Characterizing the Execution of Deep Neural Networks on Collaborative Robots and Edge Devices*

Matthew L. Merck†* Georgia Tech
Bingyao Wang* Georgia Tech
Lixing Liu* Georgia Tech
Chunjun Jia* Georgia Tech
Arthur Siqueira Georgia Tech
Qiusen Huang Georgia Tech
Abhijeet Saraha Georgia Tech
Dongsuk Lim Georgia Tech
Jiashen Cao Georgia Tech
Ramyad Hadidi‡ Georgia Tech
Hyesoon Kim Georgia Tech

ABSTRACT

Edge devices and robots have access to an abundance of raw data that needs to be processed on the edge. Deep neural networks (DNNs) can help these devices understand and learn from this complex data; however, executing DNNs while achieving high performance is a challenge for edge devices. This is because of the high computational demands of DNN execution in real-time. This paper describes and implements a method to enable edge devices to execute DNNs collaboratively. This is possible and useful because in many environments, several on-edge devices are already integrated in their surroundings, but are usually idle and can provide additional computing power to a distributed system. We implement this method on two iRobots, each of which has been equipped with a Raspberry Pi 3. Then, we characterize the execution performance, communication latency, energy consumption, and thermal behavior of our system while it is executing AlexNet.

CCS CONCEPTS

• **Computing methodologies** → *Machine learning*; *Distributed computing methodologies*; • **Computer systems organization** → *Embedded and cyber-physical systems*;

ACM Reference Format:
Matthew L. Merck, Bingyao Wang, Lixing Liu, Chunjun Jia, Arthur Siqueira, Qiusen Huang, Abhijeet Saraha, Dongsuk Lim, Jiashen Cao, Ramyad Hadidi, and Hyesoon Kim. 2019. Characterizing the Execution of Deep Neural Networks on Collaborative Robots and Edge Devices. In *Practice and Experience in Advanced Research Computing (PEARC '19), July 28-August 1, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3332186.3333049

*This work is supported by NSF CSR 1815047.
†Equal contribution to the paper.
‡All corresponds to rhadidi@gatech.edu.

PEARC '19, July 28-August 1, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-7227-5/19/07…$15.00
https://doi.org/10.1145/3332186.3333049

1 INTRODUCTION & MOTIVATION

With the recent advancement of Deep Neural Networks, we are now able to solve many previously challenging problems [24, 30, 34], such as Computer Vision [23, 36], Neural Machine Translation [3, 6], and Video Recognition [35]. At the same time, several physical domains are undergoing fast changes due to such advancements; some examples include robots [9, 13, 16, 31], Unmanned Aerial Vehicles (UAVs) [29, 37], and Internet-of-Things (IoT) devices [12, 33]. With the widespread applicability and benefits of DNNs, the fast execution of DNNs on devices that are characterized by tight resource constraints and tight real-time requirements is crucial. In fact, performing DNN computations on the edge is rapidly gaining grounds due to privacy concerns [4, 21, 25, 27]and unreliable connection of conducting computation on the cloud, strict real-time resource requirements, and increased opportunity for personalization. However, DNN advancement comes with the issue of increased requirements for computational power on devices where they are executed [18] and this rapid increase is not expected to slow down.

DNN execution on the edge is especially challenging for low-performance robots and IoT devices. However, these devices are also a perfect candidate for DNN computation because of their immediate access to local raw data (e.g. input from cameras and sensors). Although the results from processing this raw data could be extremely beneficial, single devices lack the computational power to carry out the DNN computation. Currently, users need to upload collected data to cloud services to carry out any intensive computation [11, 26], but the high-sensitivity of some data (e.g. recordings of home security cameras) raises considerable privacy concerns [4, 21, 25, 27]. As a result, it is crucial to enable these low-performance devices to carry out intensive DNN computations.

There have been endeavors to address the high requirements of DNNs on a *single device*, such as weight pruning [18, 28, 39, 40] and quantization [7, 10, 22, 38]. However, in this paper, we focus on collaboration between such devices. The reasons for this are: (1) In several scenarios, there are many devices that are already integrated within their surroundings, such as smart home cameras, autonomous vehicles with several sensors, or wide area networks with a variety of devices. (2) These devices are idle most of the time, and they contain embedded processors with unused computing power. Therefore, to achieve faster execution of DNNs, we can distribute the computations of a single inference (i.e. prediction)

on several devices. To do so, we use the model-parallelism techniques introduced by Hadidi et al. [12, 13, 15] to collaboratively execute AlexNet [23], an image-recognition DNN model, on two iRobots [20]. Our system, as shown in Figure 1, uses two iRobots that are each equipped with an additional Raspberry Pi 3 [8], which represents an edge device. To understand the effect of executing DNNs on the edge, we characterize several behaviors of DNN execution in our system. In detail, we examine execution performance, communication latency, energy consumption, and thermal behavior of our system.

2 EXPERIMENTS

2.1 Experimental Setup

Two iRobot Roomba 600s [20] are used as our robots. Each of the robots is equipped with one Raspberry Pi 3 [8], the specifications of which are shown in Table 1. The power source of each Raspberry Pi 3 is derived from the iRobot's battery with a voltage converter. The Raspberry Pi, which is connected to the iRobot's serial port, uses the iRobot Create 2 Open Interface [19] to control the iRobot. In addition to controlling the robot, we utilize each Raspberry Pi as our computing engine for executing DNNs. On each Raspberry Pi, with the Ubuntu 16.04 operating system, we use Keras 2.0 [5] with the TensorFlow 1.0 [1] backend. As shown in Figure 1, we use a USB digital multimeter, which records measurements to an excel file once every second, to measure the power consumption of the Raspberry Pi. In order to measure the power consumption of one robot, we use the iRobot Open Interface to retrieve battery voltage and amperage once every 100 ms while performing our experiments. Each experiment for energy measurement takes around 3 minutes to finish, which includes some idle time to display the baseline. Experiments for measuring communication latency are done for 10 minutes, during which the latency of each data packet is recorded.

Table 1: The specification of Raspberry Pi 3 [8].

CPU	1.2 GHz Quad Core ARM Cortex-A53
Memory	900 MHz 1 GB RAM LPDDR2
GPU	No GPGPU Capability
Price	$35 (Board) + $5 (SD Card)

2.2 Energy & Performance

To understand how DNN distribution affects the power consumption of the Raspberry Pi, in our first set of experiments, we measure the power usage of each Raspberry Pi. In one experiment, as shown in Figure 2, we measure the power consumption rate of a Raspberry Pi while it is executing the entire AlexNet model. To show the change in the average power consumption, we also include a measurement for a certain length of idle period in our graph. As shown in the graph, the average rate of power consumption is 1.95 W during the execution of DNN. The performance of one Raspberry Pi measured running AlexNet is 1.25 inferences per second. As a result, during our 3-minute-long (180 seconds) experiment period, approximately 200 inferences are made. In order to show the difference in power consumption rate of the distributed approach, we execute AlexNet on two Raspberry Pis and measure the rate of power consumption on one of the two Raspberry Pis. This time, the execution is performed by dividing the AlexNet model into two parts and executing each part on one Raspberry Pi. Figure 3 illustrates the trend in rate of power consumption in this experiment. As seen, although the rate of power consumption of the Raspberry Pi in the idle period is the same as that of previous experiment, the average rate of power consumption during the DNN execution period is now 1.53W, less than that of the previous experiment. With two Raspberry Pis computing different parts of the model in parallel, the performance now is approximately 3 inferences per second. However, the rate of power consumption of each single device in this experiment is less than when one device performed all computations. This is because, with the distribution of computing, (i) less computations are performed per device, (ii) fewer memory operations are performed per device, (iii) each device has some idle time as a result of communication latency. Although the power consumption of the total distributed system has been increased (power consumption rate for one device running the whole computing process is 1.95W, versus two devices each consuming power

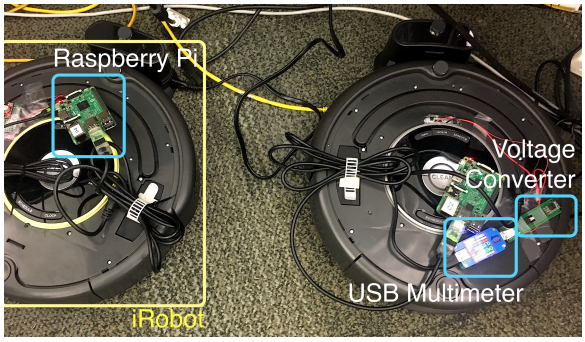

Figure 1: Two iRobot2 equipped with Raspberry Pis.

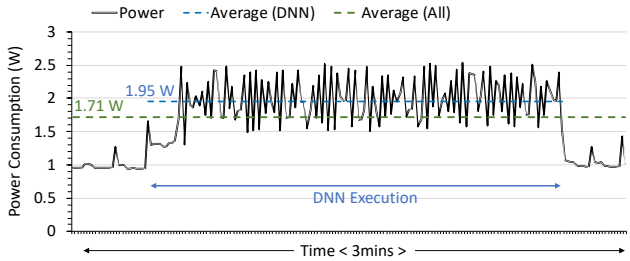

Figure 2: Power consumption of a single Raspberry Pi 3 executing whole AlexNet.

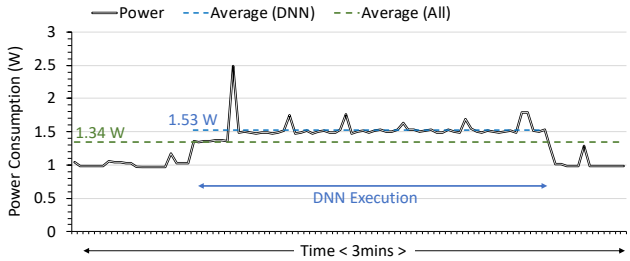

Figure 3: Power consumption of a single Raspberry Pi 3 executing AlexNet in a distributed manner (total of two Raspberry Pis).

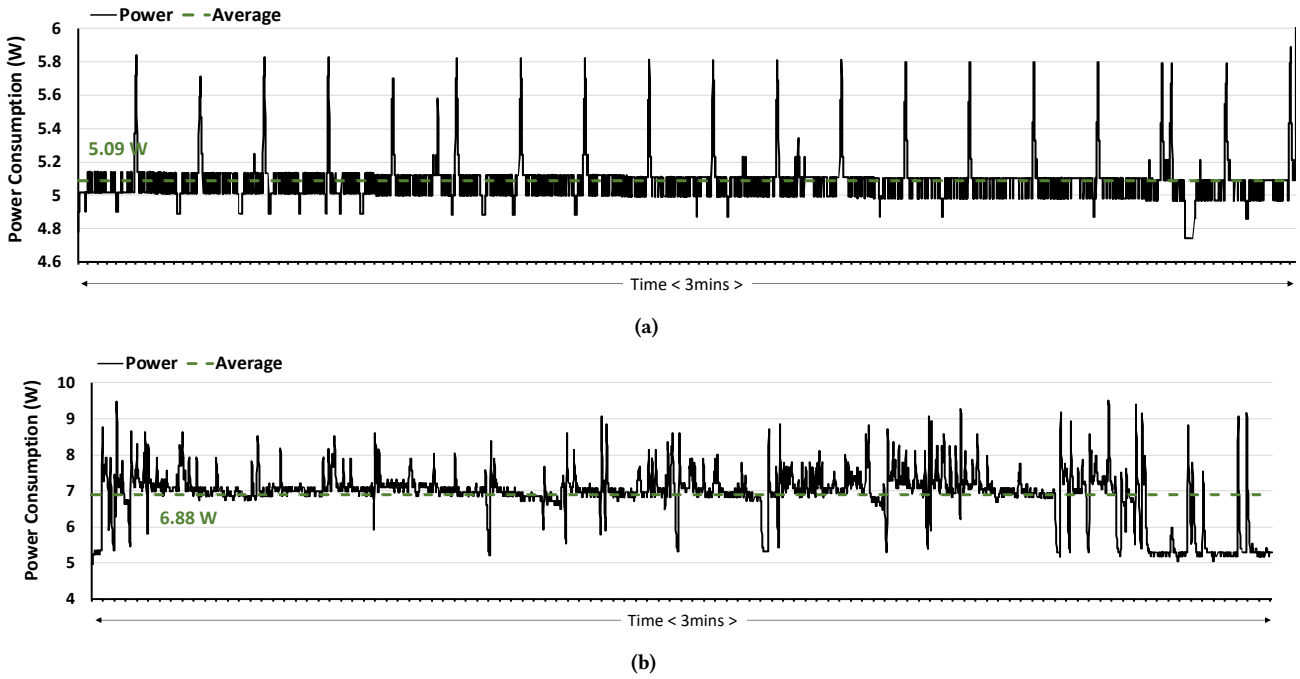

Figure 4: Power consumption of iRobot in (a) idle mode (stationary) and (b) in movement.

at 1.53W, a total of 3.0W), each individual device in the system consumes less power.

To further explore the change in power consumption rate and performance, we studied and now present the results of both the iRobot and Raspberry Pi power consumption in several common cases. As mentioned, each iRobot is equipped with a Raspberry Pi. We measure the power drawn from the iRobot's battery, which consists of the power used for Raspberry Pi computation and the power used for the iRobot's movements. The first case we studied is the idle state (no movement) with no computation. Figure 4a illustrates the power usage behavior in this case, which has an average power consumption of 5.09 W. The frequent spikes (around 0.8 W) in the figure is caused by the iRobot's frequent system checks. To see how physical activity affects power usage, we also measured the power consumption rate of the iRobot while moving, as shown in in Figure 4b. The movement profile is random, depending on the environment. As seen, the average power consumption rate is 6.88 W, around 1.8 W higher than that of the idle state. In addition, there are spikes as large as 3 W in the graph. Compared to the idle case, the trend in power consumption rate in movement is less predictable and contains larger spikes.

To measure how the power consumption rate changes when the robot is executing DNNs, we execute AlexNet on the two Raspberry Pis in our system. Figure 5a illustrates the power consumption rate of one iRobot in stationary mode, while it performs the computation of AlexNet collaboratively with the other iRobot. In this case, the average power consumption is 7.51 W, and there are spikes around 2 W. Although the profile of power consumption was expected to be similar to what we observed in Figure 3, the spikes are much larger than what we hypothesized the idle state would show. We believe this is due to the unreliability of the iRobot's battery in sustaining a constant current to the Raspberry Pi, or alternatively because of our circuitry in converting the voltage. In summary, DNN execution increases the power consumption rate from 6.88 W to 7.51 W, which is a 6% increase. Note that this increase only accounts for the dynamic power consumption during DNN execution. In fact, the addition of a Raspberry Pi increases the static power consumption of the system by approximately 41%, which is derived from the 3.5 W average idle power consumption rate of the iRobot with no Raspberry Pi, in addition to the 1.5 W average power consumption rate of the Raspberry Pi.

Figure 5b shows the power consumption rate of one iRobot in motion while executing AlexNet computations collaboratively. This case is the closest case to a real-world setting, where the robots execute the computations of DNNs in a parallel manner, which indicates that the result produced by a certain robot might depend on the computation results of another device that performed the computation on previous layers of the DNN model. As shown in the figure (Figure 5b), the average power consumption rate is 9.35 W. Compared to the consumption rate of 5.09 W (Figure 4a) in the idle case where no computation happens, there is an 85% increase. Additionally, some spikes are as large as 4.5 W in this experiment. The large spikes suggest that the execution of a DNN while moving caused greater variation in the power consumption rate compared to that in the case where the robot is moving without execution of the DNN (Figure 4b). Such variation (reflected by spikes in the figure) may limit the Raspberry Pi's capability to attain a high performance, because variation of power delivery may lead to discrepancy in power saving settings in its CPU, leading to instability.

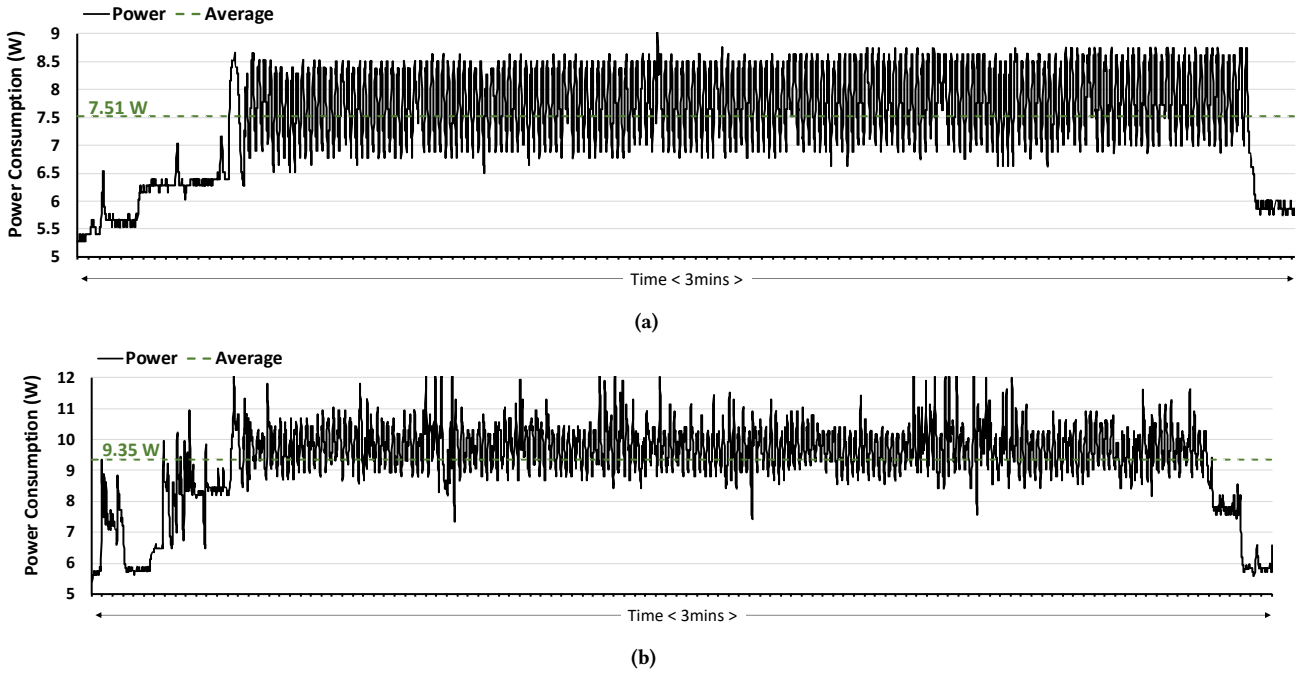

Figure 5: Power consumption of iRobot in (a) idle mode (stationary) while executing AlexNet in distributed manner and (b) in movement while executing AlexNet in distributed manner.

Table 2 summarizes our results about iRobot average power consumption in different cases. All the results include the measurements of the average power consumption rate of both the iRobot and Raspberry Pi. As seen in the table, DNN execution increases the average power consumption rate by around 2.5 W, which is 50% greater than that in the idle case. It also significantly increases the strength of spikes, which indicates greater variation in power consumption rate.

Table 2: iRobot average power consumptions.

Scenario		Average Power Consumption (W)	Spike Strength (W)
~~DNN~~	Idle	5.1	0.8
	Movement	6.9	3.0
DNN	Idle	7.5	2
	Movement	9.4	4.5

To study the performance gain and energy trends of distributed computations on Raspberry Pis, we measure the performance and energy consumption of cases where two, four, or six Raspberry Pis are executing the AlexNet collaboratively. Figure 6 shows the average energy consumption rate and performance in different cases - specifically when two, four or six Raspberry Pis execute the computation collaboratively - divided into 2 categories: static and dynamic (static energy is measured when the robot is not moving, dynamic energy is the inverse). As shown in the chart, we achieve greater performance as the number of devices involved in the parallel computing task increases. There is also a decrease in dynamic energy consumption, similar to the trend shown in Figures 2 and 3. In contrast to dynamic energy consumption rate, there is an

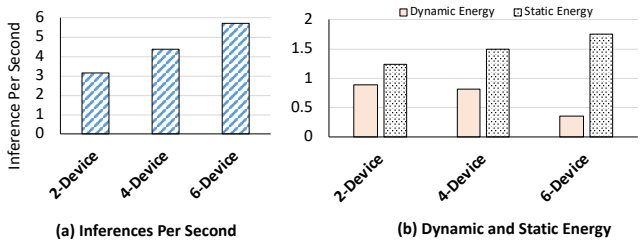

Figure 6: The inferences per second (a) and power consumption (b) of various systems.

increase in average static energy consumption rate. This is because each Raspberry Pi contains certain additional parts that consume extra amounts of energy and are unnecessary for our experiment. As we use more devices, the total number of unnecessary parts increases, so the static energy increases accordingly.

2.3 Communication Latency

Communication latency is one of the important factor that affect the performance of our collaborative DNN computation system. In the model parallelism technique, the model is divided into several parts (e.g. convolutional and fully-connected layers). A typical device's input data depends on the computation result produced by one or several devices which are responsible for the computations of previous layers. The device must wait until the work is done by the devices it depends on, then combine the computed data it receives from the other devices as the input and finally start working based on the processed input. Of course, there is a certain level of communication latency between devices. Latency causes inefficiency in computing; thus, it should be minimized. As a result, in our study,

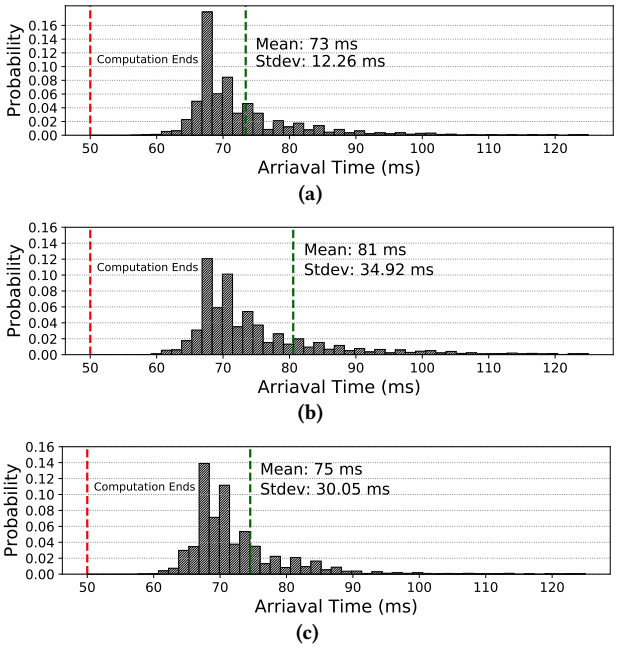

Figure 7: Histogram of communication latency (a) while robot is near the station; (b) while robot is away from the station; (c) while robot is moving.

we conduct experiments to examine the communication latency behaviours in a system with collaborative robots.

We measure the communication latency between a robot and its station under different settings. The robot is performing DNN computations that takes 50 ms. We use a WiFi router with a measured bandwidth of 94 Mbps. We gather the latency data for 10 minutes, and calculate the probability of certain levels of latency measured (in milliseconds). The distribution of communication latency is displayed in the corresponding histograms in Figure 7. In the first case, we measure communication latency for a single robot while it is near the station. As shown in Figure-7a, the mean latency is 73 ms while the standard deviation is 12 ms. Most of the latency time measurements center within 65 to 70 milliseconds, although there are some outliers which are greater than 90 ms. Generally speaking, the latency is approximately 70 ms when the robot is near the station. In the second case, we measure the communication latency while the robot is far from the station. As shown in Figure-7b, the mean latency is 81ms while the standard deviation is around 34 ms, 2 times greater than that of the near-station case. However, most of the latency time gathered centers around 70 ms. It is expected that the variation in latency is greater, because the communication is less stable as the distance increases. In the third case, we measure the communication latency when the robot is moving. As shown in Figure-7c, the mean latency is 75 ms and the standard deviation is around 30 ms, which is still around 2 times greater than that of the near-station case. The high variation in latency time when the robot is moving is caused by frequent changes in the communication environment and distance. Some obstacles might cause unpredictable shifts in latency by obstructing the communication.

To see the aggregated latency of executing a DNN model, we measure the communication latency while five devices run AlexNet [23]

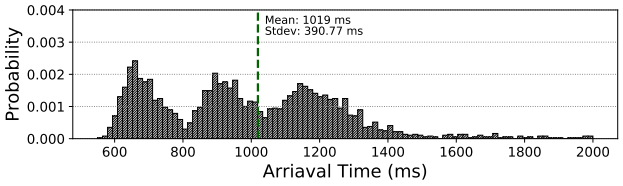

Figure 8: Execution latency histogram on six Raspberry Pi while executing AlexNet collaboratively.

collaboratively. As shown in Figure-8, the mean latency time is 1019 ms, and the standard deviation is approximately 390 ms. In the graph, there are three clusters of data: one centered around 700 ms, one around 900 ms, and one around 1200 ms. That is because the three devices are sharing the computation for a single fully-connected layer. As seen, for an entire model, the amount and range of fluctuations is high. Additionally, the reliability issue is exacerbated by the local networks of these systems, which consist of low-end equipment. Therefore, the real-time latency of DNN execution on edge devices is unreliable and varies greatly. As we observed in Figure-7, this is because of the inherent unpredictability in the latency of WiFi networks. One result of this unpredictability is the occasional severed network connection, which can lead to loss of computation results for a single inference. However, because of the continuous camera input, losing a single inference is acceptable. Additionally, there are methods that can be applied to make inferences more reliable with worsened network connections.

2.4 Device Temperature

To observe the thermal behaviour of edge devices when executing DNNs, we measured the thermal hotspots on the Raspbery Pi 3 with a thermal camera. In Figure 9, we can see the temperatures of our device in different conditions. When the device is off, the temperature of it is around 25.7 Celsius degrees. When the device is on but in an idle condition, the maximum temperature reaches 46.5 Celsius degrees. The hotspots are results of the CPU, DRAM, and LAN chips – this is why the center area produces more heat than the marginal areas. When we perform DNN computations on our device, the temperature rises up to approximately 62.2 Celsius degrees, an increase of 16 degrees Celsius. This temperature increase can correspond to problems with resolution and performance of a connected Raspberry Pi Camera, however it does not lead to decreased performance of the Raspberry Pi itself, and as the results

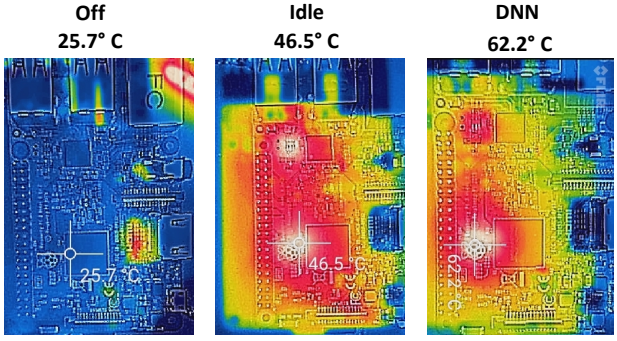

Figure 9: Thermal camera pictures of Raspberry Pi 2 in off, idle, and DNN execution conditions.

have shown, occurs with increases in the inferences per second of the system.

3 CONCLUSION

In this paper, we examined and analyzed the effects of executing deep neural networks collaboratively on edge devices. Our method uses model parallelism to distribute tasks from DNN layers to Raspberry Pis, which are mounted on and powered by iRobots. Our measurements showed that while adding additional devices to a network of edge devices increases static energy, it decreases the dynamic energy used by each device. It also increases the average number of inferences performed by each device. We found that execution of DNNs on a distributed robot system can lead to unpredictable power consumption, which may in turn worsen Raspberry Pi performance. Another unreliable aspect is the network latency, which increases and decreases often based on a variety of often uncontrollable factors. Finally, through our measurement of the Raspberry Pi temperature during DNN execution, we found that DNN computation can increase temperature by as much as 16 degrees Celsius. For future work, we plan to extend our robot system to execution YOLO [32] similar to our demo [17] on several Raspberry Pis using pruning methods [2]. Additionally, as discussed, since distributed computations are susceptible to latency and data loss, we plan to add robust DNN computations [14] in our system.

REFERENCES

[1] Martín Abadi et al. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[2] Bahar Asgari, Ramyad Hadidi, Hyesoon Kim, and Sudhakar Yalamanchili. 2019. LODESTAR: Creating Locally-Dense CNNs for Efficient Inference on Systolic Arrays. *ACM/IEE Design Automation Conference (DAC) - Late Breaking Results, Las Vegas, NV* (2019).

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR'15)*. ACM.

[4] F Biscotti, J Skorupa, R Contu, et al. 2014. The Impact of the Internet of Things on Data Centers. *Gartner Research* 18 (2014).

[5] François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

[6] Ronan Collobert and Jason Weston. 2008. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *ICML'8*. ACM, 160–167.

[7] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2014. Training Deep Neural Networks with Low Precision Multiplication. *arXiv preprint arXiv:1412.7024* (2014).

[8] Raspberry PI Foundation. 2017. Raspberry Pi 3B+. www. raspberrypi.org/products /raspberry-pi-3-model-b/. [Online; accessed 04/01/19].

[9] Alessandro Giusti, Jérôme Guzzi, Dan C Cireşan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. 2016. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters* 1, 2 (2016), 661–667.

[10] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. 2014. Compressing Deep Convolutional Networks Using Vector Quantization. *arXiv preprint arXiv:1412.6115* (2014).

[11] Binita Gupta. 2015. Discovering cloud-based services for iot devices in an iot network associated with a user. US Patent App. 14/550,595.

[12] Ramyad Hadidi, Jiashen Cao, Michael Ryoo, and Hyesoon Kim. 2018. Collaborative Execution of Deep Neural Networks on Internet of Things Device. *arXiv preprint* (2018).

[13] Ramyad Hadidi, Jiashen Cao, Michael S. Ryoo, and Hyesoon Kim. 2018. Distributed Perception by Collaborative Robots. *IEEE Robotics and Automation Letters (RA-L), and International Conference on Intelligent Robots and Systems 2018 (IROS)* 3, 4 (Oct 2018), 3709–3716. https://doi.org/10.1109/LRA.2018.2856261

[14] Ramyad Hadidi, Jiashen Cao, Michael S. Ryoo, and Hyesoon Kim. 2019. Robustly Executing DNNs in IoT Systems Using Coded Distributed Computing. *ACM/IEE Design Automation Conference (DAC) - Late Breaking Results, Las Vegas, NV* (2019).

[15] Ramyad Hadidi, Jiashen Cao, Matthew Woodward, Michael Ryoo, and Hyesoon Kim. 2018. Musical Chair: Efficient Real-Time Recognition Using Collaborative IoT Devices. *arXiv preprint arXiv:1802.02138* (2018).

[16] Ramyad Hadidi, Jiashen Cao, Matthew Woodward, Michael S. Ryoo, and Hyesoon Kim. 2018. Real-Time Image Recognition Using Collaborative IoT Devices. In *Proceedings of the 1st on Reproducible Quality-Efficient Systems Tournament on Co-designing Pareto-efficient Deep Learning (ReQuEST '18)*. ACM, New York, NY, USA, Article 4.

[17] Ramyad Hadidi, Jiashen Cao, Fei Wu, Tushar Kirshna, Michael S. Ryoo, and Hyesoon Kim. 2019. An Edge-Centric Scalable Intelligent Framework To Collaboratively Execute DNN. *Demo for SysML Conference, Palo Alto, CA* (2019).

[18] Song Han, Huizi Mao, and William J Dally. 2016. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. In *4th International Conference on Learning Representations*. ACM.

[19] iRobot Inc. 2019. iRobot Create 2 Open Interface. www.cdn-shop.adafruit.com/datasheets create_2_Open_Interface_Spec.pdf. [Online; accessed 15/03/19].

[20] iRobot Inc. 2019. iRobot Create 2 Programmable Robot. www.irobot.com/about-irobot/stem/create-2. [Online; accessed 15/03/19].

[21] Rafiullah Khan, Sarmad Ullah Khan, Rifaqat Zaheer, and Shahid Khan. 2012. Future Internet: The Internet of Things Architecture, Possible Applications and Key Challenges. In *FIT'12*. IEEE, 257–260.

[22] Urs Köster, Tristan Webb, Xin Wang, Marcel Nassar, Arjun K Bansal, William Constable, Oguz Elibol, Scott Gray, Stewart Hall, Luke Hornof, et al. 2017. Flexpoint: An adaptive numerical format for efficient training of deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*. 1742–1752.

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet Classification With Deep Convolutional Neural Networks. In *26th Annual Conference on Neural Information Processing Systems (NIPS)*. ACM, 1097–1105.

[24] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436.

[25] In Lee and Kyoochun Lee. 2015. The Internet of Things (IoT): Applications, Investments, and Challenges for Enterprises. *Business Horizons* 58, 4 (2015), 431–440.

[26] Hui Li and Xiaojiang Xing. 2015. Internet of things service architecture and method for realizing internet of things service. US Patent 8,984,113.

[27] Shancang Li, Li Da Xu, and Shanshan Zhao. 2015. The internet of things: a survey. *Information Systems Frontiers* 17, 2 (2015), 243–259.

[28] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. 2017. Runtime neural pruning. In *Advances in Neural Information Processing Systems (NIPS)*. 2181–2191.

[29] Huimin Lu, Yujie Li, Shenglin Mu, Dong Wang, Hyoungseop Kim, and Seiichi Serikawa. 2018. Motor anomaly detection for unmanned aerial vehicles using reinforcement learning. *IEEE internet of things journal* 5, 4 (2018), 2315–2322.

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[31] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. 2017. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 ieee international conference on robotics and automation (icra)*. IEEE, 1527–1533.

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.

[33] Omer Berat Sezer, Erdogan Dogdu, and Ahmet Murat Ozbayoglu. 2018. Context-aware computing, learning, and big data in internet of things: a survey. *IEEE Internet of Things Journal* 5, 1 (2018), 1–27.

[34] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[35] Karen Simonyan and Andrew Zisserman. 2014. Two-Stream Convolutional Networks for Action Recognition in Videos. In *NIPS'14*. ACM, 568–576.

[36] Karen Simonyan and Andrew Zisserman. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations*. ACM.

[37] Arti Singh, Baskar Ganapathysubramanian, Asheesh Kumar Singh, and Soumik Sarkar. 2016. Machine learning for high-throughput stress phenotyping in plants. *Trends in plant science* 21, 2 (2016), 110–124.

[38] Vincent Vanhoucke, Andrew Senior, and Mark Z Mao. 2011. Improving the Speed of Neural Networks on CPUs. In *Proceeding Deep Learning and Unsupervised Feature Learning NIPS Workshop*, Vol. 1. ACM, 4.

[39] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*. 2074–2082.

[40] Jiecao Yu, Andrew Lukefahr, David Palframan, Ganesh Dasika, Reetuparna Das, and Scott Mahlke. 2017. Scalpel: Customizing DNN Pruning to the Underlying Hardware Parallelism. In *44th International Symposium on Computer Architecture (ISCA)*. IEEE, 548–560.