Placeto: Learning Generalizable Device Placement Algorithms for Distributed Machine Learning

R. Addanki, S. B. Venkatakrishnan, S. Gupta, H. Mao, M. Alizadeh

Presenter: Qianyi Liu
Background

Key challenge for **distributed training**: split a large model across multiple heterogeneous devices to achieve the fastest possible training speed

- Typical approach: left to human experts
- A new solution: automated approach to device placement based on reinforcement learning
Prior works

- Mirhoseini et al. (2017): train an RNN to process a computation graph and predict a placement for each operation.

Key drawbacks:
- Long time
- Don’t learn generalisable device placement strategy -> requires retraining.
Placeto

- Use RL to learn an efficient algorithm for device placement for a given family of computation graphs
- Two key ideas:

  **Idea 1: Find a sequence of iterative placement improvements**
  - Simpler to learn $\rightarrow$ training efficiency $\uparrow$

  **Idea 2: use graph embeddings to encode the computation graph structure**
  - Doesn’t depend on sequential order of nodes
  - GNN + message passing
  - Generalisability $\uparrow$
Learning procedure: a Markov decision process

Placement improvement MDP steps

Step t=0
Action $a_1$: Device 2

Step t=1
Action $a_2$: Device 1

Step t=2
Action $a_3$: Device 2

Step t=3
Action $a_4$: Device 2

Final placement

End of episode
RL to learn the MDP policy – a neural network
Graph embedding step 1: Compute per-group attributes

Op group feature:
(totai_runtime, output_tensor_size, current_placement, is_node_current, is_node_done)
Graph embedding step 2: Local neighbourhood summarisation

- A sequence of message passing steps to aggregate neighbourhood information for each node

\[
x_v \leftarrow g(\sum_{u \in \xi(v)} f(x_u))
\]

- Two directions: top-down + bottom-up
Graph embedding step 3: Pooling summaries

- Create a global summary of the entire graph, from the point of view of node v
Rewards are generated from a simulator rather than actual hardware measurement during training.
# Evaluation: performance

## Metric:

1) Runtime of the best placement found
2) Time taken to find the best placement (# of placement evaluations)

<table>
<thead>
<tr>
<th>Model</th>
<th>Placement runtime (sec)</th>
<th>Training time (# placements sampled)</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>#GPUs</td>
<td>Expert</td>
<td>Scotch</td>
</tr>
<tr>
<td>Inception-V3</td>
<td>12.54</td>
<td>1.56</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1.15</td>
<td>1.74</td>
</tr>
<tr>
<td>NMT</td>
<td>33.5</td>
<td>OOM</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>OOM</td>
<td>OOM</td>
</tr>
<tr>
<td>NASNet</td>
<td>37.5</td>
<td>1.28</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.84</td>
<td>1.22</td>
</tr>
</tbody>
</table>
Evaluation: generalisability
Takeaways

Pros:

- Novelty: first attempt to use GNN to encode graph structure in device placement optimisation — learns generalisable placement policy
- Impressive performance: find better placements faster than RNN-based approach

Cons:

- Operator needs to be manually grouped based on heuristics — not an end-to-end solution
- Generalisability is limited to graphs from the same family
Discussion