Ordering Chaos
Memory-Aware Scheduling for Irregularly Wired Neural Networks on Edge Devices

Byung Hoon Ahn, Jinwon Lee, Jamie Lin, Hsin-Pai Cheng, Jilei Hou, Hadi Esmaeilzadeh
Motivation: Enabling Intelligence, Transition from Cloud to Edge

Intelligence moving from the Cloud to the Edge

- Low Latency
- Privacy
- Reliability

Intelligence is moving from **Cloud to Edge** for **Low Latency, Privacy, and Reliability**
Motivation: How to Make Deep Neural Networks More Efficient?
Motivation: Irregularly Wired Neural Networks

These **Efficient Networks** comprise of many **Irregular Wirings**
We classify them as **Irregularly Wired Neural Networks**
Motivation: Emerging Class of DNNs for Resource Constrained Scenarios

Certain class of networks require **less Resources for same Accuracy**
(a.k.a. **More Efficient Networks**)

- Multiply-and-accumulate (\#JMMJPOT)
- Top-1 ImageNet Accuracy (%)
- Number of Parameters (Millions)
Running Example: SwiftNet (ICCV-W’19)

<table>
<thead>
<tr>
<th>Size (8bits)</th>
<th>MACs</th>
<th>Peak Mem</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>249.7KB</td>
<td>57.4M</td>
<td>?</td>
<td>95.13%</td>
</tr>
</tbody>
</table>

224x224 Input Image

SwiftNet: Using Graph Propagation as Meta-knowledge to Search Highly Representative Neural Architectures: hsipaic@qti.qualcomm.com; dave.cheng@duke.edu
Running Example: SwiftNet (ICCV-W’19)

<table>
<thead>
<tr>
<th>Size (8bits)</th>
<th>MACs</th>
<th>Peak Mem</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>249.7KB</td>
<td>57.4M</td>
<td>800KB?</td>
<td>95.13%</td>
</tr>
</tbody>
</table>

Peak Memory Footprint: 800KB (> 250KB Requirement)

Today’s Frameworks are **Oblivious to "Peak Memory Footprint" Issue** When it come to **Irregularly Wired Neural Networks**
Running Example: SwiftNet (ICCV-W’19)

<table>
<thead>
<tr>
<th>Size (8bits)</th>
<th>MACs</th>
<th>Peak Mem</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>249.7KB</td>
<td>57.4M</td>
<td>200KB</td>
<td>95.13%</td>
</tr>
</tbody>
</table>

4x improvement in Peak memory footprint (c.f., today’s TF Lite scheduler = 800KB)
We cannot rely on human expert for scheduling all the time

Manual Work vs Automation

Laziness drives innovations that improve productivity
- Steven Shapiro
Our Solution
Automated Solution: Serenity (Ordering Chaos)

We propose an Automated Approach that:

1. Quickly **finds a memory-optimal schedule** for a fixed graph
2. Explores another dimension that **alleviates the memory footprint** of the graph
Search Space: Scheduling = Topological Ordering

While Conventional Network (e.g. AlexNet, ...) execution is "streamlined"
Irregularly Wired Neural Network execution is "not streamlined"
Search space is exponentially large and optimal solutions account for very very small fraction of the entire space.
Brute Force Algorithm for Topological Ordering

Many zero-indegree sets are redundant. Optimizing this eliminates redundancy.
Dynamic Programming Algorithm for Topological Ordering

Dynamic Programming-based Topological Ordering can speed-up the traversal of schedules significantly
Overlaying Problem Constraints

Overlaying these constraints gives Memory-optimal schedule of the nodes.
Dynamic Programming-based Scheduling

<table>
<thead>
<tr>
<th>Size (8bits)</th>
<th>MACs</th>
<th>Peak Mem</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>249.7KB</td>
<td>57.4M</td>
<td>200KB</td>
<td>95.13%</td>
</tr>
</tbody>
</table>

4x Improvement in Peak Memory Footprint
(c.f., today’s TF Lite scheduler = 800KB)

Output Activations In memory

SwiftNet: Using Graph Propagation as Meta-knowledge to Search Highly Representative Neural Architectures: hsinpaic@qti.qualcomm.com; dave.cheng@duke.edu
Identity Graph Rewriting

Channel-wise Partitioning

\[ \mu_{\text{peak}} = \sum \text{size}(x_i) + \text{size}(y) \]

Kernel-wise Partitioning

\[ \mu_{\text{peak}} = \max(\text{size}(x_i) + \text{size}(y)) \]

Graph Rewriting while maintaining the mathematical integrity allows further reduction in Peak Memory Footprint
Dynamic Programming-based Scheduling + Graph Rewriting

<table>
<thead>
<tr>
<th>Size (8bits)</th>
<th>MACs</th>
<th>Peak Mem</th>
<th>ACC</th>
</tr>
</thead>
<tbody>
<tr>
<td>249.7KB</td>
<td>57.4M</td>
<td>188KB</td>
<td>95.13%</td>
</tr>
</tbody>
</table>

12KB Further Improvement with Graph Rewriting  
(c.f., today’s TF Lite scheduler = 800KB)
Peak memory performance for different scheduling strategies:

<table>
<thead>
<tr>
<th>Scheduling Strategy</th>
<th>Peak Mem</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Optimization + Partial Convolution</td>
<td>200KB</td>
<td>2 days</td>
</tr>
<tr>
<td>(Automatic) Dynamic Programming-based Scheduling</td>
<td>200KB</td>
<td>?</td>
</tr>
<tr>
<td>(Automatic) Dynamic Programming-based Scheduling + Graph Rewriting</td>
<td>188KB</td>
<td>?</td>
</tr>
</tbody>
</table>
Long Compile Time is Not Good for Mental Health
Pruning without Affecting Optimality

By setting an appropriate threshold, some paths can be pruned without affecting optimality.
Adaptive Soft Budgeting finds appropriate threshold reducing the scheduling time significantly
Many Irregularly Wired Neural Networks are Hourglass-shaped that enables Divide-and-Conquer
Peak memory performance for different scheduling

<table>
<thead>
<tr>
<th>Scheduling Strategy</th>
<th>Peak Mem</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual Optimization + Partial Convolution</td>
<td>200KB</td>
<td>2 days</td>
</tr>
<tr>
<td>(Automatic) Dynamic Programming-based Scheduling</td>
<td>200KB</td>
<td>seconds</td>
</tr>
<tr>
<td>(Automatic) Dynamic Programming-based Scheduling + Graph Rewriting</td>
<td>188KB</td>
<td>minutes</td>
</tr>
</tbody>
</table>
Evaluation
### Evaluation: Benchmark Irregularly Wired Neural Networks

<table>
<thead>
<tr>
<th>Network</th>
<th>Type</th>
<th>Dataset</th>
<th># MAC</th>
<th># Weight</th>
<th>Top-1 Accuracy*</th>
</tr>
</thead>
<tbody>
<tr>
<td>DARTS [ICLR’19]</td>
<td>Neural Architecture Search</td>
<td>ImageNet</td>
<td>574.0M</td>
<td>4.7M</td>
<td>73.3%</td>
</tr>
<tr>
<td>SwiftNet [CVPR-C’19, ICCV-W’19]</td>
<td></td>
<td>Human Presence Detection</td>
<td>57.4M</td>
<td>249.7K</td>
<td>95.1%</td>
</tr>
<tr>
<td>Randomly Wired Neural Networks [ICCV’19]</td>
<td>Random Network Generators</td>
<td>CIFAR10</td>
<td>111.0M</td>
<td>1.2M</td>
<td>93.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CIFAR100</td>
<td>160.0M</td>
<td>4.7M</td>
<td>74.5%</td>
</tr>
</tbody>
</table>

* Serenity does not affect accuracy
Evaluation: Reduction in Peak Memory Footprint

Serenity reduces the Peak Memory Footprint by 1.68x without Graph Rewriting and 1.86x with Graph Rewriting
Serenity also reduces off-chip memory communication by **1.52x**, **1.49x**, **1.51x**, and **1.76x** for 32KB, 64KB, 128KB, and 256KB, respectively.
Serenity even eradicates off-chip memory communication
Evaluation: Scheduling Time

Average scheduling time of **Serenity** is under a minute for the benchmark models. Can be further improved by **Porting from Python to C/C++**.
Summary and Takeaways

1. Irregularly Wired Neural Networks are emerging class of Network Architectures with many upsides in terms of efficiency, but current deep learning frameworks are oblivious to the Peak Memory Footprint challenge they introduce.

2. We leverage Dynamic Programming-based Scheduling to find an optimal schedule; devise a Identity Graph Rewriting to further reduce Peak Memory Footprint; and develop Adaptive Soft Budgeting and Divide-and-Conquer to minimize overhead.
Future Directions

1. **Expanding Applications** or **Revisiting** the classical algorithms or compiler heuristics:
   - Problems of optimizing memory communication and inference time can also benefit from similar dynamic programming formulation

2. Using **Machine Learning** techniques to find good schedules in **one-shot**:
   - Graph Neural Networks to parse and extract information from the graph
   - Reinforcement Learning and other intelligent algorithms for scheduling

3. Exploring **Other Dimensions** of **reducing intermediate activations**:
   - Quantization and Pruning are popular compression techniques
   - Lossy/Lossless compression for intermediate activations are interesting future path