An Edge-AI Heterogeneous Solution for Real-time Parking Occupancy Detection

Tran Ngoc Thinh[1,2], Long Tan Le[1,2], Nguyen Hoang Long[1,2],
Ho Le Thuc Quyen[1,2], Ngo Qui Thu[1,2], Nguyen La Thong[1,2], Huynh Phuc Nghi[1,2]
[1]*Ho Chi Minh City University of Technology (HCMUT)*
[2]*Vietnam National University - Ho Chi Minh City (VNU-HCM)*
Ho Chi Minh City, Vietnam
{tnthinh,1870385,long.nguyen29798,quyen.holethuc,1652595,thong.nguyenla, nghihp}@hcmut.edu.vn

***Abstract*—In the digital era, building smart cities is a highly desired goal that every country strives to achieve. One of the most difficult and exciting aspect of Smart City that has been in constant development is Smart Parking. In this paper, we are putting in our best effort into studying and implementing a solution that we believe can help bring AI image recognition capability to aid in Smart Parking as well as an accompanying server and application set.**

***Index Terms*—Smart Parking, Edge Computing, BNN**

I. Introduction

Over the past decade, interest in developing Smart Cities around the world has increased dramatically. Smart parking system is one of the most critical components in Smart City architecture, promising to improve quality of life significantly by improving transportation and accessibility. One of the main reasons for causing daily urban traffic congestion is drivers trying to find a place to park their vehicles - accounting for 30% of congestion [1]. Moreover, searching for parking space is a routine activity for many people in cities around the world. This search burns through about one million barrels of the world's oil supply every day. As the global population continues to urbanize, without a well-planned, convenient solution to help managing parking slot, these problems will keep getting worse.

An effective solution for smart parking can be implemented by many kind of new technologies. With the growth of sensor technologies, studies [2] [3] [4] described parking lots that were installed with sensors to determine whether the slot is occupied or not. This approach has a high cost due to the need of installing multiple devices, especially with large or existing parking lots without the suitable infrastructure. Meanwhile, paper [24] introduced a system using IoT technology and TensorFlow, conbining with Android device for their smart parking system. However, the system were still testing on dataset and using sensor system, which is reason why it is still expensive for checking parking availability. Therefore, a camera-based solution is recently proposed to reduce the number of installed devices and deployment cost. However, detecting the parking lot's status through visual information requires a large amount of computational processing to reach optimal accuracy and efficiency.

Collecting all raw data from sensors can lead to delays in system's responsiveness and network performance so Edge Computing has been applied to extract the necessary raw data for further processing. Instead of processing all data at the central processing unit, Edge Computing system processes data at the node where the data was generated. This method focuses on processing the data as close to the source as possible. Edge AI (Artificial Intelligence) is the combination of Edge Computing and AI with the aim to accelerate the speed of processing data at the edge nodes. Instead of sending all the data to the server, local devices only need to send the result of the computed data. Edge Computing/ Edge AI is suitable for systems that do not depend much on internet connection speed and require immediate data processing (image and video analysis) such as surveillance cameras or self-driving vehicles.

Moving DNN compute to the edge devices requires a lot of optimization for the power budgets and the the performance levels available at the edge. The communication from the edge to the cloud has many issues as well that come in to play when trying to keep devices size small. Research has shown that through quantization and pruning the neural network, the computational intensity can be lowered drastically.

Many methods and implementations of identifying empty spaces in parking lots have been introduced over the years. In [5], a system uses edge computing, cameras and AI to achieve this. A set of cameras and Deep Learning techniques used to detect parking space status and edge computing devices were installed and worked together as a data analysis and managing system. Another solution was proposed in [9], where a system used cameras and lights to control parking. The light system toggles based on the driving direction of the cars, which are detected by the Deep Learning Algorithm. The controller and algorithm was implemented inside an SoC board. Tests were run on a dataset and the system returned positive results.

In this paper, we propose a solution for heterogeneous System-on-Chip platforms fast inference accelerator, scalable real-time interpretation of image sequences which helps to automate the detection of parking spaces.

The rest of this paper is organized as follow. Section 2 outlines the background knowledge and related works. Section 3 presents our proposed methodology. The experiment results will be discussed in Section 4., and the conclusion of our

research work is delivered in Section 5.

II. Background and Related Work

In this section, we will first describe some basic concepts , and then outline some other works that is related to our research works.

A. Edge Computing

In order to optimize the latency and scalability of Smart Parking systems, machine learning algorithms are implemented in edge devices. Support vector machine (SVM), deep learning and neural networks (NNs) are the most deployed algorithms corresponding to different applications. Bringing those techniques to devices requires the need to meet the model design and compression which are adapted to the hardware.

Many hardware platforms have been developed and brought to the market by big companies in the industry such as Intel, Xilinx, NVIDIA, etc. to ensure that edge computing operates AI in an efficient way and meets system's requirements. There are many popular types of hardware including central processing unit (CPU), graphics processing unit (GPU) and field-programmable gate array (FPGA) which have been adapted to perform AI on edge devices. For general-purpose applications, Raspberry Pi is used as an edge device. The Raspberry Pi3 was run in [14] to detect and count the number of vehicles via video stream on the street using OpenCV5 (Open Source Computer Vision Library). Although Raspberry Pi is widely chosen in certain AI systems because of its flexibility, it is still not powerful enough for advanced deep learning and AI algorithms. To speed up the calculation and increase the performance, NVIDIA Jetson Nano has been available with GPU acceleration. One of the many famous applications running on Jetson board is the Jetbot robot. This robot, with the help of AI, can find and avoid many obstacles. For example, Jetbot was used to deliver agricultural tools to its owner while avoiding many objects that were determined as obstacles [15]. Apart from using GPUs, Ultra96-v2 - the Xilinx Zynq UltraScale+ MPSoC - can also accelerates machine learning algorithms with FPGA. Taking the advantage of this architecture, in [16], video processing using deep learning for smart IoT systems was implemented on Ultra96-v2.

B. Binary Neural Networks

Binary Neural Networks (BNN) has emerged as a suitable way for edge devices to save the storage and computation power. The main idea behind BNN is to replace floating point operations with bitwise boolean operations, thereby introducing a new approach for training and inferencing neural networks, where weight's and activation's parameters are binarized at training time, and then used to compute the parameter's gradients. Binarization is a 1-bit quantization where data can only have two possible values, namely -1 (0) or +1. There are two popular methods for binarization: deterministic function or stochastic function:

$$x^b = \begin{cases} +1 & \text{if x} \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{1}$$

C. Related Works

Smart parking system has received a lot of study in recent years due to its necessity in the modern world. Convolutional Neural Network (CNN) has been applied to identifying the occupancy status of the parking space by [13]. The authors tested two CNN architectures, namely mAlexNet and mLeNet, both of which have less than five trainable layers and take a 224x224 RGB image as input. To run these deep learning techniques, the solution was to installed the Raspberry Smart Cameras, which is Raspberry Pi 2 model B equipped with the standard Raspberry Pi camera module, and mounted it in an outdoor camera box. The papers above introduced solutions that were only tested on developed countries with suitable infrastructure for smart parking. Some papers described projects that were only tested using the CNRPark dataset.

Bura et al. [6] proposed a fully end-to-end application parking occupancy detection system using top-view camera images utilizing custom AlexNet. The dataset were trained and validated using 150,000 images and in terms of reference time. OpenALPR was used for license plate recognition and Tiny Yolo was used for parking spot classification [6]. The results of [6] showed that there was a dramatic decrease in inference time for the custom network model.

Regarding classification problems, [7] provided a solution that was divided into offline-training and real-time operation, then a SVM classifier was used to extract vehicle and non-vehicle parts from regions of interest. The tests performed on their dataset provided over 91% accuracy in detection. [8] presented a solution that only requires low-cost and low-power requirement using Internet of Things (IoT). The experiment collected data from Melbourne (Australia) and San Francisco (USA) as input. They used three models, Mean square error (MSE), Mean absolute error (MAE) and Coefficient of determination R2, to measure performance of machine learning models using Regression Tree, Neural Network and Support Vector Regression respectively. The evaluation revealed that Regression was best at predicting parking availability based on the dataset. [7] and [8] got results for object detection but they did not focused on implementing a system for parking.

Model of [10] detects edges of objects in parking lot using Canny algorithm and track object by analyzing the changing of pixels in captured images using Blob detection techniques, image subtraction between video frames in grey-scale to find the differences and turn image into black and white then focus on the changing of object.

[11] introduced a system utilizing cameras and ultrasonic sensors, which is also an effective solution. The system consists of several essential parts: the detection of license plate number, the central information processing system, the networking and the mobile application. The system implemented Node-RED and OpenALPR. Cameras are used to capture vehicles' presence as well as their license plates. Ultrasonic

sensors are used to confirm parking activities. Raspberry PI 3 boards were used as Wi-fi Routers and Hubs to connect the IoT devices together. Raspberry PI 4G + GPS Shield was installed on the Raspberry PI to add 4G capability. Edimax wireless nano USB Adapter was attached to enable the Raspberry PI to become a Wi-fi Router with speed up to 150Mpbs according to their paper.

In [12], the paper proposed a system that can predict future congestion due to parking and from that, recommend drivers the best route to avoid congestion. They introduced an LSTM (Long Short-Term Memory) model for multivariate time series forecasting of parking lots. The project used the Keras library's provided tools for building and learning neural networks and TensorFlow for the backend.

Some other researches focused on implementing camera-based solution for parking detection. System [5] consists of a Camera, a Raspberry Pi, a Firebase Database Cloud Computing platform and a user mobile application. The whole process starts by taking real-time images then converting the image color to binary or black and white to more easily identify the object and background. The Raspberry Pi 3B was programmed with the Haar-Cascade and AdaBoost Learning Algorithm, which processed the data and then send it to the FireBase Database Cloud platform.

Paper [9] introduced a video-based smart control system which also utilized a street-light controller. The system detects vehicles using streetlight cameras and then suggests parking space on the side of the road to minimize time needed to find a parking space. A YOLO v3 detection model with MobileNet v2 was implemented on a Jetson TX2 board, yielding over 90% accuracy in any weather condition.

Lastly, we take a look at some related works that aim to accelerate and improve machine learning capability of FPGAs. The authors of paper [17] introduced their result on optimize BNNs (Binary Neural Networks) and 1-bit CNNs - "extremely compressed version of BNNs" - through six aspects, they minimizing the quantization error, improving the loss function, or reducing gradient error. Their first result was MCN, with the end-to-end framework, approach the unbinarized filters. Base on MCN (modulated convolutional networks), they developed CBCN (mirculant binary convolutional networks), RBCN (rectified binary convolutional networks), BONN (Bayesian optimized 1-bit cnns) for improving training process. Paper [18] proposed FINN as a framework for building scalable and fast BNN inference accelerators on FPGAs. According to their numbers, it achieved the fasted reported neural network inference implementation on MNIST, CIFAR-10 and SVHN benchmark dataset. Using their novel parameterizable dataflow architecture and optimizations, the authors presented unprecedented classification rates along with minimal power usge and latency which are paramount for real-time embedded surveilance and monitoring system.

In this paper, we will focus on how to implement a smart parking system on developing countries such as Vietnam. The system should be low-cost, be able to operate in real-time and has good accuracy.

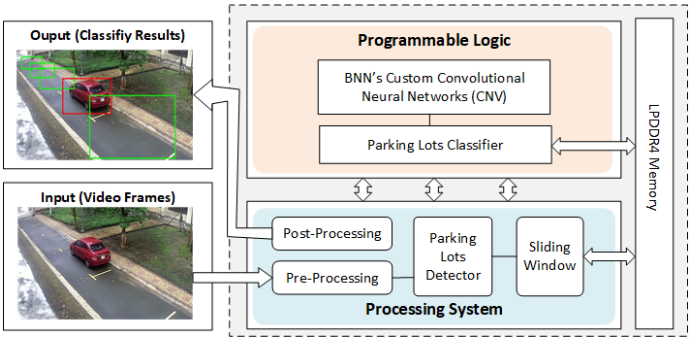

Fig. 1. Parking lot occupancy detection based on a heterogeneous platform.

III. Methodology

In this research, we propose a parking lot occupancy detection solution based on a heterogeneous computing platform. Our solution allows the integration of surveillance camera systems as well as sensor devices to collect and real-time process data for smart parking systems.

The architecture of our solution is depicted in Fig. 1. In this solution, we design an environment for software and hardware co-processing based on two significant components of a heterogeneous platform : the *Field Programmable Gate Array (FPGA) Programmable Logic (PL)* and the *Multi-processor Processing System (PS)*. Where the PL accessing programmable logic overlays to perform the machine learning acceleration providing hardware acceleration for computationally intensive arithmetic, and the PS gets the data from sensors and passes to the memory where the PL can access. Moreover, we also leverage the I/O interfaces peripheral parking components such as surveillance camera systems, parking meters, etc.

The co-processing system is responsible for handling two main tasks: Parking space detection and parking lots occupancy classification. The main objective of the first task is to identify the coordinate of parking spaces, the second task to classify whether parking spaces is empty or not.

We assumed that the camera is fixed at its installed place therefore the parking slot's position will be unchanged. Because of this property, we only need to determine the position once. At first, it requires an image of parking lot captured from the camera placement before starting the detection. Then, the parking spot will be determined and we will save its width, height and coordinates. Furthermore, each spot is also marked by a specific ID number to distinguish with others.

The device will identify the possible parking spots in the input frames collected from cameras, and extract images of spots for classification. Depending on the coordinates of the spots which are determined before, the spots will be cropped out. Since the model we use for classification requires the input of 32x32 image, all the cropped spots is resized to 32x32 images and passed to the PL for further processing.

For the classification task, we employ Binary Neural Networks (BNN). Our model is leveraged BNN's custom con-

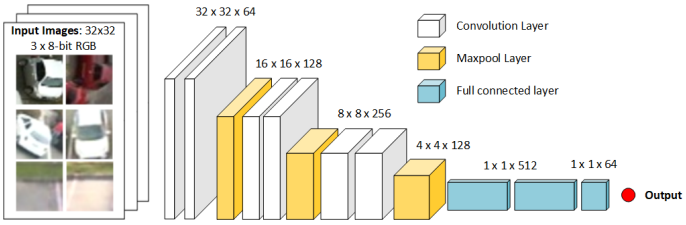

Fig. 2. BNN's custom convolution neural networks (CNV) topology

volution neural networks (CNV). The characterization of the BNN-CNV is described as follows:

- 3x3 convolution
- 2x2 maxpool
- 512 neurons fully connected
- popcount for accumulation
- threshold for batchnorm normalization and activation function
- boolean OR for max-pooling
- folding for matrix–vector multiplication

CNV contains a succession of (3x3 convolution, 3x3 convolution, 2x2 maxpool) layers repeated three times with 64-128-256 channels, followed by two fully connected layers of 512 neurons each and a fully connected layer for classification, as shown in Fig. 2.

The convolutional operation of CNN which is multiply-accumulation (MAC) can be transformed to binary multiply-accumulation (BMAC) operation by applying XNOR followed by popcount operation in which set bit represents +1 while unset bit represents -1. After the XNOR operation performs the multiplication of binary values, instead of accumulation with signed arithmetic as in CNN, a popcount operation implements the summary of binary dot product by counting the number of bit 1. Performing the BNN on FPGAs is significantly efficient since it is easy to map and parallel the XNOR and popcount operations on the LUTs (Look-up Tables) and FFs (Flip-flops).

The output of the XNOR and popcount operations is then fed to a single thresholding function to determine output activation. As batch normalization normalizes data with the mean of 0 and variance of 1, it can be computed from the data before normalization which value will equal 0 after normalization. Moreover, the activation function in BNN actually is the thresholding function with the threshold of 0. Hence, using threshold can achieve the same output as batch normalization and activation function.

The max-pooling operation aims to get the maximum value. In this model, the value is represented by set and unset bits, therefore, the maximum value of the set of these bits is calculated through the Boolean-OR.

Once the PS has received the classification result from the PL, the parking spaces are immediately determined whether they are occupied or not.

IV. Evaluation

To evaluate our solution, we take parking spaces data from two dataset: PKLot and CNRPark+EXT. Because this problem focuses on car parking occupation, the dataset will include occupied and empty parking slots.

CNRPark+EXT is a dataset for visual occupancy detection of parking lots of roughly 150,000 labeled images (patches) of vacant and occupied parking spaces, built on a parking lot of 164 parking spaces. CNRPark+EXT extends CNRPark, a preliminary dataset composed by 12,000 images collected in different days of July 2015 from 2 cameras. The additional subset, called CNR-EXT, is composed by images collected from November 2015 to February 2016 under various weather conditions by 9 cameras with different perspectives and angles of view. CNR-EXT captures different situations of light conditions, and it includes partial occlusion patterns due to obstacles (trees, lampposts, other cars) and partial or global shadowed cars[1]. On the other hand, PKLot dataset contains 12,417 images (1280X720) captured from two different parking lots in sunny, cloudy and rainy days. The first parking lot has two different capture angles. Each image of the database has a XML file associated including the coordinates of all the parking spaces and its label (occupied/vacant). By using the XML files to segment the parking space, you will be able to get around 695,900 images of parking spaces[2].

We randomly divided the images from CNRPark + EXT and PKLot dataset into three sets namely training, validation and testing set to evaluate the performance of the model under different circumstances. The number of images in each set is indicated in the Table I,II & III.

We set up the experiments with four training, validation and testing set's combinations from the CNRPark + EXT and PKLot dataset, as shown in Fig 3-6, for evaluating the performance of training process. The training and validation set use the same dataset while the testing set can be tested on the same or different dataset. All experiment's results are displayed in Fig 7 where each result is corresponding to its dataset setup.

First of all, we trained the model with two common optimizers, RMSProp and Adam. RMSProp which is a gradient-based optimization technique used in training neural was developed as a stochastic technique for mini-batch learning, whereas the Adam optimizer makes use of a combination of ideas from other optimizers networks - AdaGrad and RMSProp. Currently the Adam optimizer is the preferred optimizer for use with deep learning models. In our experiments, despite the differences in dataset and batch size, Adam had a lower training loss and better testing accuracy compared to RMSProp.

Besides that, we also evaluated the performance of two above parking car dataset. When all data was taken from only one dataset, either CNRPark + EXT or PKLot, we observed the low training loss which were less than 0,001 in most cases and high accuracy which were all nearly 100%. In contrast, a significant decrease of accuracy, which varied from over 70% to 85%, was witnessed when using different dataset to test the model (experiment 10 to 15). Moreover, the training process

[1]http://cnrpark.it/

[2]https://web.inf.ufpr.br/

using PKLot was better as it got a lower training loss and a higher testing accuracy. As can be seen, individual dataset can not cover all the possible views (or slots) in the parking place, therefore it's necessary to train the data from both dataset.

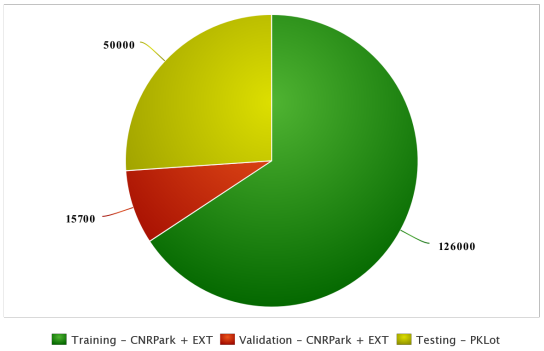

Fig. 3. Experiment 1, 2, 3 & 4 dataset setup

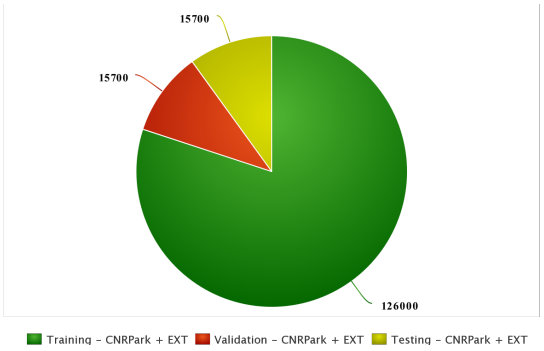

Fig. 4. Experiment 4, 5, 6, 7, 8 & 9 dataset setup

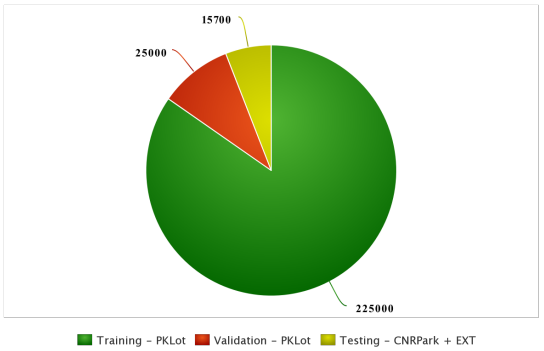

Fig. 5. Experiment 10, 11, & 12 dataset setup

During training, the differences in accuracy between three cases of weight and activation with the same setup of dataset and batch size are under 5% and even nearly equal in some experiments. When deploying the model on Ultra96-v2 development board, it showed that the model using 1-bit weight along with 1 or 2-bit(s) activation gave the result about 12% faster than using 2-bit Weight 2-bit Activation in inference time which was calculated from when the data is transferred to the PL to when the result is sent back to the PS.

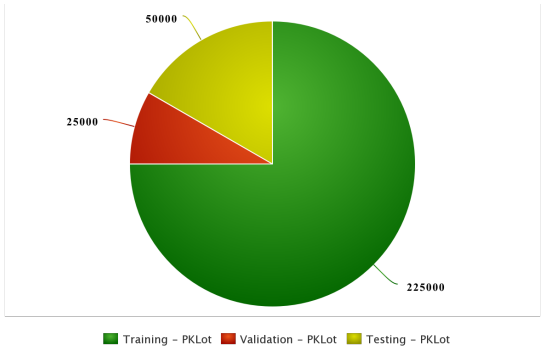

Fig. 6. Experiment 13, 14, & 15 dataset setup

TABLE I
ULTRA96-V2 ON-BOARD INFERENCE TIME

No	Weight	Activation	Inference time	FPS
1	1 bit	1 bit	0,0160s	62,412
2	1 bit	2 bits	0,0160s	62,021
3	2 bits	2 bits	0,0182s	54,993

TABLE II
ENERGY CONSUMPTION WHEN INFERENCING CNV MODEL ON ULTRA96V2 AND VGG MODEL ON JETSON NANO

Platform	Latency (ms)	FPS	On-Chip Power (W)	Power per image (W)
Ultra96v2	12.99	77	2.967	0.039
Jetson Nano	91.02	11	5	0.455

V. CONCLUSION

In this paper, we presented a real-time HW/SW co-design methodology to binary neural network. Utilizing such a heterogeneous platform as the Zynq SoC allows the inference network to be implemented within the programmable logic of the device, providing a significant increase in performance while keeping power consumption low. The low cost to implement and run this advanced detection capability demonstrates that such a system may be accommodated within a platform for immediate application, and in support of more complex operations of high value missions or payloads.

ACKNOWLEDGEMENT

This research is funded by Ho Chi Minh City University of Technology - VNU-HCM under grant number To-KHMT-2020-xx. We acknowledge the support of time and facilities from Ho Chi Minh City University of Technology (HCMUT), VNUHCM for this study.

REFERENCES

[1] R. Arnott, T. Rave, and R. Schob, "Alleviating Urban Traffic Congestion". Cambridge, MA, USA: MIT Press, 2005.
[2] L. Mainetti, L. Patrono, M. Stefanizzi, R. Vergallo, "A Smart Parking System based on IoT protocols and emerging enabling technologies", IEEE 2nd World Forum on Internet of Things (WF-IoT), 2015.
[3] J. K. Suhr and H. G. Jung, "Sensor fusion-based vacant parking slot detection and tracking," IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 1, pp. 21–36, Feb 2014.
[4] Sadhukhan, Pampa. "An IoT-based E-parking system for smart cities," 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017.

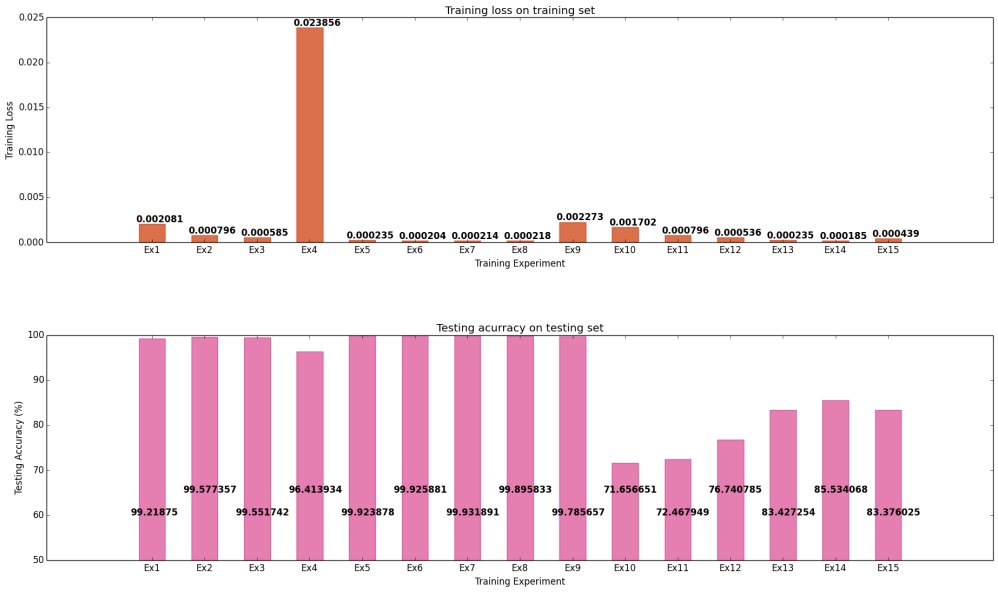

Fig. 7. Experiments result with training loss (above) and testing accuracy (below)

[5] I. M. Hakim, D. Christover, and A. M. Jaya Marindra, "Implementation of an Image Processing based Smart Parking System using Haar-Cascade Method," 2019 IEEE 9th Symposium on Computer Applications Industrial Electronics (ISCAIE), pp. 222—227, 2019.

[6] H. Bura et al, "An Edge Based Smart Parking Solution Using Camera Networks and Deep Learning," 2018 IEEE International Conference on Cognitive Computing (ICCC), pp. 17–24, 2018.

[7] Orhan Bulan et al, "Video-based real-time on-street parking occupancy detection system," Journal of Electronic Imaging 22, 2013.

[8] Yanxu Zheng, S. Rajasegarar, and C. Leckie, "Parking availability prediction for sensor enabled car parks in smart cities", 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 1–6, 2015.

[9] Qiyang Chen et al, "Video-Based Parking Occupancy Detection for Smart Control System", Applied Sciences 10, p. 1079, 2020.

[10] Daniel Ng Chiu Loong, Suhaila Isaak, and Y. Yusof, "Machine vision based smart parking system using Internet of Thing", TELKOMNIKA Telecommunication Computing Electronics and Control 17, pp. 2098–2106, 2019

[11] J. Ruili et al, "Smart Parking System Using Image Processing and Artificial Intelligence", 2018 12th International Conference on Sensing Technology (ICST), pp. 232–235, 2018.

[12] Petr Fedchenkov et al, "An Artificial Intelligence Based Forecasting in Smart Parking with IoT," 18th International Conference, NEW2AN 2018, and 11th Conference, ruSMART 2018, St. Petersburg, Russia, August 27–29, 2018, pp. 33–40.

[13] G. Amato et al, "Car parking occupancy detection using smart camera networks and Deep Learning", 2016 IEEE Symposium on Computers and Communication (ISCC), pp. 1212–1217, 2016.

[14] Anandhalli, Mallikarjun and V. Baligar, "A novel approach in real-time vehicle detection and tracking using Raspberry Pi", alexandria engineering journal 57 (2017), pp. 1597-1607.

[15] S. Kawakura and R. Shibasaki, "Deep Learning-Based Self-Driving Car: JetBot with NVIDIA AI Board to Deliver Items at Agricultural Workplace with Object-Finding and Avoidance Functions", European Journal of Agriculture and FoodSciences (EJFOOD), vol. 2, no. 3, Jun. 2020.

[16] O. Eldash, A. Frost, K. Khalil, A. Kumar and M. Bayoumi, "Dynamically Reconfigurable Deep Learning for Efficient Video Processing in Smart IoT Systems," 2020 IEEE 6th World Forum on Internet of Things (WF-IoT), New Orleans, LA, USA, 2020, pp. 1-6

[17] W. Zhao, T. Ma, X. Gong, B. Zhang and D. Doermann, "A Review of Recent Advances of Binary Neural Networks for Edge Computing," in IEEE Journal on Miniaturization for Air and Space Systems, vol. 2, no. 1, pp. 25-35, March 2021, doi: 10.1109/JMASS.2020.3034205.

[18] Yaman Umuroglu, Nicholas J. Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre and Kees Vissers, "FINN: A Framework for Fast, Scalable Binarized Neural Network Inference," in Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (ACM), Feb 2017, doi: 10.1145/3020078.3021744.

[19] Author, F.: Article title. Journal **2**(5), 99–110 (2016)

[20] Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016), doi: 10.10007/1234567890.

[21] Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)

[22] Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)

[23] LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017.

[24] M. O. Hasan, M. M. Islam and Y. Alsaawy, "Smart Parking Model based on Internet of Things (IoT) and TensorFlow," 2019 7th International Conference on Smart Computing & Communications (ICSCC), 2019, pp. 1-5, doi: 10.1109/ICSCC.2019.8843651.