
Haichen Shen (haichen@cs.washington.edu)
with Seungho Han, Matthai Philipose,
Sharad Agarwal, Alec Wolman, Arvind Krishnamurthy

University of Washington Microsoft Research
Wearable computing $\rightarrow$ more data
When computer vision meets wearable

“That drink will get you to 2800 calories for today”

“I last saw your keys in the store room”

“Remind Tom of the party”

“You’re on page 263 of this book”

Consumer  Manufacturing  Public Safety
Deep learning makes vision work

But...

<table>
<thead>
<tr>
<th>Recognition Task</th>
<th>face</th>
<th>scene*</th>
<th>object*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>97%</td>
<td>88%</td>
<td>92%</td>
</tr>
<tr>
<td>Compute/frame (FLOPs)</td>
<td>1.00G</td>
<td>30.9G</td>
<td>39.3G</td>
</tr>
<tr>
<td>Compute@1-30fps (FLOPS)</td>
<td>1-30G</td>
<td>30-900G</td>
<td>40G-1.2T</td>
</tr>
</tbody>
</table>

Do we have enough resources to run deep learning?

* top-5 accuracy is shown in the table
Resource usage for continuous vision

- Omnivision OV2740: 90mW
- Tegra K1 GPU: >800mW, 290GOPS@10W = 34pJ/OP
- Qualcomm SD810 LTE Atheros 802.11 a/g: 15Mbps@700mW= 47nJ/b

Amazon EC2
- CPU c4.large: 2x400GFLOPS, $0.1/h
- GPU g2.2xlarge: 2.3TFLOPS, $0.65/h

- Cloud cost: $10 person/year

Workload: Deep learning 300GFLOPS @ 30GFLOPs/frame, 10fps

Budget
- Device power: 30% of 10Wh for 10h = 300mW
- Cloud cost: $10 person/year

Compute power
- 9GFLOPS
- 3.5GFLOPS (GPU) / 8GFLOPS (CPU)

Huge gap between workload and budget
Neural network

(classes + scores)

(img_R)(img_G)(img_B)

(c) convolution
(f) fully connect
(m) max pool
(r) relu
(s) softmax
Neural network $\approx$ matrix multiplications

Architectural changes (J. Ba, et al. 2014)
Low rank approximation (Y. Kim, et al. 2016)
Matrix sparsification (S. Han, et al. 2015)
Managing the approx. / resource trade-off

- Detailed characterization of the approximation / resource trade-off for many optimizations

- Two new optimizations for streaming, multi-application settings

- New scheduling problem, **Approximate Model Scheduling**, with a heuristic solution
Outline

- Detailed characterization of the approximation / resource trade-off for many optimizations

- Two new optimizations for streaming, multi-application settings

- New scheduling problem, Approximate Model Scheduling, with a heuristic solution
Memory / accuracy trade-off

![Memory vs Accuracy Graph]

- **Accuracy (%)**
- **Memory (MB)**

Legend:
- **Obj**
- **Scene**
- **Face**
Memory / accuracy trade-off

Substantially reduce memory use with gradual accuracy loss
Energy / accuracy trade-off

- Always execute locally
- Can execute locally under energy budget
- Exceed energy budget when execute locally

energy budget = total energy / total time(10h) / requests per second(1 req/sec)
Outline

- Detailed characterization of the approximation / resource trade-off for many optimizations

- Two new optimizations for streaming, multi-application settings
  - Specialization
  - Model sharing

- New scheduling problem, Approximate Model Scheduling, with a heuristic solution
Exploiting stream locality by specialization

• Standard deep neural network recognizes 4000 people
• Most of videos are dominated by less than 10 faces over minutes

Timeline

Produce more compact models for skewed classes
Specialization runtime

(full model) → (check for "other") → (compact model) → (input)

(with skewed distribution)
Better resource/accuracy trade-off

![Bar charts showing resource/accuracy trade-off for different categories: Face, Object, and Scene. The x-axis represents categories, and the y-axis represents accuracy and energy consumption. The charts compare Full model, Compact model, and Compact model with specific (7 cls) for accuracy and energy.](image)
Outline

- Detailed characterization of the approximation / resource trade-off for many optimizations
- Two new optimizations for streaming, multi-application settings
- New scheduling problem, Approximate Model Scheduling, with a heuristic solution
Approximate model scheduling

Goal: maximize the overall accuracy

Model Pool

Mobile device

Cloud

Accuracy

Energy 14

Memory 10

Cost 14

Task 1
model 1 (90%) 8 2 2
model 2 (80%) 5 1 1

Task 2
model 3 (80%) 9 3 3
model 4 (70%) 6 2 2

Memory
energy
cost

Accuracy

Cost

Energy

Task 1

Task 2

model 1
model 2
model 3
model 4

90%
80%
80%
70%

model 1
model 2
model 3
model 4

8
5
9
6

2
1
3
2

2
1
3
2

Goal: maximize the overall accuracy
Approximate model scheduling

Model Pool

- Task 1
  - model 1 (90%)
  - model 2 (80%)
    - model 1
    - model 2

- Task 2
  - model 3 (80%)
  - model 4 (70%)
    - model 3
    - model 4

Mobile device

- model 4
- model 4

Energy 14

Cloud

- model 1
- model 1
- model 2

Cost 14

Accuracy

- 1
- 2
- 3
- 4
- 5

Memory 10

Packing problem:

1. task 1 → cloud, model 1
2. task 2 → device, model 4
3. task 1 → cloud, model 1
4. task 1 → cloud, model 2
5. task 2 → device, model 4
Approximate model scheduling

Model Pool

<table>
<thead>
<tr>
<th>Task 1</th>
<th>Task 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>model 1 (90%)</td>
<td>model 3 (80%)</td>
</tr>
<tr>
<td>model 2 (80%)</td>
<td>model 4 (70%)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8</th>
<th>2</th>
<th>2</th>
<th>5</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
</table>

| 9 | 3 | 3 | 6 | 2 | 2 |

Mobile device

<table>
<thead>
<tr>
<th>model 4</th>
<th>model 4</th>
</tr>
</thead>
</table>

Energy 14

Cloud

<table>
<thead>
<tr>
<th>model 3</th>
<th>model 1</th>
</tr>
</thead>
</table>

memory | energy | cost

Accuracy

1 2 3 4 5

Requests:
1. task 1 → cloud, model 1
2. task 2 → device, model 4
3. task 1 → cloud, model 2
4. task 1 → cloud, model 1
5. task 2 → device, model 4
6. task 1

Paging problem
Approximate model scheduling

• Packing problem: pick versions that satisfy energy/cost budgets
  \[ \sum_t e_i x_{it} \leq E, \sum_t c_i x'_{it} \leq C \ (x_{it}, x'_{it} \in [0,1], x_{it} \cdot x'_{it} = 0) \]

• Paging problem: pick versions that fit in memory
  \[ \forall 1 \leq t \leq T, \sum_{i=1}^n s_i x_{it} \leq S \]

• Goal: maximize the accuracy
  \[ \max_x \sum_t \sum_i a_i (x_{it} + x'_{it}) \]

No known optimal online algorithms
Heuristic scheduler

• Estimate future resource use and compute the budget for each request

• Account for paging cost to reduce oscillations

• Use increasingly more accurate versions of more heavily used models
Trace-driven evaluation

Lose connectivity

Run out of battery
MCDNN framework

Development time

Input type
Model schema
Training/validation data

Compiler

trained model catalog

Run time

Input

Device runtime
Scheduler
Data router

Cloud runtime
Scheduler
Data router
Profiler

Input
Classes

Device
catalog

Input
classes

Device
MCDNN framework

Development time:
- input type
- model schema
- training/validation data

Specialization time:
- compiler
  - trained model catalog
- specializer
  - specialized models
  - stats

Run time:
- device runtime
  - scheduler
  - data router
- cloud runtime
  - scheduler
  - data router
  - profiler
- input
- classes
- apps
- device
- cloud
Conclusion

• MCDNN makes efficient trade-offs between resource use and accuracy
• Formulate the approximate model scheduling problem and devise a heuristic algorithm
• Design a generic approximation-based execution framework for continuous mobile vision

Thank you! Questions?
Backup Slides
Cloud cost / accuracy trade-off

Latency budget = cost budget / cost per hour / #requests

cloud CPU latency budget (7645ms)
@ $10/yr, 1 request/min
@ AWS c4.large

cloud GPU latency budget (582ms)
@ $10/yr, 1 request/min
@ AWS g2.2xlarge

cloud GPU latency budget (9.7ms)
@ $10/yr, 1 request/s
@ AWS g2.2xlarge
Model sharing

face
ID
race
age
gender
input

intermediate values

model-fragment cache

...
Dynamically-sized caching scheme