Adaptive Distributed Convolutional Neural Network Inference at the Network Edge with ADCNN

17 August 2020
ICPP 2020

Sai Qian Zhang, Jieyu Lin, Qi Zhang
Executing DNN Inference Tasks for End Users

Option 1: edge only

- Image
- Audio
- Video

Edge devices

Limited computing capability

Option 2: cloud only

- Image
- Audio
- Video

Cloud data center

Large communication overhead

- Using edge device to handle the end user data leads to a long processing time, while using cloud server to process the end user data acquires a large communication delay.
Motivation

- Edge devices
  - Resource-limited
  - Pervasive

- **Adaptive Distributed Convolutional Neural Network (ADCNN)**
  - We propose a framework for agile execution of inference tasks on edge clusters for Convolutional Neural Networks (CNNs)

- Challenges
  - Reduce the inference latency while keeping the accuracy performance
  - Device heterogeneity and performance fluctuation
  - Applicable to different CNN models
Agenda

- Background
- CNN partitioning strategies
- ADCNN framework
- Modification on CNN architecture
- Evaluation
- Conclusion
The weight filters slide across the ifmaps. The dot product between the entries of each ifmap and weight filter are calculated at each position.
Background -- CNN Workload Characteristics

- Earlier layers take much longer to process than the later layers.
In channelwise partition, each node needs to exchange their partially accumulated ofmaps to produce final ofmaps, which may lead to a significant communication overhead.
In spatial partition, each tile needs to transmit their data halo in order to compute the correct result.
The cross-tile information transfer can be eliminated by padding the edge pixels with zeros.
ADCNN Framework

Step 1

Progressive Retraining

Original CNN model → Output CNN model

Step 2

Edge device cluster

Conv node → Central node → Dog

Tiles → Conv node → Conv node
The Conv nodes need to transmit the intermediate results to the Central node, which may still cause a significant communication overhead.
We modify the CNN model for reducing this communication overhead.

We adopt progressive retraining by adding the modification on the CNN architecture.

![Diagram of CNN Topology modifications](image)

1. Apply clipped ReLU
2. Quantization
3. Unroll the neurons
4. RLE

<table>
<thead>
<tr>
<th>Output from the CONV nodes</th>
<th>Apply clipped ReLU</th>
<th>Quantization</th>
<th>Unroll the neurons</th>
<th>RLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3 0.5 2.3 0.1</td>
<td>1.1 0.0 1.8 0.0</td>
<td>1.0 0.0 2.0 0.0</td>
<td>[1,0,2,0,1,0,0,0,0,0,0,2,0,0,0]</td>
<td>[1,1,2,1,7,2,3]</td>
</tr>
<tr>
<td>1.2 -0.2 0.1 0.1</td>
<td>1.0 0.0 0.0 0.0</td>
<td>1.0 0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-3.8 0.3 0.1 -2.2</td>
<td>0.0 0.1 0.0 0.0</td>
<td>0.0 0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.5 0.1 0.2 -1.3</td>
<td>1.8 0.0 0.0 0.0</td>
<td>2.0 0.0 0.0 0.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ADCNN Architecture

ADCNN takes advantage of the fine-grained, fully independent tiles generated by FDSP and adapt it to dynamic conditions, allowing it to achieve fine-grained load balancing across heterogeneous edge nodes.
We evaluate different CNN models from different applications.

Accuracy degradations are around 1% for 8 by 8 FDSP on the input sample.
Inference Latency Comparison

- We implement ADCNN system with nine identical Raspberry Pi devices which simulate the edge devices. Among these nine devices, eight are used as Conv nodes, and the rest one is used as the Central node.
- Baselines:
  - Single device scheme
  - Remote cloud scheme
- ADCNN decreases the average processing latency by 6.68x and 4.42x, respectively.
ADCNN Performance in Dynamic Environment

- We adjust the CPU processing speed on four of the Conv nodes (node 5,6,7,8) in the middle of the processing 50 input images, and detect its impact on tile assignment and overall inference latency.

- ADCNN can handle the dynamic condition on the node performance effectively.
Conclusion

- We introduce ADCNN, a distributed inference framework which jointly optimize CNN architecture and computing system for better performance in dynamic network environments.

- ADCNN applies FDSP to partition the compute-intensive convolutional layers into many small independent computational tasks which can be executed in parallel on separate edge devices.

- ADCNN system can take advantage of the fine-grained, fully independent tiles generated by FDSP and adapt it to dynamic conditions, allowing it to achieve fine-grained load balancing across heterogeneous edge nodes.

- Compared to existing distributed CNN inference approaches, ADCNN provides up to 2.8x lower latency, while achieving a competitive inference accuracy. Additionally, ADCNN can quickly adapt to the variations on edge device performance.