FoggyCache: Cross-Device Approximate Computation Reuse

Peizhen Guo, Bo Hu, Rui Li, Wenjun Hu

Yale University
Emerging trend
Emerging trend

Computation intensive: incurring offloading latency, draining battery
Same, popular apps run on nearby devices

Example: landmark recognition
Redundancy across nearby devices

“Sterling Library”

Example: landmark recognition
Redundancy across nearby devices

“Sterling Library”

Example: landmark recognition

Up to 82% input generate the same result
More Examples: smart home scenarios

“Play some music.”

“Turn on the light.”

“Play some music.”

“Turn on the light.”
Can we eliminate this redundancy?
Can we eliminate this redundancy?

Reuse previous computation results
Traditional computation reuse

\[ \text{Cache} \]

\[
\begin{array}{|c|c|}
\hline
\text{Key} & \text{Value} \\
\hline
\text{“Sterling Library”} & \\
\hline
\end{array}
\]

\( \text{Recognize( )} \Rightarrow \text{“Sterling Library”} \)
Traditional computation reuse

\[
\text{Cache} = \begin{array}{|c|c|}
\hline
\text{Key} & \text{Value} \\
\hline
\text{"Sterling Library"} & \text{"Sterling Library"} \\
\hline
\end{array}
\]

No reusable data
Ideal computation reuse

\[
\text{Cache} = \begin{array}{c|c}
\text{Key} & \text{Value} \\
\hline
\text{"Sterling Library"} & \text{"Sterling Library"}
\end{array}
\]

\[
\text{Recognize}(\cdot) = \text{"Sterling Library"}
\]

\[
\text{Recognize}(\cdot) \approx \text{"Sterling Library"}
\]
Approximate Computation Reuse
Our goals

- Algorithms for *approximate computation reuse*

- A system to eliminate redundancy across devices
Reuse process
Reuse process

Input data

Lookup computation records with similar input

Reuse process

Input data

Lookup computation records with similar input

[ [ , “Bass Library” ] ]

[ [ , “Sterling Library” ] ]

Determine reuse outcome

Not reusable or “Bass Library” or “Sterling Library”
The rest of the talk...

- Algorithms for *approximate computation reuse*
  - A-LSH – fast lookup
  - *H-kNN* – reuse with accuracy guarantee
- FoggyCache system for cross-device reuse
Handwritten digits from MNIST dataset
A-LSH: strawman

Locality sensitive Hashing (LSH)
A-LSH: strawman

Locality sensitive Hashing (LSH)

More similar data stay in the same bucket with higher probability
LSH is not enough
LSH is not enough
LSH is not enough
LSH is not enough

Dense data

Fixed bucket size
LSH is not enough

Dense data
Increase lookup time
LSH is not enough

Dense data
Increase lookup time

Sparse data

Fixed bucket size
LSH is not enough

Dense data
Increase lookup time

Sparse data
Miss actually similar records

Fixed bucket size
LSH is not enough

Dense data
Increase lookup time

Sparse data
Miss actually similar records

Fixed bucket size

LSH configuration is static

Data distribution is dynamic
Adaptive-LSH

adapt the bucket size to data distribution
Adaptive-LSH

adapt the bucket size to data distribution

Proper bucket setting
Adaptive-LSH

Step 1: Use the ratio $c = R_2/R_1$ to characterize input data distribution
Adaptive-LSH

Step 2: Adapt bucket size according to $c$ and the lookup time target

Recall: the bucket size also affects lookup time
Reuse process

- Input data
- Lookup computation records with similar input
- Determine reuse outcome

Not reusable or "Bass Library" or "Sterling Library"
Reuse process

Input data

Lookup computation records with similar input


Determine reuse outcome

Not reusable or “Bass Library” or “Sterling Library”
H-kNN: strawman

Basic idea
\textit{k Nearest Neighbor}
H-kNN: strawman

Basic idea

\textit{k Nearest Neighbor}
H-kNN: strawman

Basic idea

\( k \) Nearest Neighbor

Take the result label of the largest cluster as the reuse outcome
kNN not enough
kNN not enough

Label of the largest cluster is not always the desirable reuse result
kNN not enough

Label of the largest cluster is not always the desirable reuse result
kNN not enough

Need high accuracy

Prefer less computation

Pill recognition

Google Lens

kNN does not give us control over the trade-off
kNN: what is needed?
kNN: what is needed?

Need to gauge dominance level of clusters
Why dominance level matters?
Why dominance level matters?

![Diagram showing why dominance level matters.](image)
Why dominance level matters?

A more dominant cluster → more confidence of accurate reuse
kNN: what is needed?

Need to gauge dominance level of clusters

Can then customize reuse trade-off
Homogeneity factor

kNN
Homogeneity factor

kNN

Clusters

<“6”, 1 >
<“9”, 1 >
<“4”, 3 >
A high $\theta$ ⇒ a large dominant cluster label (i.e., “4”) ⇒ a high confidence of correct reuse.
Homogemized-kNN (H-kNN)

1. Calculate homogeneity factor $\theta$
2. $\theta > \text{threshold } \theta_0$?
   - Yes: Reuse
   - No: Compute
Approximate computation reuse

• Algorithms for approximate computation reuse
  • A-LSH – fast lookup
  • H-kNN – reuse with accuracy guarantee
• FoggyCache system for cross-device reuse
FoggyCache architecture

- FoggyCache intercepts at library level
FoggyCache architecture

- FoggyCache intercepts at library level
FoggyCache architecture

- Cache is deployed at both edge server and client

Accelerate local reuse

Central point
System optimizations

**Stratified sampling:** Populate client cache to maximize reuse opportunity

**Speculative execution:** Precompute results that might be reused.

Details in the paper
Performance
General setup

**Devices**
- Linux desktop
- Google Nexus 9

**Visual workloads & datasets**
- Plant recognition: *ImageNet* subset
- Landmark recognition: *Oxford Buildings*, video feeds

**Audio workloads & datasets**
- Speaker identification: TIMIT acoustic dataset
Reuse accuracy vs saving computation

![Graph showing the relationship between homogeneity threshold and time saved.](image-url)
Reuse accuracy vs saving computation

Accuracy (%) vs Time saved (%) vs Homogeneity threshold (0.35 to 1)
# End-to-end performance

<table>
<thead>
<tr>
<th>Application</th>
<th>Latency (ms)</th>
<th>Energy (mJ)</th>
<th>Accuracy loss (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w/o</td>
<td>w/</td>
<td>w/o</td>
</tr>
<tr>
<td>Speaker Identification</td>
<td>13.1</td>
<td>4.2</td>
<td>30.4</td>
</tr>
<tr>
<td>Landmark Recognition</td>
<td>102.4</td>
<td>27.9</td>
<td>1315</td>
</tr>
<tr>
<td>Plant Recognition</td>
<td>269.6</td>
<td>99.8</td>
<td>3132</td>
</tr>
</tbody>
</table>
## End-to-end performance

<table>
<thead>
<tr>
<th>Application</th>
<th>Latency (ms)</th>
<th>Energy (mJ)</th>
<th>Accuracy loss (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w/o</td>
<td>w/</td>
<td>w/o</td>
</tr>
<tr>
<td>Speaker Identification</td>
<td>13.1</td>
<td>4.2</td>
<td>30.4</td>
</tr>
<tr>
<td>Landmark Recognition</td>
<td>102.4</td>
<td>27.9</td>
<td>1315</td>
</tr>
<tr>
<td>Plant Recognition</td>
<td>269.6</td>
<td>99.8</td>
<td>3132</td>
</tr>
</tbody>
</table>

**Over 3x latency reduction**
## End-to-end performance

<table>
<thead>
<tr>
<th>Application</th>
<th>Latency (ms)</th>
<th>Energy (mJ)</th>
<th>Accuracy loss (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w/o</td>
<td>w/</td>
<td>w/o</td>
</tr>
<tr>
<td>Speaker Identification</td>
<td>13.1</td>
<td>4.2</td>
<td>30.4</td>
</tr>
<tr>
<td>Recognition</td>
<td>102.4</td>
<td>27.9</td>
<td>1315</td>
</tr>
<tr>
<td>Plant Recognition</td>
<td>269.6</td>
<td>99.8</td>
<td>3132</td>
</tr>
</tbody>
</table>

**Up to 10x battery usage reduction**
## End-to-end performance

<table>
<thead>
<tr>
<th>Application</th>
<th>Latency (ms)</th>
<th>Energy (mJ)</th>
<th>Accuracy loss (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>w/o</td>
<td>w/</td>
<td>w/o</td>
</tr>
<tr>
<td>Speaker Identification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Landmark Recognition</td>
<td>102.4</td>
<td>27.9</td>
<td>1315</td>
</tr>
<tr>
<td>Plant Recognition</td>
<td>269.6</td>
<td>99.8</td>
<td>3132</td>
</tr>
</tbody>
</table>

Less than 5% accuracy loss
Conclusion

• **FoggyCache**: cross-device approximate computation reuse
  • Effectively eliminates fuzzy redundancy
• **Approximate computation reuse**
  • Promising new direction for optimizations
  • Algorithms are applicable to other scenarios
Thank you