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

Matthew Merck, Bingyao Wang, Lixing Liu, Arthur Siqueria, Qiusen Huang, Abhijeet Saraha, Jiashen Cao, Ramyad Hadidi, Hyesoon Kim

Georgia Tech

comparch

Introduction: Rapidly Advancing Neural Network (NN) & Robotics Technologies

- Computer vision
- Neural machine translation
- Video recognition
- Unmanned aerial vehicles (UAVs)
- Internet-of-Things devices (IoT)

Executing Neural Networks (NNs) on the Edge?

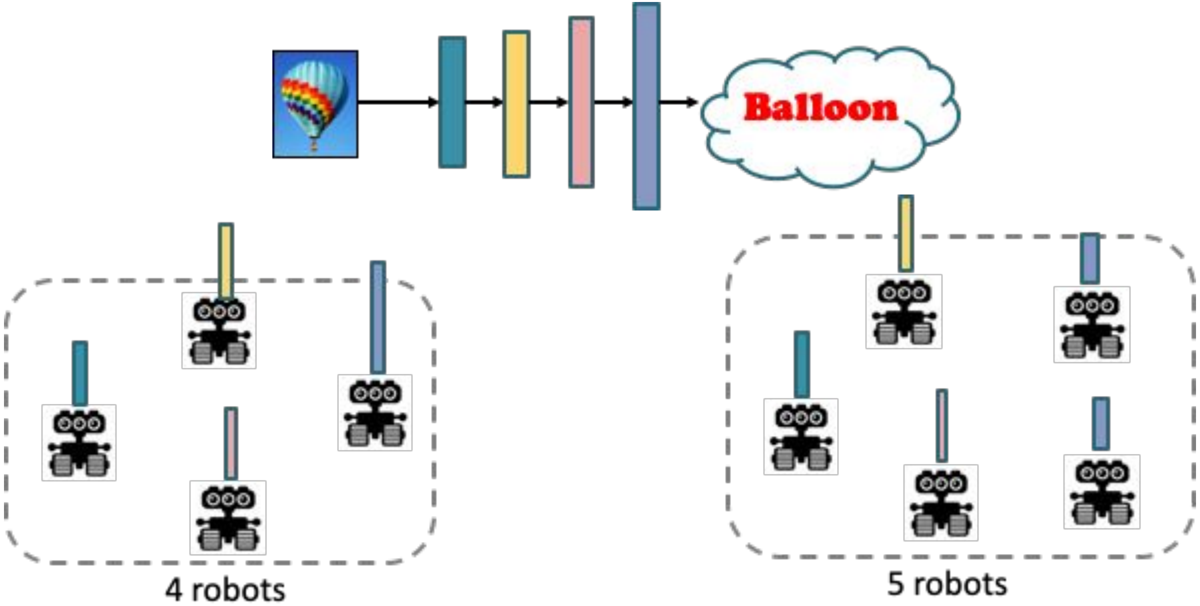
- Quick usage definition: an edge device is a device located closer to client machines than the network core
  - Often characterized by tight resource constraints
  - Provides entry point into core networks
- **Why should we execute NNs on the edge?**
  - Privacy concerns related to cloud computing
  - Strict real-time resource requirements
  - Unreliable connection of cloud computing
  - Edge devices have immediate access to raw data

Executing Neural Networks on the Edge? Continued

- Challenges:
  - Neural networks typically require greater computational resources than many individual edge devices can offer
- Alternatives:
  - Upload data to a cloud service and perform computation there
  - Weight prune or quantize a neural network to lower resource requirements and enable it to be ran on a single edge device

Distributed DL on Robotics Overview [IROS' 18]

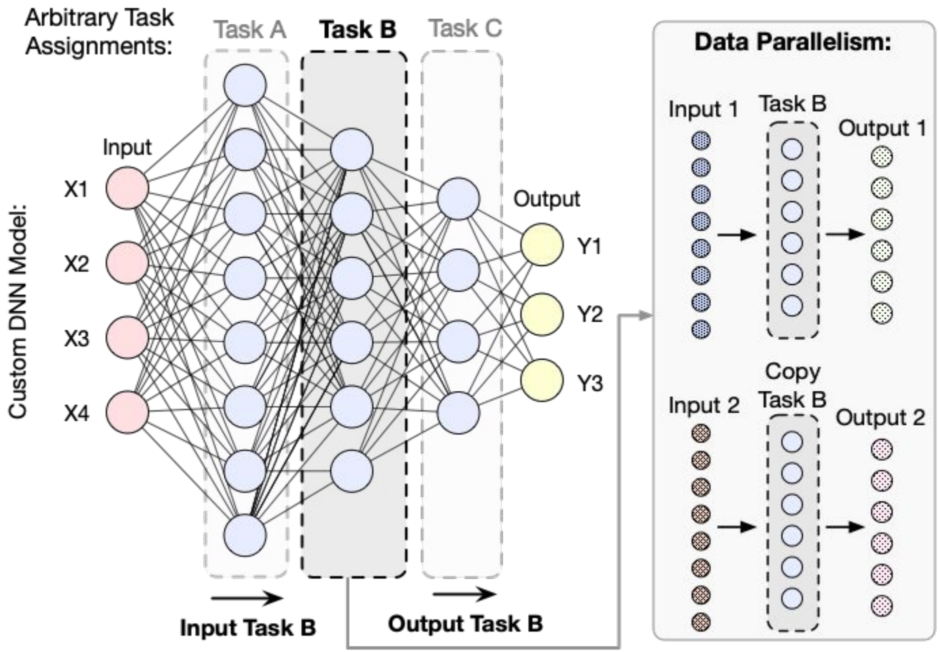
- We have proposed a technique to efficiently distribute DNN-based recognition



Distribution Method: Data Parallelism

Data parallelism is providing the next input to multiple devices in a network

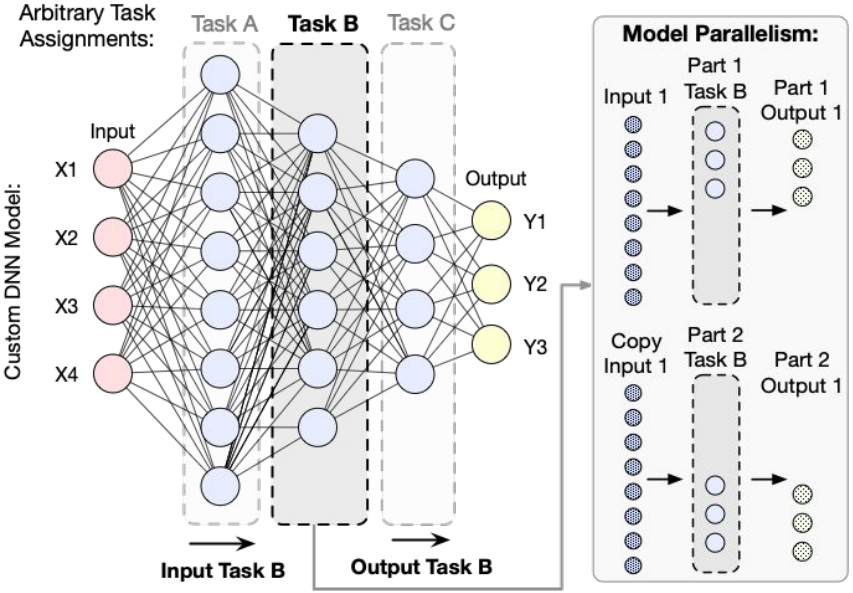
- Performing same computation of different data inputs



Distribution Method: Model Parallelism

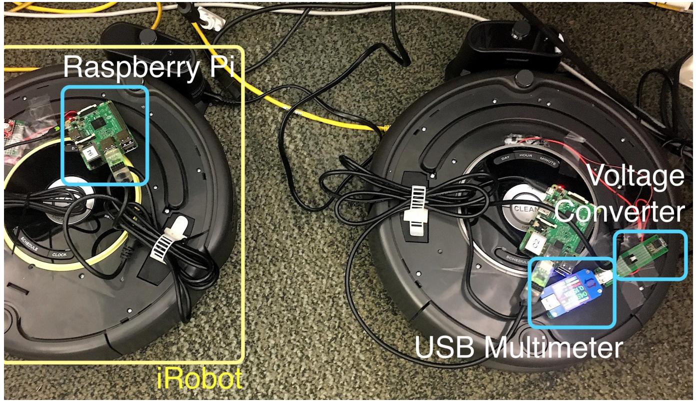
Model parallelism is splitting parts of a given layer or group of layers over multiple devices

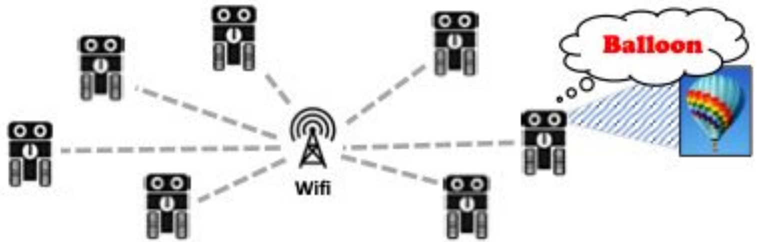
- Divide the computation on the same data input across devices



System Overview

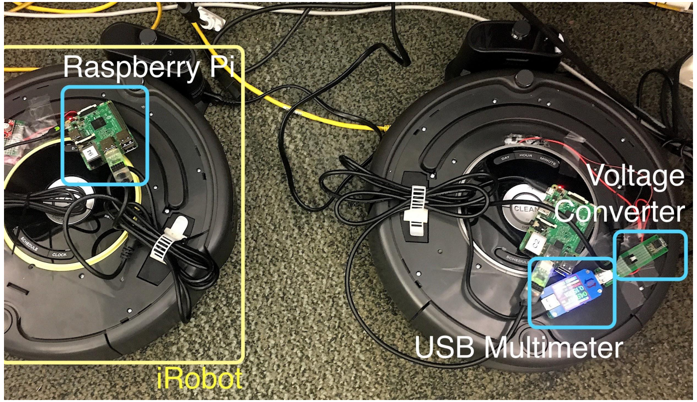
- We proposed an algorithm for deploying the distributed robot system only with Raspberry Pis.
- We used AlexNet, VGG16, and a video recognition (low resolution) model as example models.





Our Experiments in This Paper

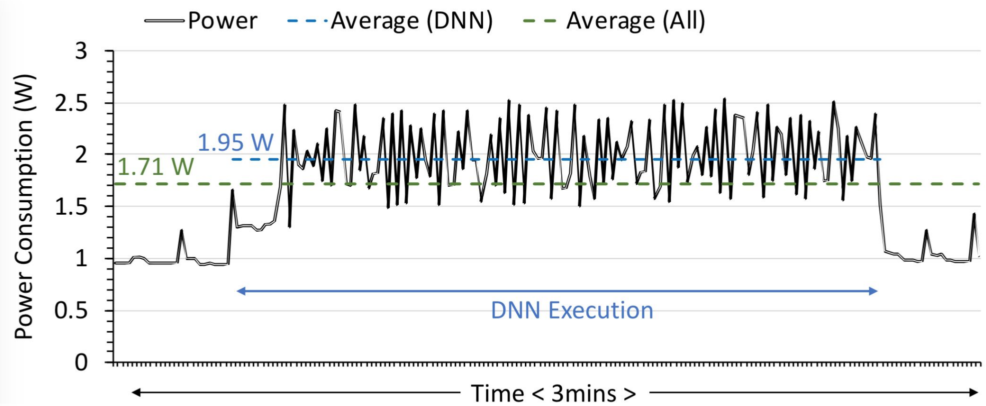
- Two iRobot Roomba 600s are each equipped with one Raspberry Pi 3 through a serial port connection
- iRobot Create 2 Open Interface is used to control the iRobot through the serial connection
- The iRobot power consumption is measured with the Open Interface
- Raspberry Pi power consumption is measured with a USB digital multimeter
- We use Keras 2.0 with the TensorFlow 1.0 backend
- Tests were ran for 3 minutes for power tests, and 10 minutes for latency tests



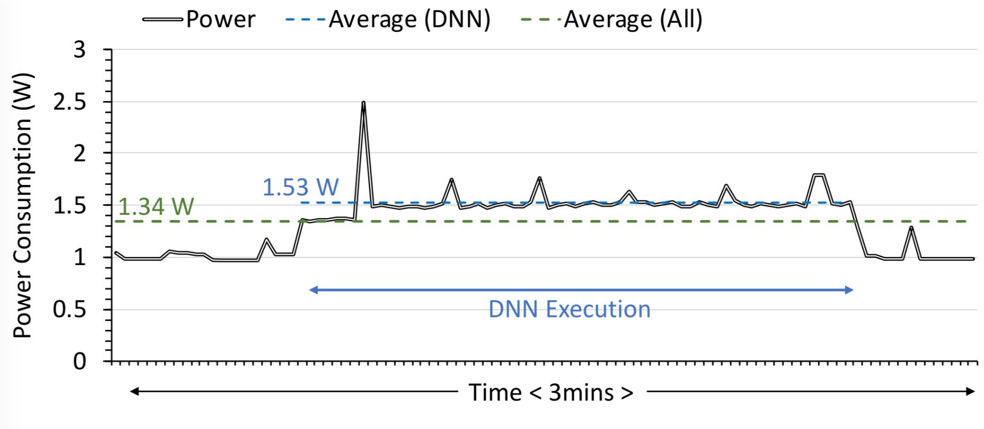
Raspberry Pi Specifications

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)

Raspberry Pi Power Measurements

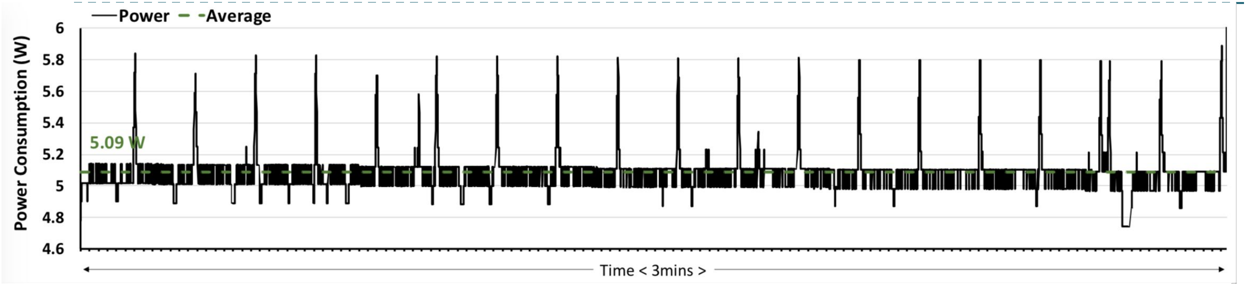


Power consumption of a single Raspberry Pi 3 executing the entire AlexNet neural network

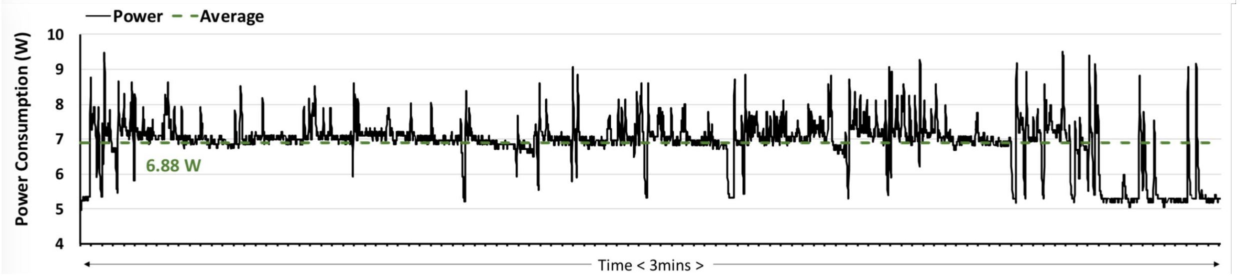


Power consumption of a single Raspberry Pi 3 while AlexNet is being executed in a distributed manner (total of two Raspberry Pis).

Results: Power Consumption of the Raspberry Pi + iRobot System (No Computations)



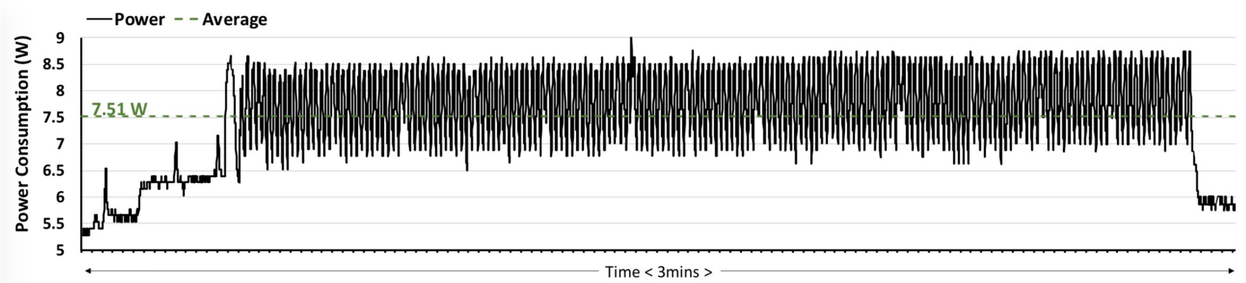
Power consumption of a stationary iRobot with no computation



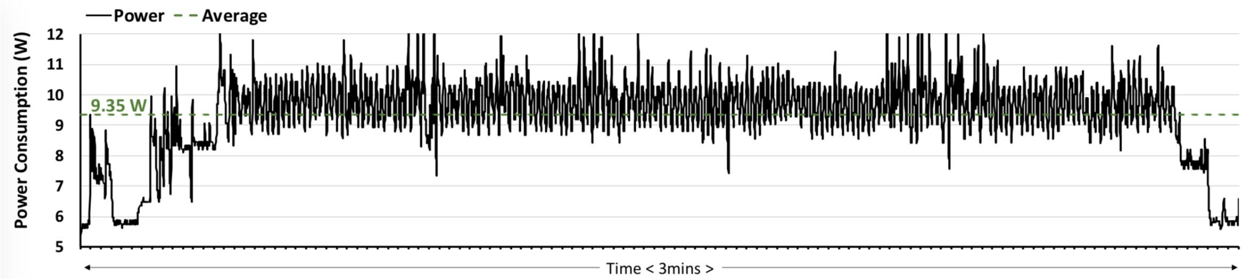
Power consumption of moving iRobot with no computation

The spikes are likely caused by the frequent system checks of the iRobot

Results: Power Consumption of the Raspberry Pi + iRobot System (With Computations)



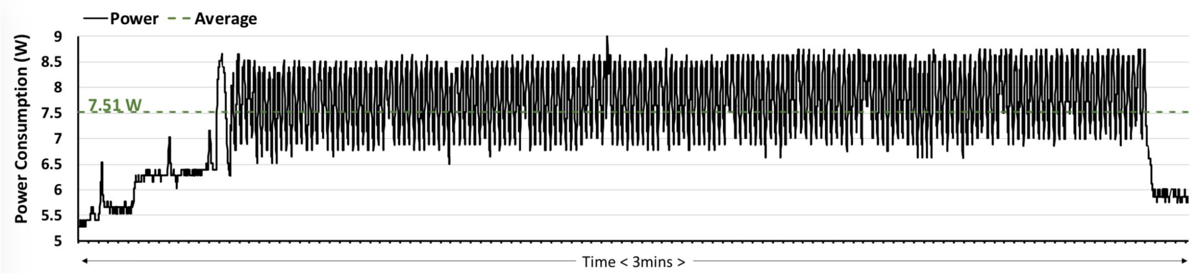
Power consumption of a stationary iRobot with computation



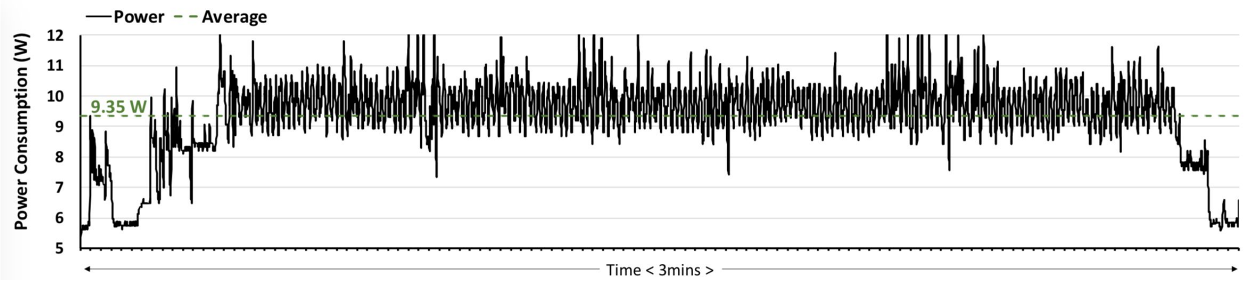
Power consumption of moving iRobot with computation

Execution of DNNs on a distributed robot system may lead to unpredictable power consumption, which can worsen Raspberry Pi performance, because variation of power delivery may lead to discrepancy in power saving settings in its CPU, leading to instability

Results: Power Consumption Spikes

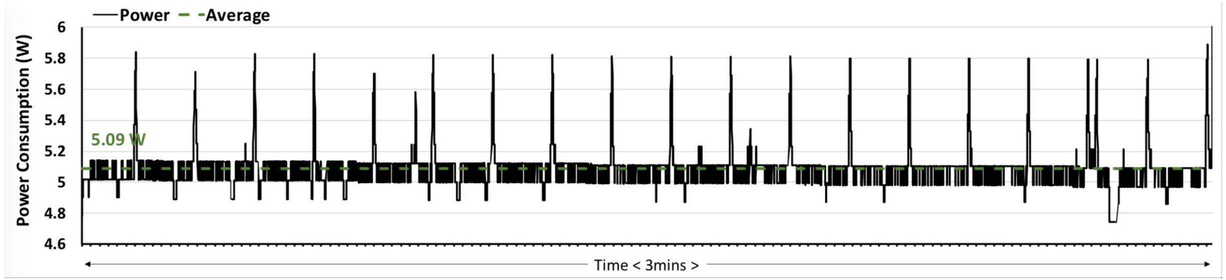


The spikes in this power consumption graph of a stationary iRobot with DNN execution were larger than hypothesized. We believe this to be due to unreliability of the current from the iRobot to the Raspberry Pi, or because of our circuitry.



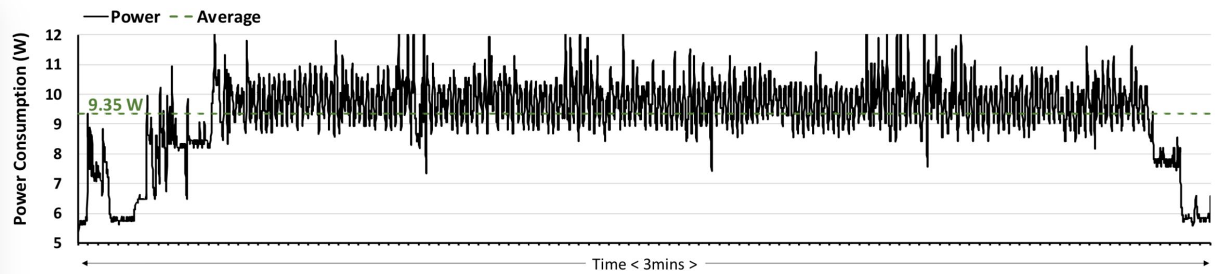
Some spikes in this graph are as large as 4.5 W. The execution of the DNN + iRobot movement causes great variation

Results: Overall Power Consumption



iRobot + Raspberry Pi system
with no movement and no computation

iRobot + Raspberry Pi system
with movement and computation

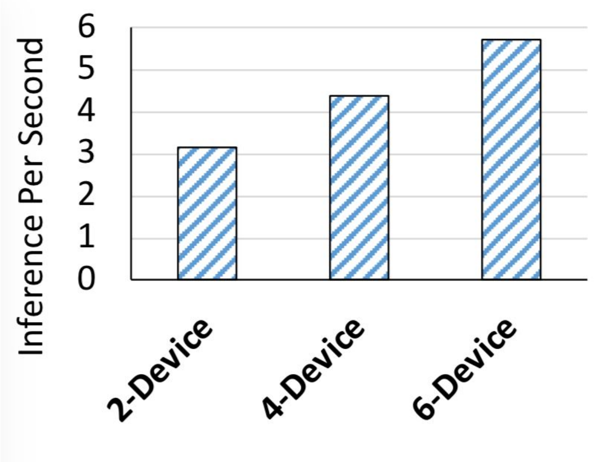


5.09 W average power consumption of the iRobot + Raspberry Pi system with no movement and no computation vs. 9.35 W average power consumption of the iRobot + Raspberry Pi system with movement and computation

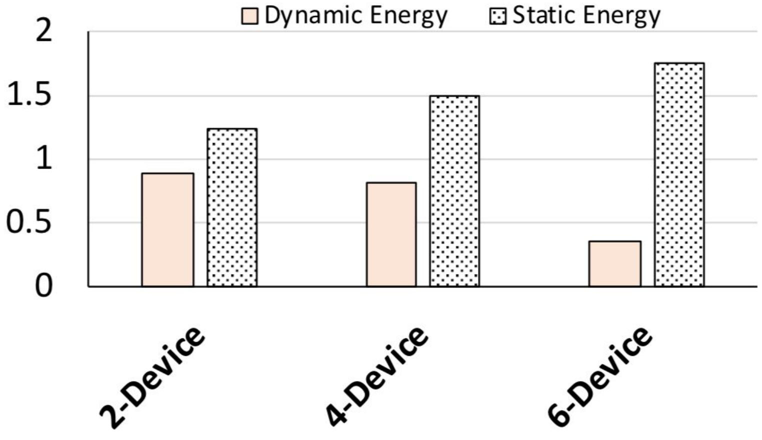
Raspberry Pi + iRobot Average Power Consumption w/ Spike Strength

	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

Results: Execution Performance



(a) Inferences Per Second



(b) Dynamic and Static Energy

Inferences per second and **power** (watts) consumption of various systems

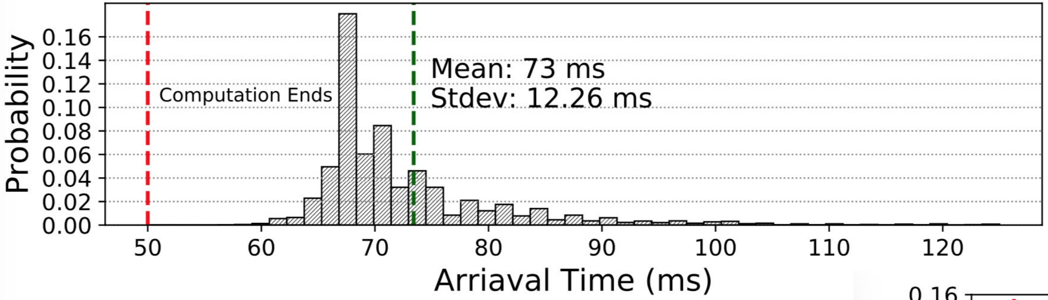
While adding additional devices to a network of edge devices increases static energy, it decreases the dynamic energy used by each device and also increases the inferences per second of each device

Why Communication Latency is Important in Distributed Systems

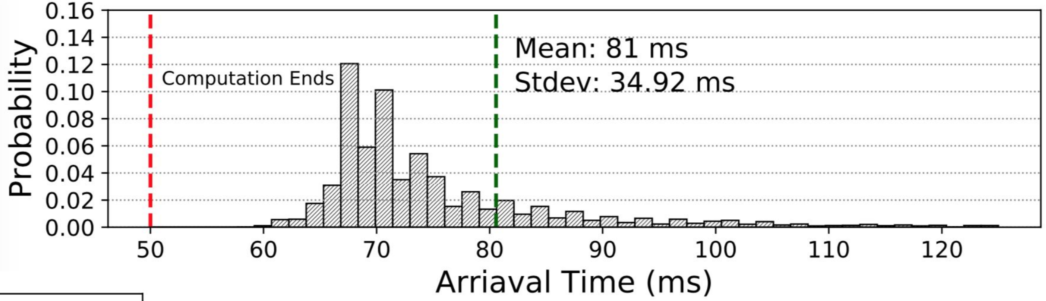
- The input data of a device usually depends on computation results produced by other devices in previous layers

  - These results must be transferred between devices using the network, which can incur delays and latency issues depending on the physical positioning of the devices, and can compound
- We used a WiFi router with a bandwidth of 94 Mbps

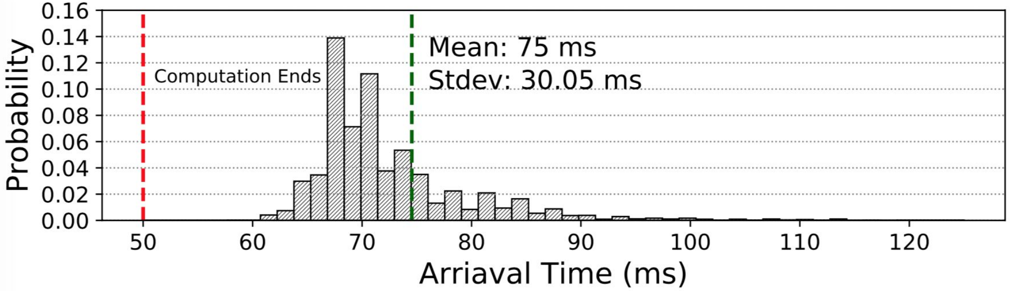
Results: Communication Latency



Histogram of communication latency while robot is near the station

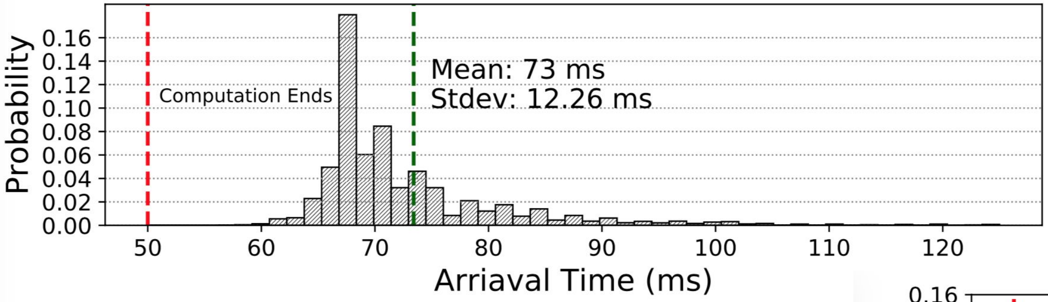


Histogram of communication latency while robot is away from the station

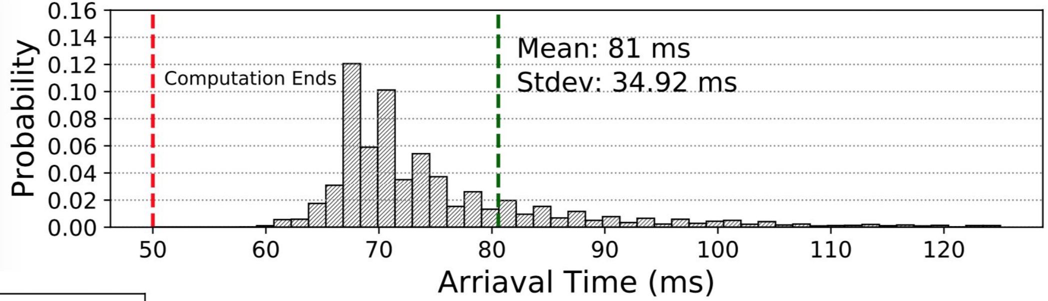


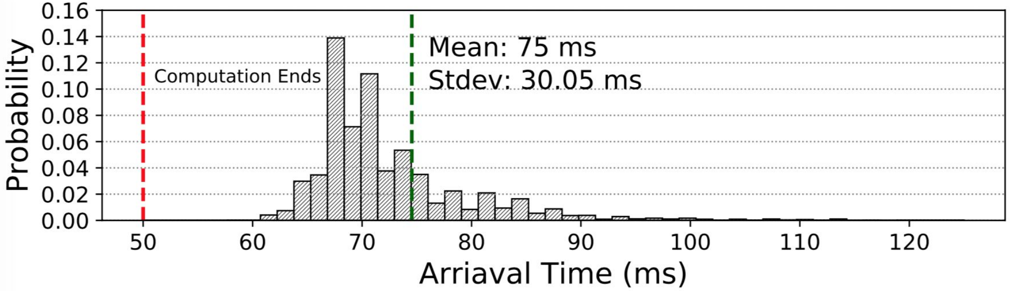
Histogram of communication latency while robot is moving (random movement profile)

Results: Communication Latency Continued



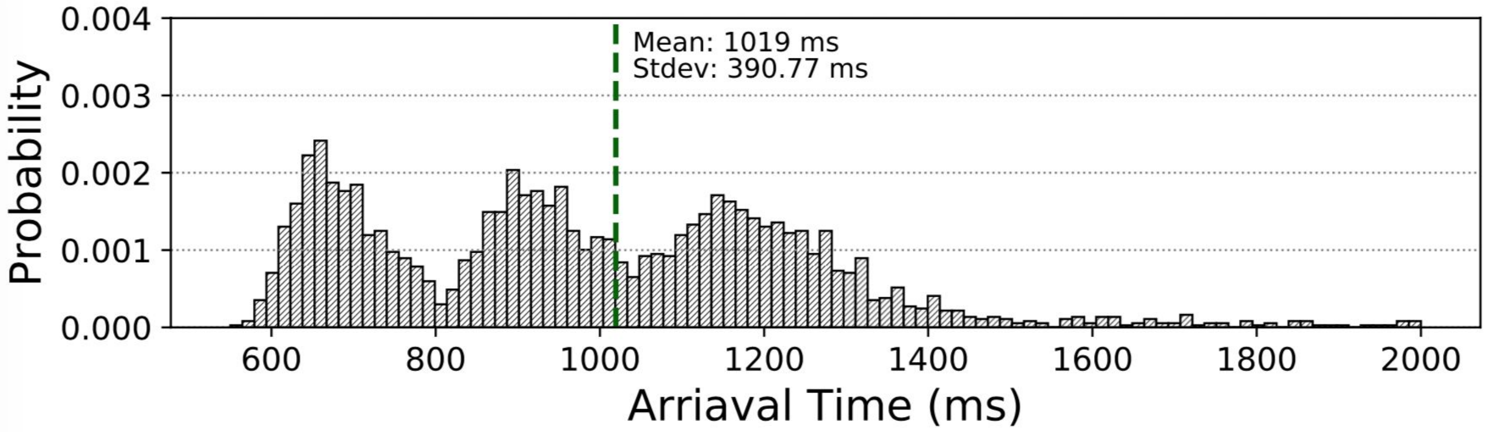
Standard deviation is 2x greater than that of above, with a mean increase of 8ms. Communication is less stable as distance increases





High variation - obstacles and distance change frequently, so there are unpredictable shifts in latency because of communication obstruction

Results: Communication Latency Aggregated



Execution latency histogram on six Raspberry Pis while executing AlexNet collaboratively.

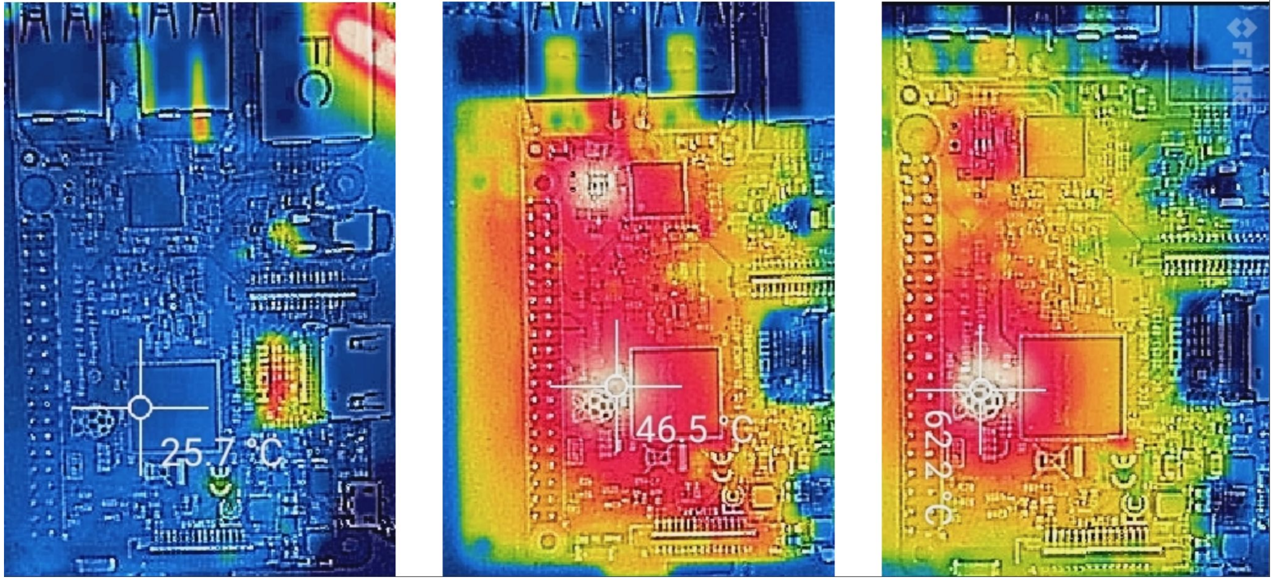
Computation is finished within 500ms

Three clusters: three devices are sharing computation for a single fully-connected layer in the NN.

Unpredictability is exacerbated by low end network equipment

Results: Device Temperature

Off	Idle	DNN
25.7° C	46.5° C	62.2° C



Thermal camera pictures of Raspberry Pi 3 in off, idle, and DNN execution conditions

Results: Device Temperature Effects

Why device temperature matters for our use case:

- High device temperature affects resolution and performance of Raspberry Pi cameras and other devices
- Extremely high temperatures can worsen performance if CPU overheats and must be slowed

Outcomes and Conclusions

- While adding additional devices to a network of edge devices increases static energy, it decreases the dynamic energy used by each device
- Adding devices also increases the inferences per second of each device
- Execution of DNNs on a distributed robot system may lead to unpredictable power consumption, which can worsen Raspberry Pi performance
- As expected, network latency varies and is uncontrollable in our system
- DNN computation can increase temperature by as much as 16 degrees Celsius

Future Work Directions

- Extend our system to YOLO (You Only Look Once) real time object detection execution on several Raspberry Pis using pruning methods
- Add robust DNN execution to the system, to reduce redundant computations that can result from high latency and data loss
  - Additionally, design models that are less communication heavy to combat latency issues

Thank you!

HPArch Spring 2019 group