Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 7029–7038, 2019, doi: 10.1109/CVPR.2019.00720.

[11] H. Xu, L. Yao, Z. Li, X. Liang, and W. Zhang, "Auto-FPN: Automatic network architecture adaptation for object detection beyond classification," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-October, pp. 6648–6657, 2019, doi: 10.1109/ICCV.2019.00675.

[12] L. Yao, H. Xu, W. Zhang, X. Liang, and Z. Li, "SM-NAS: Structural-to-modular neural architecture search for object detection," *AAAI 2020 - 34th AAAI Conference on Artificial Intelligence*, pp. 12661–12668, 2020, doi: 10.1609/aaai.v34i07.6958.

[13] N. D. Thuan, N. T. Hue, P. Q. Vuong, and H. S. Hong, "Intelligent bearing fault diagnosis with a lightweight neural network," *2022 11th International Conference on Control, Automation and Information Sciences, ICCAIS 2022*, pp. 261–266, 2022, doi: 10.1109/ICCAIS56082.2022.9990211.

[14] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 4335–4344, 2019, doi: 10.1109/CVPR.2019.00447.

[15] H. Li, H. Samet, A. Kadav, I. Durdanovic, and H. P. Graf, "Pruning filters for efficient convnets," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, 2017.

[16] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 2755–2763, 2017, doi: 10.1109/ICCV.2017.298.

[17] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018, doi: 10.1109/CVPR.2018.00286.

[18] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or -1," 2016, [Online]. Available: http://arxiv.org/abs/1602.02830.

[19] 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, doi: 10.1109/TPAMI.2016.2577031.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-December, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[21] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 10691–10700, 2019.

[22] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," *Proceedings - 2019 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking, ISPA/BDCloud/SustainCom/SocialCom 2019*, pp. 1500–1504, Dec. 2016, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00217.

[23] X. Xie, Y. Ma, B. Liu, J. He, S. Li, and H. Wang, "A deep-learning-based real-time detector for grape leaf diseases using improved convolutional neural networks," *Frontiers in Plant Science*, vol. 11, 2020, doi: 10.3389/fpls.2020.00751.

[24] W. Liu *et al.*, "SSD: Single shot multibox detector," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.

[25] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2019-October, pp. 9626–9635, 2019, doi: 10.1109/ICCV.2019.00972.

[26] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018, [Online]. Available: http://arxiv.org/abs/1804.02767.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018, doi: 10.1109/CVPR.2018.00474.

BIOGRAPHIES OF AUTHORS

Nguyen Duc Thuan received his B.S. degree from Hanoi University of Science and Technology (Vietnam) in 2021. He is pursuing an M.S. degree in Electrical and Electronic Engineering at Hanoi University of Science and Technology. His current major research fields include Signal Processing, Energy Harvesting, Fault Detection and Embedded Artificial Intelligence (EAI). He can be contacted at email: thuansvk@gmail.com.

Nguyen Thi Lan Huong received her M.S. degree in Measurement and Information from Kharkov Polytechnic University (Ukraine) in 1996 and her Ph.D. in Electrical Measurement from Hanoi University of Science and Technology (Vietnam) in 2005. From 1996 until now, she has been a lecturer at the School of Electrical and Electronic Engineering, Hanoi University of Science and Technology. The main research direction is the measurement and control systems, measurement signal processing, and research on sensory environments. She can be contacted at email: huong.nguyenthilan@hust.edu.vn.

Hoang Si Hong received his B.E. and M.E. degrees in Electrical Engineering from Hanoi University of Technology, Hanoi, Vietnam, in 1999 and 2001, respectively, and his Ph.D. degree from the University of Ulsan, Ulsan, South Korea, in 2010. Now, he is an Associate Professor at the School of Electrical and Electronic Engineering, Hanoi University of Science and Technology. His research interests include piezoelectric material, gas sensors, Graphene-based on SAW sensors, wireless sensors and harvesting energy and advanced measurement using Artificial Intelligence technology. He can be contacted at email: hong.hoangsy@hust.edu.vn.