Hue University Journal of Science: Techniques and Technology; pISSN 2588-1175 | eISSN 2615-9732

Vol. 133, No. 2B, 2024, 37–48; DOI: 10.26459/hueunijtt.v133i2B.7495

Crowd Counting using Deep Learning Model on FPGA Card

Thi Thu Thao KHONG[1]*, Van Loc TRAN[2], Hai-Phong PHAN[1], Duc-Hung DUONG[3]

[1] Faculty of Electrics, Electronics and Material Technology, Hue University of Sciences, Hue University, 77 Nguyen Hue, Hue, Vietnam

[2] Brycen Viet Nam Co., LTD, 25 Nguyen Van Cu, Hue, Vietnam

[3] Hue University, 3 Le Loi, Hue, Vietnam

Abstract. Machine learning and deep learning are becoming important tools for processing video in artificial intelligence applications, especially real-time tasks that require speed, accuracy, and flexibility. For this reason, we introduce a crowd counting and detecting system from RTSP video streams using a deep learning model. Our system uses FPGA cards, i.e. Xilinx Alveo U30 and U200, to accelerate the transmission of video streams and the deep learning inference. In the input and output stream, Vitis Video Analysis SDK GStreamer is utilized to leverage the features of Alveo U30 for streaming RTSP videos. In the deep learning inference, we apply the pre-trained YOLOX model to detect "person" objects with bounding boxes on video frames. We build a counting variable to count the number of bounding boxes of "person" objects displayed on each frame. The pre-trained YOLOX is accelerated by Alveo U200 based on the Mipsology Zebra framework. The proposed system not only processes multiple streams but also achieves faster inference and lower CPU usage than the system that just uses CPU for deep learning inference.

Keywords: Deep learning, YOLO, Crowd counting, Video processing, FPGA card

1 Introduction

With the exponential growth of available data, artificial intelligence (AI) applications have been developed faster and more efficiently. Object detection is a foundation problem of AI field, which has many viral algorithms. At present, there are a lot of deep learning models for the object detection issue. One of them is YOLO [1]: you only look once at an image to predict. It is a convolutional neural network (CNN) [2] to predict multiple bounding boxes and class probabilities, as seen in Fig. 1. In the past years, YOLO models had several versions. They are being improved and focused on the main features of real-time applications with the speed and accuracy trade-off [3]. In 2021, Ge et al. [4] proposed YOLOX which is a new high-performance detector in the YOLO family. YOLOX has been boosted by YOLOv3 [5] which is one of the most widely used detectors in the industry due to its compatibility. Therefore, we utilize YOLOX for the human detection problem from video frames.

* *Corresponding: kttthao@hueuni.edu.vn*

Received: 03–05–2024; Accepted: 11–06–2024

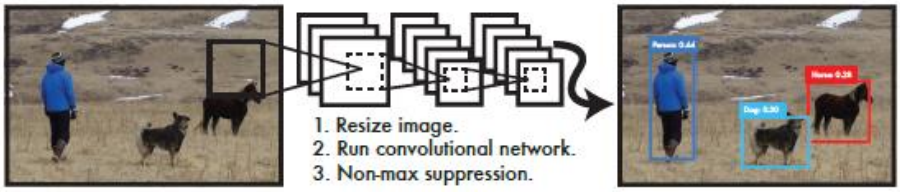

Fig. 1. A YOLO system detects the objects on an image with bounding boxes [1]

Crowd counting is one of the main areas of crowd analysis from videos and images, which is applied in traffic monitoring, congestion, public safety, urbanization, etc [6]. Detecting and counting people is an important task of object detection, hence many studies have been conducted in recent years, especially real-time object detection. YOLO models are frequently used for this application [6-8] with an accuracy of up to 95%.

Deep learning (DL) algorithms are usually implemented on powerful computation devices, like CPUs and GPUs [9,10]. CNN models require numerous computations of convolutions and matrix multiplications, and external memory accesses with their parallel nature, which are able to be conducted on GPUs easily. Nonetheless, GPUs become inefficient in optimization, such as performance and energy consumption. Therefore, Field-Programmable Gate Arrays (FPGAs) have been used to accelerate the deep learning models [11]. FPGAs are known as the high throughput and power efficiency hardware accelerators of YOLO for real-time object detection [12].

In this article, we apply the video-processing system [13] to transfer and analyze the video stream, particularly detecting and counting people using YOLOX embedded into FPGA cards, i.e. Alveo U30 and Alveo U200. We can summarize this work as follows:

- The architecture of a video-processing system is introduced, which includes cameras, an input stream block, deep learning processing, and a composite & output stream.
- Real-time streaming protocol (RTSP) and GStreamer are used to control the delivery of input and output video streams.
- Alveo U30 and U200 cards are utilized for video streaming transcoding and DL inference, respectively. Moreover, Mipsology Zebra is applied to accelerate YOLOX on Alveo U200, which implements crowd counting and detection.

The rest of this paper is ordered as follows. Section 2 introduces related work in YOLOX, crowd counting, RTSP, GStreamer, and Alveo cards. Section 3 shows the architecture of a video-processing system for detecting and counting people using YOLOX on Alveo cards. Experimental results are represented in Section 4. The conclusion is in Section 5.

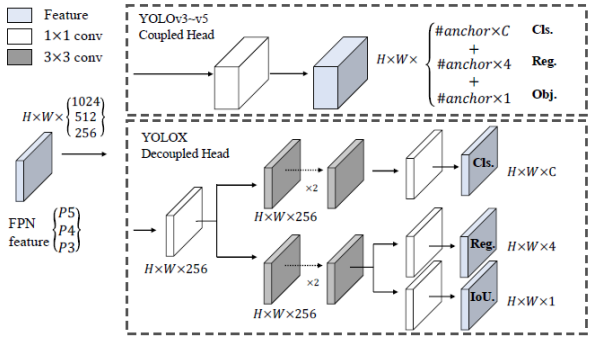

Fig. 2. The improvement of YOLOX is compared to YOLOv3~v5 [4]. The decoupled head converges much faster and achieves better performance than the coupled head in the same model

2 Related work

2.1 YOLOX and crowd counting

YOLO series have been developed to improve the speed and accuracy trade-off for real-time object detection applications [3]. A basic YOLOX is based on YOLOv3-DarkNet53 [5] with the anchor-free technique, the decoupled head, and the leading label assignment strategy SimOTA, as shown in Fig. 2. These are advanced detection techniques that help YOLOX overcome other models, such as YOLOv3, YOLOv4, YOLOv5. We use YOLOX-DarkNet53 which has been trained on the COCO dataset [14] 640×640 resolution with 47.4% AP, for the inference on input streams to detect and count people.

Crowd counting from videos and images is an essential task of object detection. It is popularly applied in video surveillance systems to control the maximum number of people in many places. There have been a lot of studies in recent years, especially during the COVID-19 pandemic, in which YOLO models have been used to detect and count people and measure the distance between them [15].

2.2 RTSP and GStreamer

RTSP [16] is an application-level protocol to control the on-demand delivery of real-time data, e.g. audio or video. This protocol is designed to provide the rules for transmitting video over the Internet, and also choose delivery channels such as UDP and TCP. The data sources of RTSP are both live data and stored videos.

GStreamer [17] is a flexible, fast, and multiplatform multimedia framework for creating streaming media applications. It connects processing elements into a pipeline and uses a plug-in architecture which is registered and loaded as shared libraries. This paper utilizes Vitis Video Analytics SDK (VVAS) GStreamer plug-ins which are provided by Xilinx Video SDK [18] to leverage the hardware-accelerated features of Alveo U30 cards for streaming video services.

2.3 Alveo U30 and U200 cards

The Xilinx Alveo U30 [19] is a media accelerator card with the lowest cost per channel and the lowest power consumption for live video streams. It is intended for multi-stream transcoding video applications and is capable of decoding, scaling, and encoding up to eight 1080p60 streams. Therefore, Alveo U30 is used to process video streams based on VVAS GStreamer plug-ins in our experiment.

Fig. 3. Xilinx Alveo U30 and U200 cards [19, 20]

In the DL inference, we conduct the people detection by YOLOX that is embedded into the Xilinx Alveo U200 [20] based on the Mipsology Zebra framework [21]. Alveo U200 provides up to 90X performance compared to CPUs for machine learning inference, video transcoding, and database analytics. Moreover, Zebra is developed to compute CNN models on FPGAs faster with lower costs and lower power consumption than CPUs/GPUs.

3 System architecture

The architecture of our video-processing system is shown in Fig. 4. The system includes an input stream block, DL processing (DL Overlay and DL Server), a composite, and an output stream block.

3.1 Input stream

We can use live cameras or prepared video clips that are transcoded into a suitable format, e.g. transport stream (TS) file, for RTSP streaming. All RTSP streams with a resolution of 1080 pixels and 15 frames per second (fps) are aggregated to the RTSP source. Data is then moved to a decoder for compressed decoding and is resized properly by a scaler. These video processes are implemented on Alveo U30 via a configured file that provides the parameters of VVAS GStreamer plug-ins. After decoding and scaling, GStreamer data is processed in a pipeline by an app sink and prepared for the next steps.

3.2 DL processing

DL processing includes two blocks: (1) DL Overlay plays the role of a client that receives data from the input stream, formats the proper data, and carries out a process requirement sent to DL Server; (2) DL Server gets the request of DL Overlay, performs the DL inference for object

detection, and then returns the result to DL Overlay. A two-way communication between Client and Server is handled by Socket which allows two programs to run on the network.

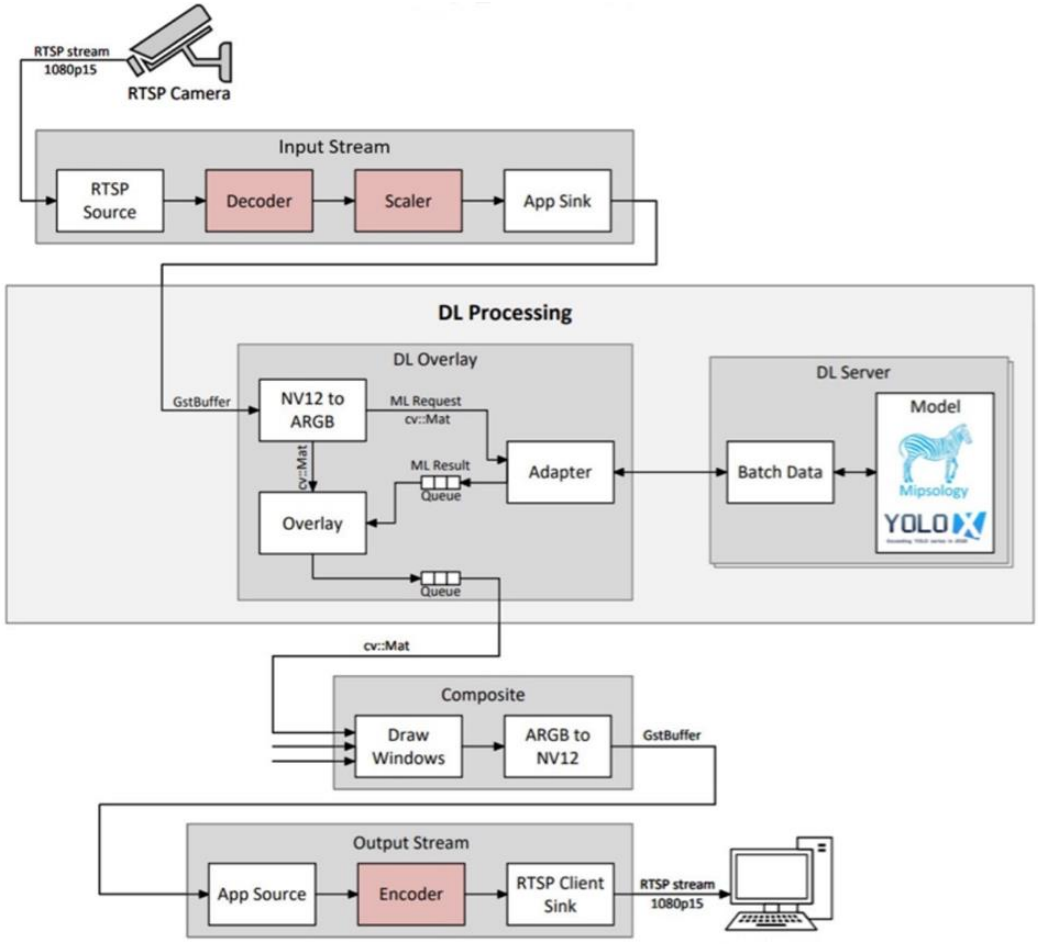

Fig. 4. The architecture of proposed video processing system for crowd counting based on the video analysis system [13]

In the first step of DL Overlay, data is converted to a suitable color format from NV12 of video cameras to ARGB standard for DL processing. Next, Adapter gets the prepared image data for transferring to Server. Also, it receives the results from Server including parameters of bounding boxes and the number of people counting on the image that are detected by YOLOX. These results are sent to Overlay to combine with the preprocessed image for displayed aggregation.

When DL Server accepts the connected request of DL Overlay, it will wait to receive frames (images) of each video on each stream. These images are added to Batch Data until a video of a stream is full, then they are moved to YOLOX. The trained YOLOX model detects people on images, and returns parameters of bounding boxes and a people counting variable. YOLOX

implements the inference on Alveo U200 card based on the Zebra framework for acceleration. After the inference, Server returns processed data to Client with the same stream of each video. The system only operates the DL inference when DL Overlay connects to DL Server via Socket. We have used Stream Socket with TCP/IP protocol for the connection, as seen in Fig. 5.

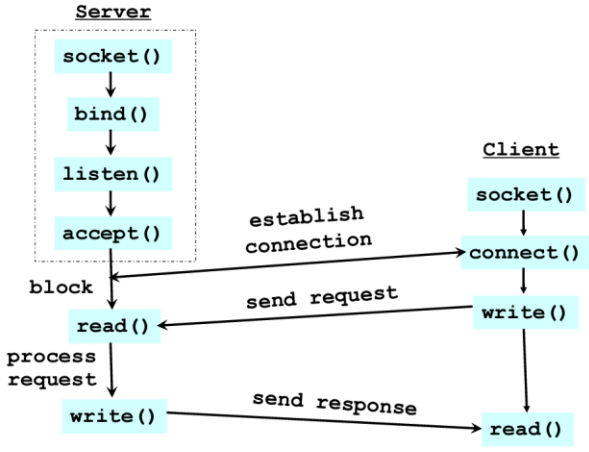

Fig. 5. The connection between Client and Server uses a stream socket with TCP/IP protocol [22]

3.3 Composite and Output stream

The function of composite is the aggregation of video streams and the color format conversion from ARGB to NV12 to return the standard video for the output stream. Based on the queue size of video streams from Overlay, Draw Windows divides the displayed screen into different windows, in which each window corresponds to a video stream of an input camera. After that, video data is converted to NV12 color format which is efficient for compression and transmission.

Thereafter, data is composited into one stream to move to App Source of Output Stream. App Source is seen as a data buffer prepared for the encoder of VVAS GStreamer. The function of Encoder is a compressed encoding to optimize data, increase video quality, and reduce bandwidth for transmission. After encoding, video data is stored at RTSP Client Sink. Based on IP address of RTSP Server, RTSP Client transmits video streams with a resolution of 1080p15 to RTSP Server to display the video with bounding boxes and the number of people on frames. Output Stream is implemented by GStreamer pipeline on Alveo U30.

4 Experiment and Result

4.1 Experiment Setup

We carry out the experiment on Intel Core i5-8600K CPU 4.3Ghz with Ubuntu 20.04.6 LTS x86_64 OS. Our experiment is accelerated by the Alveo U30 card (Xilinx Video SDK) for video transmission and the Alveo U200 card (Vitis AI, Zebra) for the DL inference.

In the inference phase, we use the pre-trained YOLOX on the COCO dataset which comprises 80 object categories, such as person, bicycle, car, cat, and dog, … for the object detection task. In crowd counting, YOLOX only focuses on detecting the person category with exact parameters of bounding boxes and draws bounding boxes of "person" objects. After that, it returns parameters of bounding boxes of "person" objects that are used for a counting variable to count the number of people on video frames.

In the experiment, we utilize stored videos and transcode them into .ts format for RTSP streaming. Hence, its property is as RTSP cameras in the input.

4.2 Experiment Results

We use VLC Media Player to open the output RTSP stream with IP address of RTSP server, as shown in Fig. 6. The displayed interface of the output video on VLC includes camera identity (ID), bounding boxes of person objects, and the number of detected people on transferred video frames, as seen in Fig. 7.

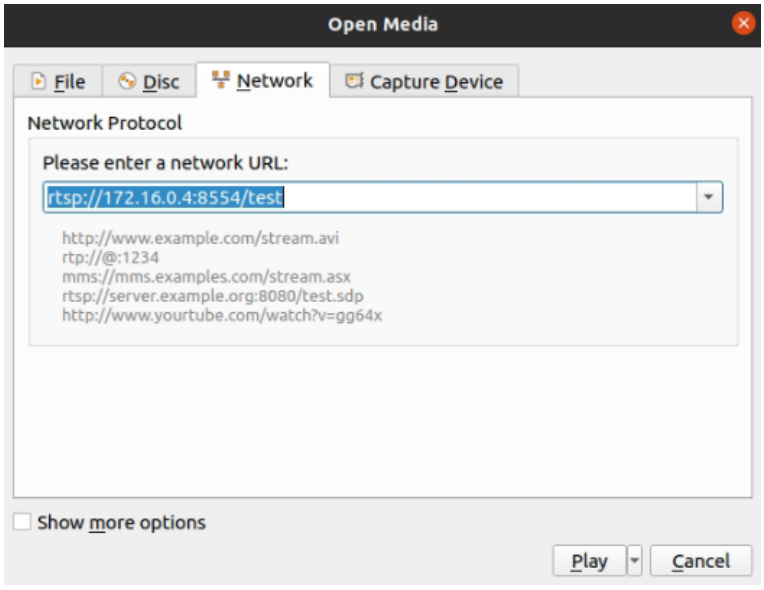

Fig. 6. Using VLC Media Player to open RTSP stream with IP address of RTSP server

Due to the limitation of devices, we have just conducted 4 input video streams and got 4 output streams. The result is illustrated in Fig. 8. The pre-trained YOLOX can recognize the person with parameters of bounding boxes and draw bounding boxes exactly to objects on clear video frames. However, the system is still limited by live videos with noise. Moreover, because YOLOX trained on the normal COCO dataset, detecting small objects has still been hard for our system.

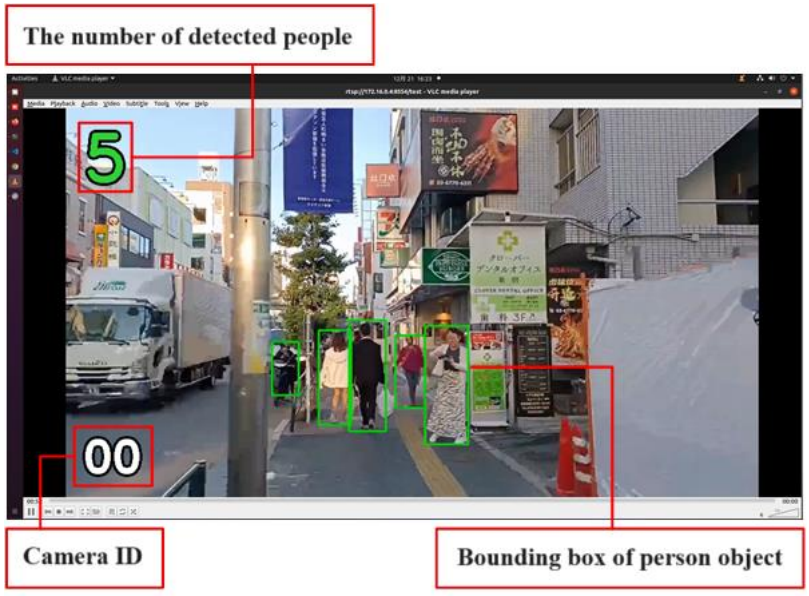

Fig. 7. An output video and detected results are displayed by VLC Media Player

Fig. 8. The output result of 4 video streams

For real-time DL applications, the inference time and CPU performance are needed to optimize and maintain the system's stability. In this experiment, we compare the inference time and CPU performance between CPU and Alveo U200 for DL inference. We measure the inference time of 100 frames of 1 video stream and 100 frames of 4 streams, as shown in Fig. 9 and Fig. 10 respectively. The inference time of Alveo U200 is always less than CPU, which is about 60% of 1 video and 45% of 4 videos on average, as seen in Fig. 11. When the system handles 4 video streams, the inference time of Alveo U200 is more stable than CPU. Hence, our system is compatible with the process of multiple streams.

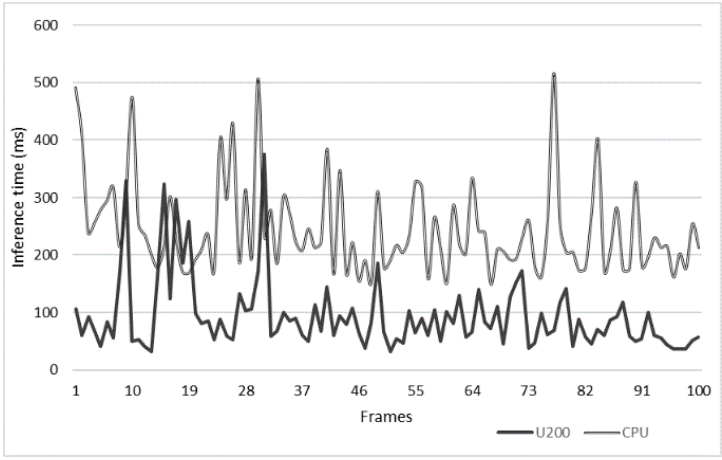

Fig. 9. The inference time of 100 frames of 1 video stream on Alveo U200 and CPU

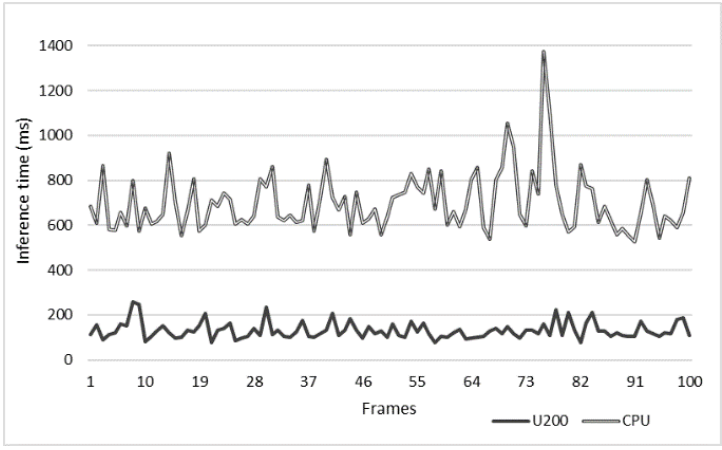

Fig. 10. The inference time of 100 frames of 4 video streams on Alveo U200 and CPU

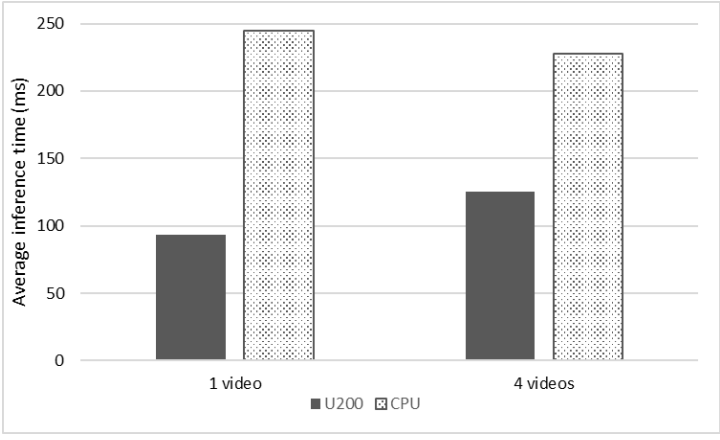

Fig. 11. The average inference time of 100 frames of 1 video stream and 4 streams on Alveo U200 and CPU

Besides the measurement of the inference time, we also achieve the average CPU performance of the system on 100 frames of 1 video stream and 4 video streams. Table 1 indicates that using CPU for the DL inference consumes more CPU resources than using Alveo U200. With 1 video, the average CPU performance for the inference on Alveo U200 declines 11.82% compared to the inference on CPU. This decrease is 7.07% on average when the system processes 4 video streams.

Table 1. The average CPU performance is measured on 100 frames of 1 video and 4 videos when the DL inference block is CPU and Alveo U200

Average CPU performance (%)	Alveo U200	CPU
1 video stream	65.56	77.38
4 video streams	80.92	87.99

Based on the demo of a video processing system [13], we have successfully built the system that utilizes YOLOX on Xilinx Alveo U200 to detect and count the crowd. Compared to the demo [13], which used YOLOv3 on Xilinx Alveo U50, our system has applied the improved deep learning model – YOLOX. We also make the comparison of latency between Alveo U200 and U50 when they conduct the inference on Pascal VOC dataset by YOLOv3. Table 2 shows that Alveo U200 is faster than Alveo U50 in deep learning inference.

Table 2. The comparison of latency between Alveo U200 and Alveo U50 when they are implemented by YOLOv3

	Alveo U200	Alveo U50
Latency (ms)	19.90	122.24

5 Conclusion

In this paper, we have built the crowd counting and detecting system from RTSP video streams. Our system uses YOLOX model for person detection, which is integrated into the Mipsology Zebra framework on Xilinx Alveo U200. We also utilize Xilinx Alveo U30 for the transmission of RTSP video streams in the input and output. Using Alveo cards has accelerated the deep learning inference that can be suitable for real-time deep learning applications. In particular, the inference time of the proposed system is less about 60% than the inference time of the system using CPU, which is calculated on average of 100 frames of 1 video. Moreover, our system also saves CPU resources for inference when it decreases 11.82% of average CPU performance with the process of 1 video. This system can handle multiple streams and it is thus expected to accelerate the transfer of live videos and real-time deep learning inference. The system will also be deployed to process special live videos, such as noise, small objects, etc. when it is used for real applications in the future.

Acknowledgment

This research was supported by Brycen Viet Nam Co., LTD, 25 Nguyen Van Cu, Hue, Vietnam.

References

1. Redmon J, Divvala S, Girshick R, and Farhadi A. You Only Look Once: Unified, Real-time Object Detection. Proceedings of the IEEE conference on computer vision and pattern recognition. 2016; 779-788.
2. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, and Rabinovich A. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition. 2015; 1-9.
3. Jiang P, Ergu D, Liu F, Cai Y, and Ma B. A Review of Yolo Algorithm Developments. The 8th international conference on information technology and quantitative management. 2022; 1066-1073.
4. Ge Z, Liu S, Wang F, Li Z, and Sun J. YOLOX: Exceeding YOLO Series in 2021. arXiv preprint arXiv:2107.08430 (2021).
5. Redmon J, Farhadi A. YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018).
6. Gündüz MS, Isik G. A new YOLO-based method for real-time crowd detection from video and performance analysis of YOLO models. Journal of Real-time Image Processing. 2023; 20(1), 5.
7. Ren P, Fang W, Djahel S. A novel YOLO-based real-time people counting approach. In 2017 International Smart Cities Conference (ISCC). IEEE 2017; 1-2.
8. Ahmad M, Ahmed I, Adnan A. Overhead view person detection using YOLO. In 2019 IEEE 10th annual Ubiquitous Computing, Electronics & Mobile Communication Conference. 2019; 0627-0633.
9. Khong TTT, Nakada T, Nakashima Y. A Hybrid Bayesian-Convolutional Neural Network for Adversarial Robustness. IEICE Transactions on Information and Systems. 2022; 105(7):1308-1319.
10. Khong TTT, Nakada T, Nakashima Y. Flexible Bayesian Inference by Weight Transfer for Robust Deep Neural Networks. IEICE Transactions on Information and Systems. 2021; 104(11):1981-1991.
11. Kljucaric L, George AD. Deep-learning inferencing with high-performance hardware accelerators. In 2019 IEEE High Performance Extreme Computing Conference (HPEC). 2019; 1-7. IEEE.
12. Nguyen DT, Nguyen TN, Kim H, and Lee HJ. A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection. IEEE Transactions on Very Large Scale Integration (VLSI) Systems. 2019; 27(8):1861-1873.
13. Online. https://github.com/anjn/alveo-video-analytics-demo/?tab=readme-ov-file
14. https://cocodataset.org/#home
15. Punn NS, Sonbhadra SK, Agarwal S, Rai G. Monitoring COVID-19 social distancing with person detection and tracking via fine-tuned YOLOv3 and Deepsort techniques. arXiv:2005.01385v4. 2021
16. RFC 2326. Real Time Streaming Protocol (RTSP). IETF. 1998.
17. https://gstreamer.freedesktop.org/

18. https://xilinx.github.io/video-sdk/v2.0/index.html
19. https://www.xilinx.com/products/boards-and-kits/alveo/u30.html
20. https://www.xilinx.com/products/boards-and-kits/alveo/u200.html
21. Mipsology. Zebra by Mipsology: Accelerating Neural Network Inference. Solution Brief.
22. Habibi P. Tutorial on Socket Programming. Computer Networks, Department of Computer Science, University of Toronto.