Distributed Perception by Collaborative Robots

Ramyad Hadidi*, Jiashen Cao*, Matthew Woodward*, Michael S. Ryoo**, and Hyesoon Kim*

*Georgia Institute of Technology

**EgoVid Inc.
Robots and Their Environment

- Robots need to process lots of raw data in their environment.
  - Visual, Sounds, Temperature, ...
  - They need to understand it, to act upon it
- For instance:
  - Drones that study an area after a disaster
  - Smart security system with lots of cameras
  - Swarm robots
Deep Learning (DL) and Robots

- How they should process complex raw data?
  - Use deep learning!
  - It can help in many tasks:
    - Object detection
    - Scene recognition
    - Action recognition
    - Speech recognition
DL Computation is Heavy

- But DL models are computationally intensive and resource hungry specially for cheap robots.

An example of 3x dense layers on resource constrained device:

- Memory Usage:
  - 3x Dense (8k): 1.5 GB
  - 3x Dense (4k): 0.5 GB

- Inference time:
  - 3x Dense (8k): Not Possible
  - 3x Dense (4k): 0.728 s
DL Computation is Heavy

- But DL models are computationally intensive and resource hungry, especially for cheap robots.

An example of 3x dense layers on Raspberry PI:

<table>
<thead>
<tr>
<th>Memory Usage</th>
<th>Inference time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.728 GB</td>
<td>Not Possible</td>
</tr>
</tbody>
</table>

It is difficult to execute DL models on all kinds of robots because:

1) Usually good models have large memory footprint.
2) For a robot, latency for single inference is important.
DL Computation is Heavy

- So robots need the result fast and in **real time**!
- Then how resource-constrained robots can use DL to understand their surroundings?
Let’s Collaborate

- Usually, many cheap robots share their environment.
- Not all robots need to perform computations at the same time.
- So What if they share their knowledge and help each other?

(a) Single Robot

(b) Collaborative Robots
Our Work Overview

- We have proposed a technique to efficiently distribute DNN-based recognition.
Our Work Overview

- We proposed an algorithm for deploying the distributed robot system only with Raspberry Pis.

- We used AlexNet, VGG16, and a video recognition model as example models.
Outline

Introduction & Motivation

Data and Model Parallelism
  - Fully Connected and Conv Layers

Distributing DNN
  - Algorithm

System Evaluations

Conclusions
Model & Data Parallelism

Assume a custom DNN model, divided layer by layer:

Arbitrary Task Assignments:
- Task A
- Task B
- Task C

Custom DNN Model:
- Input X1, X2, X3, X4
- Output Y1, Y2, Y3

Data Parallelism:
- Task B
- Copy Input 1
- Input 2

Model Parallelism:
- Input 1
- Copy Part 1
- Part 2
- Output 1
- Part 2
- Output 1
Data Parallelism

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

Model parallelism is splitting parts of a given layer or group of layers over multiple devices.
Fully Connected (FC) Layer

- Every output element is a summation of weighted inputs
- Each output element have its own set of weights
- A matrix multiplication

\[ a_j = \sum_i x_i w_{ij} \]
Model & Data Parallelism for FC

- Every output element is a summation of weighted inputs
- Each output element has its own set of weights
- A matrix multiplication

![Graph showing speedup vs. size of the fc layer for inference (single device), data parallelism, and model parallelism with two devices.](image-url)
Model & Data Parallelism for FC

![Speedup vs Size of the fc layer Graph]

- **Inference (Single Device)**
- **Data Parallelism**
- **Model Parallelism**

**Size of the fc layer**

- 512
- 1024
- 2048
- 4096
- 8192
- 10240
- 12288
- 14336
- 16384

**Speedup**

**Two Devices**
Model & Data Parallelism for FC

![Graph showing speedup for different sizes of fc layer with Inference (Single Device), Data Parallelism, and Model Parallelism compared to Two Devices.](image)

- **Size of the fc layer**
  - 512, 1024, 2048, 4096, 8192, 10240, 12288, 14336, 16384

- **Speedup**
  - Inference (Single Device)
  - Data Parallelism
  - Model Parallelism

Distributed Perception by Collaborative Robots

IROS 2018
Model & Data Parallelism for FC

Model parallelism reduces memory footprints.
So, we avoid slow hard drive accesses
Model & Data Parallelism for FC

Fully Connected Layers: Either data or model parallelism depending on size of the layer, input, and memory
Convolution Layer

- Every channel in the output is derived from applying the same filter on the input.
- Memory footprint size is smaller \( \rightarrow \) fit into device memory size.
- Model parallelism: split based on filters, and one more merging node at the end.

\[
\text{Input: } H_i \times W_i \times C_i \\
\text{Output: } H_o \times W_o \times C_o
\]

Each filter creates a channel in output.
Model & Data Parallelism for Conv

Inference (Single Device)  Data Parallelism  Model Parallelism

Number of conv layer filters

Speedup

Two Devices

Distributed Perception by Collaborative Robots  IROS 2018
Convolution Layers: Data parallelism is better
Outline

Motivation

Background
- ML Models Overview

Data and Model Parallelism
- Fully Connected and Conv Layers

**Distributing DNN**
- Algorithm overview

System Evaluations

Conclusions
Distributing a DNN Model

(i) Profiling DNN Layers

Profiling Hardware Phase

(ii) Gather Data on Environment and DNN Model

Environment and DNN Model Inspection

(iii) Generate Task Assignments

Task Assignment Phase

Task Assignments for {1...n} devices

Distribution

Details are in the paper
Outline

Motivation

Background
  • ML Models Overview

Data and Model Parallelism
  • Fully Connected and Conv Layers

Distributing DNN
  • Algorithm overview

System Evaluations

Conclusions
Software & Hardware

Software:

- Keras 2.1
  - With Tensorflow backend for Raspberry Pis
  - With Tensorflow-GPU backend for TX2
- Apache Avro for procedure call (RPC) and data serialization
Hardware Overview

Raspberry PI 3:
- Cheap and accessible platform
- Connected via a Wifi router
- No GPU
- $40

Nvidia Jetson TX2:
- High-end embedded platform
- Has a GPU
- $600

Moreover, we measured whole system power with a power analyzer.
AlexNet Distribution I

Five-device system:

Tasks of A
- Input Stream

Tasks of B
- CNN Layers

Task of C
- fc_1 (2k)
- Model Parallelism

Task of D
- fc_1 (2k)
- Model Parallelism

Task of E
- fc_2 (4k)
- fc_3 (1k)

Convolution (CNN) Layers

Distributed Perception by Collaborative Robots

IROS 2018
AlexNet Distribution II

Six-device system:

- Tasks of A
  - Input Stream
  - CNN Layers
  - Data Parallelism

- Tasks of B & C
  - CNN Layers

- Task of D
  - fc_1 (2k)
  - Model Parallelism

- Task of E
  - fc_1 (2k)
  - Model Parallelism

- Task of F
  - fc_2 (4k)
  - fc_3 (1k)

Convolution (CNN) Layers

Distributed Perception by Collaborative Robots

IROS 2018
AlexNet Results

Comparable IPS with TX2 (-30%)
Lower dynamic energy consumption
VGG16 Distribution I

Nine-device system:

Tasks of A
Block 1

Data Parallelism

Tasks of B
Block 2, 3, 4

Tasks of C, D, & E

Task of F
fc_1 (2k)

Task of G
fc_2 (4k)
fc_3 (1k)

Task of I
fc_1 (2k)

Model Parallelism

Model Parallelism

Tasks of H

Input Stream

Merge

Distributed Perception by Collaborative Robots

IROS 2018
VGG16 Distribution II

11-device system:

Task of J
\( \text{fc}_1(2k) \)  
Model Parallelism

Task of L
\( \text{fc}_2(4k) \)  
\( \text{fc}_3(1k) \)

Task of K
\( \text{fc}_1(2k) \)  
Model Parallelism

Input Stream

Tasks of A

Tasks of B, C, D, E
F, G, & H

Block 1, 2, 3, 4, 5

Data Parallelism

Block 1, 2, 3, 4, 5

Distributed Perception by Collaborative Robots  
IROS 2018
Comparable IPS with TX2 (-15%)
We achieve 2.3x speedup, by reassigning CNN blocks
Outline

Motivation
Background
  ▸ ML Models Overview
Data and Model Parallelism
  ▸ Fully Connected and Conv Layers
Distributing DNN
  ▸ Algorithm overview
System Evaluations

Conclusions
Conclusions

- We used a farm of Raspberry PIs for DNN processing
- Our technique achieves acceptable real-time performance
  - Compared to TX2 CPU, we achieve similar performance with 6 robots for AlexNet
  - 11 robots for VGG-16 compared to TX2 GPU

Future Work:
- Study the robustness of such systems
- Apply our technique to more DNN models
- Implement our model on distributed robot systems
Backup Slides
Layers of ML Models

- Convolution: Applies several filters to the input
  - Compute bound, more locality
- Activation: Introduces non-linearity
  - e.g., ReLU \( f(x) = \max(0, x) \), not compute intensive
- Fully Connected (Dense)
  - i.e., matrix multiplication, bandwidth bound
- Pooling
  - Reduces dimensions, simple doing max, average, and ... on a subset of input
Image Recognition Models (I)

- Single-stream AlexNet

Convolution (CNN) Layers

- Input
- Conv2D
- Maxpool
- Conv2D
- Maxpool
- Conv2D
- Conv2D
- Maxpool
- Conv2D
Image Recognition Models (II)

- VGG16

![Diagram of VGG16 architecture]
Vide Recognition Model

- i.e., Action recognition model
- Based on the two-stream model by Ryoo et al.\cite{ryoo}

Sliding Window

Input Frames

- Frame 15
- ...
- Frame 24
- Frame 25
- Frame 26

Recorder Node

One Frame

- Spatial CNNs
  - Frame 24
  - Frame 25
  - Frame 26

Ten Frames

- Temporal CNNs
  - Frame 15 – 24
  - Frame 16 – 25
  - Frame 17 – 26

256 Elements per one frame

256 Elements per ten frames

Temporal Pyramid & Dense Layers

Frames 1 – 10 temporal

Frames 15 – 24 temporal

Frames 16 – 25 temporal

Frames 17 – 26 temporal

Pyramid 1

IROS 2018
Distributed Perception by Collaborative Robots
Algorithm 1 Task Assignment Algorithm.

```plaintext
1: function TASK_ASSIGNMENT(dnn, nmax, comm, memsize)
    Inputs: dnn: DNN model in form of layers [type, size, inputsize, β]
            nmax: Maximum number of the devices
            comm: Communication overhead model (comm(size_data))
            memsize: Device memory size
2:    L := EXTRACT_MODEL_TO_LAYERS(dnn)
3:    for n from 1 to nmax do
4:        tasks_final[n] := ∅
5:    for n from 1 to nmax do
6:        TG, noFit := FIND_INITIAL_TASKGROUP(L, memsize)
7:        if sizeof(TG) > n then
8:            tasks[n] := COMBINE_TASKS(TG, memsize, nmax, n)
9:        if sizeof(TG) = n then
10:           tasks[n] = TG
11:    if sizeof(TG) < n or noFit == True then
12:        while sizeof(TG) ≠ n do
13:            task_variant := ∅
14:            for every \( i \in TG \) do
15:                [task_variant] += PROFILED_VARIANTS(i, comm)
16:            task_replaced, task_new = SELECT_LOWEST([task_variant])
17:            TG = TG - task_replaced + task_new
18:        tasks_final[n] = TG
19:    return tasks_final
```
Hardware Overview

Raspberry PI 3:
- Cheap and accessible platform
- Connected via a Wifi router
- No GPU
- $40

Nvidia Jetson TX2:
- High-end embedded platform
- Has a GPU
- $600

Moreover, we measured whole system power with a power analyzer
Video Recognition on Single PI

(a) Loading Time

(b) Memory Usage

(c) Inference Time

(d) Energy Per Inference
Video Recognition Distributions (I)

Node A Tasks
- Spatial CNN
- Recording Optical Flow

Node B Task
- Spatial CNN

Node C Task
- Temporal CNN

Node D Tasks
- Maxpool
- fc_1 (8k)

Node E Tasks
- fc_2 (8k)
- fc_3 (51)

5 Nodes

Tasks of A
- Spatial CNN
- Recording Optical Flow

Task of B
- Spatial CNN

Task of C
- Temporal CNN

Tasks of D
- Maxpool
- fc_1 (4k)

Task of F
- fc_2 (4k)

Task of G
- fc_2 (4k)

Task of H
- fc_3 (51)

8 Nodes
Video Recognition Distribution (II)

Tasks of A
- Recording Optical Flow
- Spatial CNN

Tasks of B & C
- Spatial CNN

Tasks of D & E
- Temporal CNN

Tasks of F
- Maxpool
- fc_1(4k)

Tasks of G
- Maxpool
- fc_1(4k)

Tasks of H
- fc_2(4k)

Tasks of I
- fc_2(4k)

Tasks of J
- fc_3(51)

Tasks of K
- fc_3(51)

10 Nodes

12 Nodes

Distributed Perception by Collaborative Robots

IROS 2018
Video Recognition Results (2)

Latency of one Frame (Seconds)

System Architecture

- HPC (GPU)
- HPC (CPU)
- TX2 (GPU)
- TX2 (CPU)
- 1-Node
- 5-Node
- 8-Node
- 10-Node
- 12-Node

Latency of one Frame (s)

- Computation
- Communication
- Reloading

<table>
<thead>
<tr>
<th>System Architecture</th>
<th>0.01</th>
<th>0.07</th>
<th>0.19</th>
<th>0.51</th>
<th>1.32</th>
<th>1.32</th>
<th>1.32</th>
<th>0.68</th>
<th>1.05</th>
<th>1.19</th>
<th>1.65</th>
<th>2.32</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPC (GPU)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HPC (CPU)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX2 (GPU)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TX2 (CPU)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-Node</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-Node</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-Node</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-Node</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12-Node</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Video Recognition Results (3)

Energy:

![Energy Bar Chart]

- HPC (GPU): 8.18 J
- HPC (CPU): 1.80 J
- TX2 (GPU): 1.96 J
- TX2 (CPU): 0.94 J
- 1-Device: 2.10 J
- 5-Device: 5.22 J
- 10-Device: 1.27 J
- 12-Device: 1.12 J

**System Architecture**

**Static Energy**
- HPC (GPU): 8.18 J
- HPC (CPU): 1.80 J
- TX2 (GPU): 1.96 J
- TX2 (CPU): 0.94 J
- 1-Device: 2.10 J
- 5-Device: 5.22 J
- 10-Device: 1.27 J
- 12-Device: 1.12 J

**Dynamic Energy**
- HPC (GPU): 198.90 J
- HPC (CPU): 40.80 J
- TX2 (GPU): 1.38 J
- TX2 (CPU): 0.72 J
- 1-Device: 1.38 J
- 5-Device: 0.72 J
- 10-Device: 0.61 J
- 12-Device: 0.55 J