Transfer Learning for Performance Modeling of Deep Neural Network Systems

Md Shahriar Iqbal

Lars Kotthoff

Pooyan Jamshidi
Performance can change across systems

Can we expect similar performance in new environment?

Environment 1.0
Dev/Staging

Push

Environment 2.0
Production

- Inference time
- Energy consumption
Case study: Performance behavior changes across environments and so does optimal configuration.

How can the developer tune parameters for energy consumption during source code development?
DNN system stack allows developers to tune parameters for energy consumption

- Network Level
- Model Compiler Level
- Deployment Level
- Hardware/OS Level

Core status, core frequency, gpu status, gpu frequency, memory controller frequency etc.

$2^{100}$ configurations possible

DNN Systems are highly configurable
Building configuration space of DNN systems

<table>
<thead>
<tr>
<th>Hardware/OS Level</th>
<th>Core status</th>
<th>Core status</th>
<th>gpu status</th>
<th>gpu frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ C = O_1 \times O_2 \times O_3 \times O_4 \times \ldots \times O_n \]

\[ c_1 \in C \quad 0 \times 1 \times 204000 \times 1 \times \ldots \times 4996000 \]

Inference time  \[ f_c(c_1) = 36.12 ms \]

Energy consumption  \[ f_c(c_1) = 1456.98 mW \]
A typical approach to understand system behavior is sensitivity analysis.

\[ c_1 \times 0 \times 0 \times 0 \times 0 \ldots \times 1 \quad y_1 = f(c_1) \]
\[ c_2 \times 0 \times 1 \times 0 \times 0 \ldots \times 1 \quad y_2 = f(c_2) \]
\[ c_3 \times 1 \times 0 \times 0 \times 0 \ldots \times 0 \quad y_3 = f(c_3) \]

\[ \ldots \ldots \ldots \]

\[ c_n \times 1 \times 1 \times 1 \times 0 \ldots \times 1 \quad y_n = f(c_n) \]

Training Set \[ f^* \rightarrow f \]

learn
Performance models could be in any form of black box models

\[ C_0 x_0 x_1 x_0 x_2 x_3 x_0 \ldots x_0 x_{20} \]

\[
\begin{align*}
C_1 & : 0x0x0x0x0\ldots x_1 \\
C_2 & : 0x0x1x0\ldots x_1 \\
C_3 & : 1x0x0x0\ldots x_0 \\
& \ldots \ldots \\
C_n & : 1x1x1x0\ldots x_1 \\
\end{align*}
\]

\[ \begin{align*}
y_1 &= f(c_1) \\
y_2 &= f(c_2) \\
y_3 &= f(c_3) \\
& \ldots \ldots \\
y_n &= f(c_n) \\
\end{align*} \]

\[ \text{Training Set} \quad \rightarrow \quad f^* \rightarrow f \]
Evaluating a performance model

\[ c_0 x_0 x_0 x_1 x_0 \ldots x_0 x_2 \]

\[ y_1 = f(c_1) \]
\[ y_2 = f(c_2) \]
\[ y_3 = f(c_3) \]

\[ \text{MAPE}(f^*, f) = \frac{|f^* - f|}{f} \times 100 \]

Training Set \[ f^* \rightarrow f \]
A Performance Model contains useful information about option interactions

\[ f: \mathbb{C} \rightarrow \mathbb{R} \]

\[ f(.) = 1.2 + 3O_1 + 5O_2 + 7O_7 + 8O_{12} - 3.4O_1O_7 + 5O_3O_7 + 11.2O_{12}O_7 \]
What are the roadblocks for performance modeling of DNN systems?

- Configuration space grows exponentially with addition of new options.
- Performance measurements are costly.

\[ 2^{20} = 1048576 \] configurations takes nearly 2 years for measurement considering only 60s on average per configuration.

Transfer learning can help to reuse the cheap measurements from old environments to new environments to quickly learn system behavior using performance modeling.
Direct Model Transfer Learning

Model learned in source environment is used directly in target environment
Linear/Non-Linear Model Shift Transfer Learning

Linear and non-linear model shift are learned using source model (all samples) and target model (random sub-samples).
Guided Sampling Transfer Learning

Influential configurations are sampled in target environment by studying options interactions from source environment to build performance model.
Experimental Setup

$\textbf{f}(s) = a_1O_1 + a_2O_3 + a_3O_7 + a_4O_1O_3 + a_5O_1O_7 + \ldots + a_7O_1O_3O_7$
## Results: Inference Time

<table>
<thead>
<tr>
<th>Type</th>
<th>Cost</th>
<th>Reduction of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Model Transfer</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Linear Model Shift</td>
<td>2.48 hours (0.15%)</td>
<td>3.48</td>
</tr>
<tr>
<td>Non-Linear Model Shift</td>
<td>2.48 hours (0.15%)</td>
<td>5.76</td>
</tr>
<tr>
<td>Guided Sampling</td>
<td>2.48 hours (0.15%)</td>
<td>22.93</td>
</tr>
</tbody>
</table>
Results: Energy Consumption

<table>
<thead>
<tr>
<th>Method</th>
<th>Cost</th>
<th>Reduction of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Model Transfer</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Linear Model Shift</td>
<td>2.48 hours (0.15%)</td>
<td>7.19</td>
</tr>
<tr>
<td>Non-Linear Model Shift</td>
<td>2.48 hours (0.15%)</td>
<td>16.31</td>
</tr>
<tr>
<td>Guided sampling</td>
<td>2.48 hours (0.15%)</td>
<td>42.85</td>
</tr>
</tbody>
</table>
Useful tool for practitioners

- To build performance models in new environments using only 2.44% of the configuration space.
- To avoid invalid configurations to quickly find optimal configurations.
- To learn the performance landscape of a system for performance debugging e.g., performance regression etc.
Summary

Case study: Performance behavior changes across environments and so does optimal configuration

How can the developer tune parameters for energy consumption during source code development?

Results: Inference Time

<table>
<thead>
<tr>
<th>Type of Hardware Change</th>
<th>Cost</th>
<th>Performance Boost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Model Transfer</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Linear Model Shift</td>
<td>2.48 hours (0.15%)</td>
<td>3.48</td>
</tr>
<tr>
<td>Non-Linear Model Shift</td>
<td>2.48 hours (0.15%)</td>
<td>5.70</td>
</tr>
<tr>
<td>Guided Sampling</td>
<td>2.48 hours (0.15%)</td>
<td>22.93</td>
</tr>
</tbody>
</table>

Results: Energy Consumption

<table>
<thead>
<tr>
<th>Type of Hardware Change</th>
<th>Cost</th>
<th>Performance Boost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct Model Transfer</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Linear Model Shift</td>
<td>2.48 hours (0.15%)</td>
<td>7.19</td>
</tr>
<tr>
<td>Non-Linear Model Shift</td>
<td>2.48 hours (0.15%)</td>
<td>15.31</td>
</tr>
<tr>
<td>Guided Sampling</td>
<td>2.48 hours (0.15%)</td>
<td>42.85</td>
</tr>
</tbody>
</table>

Guided Sampling
Transfer Learning

Influential configurations are sampled in target environment by studying options interactions from source environment to build performance model.