Joint Optimization of DNN Partition and Scheduling for Mobile Cloud Computing

Yubin Duan and Jie Wu
Temple University
Background

• DNN Inference on Mobile Devices
  • Inference *latency* matters
  • Use pre-trained DNNs
  • Execute forward propagation

• Cooperative Deep Inference
  • Powerful cloud servers
  • Fast communication channels
  • Offload computation workload
Motivation

• Duplicated Inference Tasks
  • Simultaneously arrive
  • Auto pilot, AR/VR

• Cooperative Inference Pipeline
  • Reduce inference latency
  • Hide comm. behind mobile comp.
  • Cloud comp. is negligible
  • Partition and scheduling problem
    • Minimize overall latency for duplicated inference tasks
Challenges

• Exponential Partition Plans
  • Each DNN has # layers cut points
  • Partitions are individual

• Partition-Scheduling Correlation
  • Best individual partition
  • Best pipeline scheduling ≠
  • Optimal overall latency
Model

- Chain Structure DNNs
  - E.g.: AlexNet
  - Job shop scheduling
  - NP-hard
- DAG Structure DNNs
  - E.g.: GoogLeNet
  - DAG shop scheduling
  - NP-hard

Comp.: forward propagation before cut-point layers
Comm.: send outputs of cut-point layers
Chain Structure

- **Observations**
  - Mobile comp. time increases with layers
    - comp. workloads of layers are similar
  - Comm. time decreases with layers
    - pooling usually reduces tensor size by half
    - group layers if tensor sizes are not reduced
  - Cloud comp. time is negligible

- **Observations Diagram**
  - Mobile comp. time increases with layers
  - Comm. time decreases with layers
  - Cloud comp. time is negligible
Chain Structure (cont’d)

• Assumptions
  • Mobile comp. time: linear functions
  • Comm. time: convex functions

• Relax Problem to Continuous Domain
  • Optimal cut-point \( x^* \) for all DNNs:
    • mobile comp. time = comm. time

• Original Problem in Discrete Domain
  • Two types of cuts:
    • Comm.-heavy: left to \( x^* \)
    • Comp.-heavy: right to \( x^* \)
    • Adjust ratio of two types of cuts to fill the gap
DAG Structure

- Convert to Multi-Path DAG
  - Duplicate fork and join nodes

- Partition and Schedule Each Path
  - Treat each path as an independent task
  - Include duplicated nodes for scheduling
  - Exclude duplicated nodes for execution
    - Memorize layer outputs with hash tables
Experiment

• Testbed
  • Mobile:
    • Raspberry Pi 4 model B
  • Cloud:
    • Lab server (i7, GTX1080, 32GB)

• Comparison Algorithms
  • LO: local only
  • CO: cloud only
  • PO: partition only (state-of-the-art)
  • JPS: joint partition and scheduling

<table>
<thead>
<tr>
<th>Model</th>
<th>3G PO</th>
<th>JPS</th>
<th>4G PO</th>
<th>JPS</th>
<th>Wi-Fi PO</th>
<th>JPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlexNet</td>
<td>0</td>
<td>22.06</td>
<td>33.33</td>
<td>42.11</td>
<td>63.91</td>
<td>73.43</td>
</tr>
<tr>
<td>MobileNet-v2</td>
<td>27.60</td>
<td>56.73</td>
<td>60.00</td>
<td>78.83</td>
<td>82.81</td>
<td>84.69</td>
</tr>
<tr>
<td>GoogLeNet</td>
<td>0</td>
<td>52.83</td>
<td>56.13</td>
<td>71.93</td>
<td>66.63</td>
<td>72.17</td>
</tr>
<tr>
<td>ResNet18</td>
<td>0</td>
<td>0.73</td>
<td>1.46</td>
<td>28.22</td>
<td>58.52</td>
<td>58.52</td>
</tr>
</tbody>
</table>

Latency reduction ratio compared with LO
Experiment Results

- Significantly reduced latency with negligible scheduling overhead

![Graphs showing time (ms) for different models under various conditions.](image)

- Wi-Fi (18.88 Mbps)
- 3G (1.1 Mbps)
- 4G (5.85 Mbps)
- JPS overhead
Conclusion

- Joint Partition and Scheduling
  - Cooperative DNN inference
  - Reduce DNN inference latency

- Chain-Structure DNNs
  - Relax problem with optimal solution
  - Two types of DNN cuts

- DAG-Structure DNNs
  - Multi-path DNN conversion
Q&A

yubin.duan@temple.edu