Volume 1, Issue 5, February 01, 2018

# TC-CPS Newsletter

**Technical Articles**

- Kin Sum Liu, Sirajum Munir, Shan Lin and Charles Shelton: *"Understanding Occupancy Patterns in a Commercial Space"*.
- Xiaowei Xu: *"On the Quantization of Cellular Neural Networks for Cyber-Physical Systems"*.
- Devki Nandan Jha, Yinhao Li, Prem Prakash Jayaraman, Saurabh Garg, Paul Watson, Rajiv Ranjan: *"Challenges in Automatic Deployment and Configuration Management in Cyber Physical System"*.
- Long Chen, Xuemin Hu, Wulin Huang: *"A New Planning Framework for Self-driving Vehicles against Emergencies"*.

**Summary of Activities**

**Call for Contributions**

IEEE TC-CPS Cyber-Physical Systems

# Understanding Occupancy Patterns in a Commercial Space

Kin Sum Liu[1], Sirajum Munir[2], Shan Lin[1], and Charles Shelton[2]

[1]Stony Brook University, Stony Brook, NY
[2]Bosch Research and Technology Center, Pittsburgh, PA

## 1 Introduction

Heating, ventilation, and air conditioning (HVAC) is a major source of energy consumption in the US. In 2006, approximately 35% of energy in the US was used for HVAC [1]. Usually building operators use a static schedule for controlling HVAC systems without having a deeper understanding of how many people use the building at different times of the day. In addition, HVAC systems operate by assuming maximum occupancy in each room, which leads to a significant energy waste, e.g., an HVAC system providing ventilation for 30 people when there are only 10 people in a room [9]. Understanding how many people use different rooms at different times of the day is very crucial for achieving building energy efficiency and providing occupant comfort.

There have been several attempts to collect long term occupancy patterns from office buildings, e.g., Mitsubishi's Electronic Research Lab (MERL) dataset [14] and Colorado School of Mines (CSMBB) dataset [4]. However, PIR motion detectors are used in these projects for sensing occupancy and hence these datasets only reflect whether a room is occupied or not without revealing the actual person count. Similar datasets are also collected from households [2][11]. Only a few datasets from commercial spaces capture actual person count, and even in these cases, the data collection period was very limited, e.g., in [3] only 5 days of occupancy data is collected as ground truth. We are the first to collect long term (9 months) occupancy data (people count) from an 11,000 square foot commercial office (second floor of Bosch Research and Technology Center Pittsburgh). In this paper, we outline some statistical results from our empirical study and explore their implications for saving energy.

The findings from the empirical study are about understanding and predicting occupancy patterns (1, 2, 3, 4), understanding the usage of a common office space (5, 6) and conference rooms (7, 8), and weekend behavior (9) as described below.

1. The daily occupancy pattern of the common office space is regular. However, the percentage change of the peak (maximum occupancy) varies from 0% to 730% with an average of 18.4%.
2. The daily occupancy pattern in conference rooms is very irregular.
3. Simple solution leveraging occupancy count from the previous day for predicting future occupants suffers a percentage error from 0% to 1100% with an average of 30.1% for the common office space. It is much worse for the conference rooms, e.g., in one conference room (Warhol) it varies from 0% to 2100% with an average of 104%.
4. The number of people in the initial 3 minutes is a useful indicator for predicting number of participants in a meeting. Such a simple predictor achieves 2.6 people RMSE error on average.
5. Arrival time of the earliest person is at 7:25 AM on average, which is 1.5 hours before the arrival of the earliest group of people, which is 10 people in this analysis.

6. Departure time of the latest person is 8:43 PM on average which is 3 hours after the latest group.

7. In conference rooms, the average and median number of participants in meetings are 3.85 and 3, respectively. We find that in 99% cases, the number of participants does not reach the maximum occupancy capacity of the conference rooms. In fact, the number of participants does not exceed half of the room capacity in 95% cases. The average and median durations of meetings are 63 minutes and 48 minutes, respectively.

8. The utilization of conference rooms varies from different times of the day with a mean of 55.6% between work hours (9 AM to 6 PM), which means conference rooms are unused in 44.4% time during work hours.

9. The office space is not completely unused during weekends. The chance of occupancy in a day for the common office space is 57.9% and the utilization of conference rooms is 4.8% on average during weekends.

We published some of these findings in a previous work briefly [5]. In this paper, we describe the findings in more details. The rest of the paper is organized as follows. In Section 2, we describe the deployment and data collection procedure. Then, we describe the findings in detail in Section 3. Finally, we discuss about the findings, their implications for saving energy, and our future work in Section 4.

## 2 Deployment and Data Collection

There are several solutions for occupancy estimation using IR-array sensors [8], ultrasonic sensors [13], and RGB cameras [12]. We use a depth sensor (Kinect for XBox One) for counting number of people in a room. The depth sensor is mounted on the ceiling looking downwards near to a doorway as it accurately counts the number of people entering and exiting through the door. The solution is called FORK (Fine grained Occupancy estimatoR using Kinect) and the detailed algorithm is described in [9, 10]. We have deployed five instances of FORK at a Bosch office to cover the *common office space* (requires monitoring at two entrances: Main Gate and Remote Gate), two *conference rooms* (Warhol and Clemente), and a lab. Warhol and Clemente conference rooms are the most used and largest conference rooms of this office that can accommodate 25 and 20 people, respectively. The floor plan of the office is shown at Figure 1. Four FORK units were deployed on August 24th 2015. The Remote Gate was not used at that time often, but later a group migrated to the other side of the office and hence the Remote Gate unit was deployed on January 27th 2016. After collecting over 9 months of data, we stopped the data collection on June 7th 2016. During this period, there were 64,925, 18,477, 9,938 entrance and exit events in the common office space (Main Gate and Remote Gate), Warhol, and Clemente conference rooms, respectively. The entire dataset is available at [6].

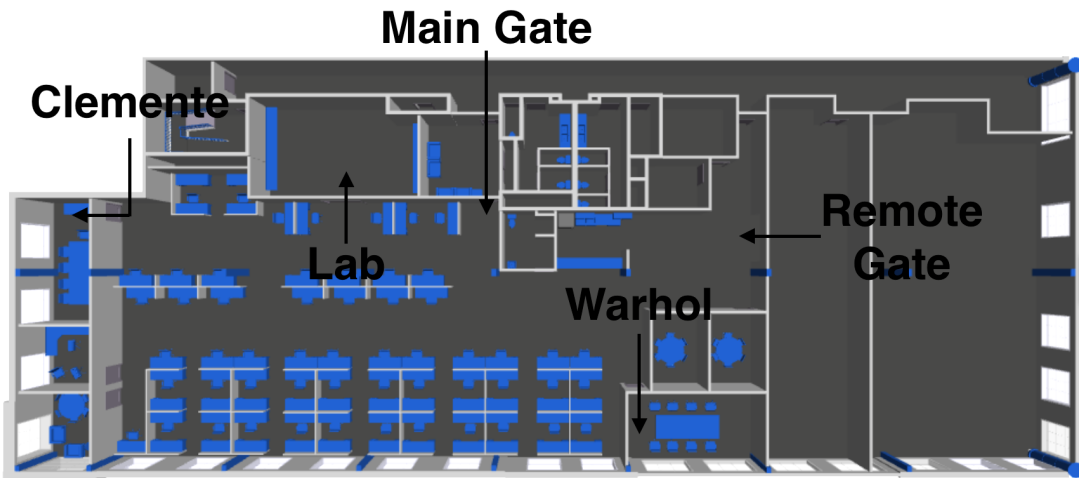



Figure 1: Floor plan of the office

## 3 Findings in Detail

In this section, we describe the findings from the long term empirical study in detail.

**3.1 Occupancy patterns in the common office space:** Occupancy count of the common office space from April 2016 is shown in Figure 2. It shows that the shape of waveform is consistent across days and weeks in the sense that the influx and outflux of people happen at similar times. However, the magnitude (maximum people count in a day) varies, which is shown in Figure 3.

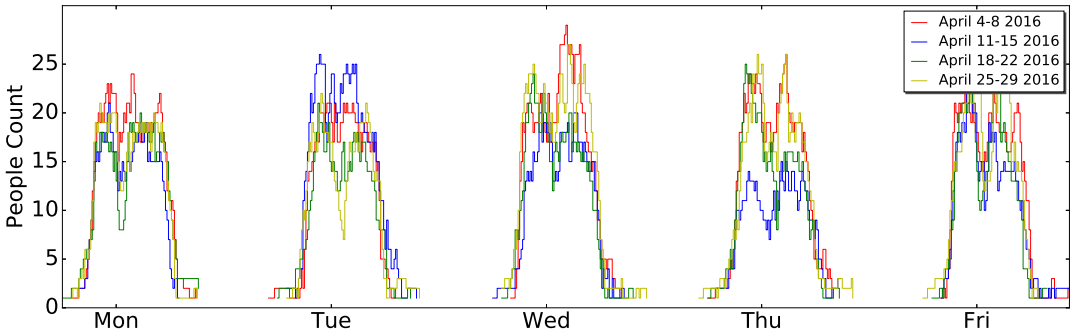



Figure 2: People count in common office space in April 2016

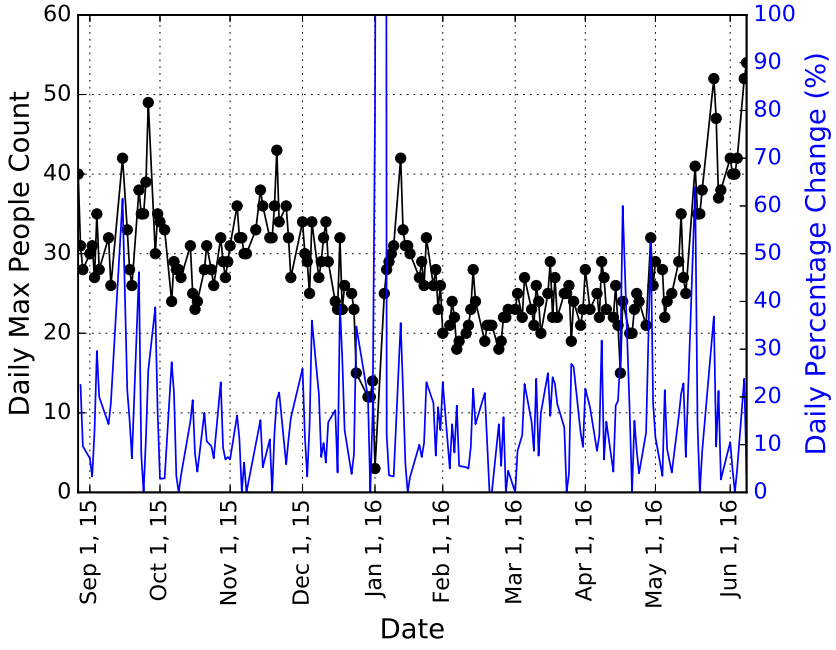



Figure 3: Variation of maximum people count in the common office space

The black curve shows a large variation of daily maximum people count with an average day-to-day change of 4.1 persons. The percentage change is 18.4% on average with a maximum of 730% on December 31st 2015 (a cut off portion is shown in the figure) and a minimum of 0%. The reason for this variation is interns joining and leaving, joining of new employees, arrival of visitors, arrival of interview candidates, employees travelling and on vacation, and different teams migrating to different parts of the building.

**3.2 Occupancy patterns in conference rooms:** Occupancy pattern of a conference room (Warhol) for the same month (April) is shown in Figure 4. It shows that occupancy patterns are not consistent even across days. The tim-

ing, frequency, size and duration of meetings vary significantly in different days.

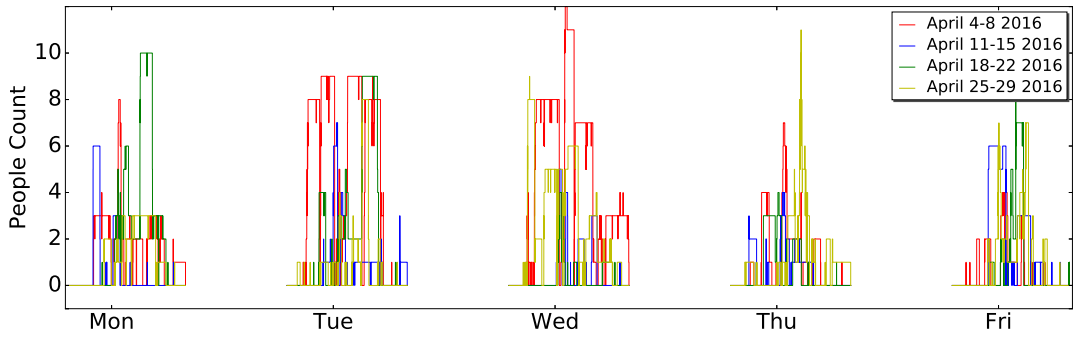
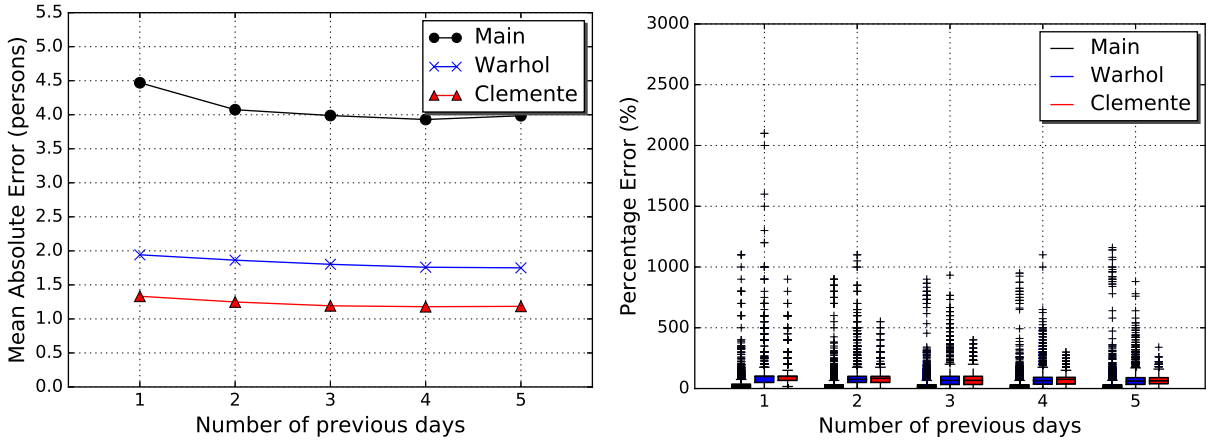
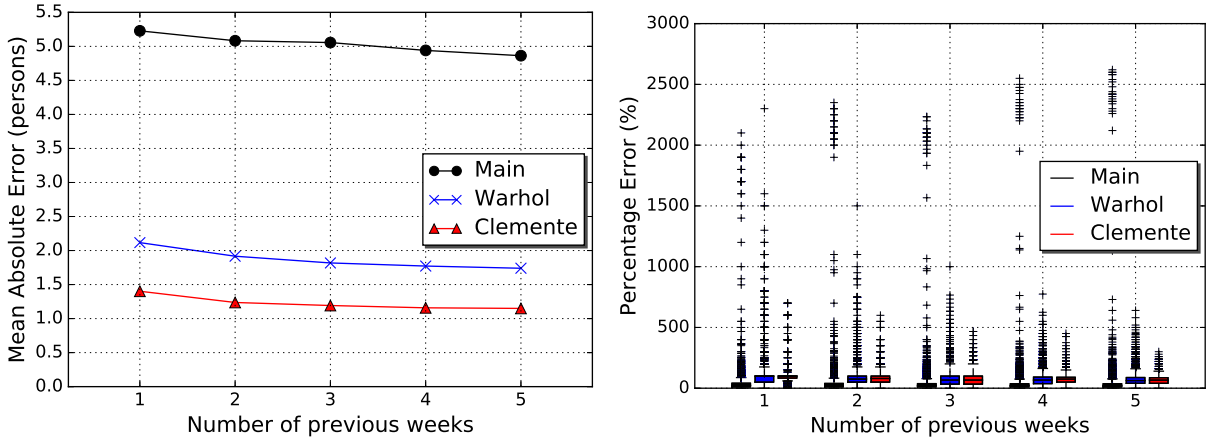



Figure 4: People count in a conference room (Warhol) in April 2016



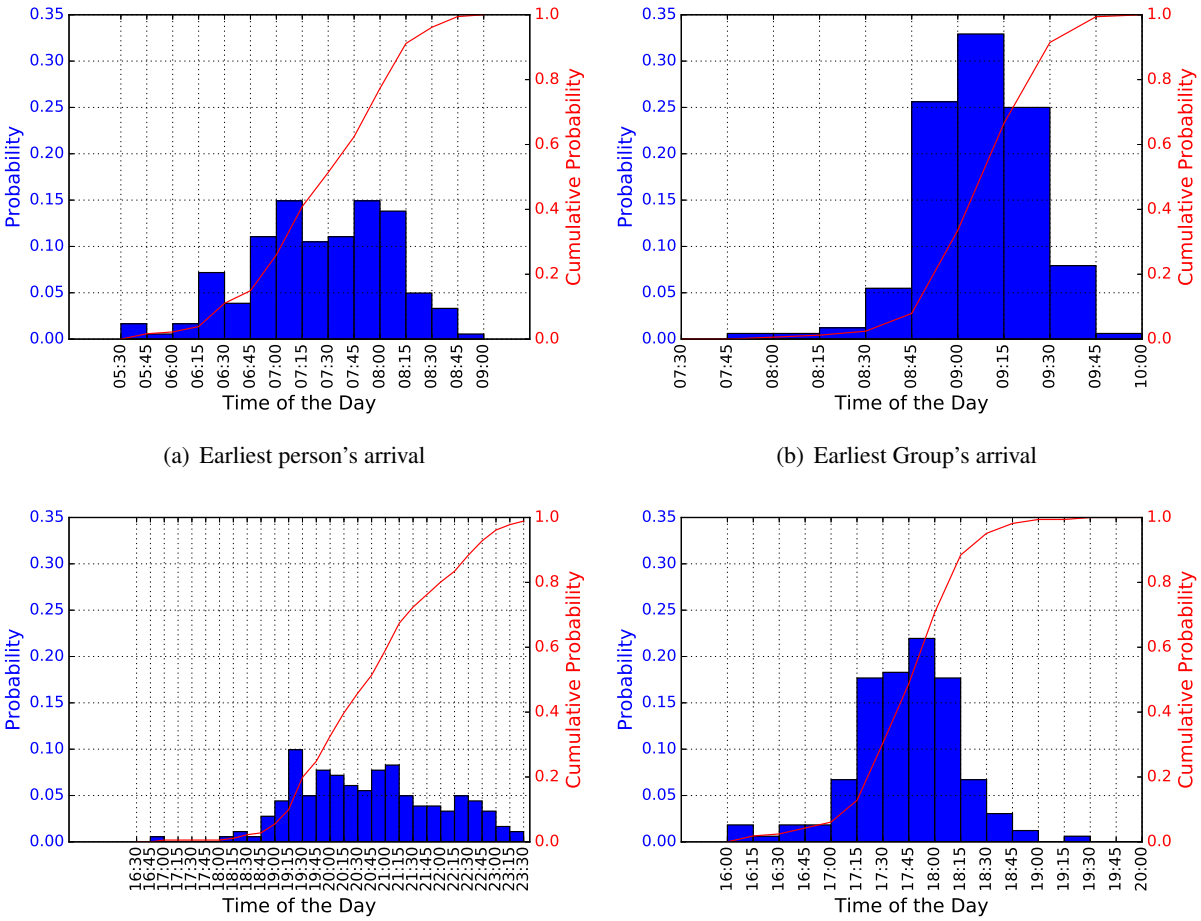

(a) Using previous days for prediction




(b) Using previous weeks for prediction

Figure 5: Prediction errors using previous data points to predict future occupancy count from 9:00 to 18:00

**3.3 Prediction of future occupancy:** One simple way to predict occupancy pattern is by averaging the historical occupancy data of previous days. We define a time slot as a 15 minute interval and predict the number of people in a particular slot by averaging the occupancy count of the same slot in the previous $n$ days (Figure 5(a)) and same weekdays in the previous $n$ weeks (Figure 5(b)). We vary $n$ from to 1 to 5 in the $x$-axis of both figures. We show both




(a) Earliest person's arrival

(b) Earliest Group's arrival

(c) Latest person's departure

(d) Latest Group's departure

Figure 6: Distribution of arrival and departure times in the common office space

Mean Absolute Error (offset from the actual count) and box plot for Percentage Error (percentage offset from the actual count) in the sub-figures. The figures show that the prediction for the count of common office space suffer a large error and the error for conference rooms is even more severe. Using the previous day for prediction, the means of percentage errors are 30.1%, 104% and 97.6% for common office space, Warhol and Clemente, respectively. We exclude 0.2%, 48.2%, and 61.5% instances when the actual count is 0 and hence percentage error becomes undefined. If the previous week's data is used, the means of percentage errors are increased to 38.5%, 112% and 98.1%, respectively. We exclude 0.2%, 48.4%, and 61.3% instances when the actual count is 0 and percentage error is undefined.

**3.4 Prediction of number of participants:** Figure 7 shows how error varies when number of participants at different times from the beginning of a meeting is used as a direct predictor of the maximum number of participants in the meeting. It shows that after 1 minute of when a meeting starts, the Root Mean Square Error reaches below 2 and 4 persons from the actual number of participants with a Mean Percentage Error at 29% and 35% for Clemente and Warhol, respectively. If this time interval is increased to 3 minutes, RMSE is reduced to 1.6 and 3.6, while the percentage error is 21% and 26% for Clemente and Warhol, respectively.

**3.5 Arrival time:** The arrival time of the earliest person ranges between [5:30, 9:00], which is shown in in Figure 6(a). For 50% of the days, we observe people arriving as early as 7:30. We define a group consisting of at least 10 people. The arrival time of the earliest group is shown in Figure 6(b), which shows that the range is much smaller

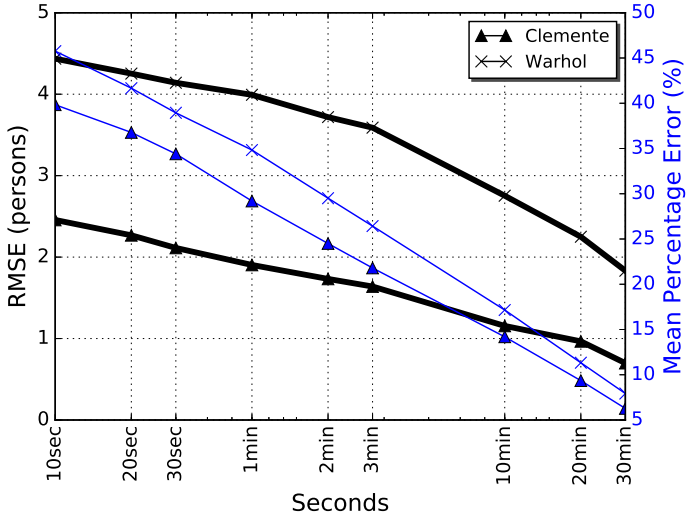



Figure 7: Prediction error on size of meetings based on number of people at the beginning of meetings

spanning [7:45, 10:00] and it happens between [8:30, 9:30] usually. On average, the earliest person arrives 1.5 hours before the earliest group.

**3.6 Departure Time:** We show the departure time of the latest person in Figure 6(c), which shows that the departure time ranges between [16:45, 23:30]. The outlier 16:45 happened on December 31st 2015. If we exclude this data point, the span is still across 4 hours. The departure time of the latest group is shown in Figure 6(d), which shows that the range is smaller spanning between [16:00, 19:30] and this happens between [17:00, 18:30] usually. On average, the latest person leaves almost 3 hours after the latest group.

**3.7 Duration and number of occupants in meetings:** The main usage of conference rooms is for meetings. We define a meeting as an occupancy of more than one person in the room for more than 1 minute. The distribution of duration and number of participants in meetings in the conference rooms (Clemente and Warhol) are shown in Figures 8 and 9, respectively. Average durations of a meeting in Clemente and Warhol are 62.2 minutes and 65.3 minutes, respectively. Number of participants in a meeting in Clemente and Warhol are 3.4 and 4.3 people on average. Both metrics have long tails that indicates large or long meetings happen infrequently but they exist. Clemente's and Warhol's capacity is 20 and 25 people respectively. For both conference rooms, 99% of meetings have not reached the maximum capacity. Also, 95% of meetings have the number of participants equal to or below 7 and 12, respectively. Therefore, for most of the times, they are occupied by less than 50% of maximum capacity. The outlier happened once on November 24th 2015 when Warhol hosted 29 people for a Thanksgiving party.

**3.8 Utilization of conference rooms:** Figure 10 shows the utilization rate of both conference rooms with respect to different times of the day over the whole observation period. It is utilized if there is at least one meeting in that hour. The average utilization rate of Clemente and Warhol for between 9:00 to 18:00 is 47.0% and 64.2% respectively. Therefore, the average utilization between work hours (9 AM to 6 PM) is 55.6%. In the early morning before 9:00 and late afternoon after 18:00, the chance of each room being used is less than 20%. The two conference rooms also have different patterns of usage. For both conference rooms, the utilization rate varies greatly throughout the day. First, Clemente is used less frequently. The utilization does not exceed 65% for any time of day. There is also a drop in utilization rate at noon because employees take their lunch break. On the contrary, Warhol is used for around 70% from 10:00 to 16:00. The peak appears at noon because some employees have their lunch in Warhol at that time.

**3.9 Occupancy patterns during weekends:** Throughout the data collection period, there is a chance of 57.9% that at least one person comes to the common office space on a day during weekends. On average, the office is used for

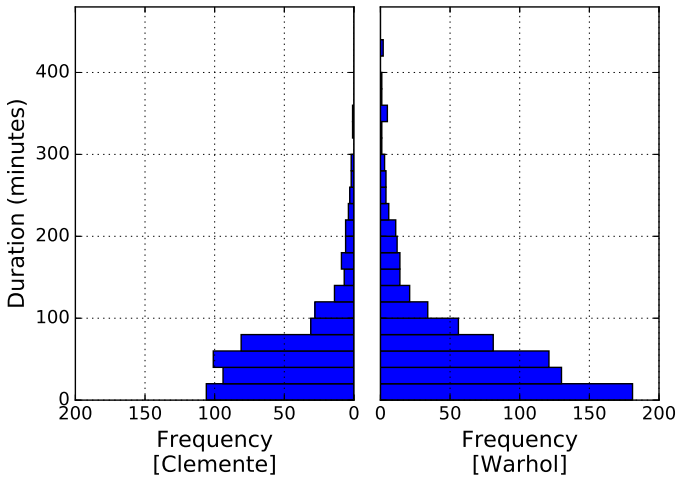



Figure 8: Distribution of duration of meetings.

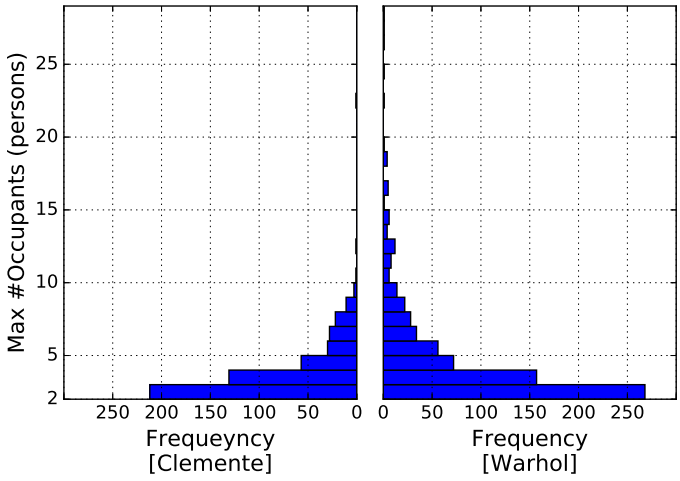



Figure 9: Distribution of number of occupants in meetings.

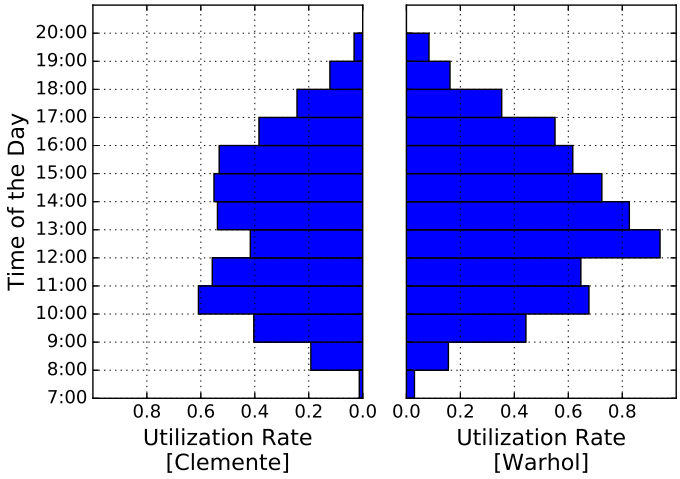



Figure 10: Hourly utilization rate of two conference rooms from 7:00 to 20:00.

187 minutes with a standard deviation of 212 minutes. The conference rooms are rarely used. The chance of usage is 2.78% and 6.85% for Clemente and Warhol, respectively. The average utilization is 4.8%.

## 4 Discussions and Future Works

In addition to detecting humans, it is useful to sense additional contextual information regarding them for an efficient HVAC control. In our recent work [7], we detect objects that are carried inside and outside of a room in a building, e.g., backpacks, boxes, laptops, phones, umbrellas, cups, and guns. In addition to improving safety and security of the occupants, object detection can be useful for controlling HVAC systems to save energy. For example, in a commercial space, if someone is leaving his office with a backpack in the afternoon, it may mean he is leaving for the day, whereas if he is leaving with a laptop, it may mean he is going out for a meeting, and leaving empty handed may mean that he is leaving for a restroom/lunch break. If someone enters in a conference room with a phone at his hand/ear, he is probably going there to make a phone call, which may not last as long as a regular office meeting. An HVAC system can potentially employ a better control strategy with such additional information.

Our empirical study suggests several findings for more energy efficient HVAC control. We see that the variance of the earliest arrival and latest departure time for an individual is much larger than that of a group behavior. Hence, if we sacrifice the comfort of the first and last few people, that will reduce energy consumption substantially. In that way, instead of running the HVAC system at a full rate from 7:00 to 23:30, we can run it at a lighter load between 6:30 to 8:30 and between 18:30 to 23:30. Occupancy count of the common space shows a periodic behavior with a variation of amplitude (maximum occupancy). Hence, an adaptive solution is useful for predicting occupancy in the common space. However, occupancy prediction is very difficult for conference rooms. We see that occupancy in the first 3 minutes is a good indicator of the number of participants in a meeting in conference rooms. Hence, a multistage HVAC system can be useful for saving energy. Another reason for using it is that conference rooms are not used for 44.4% time during office hours. Hence, instead of keeping the rooms warm/cool during the entire work day, the rooms can be put to a setback threshold and be heated/cooled based on demand. Also, we find that in 95% cases, the number of occupants does not exceed half of the capacity of conference rooms. Hence, a Variable Air Volume (VAV) HVAC system will save energy. In the future, we plan to use a simulation tool to determine the amount of energy savings possible using these techniques.

## 5 Acknowledgments

This work was supported, in part, by DOE grant DE-EE0007682 and NSF grant CNS 1536086. The opinions expressed here are those of the authors and do not necessarily reflect the views of the DOE.

## References

[1] EIA - energy information administration. http://www.eia.doe.gov/.

[2] Christian Beckel, Wilhelm Kleiminger, Romano Cicchetti, Thorsten Staake, and Silvia Santini. The eco data set and the performance of non-intrusive load monitoring algorithms. In *Proceedings of the 1st ACM International Conference on Embedded Systems for Energy-Efficient Buildings (BuildSys)*, pages 80–89. ACM, November 2014.

[3] Varick L. Erickson, Miguel Á. Carreira-Perpiñán, and Alberto Cerpa. OBSERVE: occupancy-based system for efficient reduction of HVAC energy. In *IPSN*, 2011.

[4] W. A. Ho and J. W. Howard. Activity recognition in a dense sensor network. In *International Conference on Sensor Networks and Applications (SNA)*, 2009.

[5] K. S. Liu, S. Munir, J. Francis, C. Shelton, and S. Lin. Poster abstract: Long term occupancy estimation in a commercial space: an empirical study. In *IPSN*, 2017.

[6] K. S. Liu, E. V. Pinto, S. Munir, J. Francis, C. Shelton, M. Berges, and S. Lin. Poster: COD: a dataset of commercial building occupancy traces. In *BuildSys*, 2017.

[7] N. C. Mithun, S. Munir, K. Guo, and C. Shelton. ODDS: real-time object detection using depth sensors on embedded gpus. In *IPSN*, 2018.

[8] H. Mohammadmoradi, S. Munir, O. Gnawali, and C. Shelton. Measuring people-flow through doorways using easy-to-install ir array sensors. In *DCOSS*, 2017.

[9] S. Munir, R. S. Arora, C. Hesling, J. Li, J. Francis, C. Shelton, C. Martin, A. Rowe, and M. Berges. Real-time fine grained occupancy estimation using depth sensors on ARM embedded platforms. In *RTAS*, 2017.

[10] S. Munir, L. Tran, J. Francis, C. Shelton, R. S. Arora, C. Hesling, M. Quntana, A. K. Prakash, A. Rowe, and M. Berges. Demo: FORK: Fine grained Occupancy estimatoR using Kinect on arm embedded platforms. In *BuildSys*, 2017.

[11] James Scott, AJ Bernheim Brush, John Krumm, Brian Meyers, Michael Hazas, Stephen Hodges, and Nicolas Villar. Preheat: controlling home heating using occupancy prediction. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 281–290. ACM, 2011.

[12] Jakub Segen and Sarma Pingali. A camera-based system for tracking people in real time. In *ICPR*, 1996.

[13] Oliver Shih and Anthony Rowe. Occupancy estimation using ultrasonic chirps. In *ICCPS*, 2015.

[14] C.R. Wren, Y.A. Ivanov, D. Leigh, and J. Westhues. The MERL motion detector dataset. In *Workshop on Massive Datasets (MD)*, pages 10–14, November 2007.

# On the Quantization of Cellular Neural Networks for Cyber-Physical Systems

Xiaowei Xu, University of Notre Dame

## 1 Introduction and Motivation

Cyber-Physical Systems (CPSs) have been pervasive including smart grid, autonomous automobile systems, medical monitoring, process control systems, robotics systems, and automatic pilot avionics [1]. As usually implemented on embedded devices, CPS is typically constrained by computation capacity and energy consumption. In some CPS applications such as telemedicine [23] and advanced driving assistance system (ADAS) [33], data processing on the embedded devices is preferred due to security/safety and real-time requirement. Therefore, high efficiency is highly desirable for such CPS applications.

A very powerful tool for telemedicine and ADAS is cellular neural network (CeNN), which can achieve very high accuracy through proper training. It should be noted that CeNNs are popular in image processing areas such as classification [7], segmentation [8], while convolutional Neural Networks (CNNs) are most powerful in classification related tasks. However, due to the complex nature of segmentation and other image process tasks and the associated real-time requirements in many applications, hardware implementations of CeNNs have remained an active research topic in the literature.

The structure of CeNNs makes them a natural fit for analog implementations. Many studies exist along this direction [10][24][17][2]. The advantages of analog implementations include high performance with an extremely fast convergence rate and the convenience of integrating them into image sensors for direct processing of captured data. However, these analog implementations suffer from Input/output (I/O) and data precision problems. First, they require that each input corresponds to a unique neuron cell, resulting in too many I/O ports. For example, recent implementation [2] can only support 256×256 pixels at its most, which is far from the requirement of mainstream images, e,g., 1920×1080 pixels. Second, analog circuits are prone to noise, which limit the output data precision to 7 bits or below [30]. As a result, analog implementation cannot even process regular 8-bit gray images.

In view of the above issues, digital implementations of CeNNs have been proposed, where data is quantized with approximation. Tens to hundreds of iterations are needed in the discretized process and as a result, the computational complexity of digital CeNNs is very high. For example, to process an image of 1920x1080 pixels requires 4-8 Giga operations (for 3×3 templates and 50-100 iterations), which needs to be done in a timely manner for real-time medical image segmentations.

To tackle the computation challenge, CeNN accelerations on digital platforms such as ASICs [13][16], GPUs [21] and FPGAs [4][20] [18][30][31] [19] have been explored, with FPGA among the most popular choices due to its high flexibility and low time-to-market. The work [4] presented a baseline design with several applications, while the study [20] took advantage of reconfigurable computing for CeNNs. Recently, the CeNN implementation for binary images was demonstrated [19]. Expandable and pipelined implementations were achieved on multiple FPGAs [18]. Taking advantage of the structure in [18], the work [30] implemented a high throughput real-time video streaming system, which is further improved to be a complete system for video processing [31]. All the three works share the same architecture for CeNN computation. Due to the large number of multiplications needed in CeNNs, the limited number of embedded multipliers in an FPGA become the bottleneck for further improvement. For example, in work [18] 95%-100% of the embedded multipliers are used. On the other hand, it is interesting to note that the utilization rates of LEs and registers are only 5% and 2%, respectively, which is natural to expect as not many logic operations are needed. However, in a mainstream FPGA, LEs and registers count for significantly larger portion of the total programmable resources than embedded multipliers. For example, LEs and registers occupy 95.4% of the core area while embedded multipliers only 4.6% for a EP3LS340 FPGA [28]. Such an unbalanced resource utilization apparently cannot attain the best possible speed of the CeNN being implemented, and an improved strategy is strongly desired.

A naive approach for potential improvement is to use LEs and registers to implement additional multipliers. This technique, although straightforward, is very inefficient due to the high cost. For example, it takes 676 LEs and 486 shift registers to implement an 18-bit multiplier. For an XC4LX25 FPGA, all the LEs and registers can

only contribute 42% additional multipliers. Apparently, such an approach would not lead to significant improvement and we aim to address the problem through an alternative approach, i.e., by completely eliminating the need of multipliers. From basic Boolean algebra, we know that the multiplication of any number with powers of two can simply be done with logic shift, which only requires a small number of LEs and registers to achieve. Inspired by this, we can quantize the values in CeNN templates to powers of two, so that we can make full use of the abundant LEs and registers in FPGAs. An extra benefit from this approach is that LEs and registers are much more flexible for placement and routing, leading to higher clock frequencies. While this can lead to significantly higher resource utilization rate and reduced computational complexity, many interesting questions still remain. For example, how would such quantizations affect the final CeNN accuracy? What is the impact of different quantization strategies? Note that quantization to powers of two has been explored in the context of CNNs [32], but as detailed in Section 2.3, the difference in computation structures between CeNNs and CNNs warrants a separate investigation for CeNNs. And indeed, our findings show that the answers to these questions are different for the two.

In this paper we present CeNN quantization for high-efficient processing for CPS applications, particularly telemedicine and ADAS applications. We systematically put forward powers-of-two based incremental quantization of CeNNs for efficient hardware implementation. The incremental quantization contains iterative procedures including parameter partition, parameter quantization, and re-training. We propose five different strategies including random strategy, pruning inspired strategy, weighted pruning inspired strategy, nearest neighbor strategy, and weighted nearest neighbor strategy. Experimental results show that our approach can achieve a speedup up to 7.8x with no performance loss compared with the state-of-the-art FPGA solutions for CeNNs.

The remainder of the paper is organized as follows. Section 2 introduces backgrounds and motivation of the paper. The proposed framework for CeNN and the optimized hardware implementation are presented in Section 3. Experiments and discussion are provided in Section 4 and concluding remarks are given in Section 5.

# 2 Preliminaries

## 2.1 Cellular Neural Networks

Different from the prevalent CNNs which are superior for classification tasks, the CeNN model is inspired by the functionality of visual neurons. In a CeNN, a mass of neuron cells are connected with neighbouring ones, and only adjacent cells can interact directly with each other. This is a significant advantage for hardware implementation, resulting in much less routing complexity and area overhead. CeNNs are superior at image processing tasks that involve sensory functions, such as noise cancellation, edge detection, path planning, segmentation, etc. For the widely used 2D CeNN with space-invariant templates, the dynamics of each cell state with an M×N rectangular cell array [5] are as follows:

$$\dot{x}_{i,j}(t) = -x_{i,j}(t) + \sum_{k,l=-N}^{N} (A_{k,l}(t) y_{i+k,j+l}(t) + B_{k,l}(t) u_{i+k,j+l}(t)) + I(t), \tag{1}$$

$$y_{i,j}(t) = f(x_{i,j}(t)) = 0.5 \times (|x_{i,j}(t)+1| - |x_{i,j}(t)-1|), \tag{2}$$

where $1 \leq i \leq M$, $1 \leq j \leq N$, $A_{k,l}(t)$ is the feedback coefficient template, $B_{k,l}(t)$ is the feedforward coefficient template, $I(t)$ is the bias, and $x_{i,j}(t)$, $y_{i+k,j+l}(t)$ and $u_{i+k,j+l}(t)$ are the state, output and input of the cell, respectively. Note that $A_{k,l}(t)$, $B_{k,l}(t)$ and $I(t)$ are time-variant templates, and $t$ can be removed when time-invariant templates are used. For efficient implementation on a digital platform (e.g., CPU, GPU, FPGA), discrete approximation of CeNN is obtained by applying forward Euler approximation as shown in Equations (3), (4) and (5).

$$x_{i,j}(t) \cong (x_{i,j}(n+1) - x_{i,j}(n))/\Delta t. \tag{3}$$

$$x_{i,j}(n+1) = x_{i,j}(n) + \Delta t(-x_{i,j}(n) + I(n) + \sum_{k,l=-N}^{N} (A_{k,l}(n) y_{i+k,j+l}(n) + B_{k,l}(n) u_{i+k,j+l}(n))). \tag{4}$$

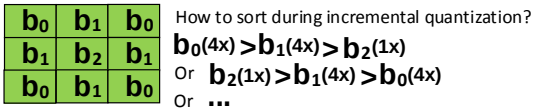



Figure 1: CeNN template for binary image noise cancellation application.

$$y_{i,j}(n) = f(x_{i,j}(n)) = 0.5 \times (|x_{i,j}(n)+1| - |x_{i,j}(n)-1|). \tag{5}$$

Delayed CeNN is a special type of CeNN described by adding $\sum_{k,l=-N}^{N}(D_{i,j}(n)g(x_{k,l}(n),y_{k,l}(n),u_{k,l}(n))$ to Equation (4), where $g$ is usually a piece-wise constant function. Please refer to [5] for details. For the mainstream image size with 1920×1080 pixels, the total complexity is $1920\times1080\times39\times100$=$8.1\times10^9$ operations with 100 iterations (19 multiplications and 20 additions in each iteration). This warrants exploration of hardware approaches to speedup CeNN computations.

## 2.2 Template Learning Algorithm and PSO Algorithm

Template learning is a widely applied method to find satisfactory templates for CeNN-based applications, in which Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) are two representatives. PSO is adopted in this paper, while GA and other template learning methods are also compatible with the framework proposed here.

PSO finds solutions (i.e., determining A, B and I templates) in a heuristic way by searching the solution space with multiple particles (swarm of potential solutions). In each iteration, PSO performs position update and object function calculation. Inspired by the social behavior of animals, the position update of each particle is affected by its past best position and the position of the current global best position as depicted by Equation (6),

$$p_{i,d}(n+1) = p_{i,d}(n) + \{w \times v_{i,d}(n) + c_1 r_1 \times (pb_{i,d} - p_{i,d}(n)) + c_2 r_2 \times (gb_d - p_{i,d}(n))\}. \tag{6}$$

where $1 \le i \le N$, $1 \le d \le D$, $N$ is the size of particles, $D$ is the dimension of each particle, $c_1$ and $c_2$ are the acceleration coefficients, and $r_1$ and $r_2$ are random numbers with uniform distribution. $p_i(n+1)$ and $p_i(n)$ are the positions of the $i$th particle in iteration $n$ and $n+1$, respectively. $pb_n$ is the best position that the $i$th particle ever searches, and $gb$ is the current best position among all particles. Inertia weight $w$ controls the balance of the search algorithm between exploration and exploitation. A bound of $[min_d, max_d]$ is introduced for $p_{i,d}$ to limit the solution space. The object function for particles taking positions as input is designed according to applications. In CeNN training, PSO will search the space constructed with A, B and I templates, and the templates with the best object function value are obtained as the learned templates.

## 2.3 Motivation

While hardware oriented memory/computation compression and optimization of CNNs have been extensively studied recently [6][11] [29][22][26][32], little has been explored for CeNNs where memory consumption is not a problem and the focus is only on computational complexity.

The main difference between CeNNs and CNNs is that in CeNNs the parameters are coupled. The weight values in a CNN tend to be all unique. However, in CeNNs some parameters share the same values. For example, in Figure 1, a CeNN template (template B) for binary image noise cancellation [14] is shown. Only three different values exist for the nine parameters. As such, in [32] the weights of CNNs are incrementally quantized in an order simply based on their magnitudes (pruning-inspired strategy). The same strategy may not work well for CeNNs, as a parameter with small magnitude may repeat multiple times thus playing a more important role than a parameter with a large magnitude but appearing only once. Furthermore, the training process of CNNs is mathematically optimal, while that of CeNNs is heuristic. This will also influence the performance of quantization strategies. Finally, the

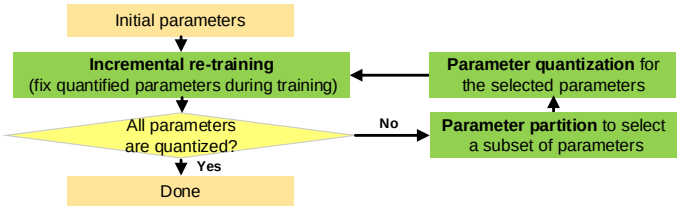


Figure 2: The flowchart of incremental quantization.

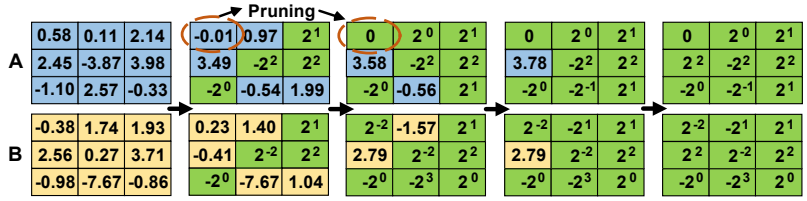


Figure 3: An example of the proposed quantization framework. In each iteration, parameter partition, parameter quantization and incremental re-training are performed sequentially. Green cells represent quantized parameters.

sparsity and repetition existing in CeNN templates provide some additional opportunity for further improvement when implemented in hardware.

# 3 CeNN Quantization and Hardware Implementation

In this section, we present the CeNN quantization framework followed by the details of the hardware implementation.

## 3.1 Incremental Quantization

The proposed incremental quantization framework is an iterative process as shown in Figure 2. Each iteration completes three tasks: parameter partition, parameter quantization, and incremental re-training. We assume that as a starting point, we have all parameters in the original templates before quantization well trained. An illustrative example of the process is shown in Figure 3 to facilitate understanding.

### 3.1.1 Parameter Partition

This task selects a subset of parameters not yet quantized (un-quantized parameters) to perform quantization. Two knobs exist in this task: parameter priority and batch size.

For the first knob, the pruning-inspired (PI) strategy has been well explored in quantization of CNNs [32], based on the consideration that weights with larger magnitudes contribute more to the result and thus should be quantized first. However, the parameters in CeNNs have some unique characteristics which have been discussed in Section 2.3. In order to tackle the problem, we propose a nearest neighbor (NN) strategy and a weighting method for the first knob. The combined weighted nearest neighbor algorithm takes the number that a parameter appears in the template, defined as its repetition quantity (rq) as the reciprocal of the weight, and uses the difference between the parameter and its nearest power-of-two as distance to perform a weighted NN algorithm (WNN). The detail explanation of WNN algorithm is shown in Algorithm 1. Other combinations such as weighted pruning-inspired (WPI) strategy adopt the same weighting method but with PI to form WPI. A total of five strategies PI, WPI, NN (WNN with all weights set to 1), WNN and a random strategy (RAN) are compared in the experimental section.

For the second knob, batch size is the number of parameters selected in each iteration, which will affect re-training speed and quality. We propose to use two batch sizes, constant and log-scale. The former selects the same number of parameters in each iteration, while the latter picks a fixed percentage from the remaining un-quantized parameters, rounded to the nearest integer. Compared with constant batch size, log-scale batch size quantizes more parameters in the first several iterations and fewer towards the end.

**Algorithm 1** Weighted nearest neighbor strategy

**Input:** un-quantized parameters $uq_i$, repeat quantity, $rq_i$, selected quantity, $N$, $1 \leq i \leq n$, $n$, the number of un-quantized parameters
**Output:** the most important $N$ parameters
$neighbor = log_2 |(uq)|$; // get the power of the absolute value of the un-quantized parameters
**for** $i = 1$ **to** $n$ **do**
  $md = (2^{floor(neighbor(i))} + 2^{floor(neighbor(i)+1)})/2$;
  **if** $md > |(uq(i))|$ **then**
    $nnDist(i) = |(uq(i))| - 2^{floor(neighbor(i))}$;
  **else**
    $nnDist(i) = 2^{floor(neighbor(i))+1} - |(uq(i))|$;
  **end if**
**end for**
$wnnDist = nnDist/rq$;
sort $wnnDist$ in ascending order;
output the first $N$ parameters;

### 3.1.2 Parameter Quantization

Before parameter quantization, the bit width should be defined first according to applications. Note that there are millions of parameters for CNN, and short bit width is always appreciated considering memory and computational consumption. However, CeNN usually has tens to hundreds of parameters (time-variant templates have more parameters than time-invariant templates), and bit width has no significant impact on memory consumption. In addition, with power-of-two conversion multiplications can be done with logic shifts, and bit width will also have little impact on computation complexity. The only impact it will have is on the resource utilization of multipliers.

Suppose the quantization set is designed as depicted in Equation (7), where $k$ and $m$ indicate the range of quantization. The corresponding bit width $bw$ is calculated as shown in Equation (8), where the extra one bit is the sign bit.

$$qs = \{\pm(2^k, ., 2^p, ., 2^m), 0\}, \;\; k \leq p \leq m, \;\; p, k, m \in \mathbb{Z}. \tag{7}$$

$$bw = Ceiling[log_2(2 \times (m - k + 1) + 1)] + 1. \tag{8}$$

With the quantization set, a parameter $uq(i)$ is quantized as shown in Equation (9). When the absolute value of a parameter is smaller than $2^{-k-1}$, it will become zero after quantization and get pruned. Lower bit width can prune more parameters, at the cost of accuracy loss.

$$uq(i) = \begin{cases} 2^p & \text{if } 3 \times 2^{p-2} \leq |uq(i)| < 3 \times 2^{p-1}; \\ & \quad k \leq p \leq m; \\ 2^m & \text{if } |uq(i)| \geq 2^m; \\ 0 & \text{if } |uq(i)| < 2^{-k-1}. \end{cases} \tag{9}$$

### 3.1.3 Incremental Re-training Algorithm

Usually, re-training algorithm is an optimal problem as shown in Equation (10), where $P$ is the set of all the parameters. In incremental re-training algorithm, the optimal problem is revised as shown in Equation (11), where $U$ and $Q$ are the sets of un-quantized and quantized parameters, respectively. $a_i$ and $b_i$ are the lower and upper bounds for both $P_i$ and $U_i$, respectively. Note that $P = Q \cup U$, and $U \cap Q =$ . In each iteration, a subset of $U$ will be quantized and added to $Q$.

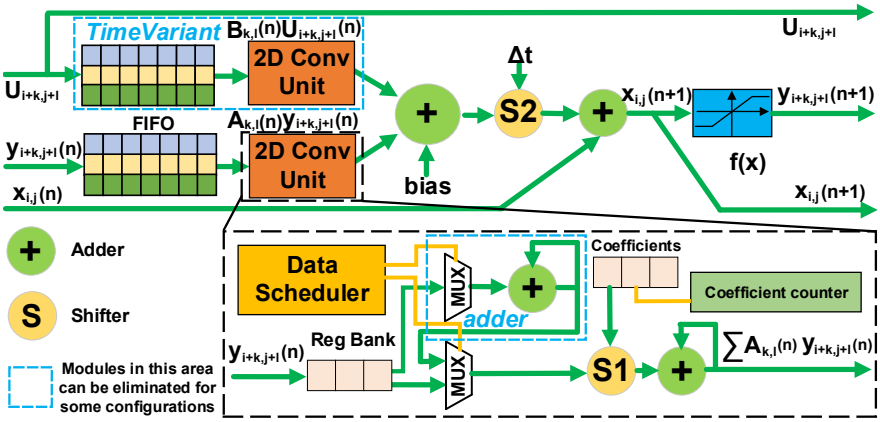



Figure 4: Architecture of the optimized stage design.

Table 1: Comparison of resource utilization between 18-bit multipliers implemented using shifter modules of various configurations $S1(m)$ and $S2(m)$ (with different $m$ as defined in Equation (7), $k$=-$m$ for $S1$, and $k$=0 for $S2$) and a direct implementation of an 18-bit multiplier using LEs and registers.

| MODULE | $S1(0)$ | $S1(1)$ | $S1(2)$ | $S1(3)$ | $S1(4)$ | $S1(5)$ | $S2(7)$ | *Multiplier* |
|---|---|---|---|---|---|---|---|---|
| LEs | 39 | 44 | 50 | 80 | 109 | 105 | 80 | 676 |
| REGISTERS | 39 | 42 | 45 | 47 | 50 | 52 | 75 | 486 |

$$f = min\ obj(P),\ s.t.\ P_i \in [a_i, b_i], 0 \leq i \leq |P|. \tag{10}$$

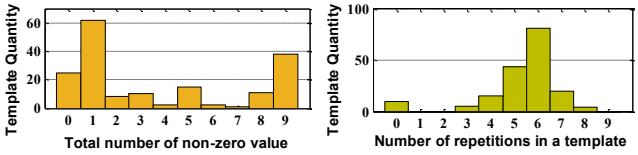
$$f = min\ obj(U, Q), s.t. U_i \in [a_i, b_i], 0 \leq i \leq |U|. \tag{11}$$

$Q$ will be fixed during the re-training process and only $U$ is used for space searching. After multiple iterations, all the required parameters are quantized. It should be noted that the bias $I(n)$ in Equation (4) for CeNN is not required to be quantized as it is not involved in multiplication. Therefore, another re-training iteration is required for the optimal bias when all the required parameters are quantized.

## 3.2 Efficient Hardware Implementations

We base our work on the state-of-the-art FPGA CeNN implementations [18][30][31], which is expandable, highly parallel and pipelined. The basic element of the architecture is the stage module which handles all the processes in one iteration corresponding to Equation (4) for $1 \leq i \leq M$, $1 \leq j \leq N$. Multiple stages are connected sequentially for multiple iterations to form a layer, which processes the input in a pipelined manner. Furthermore, multiple layers can be connected sequentially for more complex processing or be distributed in parallel for a higher throughput. Note that First In First Out (FIFO) are used between adjacent stages to store the temporary results of each stage (or each iteration), and they are configured as single-input multiple-output memories. Please refer to FPGA implementations in [18][30] for more details.

Our efficient hardware implementation focuses on the optimization of the stage design as shown in Figure 4. Two optimizations are performed: multiplication simplification and data movement optimization. First, with incremental quantization, simplification can be achieved by replacing multiplications with shift operations. The detailed hardware implementation will be discussed in Section 3.2.1. Second, when FPGA resource is extremely limited (e.g. for low-end FPGAs), data movement optimization can be performed utilizing the sparsity and repetition in CeNN templates. As will be discussed later in Section 3.2.2, in many applications CeNN templates naturally involves zero or repeated parameters. With incremental quantization, more zeros are yielded leading to higher sparsity and the small




(a) (b)

Figure 5: Illustration of (a) sparsity and (b) repetition characteristic with 174 CeNN templates.

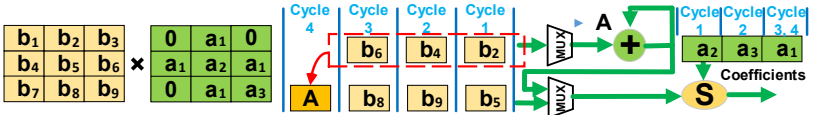



Figure 6: Illustration of sparsity-induced and repetition-induced optimizations.

quantization set introduces a larger number of repetitions. Data movement optimization can minimize the number of computations needed. The details will be discussed in Section 3.2.2.

The optimized stage can be configured for both time-invariant templates and time-variant templates. Note that the FPGA implementation [30] is dedicated to CeNN with time-invariant templates, while [18] is for time-variant. The *TimeVariant* part in Figure 4 is specific for time-variant templates, and can be eliminated in the configuration for time-invariant ones.

### 3.2.1 Shifter Module

In Figure 4, shifter $S1$ is for multiplications in CeNNs and $S2$ is for discrete approximation involved with $\Delta t$ in Equation (4). Usually $\Delta t$ is very small, and the hardware implementation of $S2$ in this paper is designed to support $\Delta t$=$2^s$, where $-7 \leq s \leq 0$, $s \in \mathbb{Z}$. Note that when $\Delta t$ is configured to $2^0$ or 1, the computation is transformed to discrete CeNN [9].

Table 1 provides an illustrative comparison of resource utilization between multipliers implemented using shifter modules of various configurations and a direct implementation of multiplier using LEs and registers. It can be noticed that the shifter module consumes much fewer resources than the general implementation, such that more multiplications can be placed on FPGAs for higher performance and speed. It should be pointed out that multiple shifters can be adopted in the 2D convolutional module.

### 3.2.2 Data Scheduler Module

Data scheduler module exploits the sparsity and repetition of parameters in CeNN templates. We analyzed 87 tasks from 79 applications [12], and totally 174 templates are examined (each task has two templates: template $A$ and template $B$). All the templates are 2D 3×3 each having nine parameters. The corresponding sparsity and repetition are shown in Figure 5(a). In Figure 5(a), we discover that a majority of templates have zero values, and more than half have only three or less non-zero parameters. Therefore, ignoring multiplications with zeros will give a significant improvement in efficiency.

Figure 5(b) depicts the histogram of the parameter repetition in all the 174 templates. We can see that in most of the templates, about 5-6 parameters are repeated values. With repeated parameters, we can also take advantage of the associative law for repetition-induced optimization, e.g., $a_1 \times b_1 + a_1 \times b_2 + a_1 \times b_3 = (b_1 + b_2 + b_3) \times a_1$, and hence three multiplications are optimized to only one.

Note that these optimizations seem to be straightforward and automatic in software synthesis, but for hardware implementations detailed attention is needed. An illustration of optimization with sparsity and repetition is shown in Figure 6. With sparsity-induced optimization, we only take the non-zero parameters into consideration, and three multiplications can be eliminated. An adder (only consumes 10 LEs in the design) is utilized to calculate the sum $A$ of $b_2$, $b_4$ and $b_6$ in parallel with the shifter module. The shifter module calculates $b_5 \times a_2$, $b_9 \times a_3$, and $b_8 \times a_1$ in the first three cycles, and computes $A \times a_1$ in the forth. Thus, totally it takes four cycles rather than nine cycles to calculate Equation (8). Specifically, sparsity-induced optimization reduces the computation time from nine cycles to six, and repetition-induced optimization reduces it from six to four.

Table 2: Configuration of PSO algorithm.

| $N$ | $c_1$ | $c_2$ | $w$ | $iteration$ | $min_d$ | $max_d$ |
|---|---|---|---|---|---|---|
| 10 | 1.4 | 1.2 | 0.8 | 500 | $-2^m$ | $2^m$ |

The power of sparsity-induced and repetition-induced optimizations varies with different applications. Note that if the number of shifters adopted in the 2D convolution module is larger than one, repetition-induced optimization can be eliminated as it contributes much less compared with the shifters. If the number of shifters equals that of the coefficients which is also the situation to achieve the highest throughput, repetition-induced optimization can also be eliminated as all multiplications can be processed in only one cycle. Therefore, the two optimizations are only for situations with very limited resources.

# 4 Experiments

In this section, we first evaluate the performance of various incremental quantization strategies discussed in Section 3 for two CPS applications: medical image segmentation for telemedicine and obstacle detection for ADAS. Then we implement the quantized CeNNs on FPGAs and compare their speed with state-of-the-art works.

## 4.1 Performance Evaluation

### 4.1.1 Experimental Setup

For incremental quantization, a total of 10 incremental quantization strategies are evaluated: five partition strategies (RAN, PI, WPI, NN (WNN with all weights set to 1), and WNN) in combination with two batch sizes (constant and log-scale). For compact presentation, we use postfix -C and -L to denote constant and log-scale batch sizes, respectively. For constant batch size, we set the size to 20% of the total parameters. While for log-scale batch size, we set it to half of the remaining un-quantized parameters. We discuss five quantization set sizes with $m$ =0, 1, 2, 3, 4 and $k = -m$ (see Equation (7)).

The parameters of PSO algorithm in Equation (6) is shown in Table 2. The object function designed according to applications will be discussed in the following sections.

### 4.1.2 Medical Image Segmentation for Telemedicine

The objective function for medical image segmentation in PSO re-training is shown in Equation (12), where $output$ and $IdealOutput$ are output images of CeNN processing on input images and desired output images, respectively, and $t$ is the number of training pairs, and $area$ is the product of the width and height of the image. We also adopts the objective function as accuracy to evaluate the quality of segmented images. The pattern structures of the 3×3 templates $A$ and $B$ are as follows: $A = \{a_0, a_1, a_2; a_3, a_4, a_3; a_2, a_1, a_0\}$, and $B = \{a_5, a_6, a_7; a_8, a_9, a_8; a_7, a_6, a_5\}$. The dataset is from the mammographic image analysis society (MIAS) digital mammogram database [27], and two images and its corresponding segmented results are selected as training images as shown in Figure 7, which is the of the same configuration with the work [25]. Totally 119 test images are used in the experiment. Note that as there is no ideal output in the MIAS database, the outputs of the template with double precision are regarded as the ideal outputs.

$$obj = accuracy = \sum_{i=1}^{t} abs(output_i - IdealOutput_i)/area. \tag{12}$$

We fix the quantization size using $m$ = 2 and $k = -m$, and evaluate all 10 incremental quantization frameworks. The results are shown in Figure 10(a). We can notice that the quantized templates achieve similar accuracy compared with the original template without quantization. The lowest accuracy is about 12% lower than that with the original templates. The highest accuracy is achieved with WNN-C strategy, which is only 3% lower than that of the original templates. Note that generally PI strategy achieves the best performance for CNNs [32]. However, WNN strategy obtains the best performance for CeNN, and NN strategy also obtains a comparable performance. We can also find

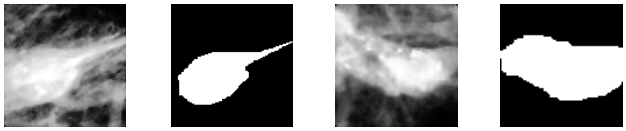
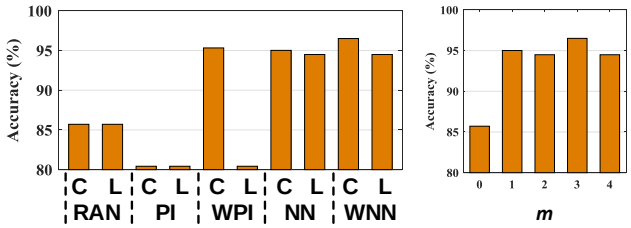



Figure 7: Two selected images and their manually segmented images taken from MIAS database to train CeNN.




(a) $m$=2, $k$=-2 (b) NN-L, $k$=-$m$

Figure 8: Performance comparison between templates with various (a) strategies and (b) quantization sizes $m$ for biomedical image segmentation.

that NN and WNN strategy are much stable than PI as NN and WNN can achieve almost the same accuracy for constant and logscale batch sizes while PI not. Even random strategy can have a better accuracy than PI in some configurations. In terms of batch size, constant seems to perform better than log-scale in most cases. It can be interesting in the future to study this in more detail and figure out a systematic way to decide the optimal strategy.

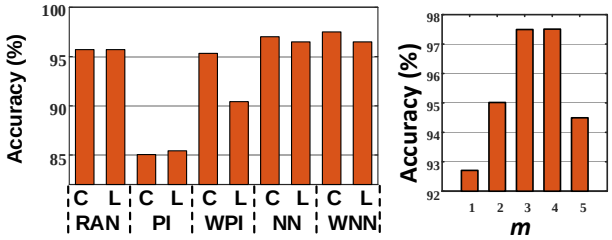
The impact of batch sizes is presented in Figure 10 with the optimal partition WNN-C. The quantization set size has an interesting relationship with the performance. First, even when the quantization set is only of three values (-1, 0, 1), the quantized template can still achieve high accuracy Second, there exists an optimal $m$ which gives the best performance and $m$=3 for medical image segmentation. Further increasing $m$ will not provide any performance gain.

### 4.1.3 Obstacle Detection for ADAS

We adopts the same objective function as medical image segmentation. The pattern structures of the 3×3 templates $A$ and $B$ are as follows: $A = \{a_0, a_0, a_0; a_0, a_1, a_0;\ a_0, a_0, a_0\}$, and $B = \{a_2, a_2, a_2; a_2, a_3, a_2; a_2, a_2, a_2\}$. The training dataset is from [33] as shown in Figure 9, which is the of the same configuration with the work [25]. For test dataset, totally 40 test images are selected from Hlevkin test images collection [34].

We fix the quantization size using $m = 2$ and $k = -m$, and evaluate all 10 incremental quantization frameworks. The results are depicted in Figure 10(a). From the figure we can observe that the quantized templates achieve similar accuracy compared with the original template without quantization. The lowest accuracy is about 15% lower than that with the original templates. Like medical image segmentation, the highest accuracy for obstacle detection is achieved with WNN-C strategy, which is only 3% lower than that of the original templates, and constant strategy also performs better than log-scale in most cases. The impact of batch sizes is presented in Figure 10 with the optimal partition WNN-C. The quantization set size has the same relationship with the performance as that for medical image segmentation.


Figure 9: Training image and the manually detected image taken from [33].




(a) $m$=2, $k$=-2 (b) NN-L, $k$=-$m$

Figure 10: Performance comparison between templates with various (a) strategies and (b) quantization sizes $m$ for obstacle detection.

## 4.2 Speed Evaluation Using FPGAs

In previous section we have evaluated the performance of our CeNN quantization framework in terms of accuracy. In this section we will evaluate its speed when implemented in FPGAs. For a fair comparison with existing works [18][30][31], we adopt the same configurations of stages and try to place the maximum possible number of stages utilizing our quantized templates. Note that all the three works share the same architecture for CeNN computation. The performance of the implementation is evaluated by equivalent computing capacity which is the product of number of stages and the computing capacity of each stage. The proposed efficient hardware implementation is implemented on an XC4LX25 FPGA. The data width of the input, state, and output ($u$, $x$, and $y$) is configured to be 18 bits. The widely-used template size $3\times3$ is adopted. Note that general CeNN is adopted for the FPGA implementation, and delayed CeNN is not considered here. Time-variant templates are configured. In the implementation, multiplication is achieved with embedded multipliers (more specifically, DSP48 modules on XC4LX25 FPGAs) at first, and shifters are used when there are no more available embedded multipliers. Considering the routability of FPGAs, the utilization rate of LEs and registers are constrained to be no higher than 80%. Note that since different quantization frameworks only affects the performance and do not show significant difference in hardware resource utilization, in this part of experiments we simply use WNN-L with m=5 and k=-5, and other frameworks should yield almost identical speed.

Three configurations of 2D convolution are discussed: one, three and nine multipliers. In Table 3, applying our quantization framework can lead to a 1.2x speedup with increased use of LEs (by 17%) and registers (by 8%) This allows an additional 4 stages to be placed, with a speedup of 1.2x.

Further taking sparsity-induced optimization into consideration, a speedup of 1.8x is achieved in the 2D convolution module with computations involving with template $A$ for binary image noise cancellation. However, no sparsity exists in template B, and there is no overall speedup, as sparsity-induced optimization can only yield speedup when sparsity exists in both templates A and B. Therefore, the speedup still remain about the same. Yet after the introduction of repetition-induced optimization, the speedup can be further increased to 1.4x with slightly reduced resource usage (due to the reduction of computations needed). Note that these conclusions are application-specific. Similar conclusions reside with texture segmentation. The proposed architecture achieves a little lower clock frequency due to the high resource utilization making placement and routing relatively more difficult.

For the configuration of 2D convolution with multiple multipliers, sparsity-induced and repetition-induced optimizations doing very limited optimizations with multiple multipliers are not involved. As shown in Table 4, the the state-of-the-art work [31] has a very low resource utilization (2%-15%) with LEs and registers. With the abundant resources, 10 and 5 more stages can be placed on FPGAs with shifters as a replacement of multipliers for the implementation configured with three and nine multipliers, respectively, resulting in a speedup of 2.6x and 3.5x.

As the CeNN architecture composed with stage modules are highly extensible, we make a reasonable projections to high-end FPGAs to see how the resources available in an FPGA affect the speedup. According to existing implementations on FPGAs and resource constraint of 80% LE and register utilization rate bound, the clock frequencies are assumed to be the same in the comparison. The configuration of 2D convolution with nine multipliers is adopted, which has the highest performance. We select four high-end FPGAs from Altera and Xilinx with about 500,000 to 1,000,000 LEs. As shown in Table 5, our implementations can achieve a speedup of 1.7x-7.8x. Note that the resource consumption of LEs and registers are almost the same for all the implementations, and the speedup

Table 3: Speed and resource utilization comparisons of the state-of-the-art work [31] and ours with **one multiplier (Mult.)/shifter (Shif.)** in 2D convolution module, with sparsity-induced optimization and repetition-induced optimization. The numbers in the brackets are the resource utilization rate.

| Implementation | State-of-the-art (1 Mult.) | Ours (1 shif.) | Ours (1 shif.+ sparsity) | Ours (1 shif.+ repetition) |
|---|---|---|---|---|
| # of stages | 24 | 28 | 28 | 24 |
| LEs ($\times 10^3$) | 14.6(60%) | 18.7(77%) | 18.7(77%) | 18.4(76%) |
| Register($\times 10^3$) | 8.8(40%) | 10.5(48%) | 10.5(48%) | 9.9(46%) |
| Embedded Mult. | 48(100%) | 48(100%) | 48(100%) | 48(100%) |
| Clock F. (MHz) | 353 | 331 | 331 | 322 |
| Cycles per pixel | 11 | 11 | 11 | 8 |
| Speedup | 1 | **1.2x** | **1.2x** | **1.4x** |

Table 4: Speed and resource utilization comparisons of the state-of-the-art work [31] and ours with **three and nine multipliers(Mult.)/shifter (Shif.)** in 2D convolution module. The numbers in the brackets are the resource utilization rate.

| Implementation | State-of-the-art (3 Mult.) | Ours (3 Shif.) | State-of-the-art (9 Mult.) | Ours (9 Shif.) |
|---|---|---|---|---|
| # of stages | 6 | 16 | 2 | 7 |
| LEs($\times 10^3$) | 3.8(15%) | 19.6(80%) | 1.4(5%) | 18.2(76%) |
| Registers($\times 10^3$) | 2.1(10%) | 6.5(30%) | 0.6(2%) | 3.6(17%) |
| Embedded Mult. | 48(100%) | 48(100%) | 46(95%) | 48(100%) |
| Clock F.(MHz) | 337 | 320 | 361 | 343 |
| Cycles per pixel | 5 | 5 | 1 | 1 |
| Speedup | 1 | **2.6x** | 1 | **3.5x** |

varies with the number of embedded multipliers, or more specifically, the ratio of LEs to embedded multipliers. A high ratio of LEs to embedded multipliers means more LEs can be used to implement shifters resulting with a high speedup. The highest speedup of 7.8x is due to the fact that the Stratix V E FPGA has the highest rate of LEs to embedded multipliers.

## 5 Conclusions

In this paper, we propose CeNN quantization for computation reduction for CPS, particularly telemedicine and ADAS. The powers-of-two based incremental quantization adopts an iterative procedure including parameter partition, parameter quantization, and re-training to produce templates with values being powers of two. We propose a few quantization strategies based on the unique CeNN computation patterns. Thus, multiplications are transformed

Table 5: Speed and resource utilization projections to high-end FPGAs of the state-of-the-art work [31] and ours with **nine multipliers/shifters** in 2D convolution module. The numbers in the brackets are the resource utilization rate.

| Implementation | VC7VX-980T | VC7VX-585T | Stratix V E | Stratix V GS |
|---|---|---|---|---|
| # of stages | 352 | 179 | 233 | 291 |
| LEs($\times 10^3$) | 780(80%) | 465(80%) | 718(80%) | 524(80%) |
| Registers($\times 10^3$) | 170(17%) | 93(16%) | 133(15%) | 128(19%) |
| Embedded Mult. | 3600(100%) | 1260(100%) | 704(100%) | 3926(100%) |
| Speedup | **2.3x** | **3.3x** | **7.8x** | **1.7x** |

to shift operations, which are much more resource-efficient than general embedded multipliers. Furthermore, based on CeNN template structures, sparsity-induced and repetition-induced optimizations for quantized templates are also exploited for situations where resources are extremely limited. Experimental results on medical image segmentation for telemedicien and obstacle detection for ADAS show that the proposed quantization framework can achieve similar performance compared with that using original templates without optimization, and the implementation with incremental quantization can achieve a speedup up to 7.8x compared with the state-of-the-art FPGA implementations. We also discover that unlike CNNs, the optimal strategy of CeNNs is weighted nearest neighbor strategy other than pruning-inspired strategy.

# References


[1] Khaitan et al., Design Techniques and Applications of Cyber Physical Systems: A Survey, IEEE Systems Journal, 2014.

[2] S. J. Carey, D. R. Barr, B. Wang, A. Lopich, and P. Dudek. Mixed signal simd processor array vision chip for real-time image processing. *Analog Integrated Circuits and Signal Processing*, 77(3):385–399, 2013.

[3] S.-H. Chae, D. Moon, D. G. Lee, and S. B. Pan. Medical image segmentation for mobile electronic patient charts using numerical modeling of iot. *Journal of Applied Mathematics*, 2014, 2014.

[4] H.-C. Chen, Y.-C. Hung, C.-K. Chen, T.-L. Liao, and C.-K. Chen. Image-processing algorithms realized by discrete-time cellular neural networks and their circuit implementations. *Chaos, Solitons & Fractals*, 29(5):1100–1108, 2006.

[5] L. O. Chua and T. Roska. *Cellular neural networks and visual computing: foundations and applications*. Cambridge university press, 2002.

[6] 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. *arXiv preprint arXiv:1602.02830*, 2016.

[7] F. Dohler, F. Mormann, B. Weber, C. E. Elger, and K. Lehnertz. A cellular neural network based method for classification of magnetic resonance images: towards an automated detection of hippocampal sclerosis. *Journal of neuroscience methods*, 170(2):324–331, 2008.

[8] M. Duraisamy and F. M. M. Jane. Cellular neural network based medical image segmentation using artificial bee colony algorithm. In *Green Computing Communication and Electrical Engineering (ICGCCEE), 2014 International Conference on*, pages 1–6. IEEE, 2014.

[9] H. Harrer and J. A. Nossek. Discrete-time cellular neural networks. *International Journal of Circuit Theory and Applications*, 20(5):453–467, 1992.

[10] H. Harrer, J. A. Nossek, T. Roska, and L. O. Chua. A current-mode dtcnn universal chip. In *Circuits and Systems, 1994. ISCAS'94., 1994 IEEE International Symposium on*, volume 4, pages 135–138. IEEE, 1994.

[11] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems*, pages 4107–4115, 2016.

[12] K. Karacs, G. Cserey, Zarndy, P. Szolgay, C. Rekeczky, L. Kek, V. Szab, G. Pazienza, and T. Roska. Software library for cellular wave computing engines. *Cellular Sensory and Wave Computing Laboratory of the Computer and Automation Research Institute*, 2010.

[13] S. Lee, M. Kim, K. Kim, J.-Y. Kim, and H.-J. Yoo. 24-gops 4.5-$mm^2$ digital cellular neural network for rapid visual attention in an object-recognition soc. *IEEE transactions on neural networks*, 22(1):64–73, 2011.

[14] H. Li, X. Liao, C. Li, H. Huang, and C. Li. Edge detection of noisy images based on cellular neural networks. *Communications in Nonlinear Science and Numerical Simulation*, 16(9):3746–3759, 2011.

[15] M. Magdalena and U. G. B. Bujnowska-Fedak. Use of telemedicine-based care for the aging and elderly: promises and pitfalls. *Smart homecare Technology & telehealth*, 3:91–105, 2015.

[16] D. Manatunga, H. Kim, and S. Mukhopadhyay. Sp-cnn: A scalable and programmable cnn-based accelerator. *IEEE Micro*, 35(5):42–50, 2015.

[17] G. Manganaro, P. Arena, and L. Fortuna. *Cellular neural networks: chaos, complexity and VLSI processing*, volume 1. Springer Science & Business Media, 2012.

[18] J. J. Martnez, J. Garrigs, J. Toledo, and J. M. Ferrndez. An efficient and expandable hardware implementation of multilayer cellular neural networks. *Neurocomputing*, 114:54–62, 2013.

[19] J. Muller, R. Wittig, J. Muller, and R. Tetzlaff. An improved cellular nonlinear network architecture for binary and greyscale image processing. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2016.

[20] R. Porter, J. Frigo, A. Conti, N. Harvey, G. Kenyon, and M. Gokhale. A reconfigurable computing framework for multiscale cellular image processing. *Microprocessors and Microsystems*, 31(8):546–563, 2007.

[21] S. Potluri, A. Fasih, L. K. Vutukuru, F. Al Machot, and K. Kyamakya. Cnn based high performance computing for real time image processing on gpu. In *Nonlinear Dynamics and Synchronization (INDS) & 16th Int'l Symposium on Theoretical Electrical Engineering (ISTET), 2011 Joint 3rd Int'l Workshop on*, pages 1–7. IEEE, 2011.

[22] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.

[23] Xiaowei Xu, Qing Lu, Tianchen Wang, Jinglan Liu, Cheng Zhuo, Sharon Hu, and Yiyu Shi. Empowering Mobile Telemedicine with Compressed Cellular Neural Networks. In *Proc. of IEEE/ACM 2017 International Conference On Computer-Aided Design*. IEEE/ACM, 2017.

[24] A. Rodrguez-Vzquez, G. Lin-Cembrano, L. Carranza, E. Roca-Moreno, R. Carmona-Galn, F. Jimnez-Garrido, R. Domnguez-Castro, and S. E. Meana. Ace16k: the third generation of mixed-signal simd-cnn ace chips toward vsocs. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(5):851–863, 2004.

[25] R. Rouhi, M. Jafari, S. Kasaei, and P. Keshavarzian. Benign and malignant breast tumors classification based on region growing and cnn segmentation. *Expert Systems with Applications*, 42(3):990–1002, 2015.

[26] H. Song, P. Jeff, T. John, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *4th International Conference on Learning Representations*, 2016.

[27] J. Suckling, J. Parker, D. Dance, S. Astley, I. Hutt, C. Boggis, I. Ricketts, E. Stamatakis, N. Cerneaz, S. Kok, et al. The mammographic image analysis society digital mammogram database. In *Exerpta Medica. International Congress Series*, volume 1069, pages 375–378, 1994.

[28] H. Wong, V. Betz, and J. Rose. Comparing fpga vs. custom cmos and the impact on processor microarchitecture. In *Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, pages 5–14. ACM, 2011.

[29] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016.

[30] N. Yildiz, E. Cesur, K. Kayaer, V. Tavsanoglu, and M. Alpay. Architecture of a fully pipelined real-time cellular neural network emulator. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 62(1):130–138, 2015.

[31] N. Yildiz, E. Cesur, and V. Tavsanoglu. On the way to a third generation real-time cellular neural network processor. *CNNA 2016*, 2016.

[32] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *5th International Conference on Learning Representations*, 2017.

[33] D. Feiden and R. Tetzlaff. Obstacle detection in planar worlds using cellular neural networks. In *Cellular Neural Networks and Their Applications, 2002.(CNNA 2002). Proceedings of the 2002 7th IEEE International Workshop on*, pages 383–390. IEEE, 2002.

[34] Hlevkin. http://www.hlevkin.com/06testimages.htm, 2017.

# Challenges in Automatic Deployment and Configuration Management in Cyber Physical System

Devki Nandan Jha[1], Yinhao Li[1], Prem Prakash Jayaraman[2], Saurabh Garg[3], Paul Watson[1], Rajiv Ranjan[1]

[1]School of Computing, Newcastle University
[2]Department of Computer Science and Software Engineering, Swinburne University of Technology
[3]School of Computing and Information Systems, University of Tasmania

## 1 Introduction

With the massive computation and communication capabilities provided by the cloud and edge datacentres, Cyber Physical System (CPS) creates a new way to interact between physical and computation systems. In addition to computation and communication technology, it also depends on control system, electronics and electrical engineering, chemical and biological advancements and other new design technologies to give a better interaction among these technologies. Rise of CPS revolutionizes the way of living by influencing the mankind in numerous ways e.g. smart healthcare, smart home, smart manufacturing, smart city, smart traffic, smart agriculture, etc.

CPS is defined as an interdisciplinary approach that combines computation, communication, sensing and actuation of cyber system with the physical system to perform time-constraint operations in an adaptive and predictive manner [1, 2, 3]. Here, feedback loops are associated with the physical system that helps embedded computers and networks to control and monitor physical processes. It helps in revising the design technique based on the previous design model and feedback from the physical system.

There are three main components of CPS: Cyber Component, Physical Component and Network Component. Physical component does not have any computation and communication capability. Various examples of physical components includes chemical process, mechanical machinery, biological process or human being. Cyber component consists of IoT devices, Edge Data Centre (EDC) and Cloud Data Centre (CDC). IoT devices acts as a bridge between physical and cyber components that collects the data from diverse physical sources (e.g. environment, communication media, business transactions, social media, etc.) and send to edge and cloud for further processing. EDC is considered to be the collection of smart devices capable of performing functions like storage or computation of diverse data at a smaller level e.g. Raspberry Pi, esp8266, etc. while CDC considers all the private, commercial and public cloud providers serving software, platform, storage, etc. in the form of on-demand services e.g. AWS, Microsoft Azure, etc. Network Component of CPS is involved in all the communication either between physical components and cyber components or among the cyber component. The interaction between all the components is shown in the Figure 1.

CPS solutions are typically application specific and are deployed and configured on the basis of hardware heterogeneity (e.g. sensors, actuators, gateways, SDN controllers, datacentres, etc.), communication protocols (standard or specific, connection less or connection oriented, etc.), data processing models (batch processing, stream processing, etc.) and data storage models (SQL, NoSQL, etc.). The deployment environments of one application differs from other application and there is no standard service management procedures available for all the applications. Managing and handling such type of system requires extensive knowledge of all the integrant technology, which is not possible for a user as the user in CPS are considered to be either non-technical or with little technical knowledge. This requires an autonomic system that performs all the deployment and configuration management procedure for a user. A user only needs to provide the requirement specification and necessary constraints using the interface (e.g. command line interface (CMI) or application program interface (API) or software defined kit (SDK), etc.) and remaining process is done automatically by the system.

## 2 Motivation

The motivation of this paper comes from the challenges that we experienced while deploying smart solutions in different application domains. Consider an example of smart home in CPS system that makes our life more convenient

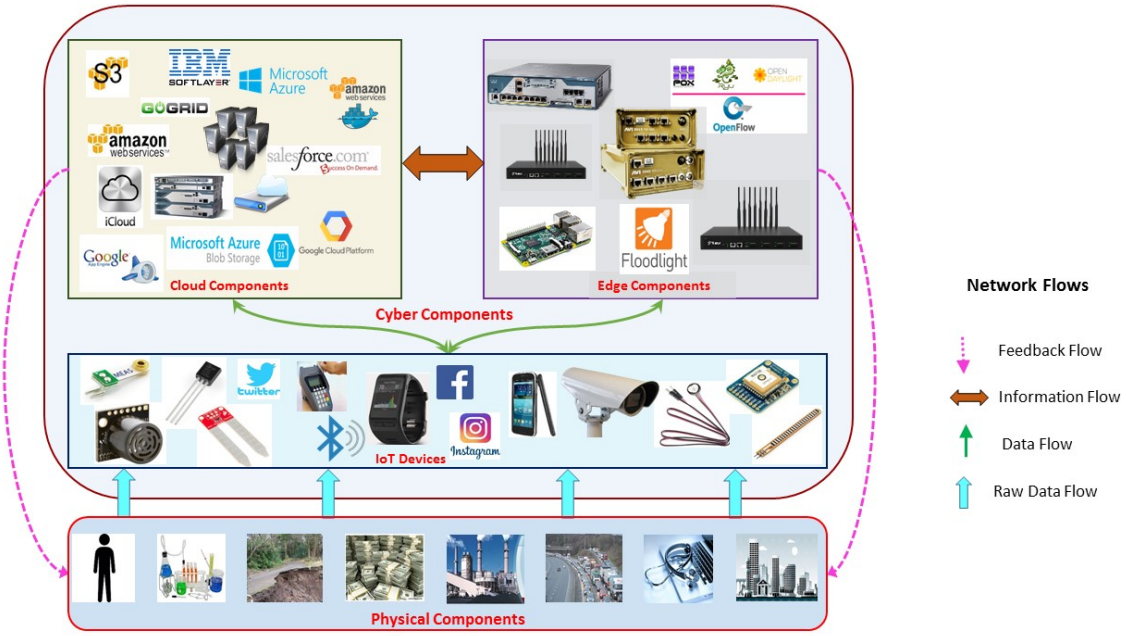



Figure 1: Components of CPS System.

and comfortable. My phone will tell the heater or air conditioner to turn on and make the house warm/cool before I reach home, the virtual assistant (e.g. Gate box, Amazon Alexa, etc.) tell me to take umbrella before I leave for office and many other things.

The processing and control of such system looks very simple but the actual operations are very complex. To do so sensors and actuators are embedded in the physical devices like refrigerators, air conditioner, heating and lighting devices so that they can communicate with a central controller entity. The embedded sensors are continuously capturing the raw data e.g. temperature of the room for air conditioner or heater, weather information for updating about rain, etc. Decision about making the heater or air-conditioner on is done by analysing the GPS data from phone and the temperature sensor data from the house along with the stored data about the time taken to make the house warm/cool. The decisive process is performed in either edge or cloud datacentre as the IoT devices does not have that much capacity to store and process diverse data. Different IoT devices have different communication standards, which makes it complicated to communicate with each other. Edge devices (e.g. Gateway, etc.) can eliminate this problem as it can easily be installed and communicate with different devices. These edge devices receives the data from different IoT devices and operates according to the demand of the application. For real time constraint applications, it performs some data analytic operations and notifies the IoT devices to perform accordingly while for other applications, it extracts the data and send it to the cloud for further storage and processing. The heavy data storage and processing is performed in Cloud and the result is send back to the IoT devices via edge devices. There may be intermediate information exchange between cloud and edge devices depending upon some factors like resource availability, time constraints, etc. The actuators embedded in the physical devices can react and respond according to the output of edge or cloud and controls the physical devices. A feedback is also send from the cloud or edge device that supervises the physical devices for self-configuration and self-adaptation.

To perform all these operations, the application handler should must be expert in all the involved technology so that he/she can deploy and manage the complex requirements of CPS applications. As the CPS application uses IoT devices, edge and cloud, the handler must know how to configure and manage data in all these environments. To provide the best service, it is not enough to select an optimal cloud resource or edge resource. It is important that the cloud, edge and IoT devices are synchronized together to provide the better service with maximum resource utilization. Given the variety of devices at each level as shown in Figure 1, it is not possible to manually configure the whole system by writing one script that manages everything. In addition to this, the continuous updation and upgradation of execution environment makes the management process more complex. It requires some solution that

performs all these operations automatically. In automated environment, user only needs to state the requirements and constraints and the remaining operations can be performed automatically.

## 3 Factors effecting Deployment and Configuration management in CPS

The complexity of data analytic activities in cloud, edge and IoT environment makes the deployment and configuration management process very challenging. To facilitate this process, we are presenting the technical dimensions that provide an intuitive view about the elements affecting the deployment and configuration management in CPS system. The factors are explained below.

a) **Dependency Graph**: Dependency graph is a directed acyclic graph that represents the dependencies of several nodes (objects/resources) towards each other. One can easily and explicitly represents the data and control flow among the nodes by using program dependency graph (PDG), which can be used to support user interaction, parameterization of models, optimization decisions, consistency checks, easy debugging and other operations. Any change in the dependency can easily be represented by adding, removing or reconfiguring them on PDG. It is easy to represent the complex dependency and requirement specification of CPS by using the dependency graph. Here, the node represents an application/service whereas the edges represents the dependency i.e. how the nodes are dependent on each other. This is also helpful for creating some similar application as we can reuse the same dependency graph with little modification.

b) **Access Mechanism**: Access mechanism signifies the methods about how to interact with the services provided by the CPS system. This is very important as it provides an abstraction for different types of users e.g. application developers, technical experts or DevOps managers accessing from different type of devices e.g. personal computers, mobile phones, tablets, etc. and facilitates the easy interaction with the CPS. There are numerous access mechanisms available e.g. command line interface (CLI), application program interface (API), software development kit (SDK), graphical user interface (GUI), etc. These mechanisms have different properties, as GUI is easier to use but have significant delay as compared to command line interface. The choice of access mechanism depends on various factors such as application support, device support, user technical knowledge, etc.

c) **Access Control**: In information technology, access control is a process by which users have granted access and privileges to systems, resources or information in a computing environment. However, application in cyber-physical system, unlike traditional applications, usually do not have well-defined security perimeters and are dynamic in nature. Traditional access control policies and mechanisms rarely address these issues and are thus inadequate for CPS. Access control for CPS depends on factors like trustworthiness of entities, environmental contexts, application contexts, etc. The overarching theme between two types of access control (physical and cyber) depends on the notion of trust. Access control models must take into account environmental factors before making access decisions.

d) **Extensibility**: Extensibility is defined as the ability to describe a system that can easily be extended or expanded from its initial state. It is an important characteristic of any software, application or programming language, which makes it adaptable for execution in frequently changing environment. Extensibility can be supported either by add-ons, plugins, packages or hooks that adds some additional functionalities or by explicitly adding macros or functionality directly to the applications. Technology is changing at a fast rate so it is necessary for the CPS application to be extensible that is how it can be adapted for a new environment.

e) **Customization**: Customization or personalization is defined as the characteristic that allows a user to customize an application based on their specific requirements. The requirements of each user is unique that may not necessarily fits with the other user application or default available applications. If the system allows a user to customize their own application, it is better from both user and system perspective as all the requirements of users' are satisfied and exact systems' resources being used reducing any resource wastage.

f) **Reusability**: Reusability refers to the reuse of existing software artefacts. There are some modules which are common in multiple applications. One way for this is to define the same modules separately for each application while the other way is to define the module only once and reuse the same module remaining times. There are a number of existing modules that can either be used directly or be used after little modification. There are some essential requirements for software/ products to be reused, some of them are consistency, adaptability, stability, flexibility and complexity. Tools like Docker have some storage repository (Docker Hub) that stores numerous

existing container. User can easily pull the image from this repository. These images can either be used directly or easily updated to satisfy some specific requirements.

g) **Deployment Environment**: Deployment environments refers to the system(s) responsible for proper execution of an application. The environment provides all the necessary resources to start, execute and stop the application. Based on the resource type and configuration, deployment environment can be categorized into on premise system, edge datacentre and cloud datacentre. Cloud datacentre is again divided into public, private or hybrid cloud. The application can be deployed on the basis of different qualitative and quantitative QoS parameters e.g. resource requirements, cost, deadline, security, etc.

h) **Virtualization Techniques**: Virtualization is the key concept of cloud computing that partitions the physical resources (e.g. compute, storage, network, etc.) into multiple virtual resources [3]. It allow multiple users to access the services provided by the cloud in an isolated manner. Mainly two types of virtualization are common in cloud perspective, hypervisor-based virtualization (e.g. Xen, KVM, etc.) and container-based virtualization (e.g. Docker, LXC, etc.). In hypervisor-based virtualization, a hypervisor is running on bare hardware machine (Type 1 hypervisor) or on host operating system (Type 2 hypervisor), but, each virtual machine must have their own separate operating system. In container-based virtualization, a container engine is added to top of host operating system that is shared by all the containers using Linux features namespace and cgropus. Cloud resources are virtualized using both hypervisor and container based virtualization while edge datacentres considers mainly containers as a virtualization mechanism. Applications running on hypervisors and containers are not compatible to each other making the deployment task in CPS complicated.

i) **Scalability**: Scalability is defined as an ability to accommodate increasing number of workload by increasing the capacity of the system. It is an important characteristic that guarantees the application can perform well with increasing workloads. Scalability can be performed either vertically (scale up) or horizontally (scale out). The main difference between them is in the way they add additional resources with increasing workload. In vertical scalability, the capacity of the node is increased by adding more resources (e.g. CPU, memory, disk, etc.) to it whereas in horizontal scalability, we can add as many independent systems (equivalent to the existing system) parallel to the existing systems to satisfy the increased workload. Scalability can be achieved through a software framework like dynamically loaded plug-ins, top-of-the-line design interfaces with abstract interfaces, useful callback function constructs, a very logical and plausible code structure, etc.

j) **Portability**: Portability is defined as the characteristic of an application that allows it to execute successfully on a variety of environment. Portability is the capability attribute of a software product, and its behaviour is manifested to a degree, and the degree of performance is closely related to the environment. The basic idea of portability is to provide an abstraction between system and application, which is possible only if the application is loosely coupled with the system architecture. It also reduces the burden of redefining the same application for different environment, which increases the reusability of the application.

Several tools have been developed by the both industry and research communities e.g. Chef [4], Puppet [5], TOSCA [6], Cloud Formation [7], Calvin [8], etc. to automate the deployment and configuration management tasks in cloud and edge environments. Some of the tools can also have support for IoT applications. Users only need to specify the requirements and these tools perform the remaining operations automatically. Frameworks such as Chef, Puppet, etc. configure and deploys the infrastructure but does not virtualizes the infrastructure. Tools like Docker [9] and Kubernetes [10] virtualizes only in the form of containers whereas Juju [11] and Cloud Formation supports both container and hypervisor-based virtualization. Tools like TOSCA and Calvin have support for IoT devices along with cloud and edge environment but they needs more development. None of the available tools can support the complicated data analytic activity of CPS while satisfying the requirements in terms of discussed dimensions.

## 4 Conclusion

Deployment and configuration management task in CPS is crucial as it need to handle the heterogeneous components (IoT devices, Edge and Cloud datacentres) in a comprehensive manner. Managing the data analytic activities across heterogeneous CPS infrastructure is very challenging because it considers the complex data and control dependencies along with infrastructure capabilities. Increasing popularity of CPS applications demand a sophisticated solution for

deployment and configuration management. In this paper, we identified a set of factors that affects the configuration of data analytics and its physical deployment in CPS system. A new framework is strongly warranted that handles the complexity of data analytic activity while automating the deployment and configuration management in CPS.

## References


[1] Wollman D. A., Reference Architecture for Cyber-Physical Systems, available online: https://www.nist.gov/programs-projects/reference-architecture-cyber-physical-systems, accessed on: 25 May 2017.

[2] Cyber-Physical Systems, available online: http://cyberphysicalsystems.org/, accessed on: 22 Jan 2018.

[3] Xing, Y. and Zhan, Y., 2012. Virtualization and cloud computing. In Future Wireless Networks and Information Systems (pp. 305–312). Springer Berlin Heidelberg.

[4] Chef, Chef Documentation Overview, available online: https://docs.chef.io/chef_overview.html, accessed on: 22 Jan 2018.

[5] Puppet, Puppet 4.10 reference manual, available online: https://docs.puppet.com/puppet/4.10/index.html, accessed on: 22 Jan 2018.

[6] Binz, T., Breitenbucher, U., Haupt, F., Kopp, O., Leymann, F., Nowak, A. and Wagner, S., 2013, December. OpenTOSCA: a runtime for TOSCA-based cloud applications. In International Conference on Service-Oriented Computing (pp. 692-695). Springer Berlin Heidelberg.

[7] AWS Cloud Formation, available online: https://aws.amazon.com/cloudformation/, accessed on: 22 Jan 2018.

[8] Calvin, Open Source release of IoT app environment Calvin, available online: https://www.ericsson.com/research-blog/cloud/open-source-calvin/, accessed on: 22 Jan 2018.

[9] Docker, Docker Documentation, available online: https://docs.docker.com/, accessed on 22 Jan 2018.

[10] Kubernetes, Kubernetes Documentation, available online: https://kubernetes.io/docs/home/, accessed on 22 Jan 2018.

[11] Juju, Getting started with Juju, available online: https://jujucharms.com/docs/stable/getting-started, accessed on: 22 Jan 2018.

# A New Planning Framework for Self-driving Vehicles against Emergencies

Long Chen[1], Xuemin Hu[2] and Wulin Huang[3]

[1]School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510275, China
[2]School of Computer Science and Information Engineering, Hubei University, Wuhan 430062, China
[3]Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China

**Abstract**

To enhance the safety in self-driving vehicles, a novel parallel planning framework is proposed in this paper. In the proposed framework, we first construct artificial scenes based on real traffic scenes, and then develop a deep planning model which can learn from both real and artificial scenes and use it to make planning decisions. To handle emergencies, a generative adversarial network (GAN) model which learns from artificial emergencies obtained from artificial traffic scenes is presented. When the self-driving vehicle is travelling on roads, the GAN model is used to generate multiple virtual emergencies based on the current real scene, and the planning model simultaneously makes different planning decisions for the virtual scenes. Thanks to parallel planning, the self-driving vehicles can timely make decisions without amounts of calculations when an emergency occurs.

## 1 Introduction

Self-driving technology has received great interest from academia and in industrial R&D departments for over decade. Decision making of self-driving vehicles is the key to autonomy using planning algorithms [1]. Traditional planning methods, including A-Star, rapidly-exploring random tree, discrete optimization-based method, and their variants, have made great achievement in the fields of intelligent robots and autonomous driving [2, 3, 4]. Nevertheless, these method are designed through making all kinds of rules based on some mathematical models, so these methods cannot be adapted to the scenes beyond the rules. In addition, only current scene is considered in existing planning methods, and potential events are ignored, which mean the planner will not have enough time to reach if an accident occurs.

Deep neural networks (DNNs) have shown their outstanding ability in the fields of feature extraction, object classification and complex scene understanding in recent years, but the performance of DNNs heavily rely on big and various training data. However, obtaining big data for training a deep planning model for self-driving vehicles, especially emergency samples, is almost an impossible process in reality. In recent years, the parallel driving framework has been stably developed [5]. An important part of the parallel framework is parallel vision, which was proposed to deal with the issue of lacking a large number of training samples in DNNs [6]. This theory which is highly appreciated by researchers creates a framework for vision perception in complex environment driven by the big data in physical and cyber spaces, combined with the technologies of computer graphics, virtual reality, machine learning, and knowledge automation.

For making self-driving vehicles drive as human drivers and handle emergencies, we propose a new framework called “parallel planning" in this paper. The word “parallel" has two meanings. One is that the model can learn from both real and artificial scenes, and the other is that the planning method can make decisions for observations in both the real scenes and the virtual emergencies generated from the current observations.

## 2 Framework Overview

The proposed framework of parallel planning is shown in Figure 1. An artificial society is built according to a real society where various of traffic scenes are designed. Then, a deep planning model that can simultaneously learn driving strategies and knowledge from both real and artificial traffic scenes is developed. The self-driving vehicle based on the proposed method is a data-driven end-to-end motion planner. The model understands and analyzes the scenes whose information is obtained from the vehicle-mounted sensors, while the human driver’s operations are provided as the desired planning strategies when training in real traffic scenes. When training in artificial traffic

scenes, the model understands and analyzes the scenes whose information is obtained from the virtual system, while the expert demonstrations are provided as the desired behaviors. The aim is not only to replicate the provided human's driving strategies or expert demonstrations in a specific scene as in a teach-and-repeat method, but also to learn collision avoidance strategies and also transfer the acquired knowledge to previously unseen real-world environment and virtual emergencies. In the stage of observation, the model is evaluated by comparing the planning parameters from planner's driving strategies and human's driving strategies in real traffic scenes. Meanwhile, a GAN model which learns from artificial emergencies obtained from artificial traffic scenes is presented. When the self-driving vehicle is travelling on roads, the GAN model is used to generate multiple virtual emergencies based on the current real scene, and the planning model simultaneously makes different planning decisions for the virtual scenes.

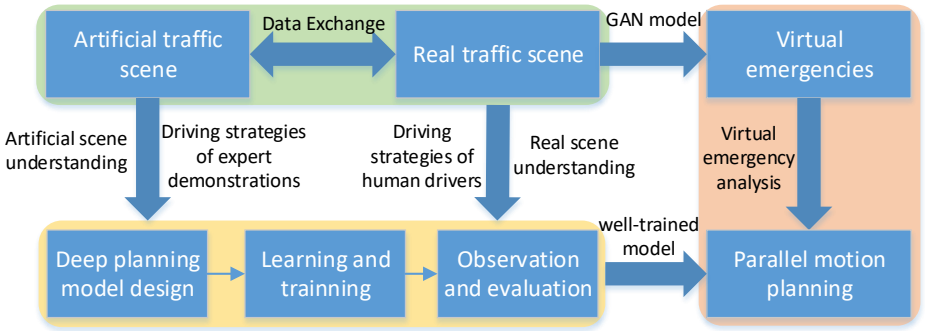



Figure 1: Framework of parallel planning

## 3 Artificial Scenes

In the artificial society, traffic scenes like real scenes are created, where the possible traffic conditions and environment are simulated. The information of precise labels is obtained automatically, and a large number of various samples are acquired from these scenes. The artificial traffic society can be seen as a virtual world which is created by the technologies of computer games, computer graphics, virtual reality, and microscopic simulation, and these technologies can be implemented using open source, commercial softwares and simulation tools [6], such as Unity, 3DS MAX, OpenGL, Google 3D Warehouse, Microsoft AirSim, Euro Truck Simulator 2, and Grand Theft Auto V (GTA5).

An artificial traffic scene is composed of many factors, including static objects, moving objects, season, whether, and light, etc. Most of moving objects such as vehicles, pedestrians and bicycles are moving in the designed artificial scene following traffic rules. However, emergencies are set to occur randomly in order to train the performance of the planning model against emergences because a reliable autonomous vehicle should make a rational decision and guarantee the safety of passengers and pedestrians even though an emergency occurs. Owing to the designed artificial traffic scenes, infinite traffic data can be provided to train the deep planning model, which meets the data requirements on largeness and variety.

## 4 End-to-End Planning Model

For making self-driving vehicles have the capabilities of perceiving environment, extracting important information and making rational decision, the planning method should learn driving knowledge and from the environment. A model is designed to describe the relationships between the observations and the decisions to make. The model is trained using the perceived information and the corresponding labels in both real and artificial scenes. In the testing stage, the model can be used to make rational decisions when new perceived information is available. To build the

relationships between the observations and the planning decisions, we try to find a function as (13).

$$c = f(o, m, \theta) \tag{13}$$

where $f$ represents the deep model, and $\theta$ is the weight parameter vector in this model. $o$ is the observations, and $m$ represents the other measurements such as the global goal, map and traffic information. $c$ is the parameter vector of the desired control decisions. The purpose of training the model is to find the parameter vector $\theta$ which can optimally explain the training data and the corresponding labels.

## 5 Planning against Emergencies

An emergency is easy to lead to an accident because existing planning systems cannot react and compute decisions in a very short time. We propose a novel parallel planning method against emergencies. The planning model makes different planning decisions for the virtual potential emergencies which are generated by a generative model.

We use the video clips of artificial emergencies in artificial scenes to train a generative model. The video clips are generated by the generative model based on the objects such as vehicles, pedestrians, and bicycles, etc., which are detected from the real traffic scenes. The well-trained planning model generates multiple planning decision candidates according to the current observations and the virtual potential emergency events. The final planning decision is made through analyzing the current scene and the virtual scenes.

Generating virtual potential emergencies based on the current real scene is significant for the proposed parallel planning method. Emergencies in both artificial and real traffic scenes can be presented using video clips, so we consider to use video generation methods to generate virtual emergencies. Since the generative adversarial network has outstanding capabilities in image generation [7], we employ the GAN as the generative model and train it using artificial emergency video clips which can be obtained from artificial scenes. After training and evaluating the generative model, we use the generator of the GAN model to generate virtual emergencies based on the observations in the current scene. To generate virtual potential emergency video clips, the current observations which are composed of image sequences of the detected objects are input into the well-trained generative model, and the well-trained planning model simultaneously makes different decision candidates for every virtual potential emergency. The final planning decision is made by the discriminator of the GAN model.

## 6 Conclusions

To handle emergencies and enhance safety for self-driving vehicles, we propose a parallel planning framework. Artificial scenes are first constructed based on the real scenes. A new deep neural network for motion planning which can learn from both artificial and real scenes is presented and used to make planning decisions from observations. To timely react when emergencies occur, a GAN model is designed and learn from the artificial emergencies. When the self-driving vehicle is travelling, the GAN model generates multiple potential virtual emergencies based on the current observations. Simultaneously, the well-trained planning model makes different decisions for the virtual potential emergency scenes. The proposed parallel planning framework is intended to treat emergencies for self-driving vehicles and extend the parallel theory to the planning field.

## Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grant 61773414.

## References

[1] S. Zhang, W. Deng, Q. Zhao, H. Sun, and B. Litkouhi, "Dynamic trajectory planning for vehicle autonomous driving," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2013, pp. 4161–4166.

[2] L. Zuo, Q. Guo, X. Xu, and H. Fu, "A hierarchical path planning approach based on a* and least-squares policy iteration for mobile robots," *Neurocomputing*, vol. 170, no. C, pp. 257–266, 2015.

[3] R. Cui, Y. Li, and W. Yan, "Mutual information-based multi-auv path planning for scalar field sampling using multidimensional rrt*," *IEEE Transactions on Systems Man and Cybernetics Systems*, vol. 46, no. 7, pp. 993–1004, 2016.

[4] X. Hu, L. Chen, B. Tang, D. Cao, and H. He, "Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles," *Mechanical Systems and Signal Processing*, vol. 100, pp. 482–500, 2018.

[5] F. Wang, N. Zheng, D. Cao, C. M. Martinez, L. Li, and T. Liu, "Parallel driving in cpss: A unified approach for transport automation and vehicle intelligence," *IEEE/CAA Journal of Automatica SINICA*, vol. 4, no. 4, pp. 577–587, 2017.

[6] K. F. Wang, C. Gou, and F. Y. Wang, "Parallel vision: An acp-based approach to intelligent vision computing," *Acta Automatica Sinica*, vol. 42, no. 10, pp. 1490–1500, 2016.

[7] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Advances in Neural Information Processing Systems*, vol. 3, pp. 2672–2680, 2014.

# 1 Conferences and Workshops

- 1st IEEE Conference on Energy Internet and Energy System Integration
- Asian Hardware Oriented Security and Trust Symposium (AsianHOST 2017)

# 2 Special Issues in Academic Journals

- Proceedings of the IEEE Special Issue on Design Automation for Cyber-Physical Systems

# 3 Book Publications

- Springer Book "Leveraging Big Data Techniques for Cyber-Physical Systems"

# Call for Contributions

## Newsletter of Technical Committee on Cyber-Physical Systems (IEEE Systems Council)

The newsletter of Technical Committee on Cyber-Physical Systems (TC-CPS) aims to provide timely updates on technologies, educations and opportunities in the field of cyber-physical systems (CPS). The letter will be published twice a year: one issue in February and the other issue in October. We are soliciting contributions to the newsletter. Topics of interest include (but are not limited to):

- Embedded system design for CPS
- Real-time system design and scheduling for CPS
- Distributed computing and control for CPS
- Resilient and robust system design for CPS
- Security issues for CPS
- Formal methods for modeling and verification of CPS
- Emerging applications such as automotive system, smart energy system, internet of things, biomedical device, etc.

Please directly contact the editors and/or associate editors by email to submit your contributions.

### Submission Deadline:

All contributions must be submitted by **July 1st, 2018** in order to be included in the August issue of the newsletter.

### Editors:

- Helen Li, Duke University, USA, hai.li@duke.edu

### Associate Editors:

- Long Chen, Sun Yat-Sen University, China, chenl46@mail.sysu.edu.cn
- Wuling Huang, Chinese Academy of Science, wuling.huang@ia.ac.cn
- Yier Jin, University of Florida, USA, yier.jin@ece.ufl.edu
- Abhishek Murthy, Philips Lighting Research, USA abhishek.murthy@philips.com
- Rajiv Ranjan, Newcastle University, United Kingdom, raj.ranjan@ncl.ac.uk
- Yiyu Shi, University of Notre Dame, USA, yshi4@nd.edu
- Bei Yu, Chinese University of Hong Kong, Hong Kong, byu@cse.cuhk.edu.hk
- Qi Zhu, University of California at Riverside, USA, qzhu@ece.ucr.edu