YOLO Based Real-Time Human Detection for Smart Video Surveillance at the Edge

Huy Hoang Nguyen
School of Electronics and Telecommunications
Hanoi University of Science and Technology
Hanoi, Vietnam
Hoang.nguyenhuy@hust.edu.vn

Thi Nhung Ta
School of Electronics and Telecommunications
Hanoi University of Science and Technology
Hanoi, Vietnam
Nhung.tt152789@sis.hust.edu.vn

Ngoc Cuong Nguyen
School of Electronics and Telecommunications
Hanoi University of Science and Technology
Hanoi, Vietnam
Cuong.nn150510@sis.hust.edu.vn

Van Truong Bui
School of Electronics and Telecommunications
Hanoi University of Science and Technology
Hanoi, Vietnam
Truong.bv154012@sis.hust.edu.vn

Hung Manh Pham
Technology Center
Vnpt Technology
Hanoi, Vietnam
manhph@vnpt-technology.vn

Duc Minh Nguyen
School of Electronics and Telecommunications
Hanoi University of Science and Technology
Hanoi, Vietnam
minh.nguyenduc1@hust.edu.vn

***Abstract*— Recently, smart video surveillance at the edge has become a trend in developing security applications since edge computing enables more image processing tasks to be implemented on the decentralised network note of the surveillance system. As a result, many security applications such as behaviour recognition and prediction, employee safety, perimeter intrusion detection and vandalism deterrence can minimise their latency or even process in real-time when the camera network system is extended to a larger degree. Technically, human detection is a key step in the implementation of these applications. With the advantage of high detection rates, deep learning methods have been widely employed on edge devices in order to detect human objects. However, due to their high computation costs, it is challenging to apply these methods on resource limited edge devices for real-time applications. Inspired by the You Only Look Once (YOLO), residual learning and Spatial Pyramid Pooling (SPP), a novel form of real-time human detection is presented in this paper. Our approach focuses on designing a network structure so that the developed model can achieve a good trade-off between accuracy and processing time. Experimental results show that our trained model can process 2 FPS on Raspberry PI 3B and detect humans with accuracies of 95.05 % and 96.81 % when tested respectively on INRIA and PENN FUDAN datasets. On the human COCO test dataset, our trained model outperforms the performance of the Tiny-YOLO versions. Additionally, compare to the SSD based L-CNN method, our algorithm achieves better accuracy than the other method.**

Keywords—Edge computing, smart video surveillance, YOLO, human detection

I. INTRODUCTION

Over the years, video surveillance has developed significantly and become indispensable in terms of safety and security applications in various areas such as agriculture, commercial objects, gastronomy, private households, retail and public infrastructure. In the past, the video monitoring task was often performed by supervisors who had to manually observe the camera system. This resulted in harmful health effects for the supervisors' eyesight as they had to spend enormous amounts of time in monitoring multiple surveillance cameras. Today, thanks to advances in science and technology, security surveillance systems have become smarter. In recent years, intelligent video analysis has emerged as a promising approach to replace traditional and largely outmoded video surveillance solutions [1]. Through the means of this video analysis, supervisors are replaced by smart camera systems, capable of performing surveillance automatically. However, the problem is that when a system needs to handle vast numbers of cameras at the same time, the system needs a high processing speed and the system configuration must be powerful. This costs a significant amount of money. Not only that, the transmission of large amounts of video data from the camera to the server for processing can also be problematic due to overloading if there are more cameras added into the surveillance system. This results in large latency which does not guarantee real-time processing for security surveillance applications.

The edge computing technology migrates more computing tasks to the connected smart things (sensors and actuators) at the edge of the network [2]. Edge computing offers a far more economical route to scalability, allowing companies to expand their computing capacity through a combination of IoT devices and edge data centres. Temporary disruptions in intermittent connectivity will not impact smart device operations just because they have lost connection to the cloud. Edge computing increases network performance by reducing latency. Since applications or services process data locally or in nearby edge data centres, the physical distance is reduced and communication delays are considerably lessened. Therefore, edge computing is a potential solution for solving the smart video surveillance problem.

Human detection is a major task for various surveillance applications including abnormal action recognition, employee safety, perimeter intrusion detection and vandalism deterrence. Moreover, it has attracted much research interest in the realm of proposing solutions for implementing this task on edge devices. In the past, utilising handcrafted features to train the human detector was the main approach for implementing the human detection on edge devices. Due to the diversity and complexity of environments, human poses, and appearance, it was tough for researchers to develop an efficient manual human feature extractor [1]. In recent years, deep learning has emerged as a potential solution to overcome the problems of traditional methods. However, while these modern approaches are good at object detection and achieve high accuracy rates, their performance relies

heavily on GPU acceleration. Therefore, it is challenging to run these human detectors on computationally limited platforms and still guarantee the real-time requirements.

From the above analysis, a good solution for the human detection problem is to build a network to achieve a good trade-off between accuracy and processing time. Accordingly, we propose a new model for human detection based on a combination of the YOLOv2 network, Residual blocks and multiple SPP. By using NNPACK [3] to optimise network computation, the proposed architecture not only has few parameters but also good accuracy and less computational costs. Through experimental results, the developed human detector can process an average of 2 frames per second (FPS) on the selected edge device, Raspberry PI 3B board. According to a study of Nikouei et al., the real-time human detection task in real-world surveillance video streams can be accomplished even if the object detector executes only two times per second [4]. Therefore, the obtained results in this study satisfy the design goal of taking into account the computationally limited edge device.

In summary, this work contains two major contributions. The first contribution is the development of a real-time human detector for the computationally limited edge device in terms of smart video surveillance. The second contribution is the implementation of an extensive experimental study including an evaluation of the proposed model on various human datasets and a comparison of the trained detector with some other human detection methods, Tiny-YOLO variants [5] and SSD based L-CNN [32].

The rest of this work is organised as follows. Section 2 delivers a brief discussion of related works regarding human detection at the edge. The details of the proposed human detection approach are described in Section 3. Section 4 manifests and discusses the experimental results of the training and testing stages. Finally, Section 5 concludes the study.

II. RELATED WORKS

In the early studies of human detection on edge devices (specifically, Raspberry Pi variants), the general approach consisted of two steps: object detection and object classification. In the first step, moving objects can be detected by various techniques such as frame difference, background subtract, and optical flow. In the second step, the classification decision is made on the detected object utilising the features extracted of and the model used to classify [6]. In a study of Dalal and Triggs, the researchers utilised Histogram of Oriented Gradients (HOG) to extract the human features [7]. Then, these extracted features were adopted to train an SVM based human classifier. Other features applied for the human feature extraction were SURF [8], LBP [9], Edgelet [10], Haar-Like [11], Shapelet [12]. Apart from the SVM method, Naive Baysesian [13] and AdaBoost [14] represent two alternatives for developing the human classifier.

In recent years, deep learning methods have greatly enhanced progress in various visual recognition tasks based on parallel computing and advances in GPU technology. The most common models in the object detection task are one-stage and two-stage detection approaches. Basically, the two-stage detection method firstly utilises the selective search or regional proposal network to generate a set of regions of interest which potentially contain target objects. Then, a classifier processes these region candidates to identify the regions representing the target objects with the highest confidence. As opposed to the two-stage detectors, the one-stage detectors skip the region proposal stage and run detection directly over a dense sampling of possible locations. Therefore, the one-stage detection approaches are faster and simpler than the two-stage detection approaches. Research reveals that the R-CNN [15], Fast R-CNN [16], Mask R-CNN [17], YOLO [18] and SDD [19] are well-known deep learning based object detection approaches.

III. METHODOLOGY

In the field of object detection, the YOLO [18] network model is well known for having the capability of detecting multiple objects in real-time. According to [20], the YOLO v2 is faster and stronger than the Fast RCNN, YOLOv1, SSD. Despite the YOLOv3 [21] providing higher accuracy than that of the YOLOv2, the frame rate of the YOLO v3 is not higher than that of the YOLOv2. Therefore, our approach is to utilise the YOLOv2 as the basic framework and then apply some modifications in the network parameters and structure to simultaneously achieve real-time performance and high accuracy on the computationally limited device.

A. Yolo based human detection

Like the YOLOv2 [20], our human detection method adopts a single deep network to predict the target location and the confidence score for each location from the entire image. Basically, the detection process is briefed in the following two steps:

Step 1: The proposed model subdivides the input image into an *m x m* grid which takes responsibility for detecting whether the person's centroid is in it. Through the network, each grid cell contains the predicted bounding boxes and their confidence scores. The information of each predicted box is expressed by (x, y, w, h). The (x, y) values denote the coordinates of the bounding box's centroid, and the (w, h) values are the bounding box's width and height, respectively. The confidence score is given by the following equations:

$$Conf\ (person) = P\ (person)\ .IOU_{pred}^{truth} \quad (1)$$

$$P\ (person) = \begin{cases} 0, & contain\ person \\ 1, & not\ contain\ person \end{cases} \quad (2)$$

$$IOU_{pred}^{truth} = \frac{area\ (box\ (truth) \cap box\ (pred))}{area\ (box\ (truth) \cup box\ (pred))} \quad (3)$$

Where, P (Person) stands for if there is a person in this grid cell, and IOU denotes the ratio of the union and the intersection of the predicted box and the ground truth box.

Step 2: To enhance the capability of detection, the predicted boxes with low confidence scores will be deleted by setting a threshold value. Additionally, in relation to the remaining bounding box results, the non-maximum suppression algorithm is implemented to eliminate the redundant predicted boxes.

B. Network design

The proposed network structure is designed by following steps. In the first step, Model 1 is obtained by modifying the

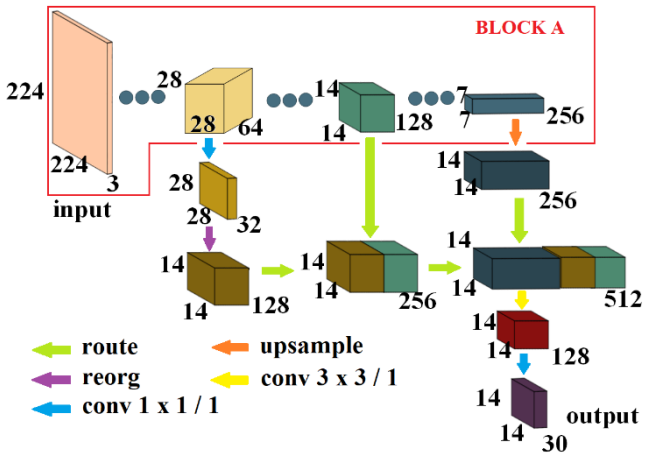


Fig. 1. The network structure of the Model 1

YOLOv2 network. In the second step, Residual blocks are integrated into Model 1 to generate Model 2. In the final step, Model 3 is a combination of the Model 2 and multiple Spatial Pyramid Pooling blocks.

1) Step 1: YOLOv2 network modification

The main idea to modify the YOLOv2 is to decrease the input size of the network model and increase the output size of the network model to achieve a good trade-off between the accuracy and the speed. To do this, we firstly keep the first 23 layers of the YOLOv2 model, and the rest of the YOLOv2 model is removed. Then, we reduce the number of filters in the convolution layers of the obtained model. Next, we add some extra layers into the obtained model (Block A). This design enables both high level information and low level information to be passed to the detection layer. As a result, the accuracy of detection and location is enhanced.

Table I and Fig. 1 show the parameter settings of the Block A and details of the Model 1, respectively when the size of the input image is 224 x 224 x 3. It can be seen from Fig. 1, aggregating feature map from multiple levels focuses on feature maps 28 x 28, 14 x 14 and 7 x 7. Feature maps 28 x 28 take to a reorg layer with a stride of 2. The reorg layer just reshapes feature maps without changing elements. The

TABLE I. THE NETWORK'S PARAMETERS OF BLOCK A IN MODEL 1

Layer	*Filter*	*Size/stride*	*Input*	*Output*
1 conv	8	3x3 / 1	224x224x3	224x224x8
2 max		2x2 / 2	224x224x8	112x112x8
3 conv	16	3x3 / 1	112x112x8	112x112x16
4 max		2x2 / 2	112x112x16	56x56x16
5 conv	32	3x3 / 1	56x56x16	56x56x32
6 conv	16	1x1 / 1	56x56x32	56x56x16
7 conv	32	3x3 / 1	56x56x16	56x56x32
8 max		2x2 / 2	56x56x32	28x28x32
9 conv	64	3x3 / 1	28x28x32	28x28x64
10 conv	32	1x1 / 1	28x28x64	28x28x32
11 conv	64	3x3 / 1	28x28x32	28x28x64
12 max		2x2 / 2	28x28x64	14x14x64
13 conv	128	3x3 / 1	14x14x64	14x14x128
14 conv	64	1x1 / 1	14x14x128	14x14x64
15 conv	128	3x3 / 1	14x14x64	14x14x128
16 conv	64	1x1 / 1	14x14x128	14x14x64
17 conv	128	3x3 / 1	14x14x64	14x14x128
18 max		2x2 / 2	14x14x128	7x7x128
19 conv	256	3x3 / 1	7x7x128	7x7x256
20 conv	128	1x1 / 1	7x7x256	7x7x128
21 conv	256	3x3 / 1	7x7x128	7x7x256
22 conv	128	1x1 / 1	7x7x256	7x7x128
23 conv	256	3x3 / 1	7x7x128	7x7x256

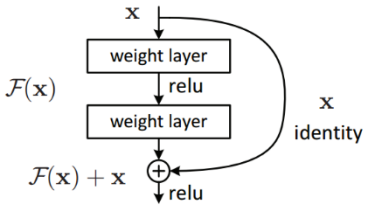


Fig. 2. Residual block

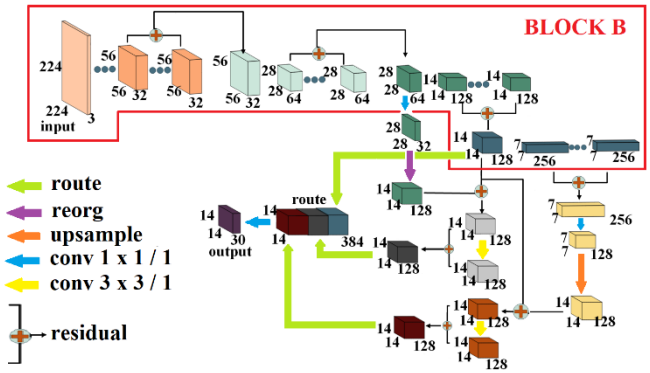


Fig. 3. The network structure of the Model 2

width and height will be decreased by 2 times and the number of channels will be increased by 4 times. Feature maps 7 x 7 take to an up-sample layer whose output is feature maps 14 x 14. The up-sample layer increases W x H resolution of input by duplicating elements, without changing the number of channels. Then, using the route layer to merge three layers with a scale of 14 x 14 in depth. Finally, the output of the route layer is taken into 2 convolution layers to obtain the final feature maps 14 x 14 x 30.

2) Residual blocks

In a network with Residual blocks, each layer feeds into the next layer and the layers about 2 – 3 hops away [22]. According to the study of Raghunandepu [23], residual connection based adding new layers guarantee that the performance of the model could increase slightly. Therefore, we combine Model 1 with some Residual blocks to obtain a new residual deep network called Model 2. To do this, the Block A of Model 1 is firstly modified to attain the Block B for Model 2. This modification includes a replacement of all max-pool layers by the convolutional layers 3 x 3 with a stride of 2 and an addition of shortcut layers before down-sampling layers. Then, we merge the multiple levels based feature maps aggregation with Residual blocks. The idea of this merging is based on [24]. Fig. 2 and 3 show the diagram of an original Residual block and the network structure of Model 2 with the input size of 224 x 224 x 3, respectively.

3) Multiple Spatial Pyramid Pooling (MSPP)

To enhance the capability of object detection, we add the Spatial Pyramid Pooling block into Model 2. Unlike existing studies of the SPP module [25], we utilise three SPP blocks instead of using only a single SPP block. Moreover, each SPP module includes 3 parallel max-pool layers with kernel sizes of 2 x 2, 3 x 3 and 4 x 4, and the stride of 1 is applied for all the max-pool layers of the SPP blocks. After concatenating three pooled feature maps and the SPP block's input, the channel pruning is utilised to reduce and refine the SPP feature channels [26]. The MSPP output is the concatenation of the three outputs of the SPP channel pruning. Details of the single SPP block and the final designed model (Model 3) with

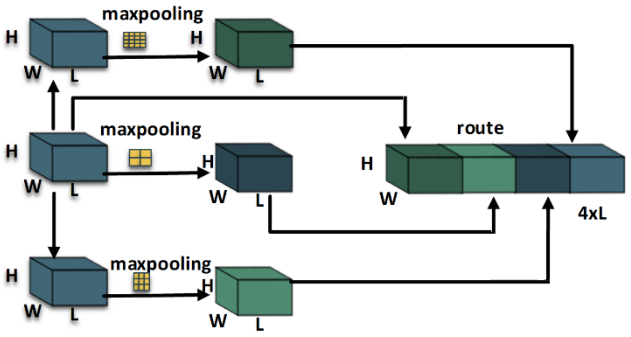


Fig. 4. The structure of the SPP block

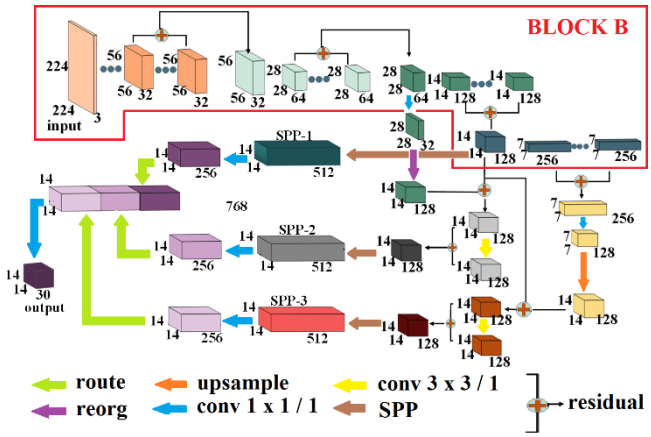


Fig. 5. The network structure of Model 3

the input size 224 x 224 x 3 are displayed in Fig. 4 and Fig. 5, respectively.

C. NNPACK based network computation optimisation

To speed up the human detection inference performance on computationally limited edge devices, we utilise NNPACK, an open source library available on GitHub. NNPACK is an acceleration package for neural network computations [3]. This package focuses on optimising high-performance implementations of convolution layers to speed up CPU inference. Moreover, this package not only supports various platforms including Linux, MAC OS X and Android but also is compatible with the Intel x86-64 processor using the AVX2 instruction set as well as the ARM v7 & v8 utilising the NEON instruction set.

TABLE II. THE NETWORK'S PARAMETERS OF BLOCK B IN MODEL 2 & 3

Layer	*Filter*	*Size/stride*	*Input*	*Output*
1 conv	8	3x3 / 1	224x224x3	224x224x8
2 conv	16	3x3 / 2	224x224x8	112x112x16
3 conv	16	3x3 / 1	112x112x16	112x112x16
4 conv	32	3x3 / 2	112x112x16	56x56x32
5 conv	32	3x3 / 1	56x56x32	56x56x32
6 conv	16	1x1 / 1	56x56x32	56x56x16
7 conv	32	3x3 / 1	56x56x16	56x56x32
8 res L 4				
9 conv	64	3x3 / 2	28x28x32	14x14x64
10 conv	64	3x3 / 1	14x14x64	14x14x64
11 conv	32	1x1 / 1	14x14x64	14x14x32
12 conv	64	3x3 / 1	14x14x64	14x14x64
13 res L 9				
14 conv	128	3x3 / 2	14x14x64	14x14x128
15 conv	128	3x3 / 1	14x14x128	14x14x128
16 conv	64	1x1 / 1	14x14x128	14x14x64
17 conv	128	3x3 / 1	14x14x64	14x14x128
18 conv	64	1x1 / 1	14x14x128	14x14x64
19 conv	128	3x3 / 1	14x14x64	14x14x128
20 res L 14				
21 conv	256	3x3 / 2	14x14x128	7x7x256
22 conv	256	3x3 / 1	7x7x256	7x7x256
23 conv	128	1x1 / 1	7x7x256	7x7x128
24 conv	256	3x3 / 1	7x7x128	7x7x256
25 conv	128	1x1 / 1	7x7x256	7x7x128
26 conv	256	3x3 / 1	7x7x128	7x7x256
27 res L 21				

D. Model training

The MS COCO 2017 [27] dataset contains 80 object classes with a total of 123287 labeled images splitting to 118k/5k for train/validation. For the problem of human detection, 66809 human images are extracted from this dataset for the purposes of this work. 64115 items of the extracted dataset are utilised for the training processing, and the rest are adopted for validation. The image background of the MS COCO dataset is complex. Additionally, the human poses, the scale of the objects and the level of occlusion are different and diverse. Therefore, the model trained on this dataset can achieve a good generalisation and deal with complicated environmental conditions when detecting individuals.

In this study, we adopt the open-source neural network framework named Darknet [28] to implement the model training. Similar to the YOLOv2, the number of anchors are kept at 5. Due to the training dataset only focusing on the human object, the anchors' parameters are recalculated by using the k-mean cluster algorithm. In order to enhance object detection with various image sizes, randomised input images are resized during the training. The human detector is trained in 500000 iterations. The basic learning rate is set at 0.001, and it will be decreased 10 times at 400000^{th} and 450000^{th} iterations. Several other training settings are reported in the Table III.

IV. EXPERIMENTAL RESULTS

A. Training and testing results on the human COCO dataset

For the human detector training stage, the proposed models were trained on a computer with an Intel Core I7 – 9700K CPU @ 3.6 GHz x 8 and 32 GB RAM running 64 bit Ubuntu 18.04 operating system. A RTX2080Ti GPU was utilised to support the training process. Fig. 6 displays training results of the Model 1, 2 and 3. In Fig. 6, the horizontal coordinate indicates the number of training iterations, while the vertical coordinate indicates the average loss values of three models during training. It can be observed from Fig. 6 that the average loss curves of the three models tend to be downward although they fluctuated during training. Among the three trained models, the Model 3 had the lowest average loss overall. This result shows that the network model employing Residual blocks and multiple SPP modules can fit the human detection task better than the model which doesn't use them.

To evaluate the detection performance of the trained models, a popular metric called Average Precision (AP) is adopted in this experiment. The AP summarises the shape of the precision/recall curve and is defined as the mean precision at a set of recall levels from 0 to 1 [29]. Testing results of the three trained models on the human COCO validation dataset are reported in Table III. In Table III, while the integration of

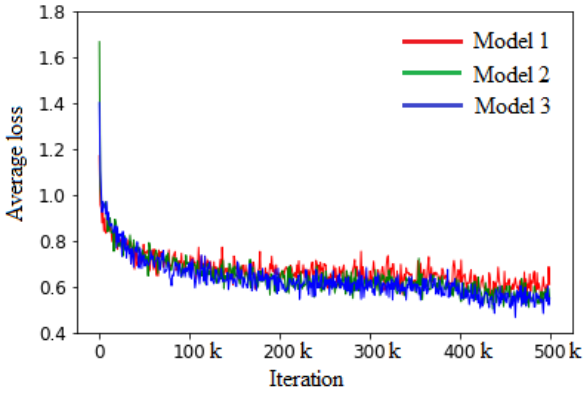


Fig. 6. Comparison of the average loss values of the three proposed models

Residual blocks and MSPP module makes the network model size bigger and requires more floating pointoperations, this design improves the accuracy of the human detector. Indeed, the AP of Model 3 is the highest, reaching 54.2 % and being higher than that of Model 1 and 2 i.e. 5.4 % and 0.9 %, respectively.

Considering how the proposed model works when the size of the image input is changed, we implement the Model 3 training with various input sizes since Model 3 is the best model among three Model 1, 2 and 3. Table IV reports the training results of Model 3 where the minimum input size was 224 x 224 and the maximum input size was 352 x 352. As shown in Table IV, the larger the image input size is, the higher that the AP of Model 3 is. However, increasing the image input size also enlarges the number of floating point operations.

B. *Runtime performance on edge devices*

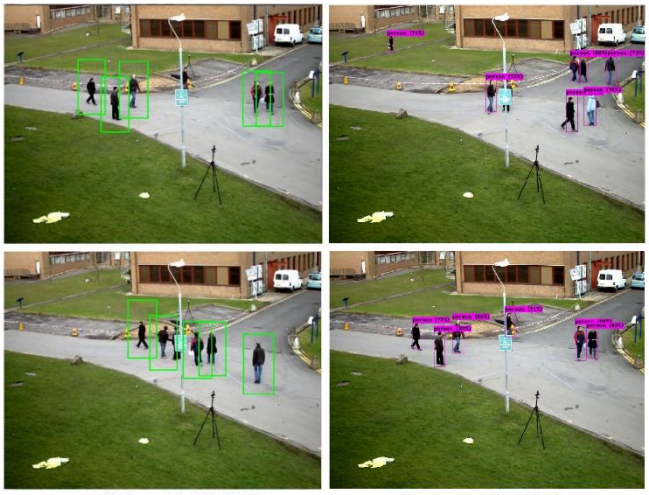
In this experiment, we evaluate our proposed approach on a resource limited edge device. Raspberry PI 3B board with an ARM v7 1.2 GHz and 1 GB RAM was chosen to implement this experiment. This is an embedded board running a full operating system and equipped with sufficient peripherals to start execution without the addition of hardware [32]. Moreover, it supports various high-level programming languages (C, C++, Python, Java, Scratch, Ruby, etc.) and Linux OS variants. Therefore, this device is a good option for edge computing. The runtime performance of Model 3 with different image input sizes is displayed in Table V. One can see from this table, the Model 3 – 224 x 224 achieves the lowest processing time of 0.22s whereas the Model 3 – 352 x 352 spends the highest processing time of 0.62s. Our goal is

TABLE III. TESTING RESULTS ON THE HUMAN COCO DATASET

Model	***AP***	***Model Size***	***BFLOPs***
Model 1	48.8 %	7.4 MB	0.596
Model 2	53.3 %	9.5 MB	0.774
Model 3	54.2 %	11.1 MB	0.935

TABLE IV. TRAINING RESULTS OF MODEL 3 WITH DIFFERENT INPUT SIZES

Model	***AP***	***BFLOPs***
Model 3 (224 x 224)	54.41 %	0.935
Model 3 (256 x 256)	57.13 %	1.222
Model 3 (288 x 288)	59.08 %	1.546
Model 3 (320 x 320)	61.18 %	1.909
Model 3 (352 x 352)	62.71 %	2.31



SSD based L-CNN [33] Our proposed method

Fig. 7. Detection results

to develop a detector so that it can process two frames per second in detecting human. From the table IV and V, the trained model with the image input size of 320 x 320 is our best trained model.

C. *Comparison with Tiny-YOLO variants*

The Tiny-YOLO variants include Tiny-YOLOv2 and Tiny-YOLOv3. To compare the performance of these methods with our propose method, we firstly modified the last convolution layer in the network of the Tiny-YOLO variants to enable them to detect a single object, then we used the same training settings of our proposed model to train these models. Table VI reports the accuracy, inference and model size of three different detection methods on the Raspberry PI 3B and three datasets including the human COCO, INRIA [30] and PENN FUDAN [31] datasets. It can be seen that our best model achieved a better performance than Tiny-YOLOv2 and Tiny-YOLOv3 in terms of the AP and FPS. The number of frames per second which our method can process is more than approximately double the other methods. Moreover, our proposed detector is of a smaller size than the Tiny-YOLO variants. This is useful for saving memory usage when running the human detection task on resource limited platforms.

D. *Comparison with SSD based L-CNN*

This experiment is implemented to compare our best human detector with the SSD based L-CNN human detector. To do this, we performed the human detection task on the same sample surveillance video in the study of the SSD based L-CNN method [32]. Fig.7 shows the results of both methods in processing this video. False Negative Rate (FNR), False Positive Rate (FPR) and Frame Per Second (FPS) are major metrics which were used for comparison purposes. Moreover, we captured the memory usage of our proposed model when

TABLE V. RUNTIME PERFORMANCE OF MODEL 3 ON RASPBERRY PI 3B

Model 3	***Raspberry PI 3B***
224 x 224	0.27 s
256 x 256	0.34 s
288 x 288	0.42 s
320 x 320	**0.51 s**
352 x 352	0.62 s

TABLE VI. COMPARISON ON THREE HUMAN TEST DATASETS

Model	*AP*			*FPS*	*Size*
	Human COCO	*INRIA*	*PENN FUDAN*		
Our best model	**61.18 %**	**95.05 %**	**96.81 %**	**1.96**	**11.1 MB**
Tiny – YOLOv2	44.97 %	89.08 %	91.84 %	1.12	44.1 MB
Tiny – YOLOv3	53.65 %	92.02 %	91.89 %	0.95	34.7 MB

TABLE VII. COMPARISON WITH THE SSD BASED L-CNN METHOD

Methods	*FNR*	*FPR*	*FPS*	*Memory Usage*
Our approach	**13.5 %**	**2.3 %**	**1.96**	**136.4 MB**
SSD based L-CNN [36]	18.1 %	6.6 %	1.79	122.5 MB

running the detector on the Raspberry PI 3B in order to compare it with that of the SSD based L-CNN. Table VII shows that both methods can process average two frames per second. Even though the memory usage of our method is slightly larger than that of the SSD based L-CNN, our solution generally performs the human detection task better than the other method. Indeed, the FNR and FPR of our method are 4.6 % and 4.3 % smaller than those of the SSD based L-CNN, respectively.

V. CONCLUSIONS

In this paper, we introduced an approach based on YOLOv2 for human detection. The proposed method is a combination of a modified YOLOv2, Residual blocks and multiple Spatial Pyramid Pooling blocks. The experimental results show that the network model employing Residual blocks and multiple SPP modules can fit the human detection task better than the model which doesn't use them. The proposed model can detect humans with high accuracy of 95.05 % and 96.81 % while testing on the INRIA and PENN-FUDAN datasets, respectively. Compare to the other methods, the proposed model outperforms the Tiny-YOLO variants and the SSD based L-CNN method.

ACKNOWLEDGMENT

This research is funded by Ministry of Science and Technology of Vietnam (MOST) under grant number 10/2018/DTCTKC.01.14/16-20.

REFERENCES

[1] H. H. Nguyen, T. N. Ta, "Turnstile jumping detection in real-time video surveillance", Image and Video Technology, PSIVT 2019, LNCS, vol 11854, C. Lee, Z. Su, A. Sugimoto, Eds. Springer, Cham.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li and L. Xu, "Edge computing: Vision and challenges", IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, 2016.

[3] M. Dukhan, "NNPACK: Acceleration package for neural networks on multi-core CPUs", 2016. [Online]. Available: https://github.com/Maratyszcza/NNPACK

[4] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B. Y. Choi, T. R. Faughnan, "Real-time human detection as an edge service enabled by a lightweight CNN", IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2018.

[5] J. Redmon, A. Farhadi, "YOLO: Real-Time object detection", 2016. [Online]. Available: https://pjreddie.com/darknet/yolo

[6] A. F. Khalifa, E. Badr, H. N. Elmahdy, "A survey on human detection surveillance systems for Raspberry Pi", in Image and Vision Computing, vol. 85, pp. 1 – 13, 2019.

[7] N. Dalal, B. Triggs, "Histograms of orirented gradients for human detection", IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), San Diego, CA, USA, 2015.

[8] H. Bay, T. Tuytelaars, L. V. Gool, "SURF: Speed Up Robust Feature", ECCV 2006, LNCS, vol. 3951, pp. 404-417, 2006.

[9] A. Oliver, X. Llad, J. Freixenet, and J. Mart, "False Positive Reductionin Mammographic Mass Detection Using Local Binary Patterns,"in MICCAI, Springer, Berlin, Heidelberg, pp. 286–293, 2007

[10] K. Bhuvaneswari, H. A. Rauf, "Edgelet based human detection and tracking by combined segmentation and soft decision", International Conference on Control, Automation, Communication and Energy Conservation, 2009.

[11] H. V. Dung, K. H. Jo, A. Vavilin, "Fast human detection based on parallelogram Haar-like features", 38th Annual Conference on IEEE Industrial Electronics Society (IECON), 2012.

[12] P. Sabzmeydani, G. Mori, "Detecting Pedestrians by Learning Shapelet Features", The IEEE Conference on Computer Vision and Pattern Recognition, June 17-22, 2007

[13] H. L. Eng, J. Wang, A. H. Kam, W. Y. Yau, "A Bayesian framework for robust human detection and occlusion handling human shape model", Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004.

[14] T. Adiono, K. S. Parkoso, C. D. Putratama, "HOG-AdaBoost Implementation for Human Detection Employing FPGA ALTERA DE2-115", International Journal of Advanced Computer Science and Applications,vol. 9, no. 10, 2018

[15] R. Girshick, J. Donahue, T. Darrell, J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation", IEEE Conference on Computer Vision and Pattern Recognition, 2014

[16] R. Girshick, "Fast R-CNN", IEEE Conference on Computer Vision and Pattern Recognition, 2015

[17] K. He, G. Gkioxari, P. Dollar, R. Girshick, "Mask R-CNN", IEEE Conference on Computer Vision and Pattern Recognition, 2018

[18] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", IEEE Conference on Computer Vision and Pattern Recognition, June 27-30, 2016.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, "SSD: Single shot multibox detector", IEEE Computer Vision and Pattern Recognition, (CVPR), 2015

[20] J. Redmon, A. Farhadi, "YOLO9000: Better, Faster, Stronger", IEEE Conf on Computer Vision and Pattern Recognition, July 21-26, 2017.

[21] J. Redmon, A. Farhadi, "YOLOv3: An Incremental Improvement", ArXiv, 2018.

[22] Sabyasachi Sahoo, "Residual blocks – Building blocks of ResNet", 2018. [Online]. Available: https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec

[23] Raghunandepu, "Understanding and implementation of Residual Networks (Resnets), 2018. [Online]. Available: https://medium.com/analytics-vidhya/understanding-and-implementation-of-residual-networks-resnets-b80f9a507b9c

[24] C. Y. Wang, H. Y. Mark, P. Y. Chen, J. W. Hsieh, "Enriching Variety of Layer-wise Learning Information by Gradient Combination", ICCV Workshop on Low-Power Computer Vision, 2019

[25] K. He, X. Zhang, S. Ren, J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition", IEEE Trans on Pattern Analysis and Machine Intelligence, vol. 37, no. 9, pp 1904 – 1916, 2015.

[26] P. Zhang, Y.Zhong, X. Li, "SlimYOLOv3: Narrower, Faster and Better for Real-Time UAV Applications", ArXiv, vol. abs/1907/11093, 2019

[27] "COCO Dataset," http://cocodataset.org, accessed: 2019-07-23.

[28] Darknet. Available: https://pjreddie.com/darknet/

[29] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, A. Zisserman, "The PASCAL Visual Object Classes (VOC) Challenge", International Journal of Computer Vision, vol 88, pp. 303-338, 2010.

[30] INRIA dataset. Available: http://pascal.inrialpes.fr/data/human/

[31] PENN-FUDAN dataset. Available: https://www.cis.upenn.edu/~jshi/

[32] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B. Y. Choi, T. R. Faughnan, "Smart surveillance as an Edge network service: from Haar-Cascade, SVM to a Lightweight CNN", IEEE 4th International Conference on Collaboration and Internet Computing (CIC), 2018.