Distributed Inference with Deep Learning Models across Heterogeneous Edge Devices

Chenghao Hu, Baochun Li
University of Toronto
Model Deployment: Cloud or Edge?

Cloud -> transmission overhead and privacy issue
Model Deployment: Cloud or Edge?

Edge -> Limited computation capacity leads to high latency

Another idea: distribute the inference workload for acceleration
How to partition the inference workload

- Sequential partition
  - Partition the model *layer-wise*
  - The computation resources are underutilized

- Parallel partition
  - Parallel paths executed simultaneously
Convolution Operation

“Sliding window” applied on the image step by step

![Convolution Operation Diagram]
Model Partition - Single Conv Layer

Step 1: decide the range of output partition

Step 2: calculate the range of the required input

Step 3: feed the input partition to the convolution layer
Model Partition - Chained Layers

Chain structured model, e.g., VGG-16

Trace all the way back to the first layer
Model Partition - Computation Graph

Directed Acyclic Graph (DAG) structured model, e.g., ResNet

Some computation graphs can be easily turned into a chain, and manually fix the layer dependency.
Model Partition - Computation Graph

Some not... like YoloV5
Our Design - EdgeFlow Overview

- Setup Phase
  - partition the computation graph into *execution units*
  - The *layer dependency is maintained* by the communication between execution units
Our Design - EdgeFlow Overview

- Inference Phase
  - Execution units collaboratively finish the inference
  - **Equivalent** result as computed on a single device
Model Partitioning

- Layer partitioning
  - Each execution unit computes *part* of the output of this layer
  - Calculate the *required part* of the input needed to complete the computation task
  - Update the forward table of preceding execution units

Problem: how to find the optimum partition scheme?
Problem Formulation

- Assume $n$ available devices, output features of current layer ranges from row 0 to row $H$

- The partition decision variables can be expressed as an integer vector $\mathbf{x} = (x_0, x_1, \ldots, x_n)$
  - device $i$ computes output ranges from row $x_{i-1} + 1$ to row $x_i$

\[
x_i \in \mathbb{Z}^+, i = 0, 1, \ldots, n
\]
\[
x_0 = 0, x_n = H
\]
\[
x_0 \leq x_1 \leq \cdots \leq x_n
\]
Problem Formulation

- Objective: finish time of the current layer $l$
  - $T_{l,i}$ denote the time that device $i$ finish its partition of layer $l$

- The optimization problem can be expressed as

\[
\begin{align*}
\min_x & \quad \max(T_{l,1}, T_{l,2}, \ldots, T_{l,n}) \\
\text{s.t.} & \quad x_i \in \mathbb{Z}^+, i = 0, 1, \ldots, n \\
& \quad x_0 = 0, x_n = H \\
& \quad x_0 \leq x_1 \leq \cdots \leq x_n
\end{align*}
\]
Problem Formulation

- $T_{l,i}$ estimation: transmission time + computation time

\[ T_{l,i} = t_{\text{trans}}(i; l) + t_{\text{comp}}(i; l) \]

- Computation time can be estimated with a pre-trained linear regression model

\[ t_{\text{comp}}(i) = Y_i(x_i - x_{i-1}; l). \]

- Transmission time can be estimated by

\[ t_{\text{trans}}(i; l) = \max_{j \in \{1, \ldots, n\}, m \in M} 1\{p_{m,i,j} > 0\} (T_{m,j} + \frac{p_{m,i,j}}{B_{i,j}}) \]
Problem Transformation

- Original problem

\[
\begin{align*}
\min_x & \quad \max(T_{l,1}, T_{l,2}, \ldots, T_{l,n}) \\
\text{s.t.} & \quad x_i \in \mathbb{Z}^+, i = 0, 1, \ldots, n \\
& \quad x_0 = 0, x_n = H \\
& \quad x_0 \leq x_1 \leq \cdots \leq x_n
\end{align*}
\]

- Step 1: introducing auxiliary variable and relax the integer constraint

\[
\begin{align*}
\min_{x, \lambda} & \quad \lambda \\
\text{s.t.} & \quad T_{l,i} \leq \lambda, i \in \{1, \ldots, n\} \\
& \quad x_i \in \mathbb{Z}^+, i = 0, 1, \ldots, n \\
& \quad x_0 = 0, x_n = H \\
& \quad x_0 \leq x_1 \leq \cdots \leq x_n
\end{align*}
\]
Problem Transformation

- Step 2: removing the indicator function

\[ 1_{\{p_{m,i,j} > 0\}}(T_{m,j} + \frac{p_{m,i,j}}{B_{i,j}}) + Y_i(x_i - x_{i-1}; l) \leq \lambda \]

The finish time of layer \( m \) at different device should be roughly the same

As long as \( T_{m,j} \) is not greater than other required devices, the constraint is loose, and won’t affect the result.
Problem Transformation

› Step 3: re-express the transmission size

\( p_{m,i,j} \) is the overlapping area between the **required input range** \((s_i, e_i)\) and the **output range** \((x_{m,j-1}, x_{m,j})\) of layer \(m\) at device \(j\)

\[
p_{m,i,j} = \min(e_i, x_{m,j}) - \max(s_i, x_{m,j-1}) \\
= \min(e_i - s_i, e_i - x_{m,j-1}, x_{m,j} - s_i, x_{m,j} - x_{m,j-1})
\]

\[
\min_{p_{m,i,j}} - p_{m,i,j} \\
s.t. \quad p_{m,i,j} \leq e_i - s_i, \quad p_{m,i,j} \leq e_i - x_{m,j-1}, \\
\quad p_{m,i,j} \leq x_{m,j} - s_i, \quad p_{m,i,j} \leq x_{m,j} - x_{m,j-1}.
\]
Problem Transformation

- Linear Programming Approximation

\[
\begin{align*}
\min_{x, \lambda, p} \quad & \lambda - \sum_{m,i,j} p_{m,i,j} \\
\text{s.t.} \quad & x_0 = 0, x_n = H \\
& x_0 \leq x_1 \leq \cdots \leq x_n \\
& T_{m,j} + \frac{p_{m,i,j}}{B_{i,j}} + Y_i(x_i - x_{i-1}; l) \leq \lambda \\
& p_{m,i,j} \leq e_i - s_i, \quad p_{m,i,j} \leq e_i - x_{m,j-1}, \\
& p_{m,i,j} \leq x_{m,j} - s_i, \quad p_{m,i,j} \leq x_{m,j} - x_{m,j-1}.
\end{align*}
\]
Model Partition

- Partition the model layer by layer in topological order
- Solve the LP problem for each layer to obtain the partition scheme
- The finish time estimation of previous layer becomes a parameter of the optimization problem of the following layers

\[ T_{m,j} + \frac{p_{m,i,j}}{B_{i,j}} + Y_i(x_i - x_{i-1}; l) \leq \lambda \]
Padding Issue

- Directly feeding the input partition to the conv/pool layer may not yield the correct output
Padding Issue

- Solution: pre-padding mechanism
  - Set the padding parameter of conv/pool layer to 0
  - Manually add paddings when necessary

\[
i_s = o_s \times \text{stride} - \text{padding},
\]

\[
i_e = (o_e - 1) \times \text{stride} + \text{kernel size} - \text{padding},
\]

\[
\text{upper padding} = \begin{cases} 
-i_s, & i_s < 0 \\
0, & \text{otherwise}
\end{cases},
\]

\[
\text{bottom padding} = \begin{cases} 
i_e - H_i, & i_e > H_i \\
0, & \text{otherwise}
\end{cases}.
\]
Inference Phase

‣ The units will be executed when the input requirements are satisfied

‣ The output will be forwarded to fulfill the requirement of next execution unit

‣ Intermediate results *flow* through execution units to finish the inference

System name: EdgeFlow
Evaluation

- 2 deep learning models
  - VGG-16: Classic image classification model in chain structure
  - YoloV5X: Latest object detection model with complicated structure
- 6 heterogeneous virtual machines
- Baselines
  - Local: deploy the model on a single device
  - Existing methods: DeepThings and CoEdge
Evaluation

Proposed method (EdgeFlow-P) achieves lowest inference latency with both models
Evaluation

- Partition scheme of YoloV5
  - DeepThings: redundant computation in the early layers
  - CoEdge: workload gradually concentrates on a single device
  - EdgeFlow: relatively even distribution among devices
Evaluation

EdgeFlow-H and CoEdge share the same partition scheme, yet still faster than CoEdge
Conclusion

- The model structure significantly affects the performance of existing distributed inference systems.

- *EdgeFlow* breaks the layer into execution units, and maintain the complicated layer dependencies by controlling the flow of intermediate results.

- Evaluation results show *EdgeFlow* has a distinct advantage, especially with complicated DAG-structured model.
Contact: ch.hu@mail.utoronto.ca