Automating Cloud Deployment for Deep Learning Inference of Real-time Online Services

Yang Li  Zhenhua Han  Quanlu Zhang  Zhenhua Li  Haisheng Tan
DNN-driven Real-time Services

Image Classification

Speech Recognition

Neural Machine Translation
Cloud Deployment

DNN Model

Low Latency

Cost Efficiency

Network transmission time
Task scheduling time
DNN inference time
......
Trade-off between execution time and economic cost

require
Cloud Deployment

DNN Model

require

Low Latency

Network transmission time
Task scheduling time
DNN inference time

......

Cost Efficiency

Trade-off between execution time and economic cost

<table>
<thead>
<tr>
<th>Model</th>
<th>Min Inference Cost</th>
<th>Max Inference Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNNLM</td>
<td>$0.17</td>
<td>$1.15</td>
</tr>
<tr>
<td>Inception-V3</td>
<td>$0.40</td>
<td>$6.39</td>
</tr>
<tr>
<td>VGG16</td>
<td>$0.58</td>
<td>$7.26</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>$0.60</td>
<td>$4.74</td>
</tr>
<tr>
<td>AlexNet</td>
<td>$0.59</td>
<td>$4.45</td>
</tr>
</tbody>
</table>

Inference cost (10000 times) of different models across different cloud configurations.
Here come the problems

I want to deploy my face recognition service on the cloud.

Given a configuration, how can I minimize the DNN inference time?

How should I choose the cloud configuration?
Choose Cloud Configurations

- Choose cloud configurations

Both of them provide over 100 types of cloud configurations!

Example: 2 series from over 40 series on Azure!

<table>
<thead>
<tr>
<th>Instance</th>
<th>Core</th>
<th>RAM</th>
<th>Temporary storage</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC6 Promo</td>
<td>6</td>
<td>56 GiB</td>
<td>340 GiB</td>
<td>1x K80</td>
</tr>
<tr>
<td>NC12 Promo</td>
<td>12</td>
<td>112 GiB</td>
<td>680 GiB</td>
<td>2x K80</td>
</tr>
<tr>
<td>NC24 Promo</td>
<td>24</td>
<td>224 GiB</td>
<td>1,440 GiB</td>
<td>4X K80</td>
</tr>
<tr>
<td>NC24r Promo</td>
<td>24</td>
<td>224 GiB</td>
<td>1,440 GiB</td>
<td>4X K80</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Instance</th>
<th>Core</th>
<th>RAM</th>
<th>Temporary storage</th>
<th>GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC6s v2</td>
<td>6</td>
<td>112 GiB</td>
<td>736 GiB</td>
<td>1X P100</td>
</tr>
<tr>
<td>NC12s v2</td>
<td>12</td>
<td>224 GiB</td>
<td>1,474 GiB</td>
<td>2X P100</td>
</tr>
<tr>
<td>NC24rs v2</td>
<td>24</td>
<td>448 GiB</td>
<td>2,948 GiB</td>
<td>4X P100</td>
</tr>
<tr>
<td>NC24s v2</td>
<td>24</td>
<td>448 GiB</td>
<td>2,948 GiB</td>
<td>4X P100</td>
</tr>
</tbody>
</table>
Reduce DNN Inference Time

• A DNN model can have hundreds to thousands of operations.
• Each operation can be placed on a list of feasible devices (e.g., CPUs or GPUs) to reduce execution time.

Example: the computation graph of Inception-V3

How to choose the optimal device placement plan?
Challenge

- Huge search space

- Inference cost
  - the price of the cloud configuration * inference time.
    - ($/hour)
    - (second/request)

How to automatically determine the cloud configuration and device placement for the inference of a DNN model, so as to minimize the inference cost while satisfying the inference time constraint (QoS)?

Black-box Optimization!
AutoDeep

• Given
  • A DNN model
  • Inference time constraint (QoS constraint)

• Goal
  • Compute the cloud deployment with the lowest inference cost

• Two-fold joint optimization
  • Cloud configuration searching
    • Black-box method: Bayesian Optimization (BO)
  • Device placement optimization
    • Markov decision process: Deep Reinforcement Learning (DRL)
Black-box Optimization

• Regard the inference cost of a given DNN model with a QoS constraint as a black-box function $f$.

Cloud Configuration Pool

select
input

$f$

goal

Minimize $f$

iterations

Optimize the DNN device placement in the selected cloud configuration and calculate inference cost (observation)

A (nearly) optimal cloud configuration with the optimized device placement plan of the DNN.
Optimize Device Placement – DRL Model
AutoDeep: Architectural Overview

1. Cloud Configuration 1 → Constrained Bayesian Optimization
2. Trial Configuration
3. RL-Based Device Placement
4. Inference Performance

Configuration Pool

Real-time Online Services
Execution Environment

Device Placement
Experiments – Device Placement

• Google RL
  • Algorithm designed by Mirhoseini et al.
  • \[\text{ICML17} \text{ Device placement optimization with reinforcement learning}\]

• Expert Designed
  • Hand-crafted placements given by Mirhoseini et al.

• Single GPU
  • Execution on a single GPU.

<table>
<thead>
<tr>
<th>CPU</th>
<th>GPU</th>
<th>GPU Number</th>
<th>Price (USD/hour/GPU)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core i7-5930K</td>
<td>GTX 980Ti</td>
<td>1-3</td>
<td>0.56</td>
</tr>
<tr>
<td>Core i7-6850K</td>
<td>GTX 1080</td>
<td>1-4</td>
<td>0.70</td>
</tr>
<tr>
<td>Xeon E5-2690 v4</td>
<td>P100</td>
<td>1-4</td>
<td>2.07</td>
</tr>
<tr>
<td>Xeon E5-2690 v3</td>
<td>K80</td>
<td>1-4</td>
<td>0.90</td>
</tr>
</tbody>
</table>
Experiments

QoS Constraint Increasing

• LCF (Lowest Cost First)
  • Try configurations in the ascending order of their unit price
• Uniform
  • Try configurations with uniform probability

Inference cost of RNNLM under varying QoS constraint

Inference cost of Inception-V3 under varying QoS constraint
Experiments

AutoDeep: Lowest search cost
Future work

- Improve learning efficiency
  - Developing a general network architecture so that re-training is not needed for new DNN inference models
  - Accelerate DRL training process
  - …

- Optimize the system efficiency
  - Over 90% of searching time is wasted to initialize the DNN computation graph
  - Allowing placing operations in a fine-grained manner (i.e., without restarting a job)

My Email: liyang14thu@gmail.com