Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy

En Li, Zhi Zhou, Xu Chen

Sun Yat-Sen University
School of Data and Computer Science
The rise of artificial intelligence

- **Deep learning** is a popular technique that have been applied in many fields

- Object Detection
- Voice Recognition
- Image Semantic Segmentation
Why is deep learning successful

- Deep neural network is an important reason to promote the development of deep learning
Deep Learning applications can **not be well supported** by today’s mobile devices due to the large amount of computation.
What about Cloud Computing?

Under a cloud-centric approach, large amounts of data are uploaded to the remote cloud, resulting in **high end-to-end latency** and **energy consumption**.
Exploiting of Edge Computing

- By pushing the cloud capacities from the network core to the network edges (e.g., base stations and Wi-Fi access points) in close to devices, edge computing enables low-latency and energy-efficient performance.
## Existing effort of Edge Intelligence

<table>
<thead>
<tr>
<th>Framework</th>
<th>Highlight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neurosurgeon (ASPLOS 2017)</td>
<td>Deep learning model partitioning between cloud and mobile device, intermediate data offloading</td>
</tr>
<tr>
<td>Delivering Deep Learning to Mobile Devices via Offloading (SIGCOMM VR/AR Network 2017)</td>
<td>Offloading video input to edge server, according to network condition</td>
</tr>
<tr>
<td>DeepX (IPSN 2016)</td>
<td>Deep learning model are partitioned on different local processors</td>
</tr>
<tr>
<td>CoINF (arxiv 2017)</td>
<td>Deep learning model partitioning between smartphones and wearables</td>
</tr>
</tbody>
</table>

Existing effort focus on data offloading and local optimization
System Design

Our Goal

- With the collaboration between edge server and mobile device, we want to tune the latency of a deep learning model inference

Two Design Knobs

- Deep Learning Model Partition
- Deep Learning Model Right-sizing
Two Design Knobs

- Deep Learning Model Partition
- Deep Learning Model Right-sizing

AlexNet Layer Latency on Raspberry Pi & Layer Output Data Size

Deep Learning Model Partition

Two Design Knobs

- Deep Learning Model Partition
- Deep Learning Model Right-sizing

AlexNet with BranchyNet[2] Structure

A Tradeoff

Early-exit naturally gives rise to the latency-accuracy tradeoff (i.e., early-exit harms the accuracy of the inference).

Problem Definition

For mission-critical applications that typically have a predefined latency requirement, our framework maximizes the accuracy without violating the latency requirement.
System Overview

◆ Offline Training Stage

◆ Online Optimization Stage

◆ Co-Inference Stage

Mobile Device → Edge Server → Mobile Device

Edge Server → DNN Optimizer

Layer Prediction Models

Exit Point 1 → Exit Point 2

a) Latency Requirement

b) BranchyNet Structure

c) Regression Models

d) Selection of Exit Point and Selection of Partition Point

a)

b)

c)

d)
System Overview

- **Offline Training Stage**
  - Training regression models for layer runtime prediction
  - Training AlexNet with BranchyNet structure

- **Online Optimization Stage**

- **Co-Inference Stage**
System Overview

- Offline Training Stage
- Online Optimization Stage
- Co-Inference Stage

- Searching for exit point and partition point

Diagram:
- a) Latency Requirement
- b) BranchyNet Structure
- c) Regression Models
- d) Selection of Exit Point and Selection of Partition Point

Flowchart:
- DNN Optimizer
- FC
- CONV
- CONV
System Overview

- **Offline Training Stage**
- **Online Optimization Stage**
- **Co-Inference Stage**

Select one exit point

Find out the partition point
Experimental Setup

- Deep Learning Model
  - AlexNet with five exit point (built on Chainer deep learning framework)
  - Dataset: Cifar-10
  - Trained on a server with 4 Tesla P100 GPU
- Local Device: Raspberry Pi 3b
- Edge Server: A desktop PC with a quad-core Intel processor at 3.4 GHz with 8 GB of RAM
Experiments

Regression Model

Table 1: The independent variables of regression models

<table>
<thead>
<tr>
<th>Layer Type</th>
<th>Independent Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>amount of input feature maps, (filter size/stride)^2*(num of filters)</td>
</tr>
<tr>
<td>Relu</td>
<td>input data size</td>
</tr>
<tr>
<td>Pooling</td>
<td>input data size, output data size</td>
</tr>
<tr>
<td>Local Response</td>
<td>input data size</td>
</tr>
<tr>
<td>Normalization</td>
<td></td>
</tr>
<tr>
<td>Dropout</td>
<td>input data size</td>
</tr>
<tr>
<td>Fully-Connected</td>
<td>input data size, output data size</td>
</tr>
<tr>
<td>Model Loading</td>
<td>model size</td>
</tr>
</tbody>
</table>
## Experiments

### Regression Model

**Table 2: Regression Models**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Edge Server Model</th>
<th>Mobile Device Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolution</td>
<td>$y = 6.03e-5 \times x_1 + 1.24e-4 \times x_2 + 1.89e-1$</td>
<td>$y = 6.13e-3 \times x_1 + 2.67e-2 \times x_2 - 9.909$</td>
</tr>
<tr>
<td>Relu</td>
<td>$y = 5.6e-6 \times x + 5.69e-2$</td>
<td>$y = 1.5e-5 \times x + 4.88e-1$</td>
</tr>
<tr>
<td>Pooling</td>
<td>$y = 1.63e-5 \times x_1 + 4.07e-6 \times x_2 + 2.11e-1$</td>
<td>$y = 1.33e-4 \times x_1 + 3.31e-5 \times x_2 + 1.657$</td>
</tr>
<tr>
<td>Local Response Normalization</td>
<td>$y = 6.59e-5 \times x + 7.80e-2$</td>
<td>$y = 5.19e-4 \times x + 5.89e-1$</td>
</tr>
<tr>
<td>Dropout</td>
<td>$y = 5.23e-6 \times x + 4.64e-3$</td>
<td>$y = 2.34e-6 \times x + 0.0525$</td>
</tr>
<tr>
<td>Fully-Connected</td>
<td>$y = 1.07e-4 \times x_1 - 1.83e-4 \times x_2 + 0.164$</td>
<td>$y = 9.18e-4 \times x_1 + 3.99e-3 \times x_2 + 1.169$</td>
</tr>
<tr>
<td>Model Loading</td>
<td>$y = 1.33e-6 \times x + 2.182$</td>
<td>$y = 4.49e-6 \times x + 842.136$</td>
</tr>
</tbody>
</table>
Experiments

Result

- Selection under different bandwidths

The higher bandwidth leads to higher accuracy

(a) Selection under different bandwidths
Experiments

- Inference Latency under different bandwidths

Our proposed regression-based latency approach can well estimate the actual deep learning model runtime latency.

(b) Model runtime under different bandwidths
Experiments

- Selection under different latency requirements

A larger latency goal gives more room for accuracy improvement

(c) Selection under different latency requirements
Experiments

- Comparison with other methods

The inference accuracy comparison under different latency requirement
Key Take-Aways

- On demand accelerating deep learning model inference through device-edge synergy
- Deep Learning Model Partition
  Deep Learning Model Right-sizing
- Implementation and evaluations demonstrate effectiveness of our framework
Future Work

- More Devices
- Energy Consumption
Future Work

- Deep Reinforcement Learning Technique

Deep Reinforcement Learning for Model Partition
Thank you

Contact:
lien5@mail2.sysu.edu.cn
zhouzhi9@mail.sysu.edu.cn
chenxu35@mail.sysu.edu.cn