Search Space Optimization in Hardware-Aware Neural Architecture Search

Dennis Rieber, Joschka Theissen*, Thomas Elsken
Bosch Research, Germany
*RWTH Aachen University, Germany

Motivation

- A search space design that is not aware of the target hardware (HW) architecture can lead to prolonged search time.
- From experience, specific HW-architectures can execute some operators more efficiently than others.
- Only searching in the part of the search space with the most promising candidates can reduce the overall search effort.

BLOX NAS Benchmark

- We use the BLOX NAS Benchmark [5] as a baseline search space. It contains ~90k possible architectures. A DNN is constructed from three subsequent blocks, where each block is one of 45 sub-networks.

Search Space Pruning

- We investigate search space pruning prior to the search.
- An “efficiency” metric is used to evaluate all building blocks.
- The metric aims to quantify the computational effort necessary for a forward-pass of a single block and its potential to improve the accuracy of the network.
- This metric is then used to prune the search space, before the search is started.
- Evaluated metric: parameters/latency: this metric is fully hardware-aware while requiring no training.

Convergence Analysis

- We employ the generational distance, which measures the distance w.r.t. the theoretically optimal pareto front to evaluate convergence speed in different search spaces.
- Pruning significantly accelerates the speed of convergence.
- Very aggressive pruning (i.e., 90%) leads to very fast early convergence, but eventually to sub-optimal result.
- Comparing HW-aware metric-based pruning with random pruning: significantly worse performance than HW-aware pruning as well as un-pruned, baseline search space.

NAS Convergence with Search Space Pruning - Adreno 645 GPU

- The following table shows the generational distance at different stages during the evolution for three different devices and different pruning strategies.

<table>
<thead>
<tr>
<th>Evolutions</th>
<th>Adreno 645 Baseline</th>
<th>Adreno 645 50% Pruned</th>
<th>Adreno 645 30% Pruned</th>
<th>Adreno 645 10% Pruned</th>
<th>Adreno 645 0% Pruned</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.06</td>
<td>0.05</td>
<td>0.04</td>
<td>0.04</td>
<td>0.04</td>
</tr>
<tr>
<td>100</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>1000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>10000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>50000</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Results

- NAS yields similar results after just 200 evolutions on a pruned search space compared to 2000 evolutions on the baseline search space, while significantly outperforming the baseline search space at 200 evolutions:

Post-Pruning Solution Space

- Looking at all discovered architectures reveals, that search space pruning shifts the discovered space towards more promising regions with architectures of lower inference latency and higher test accuracy.

Conclusion

- Search Space Pruning demonstrates promising results across different hardware targets.
- Drastically reduced search time.
- NAS on unpruned search spaces also discovers as good or better solutions in theory, but doesn’t reach these levels within reasonable search time.
- Trade-off between quality and time-to-solution.

However:

- BLOX NAS is a “closed” problem, with a finite set of reasonably good solutions on a well-known task.
- In real-world applications, even the coarse structure of a good solution is not always known.
- Thus “open problems” with potentially infinite search spaces are a reality that needs to be faced.
- How well offline-methods work on open problems remains to be investigated.

References