SwapNet: Efficient Swapping for DNN Inference on Edge AI Devices Beyond the Memory Budget

Kun Wang, Jiani Cao, Zimu Zhou, Zhenjiang Li
City University of Hong Kong
Traffic condition in Hong Kong

Very **complex** road conditions

Very **high** traffic densities
Vehicle-to-Everything (V2X)
Roadside Unit - Edge AI Device

NVIDIA Jetson Board

HUAWEI Atlas Board

Only Around 25% memory remain in NVIDIA Jetson Nano
Challenge #1 - Memory Scarcity Problem

AI apps need to compete memory with other apps

Model Size: > 2GB

Available RAM: < 2GB
How to run large model in small memory?
Existing Methods

① Upgrade Memory

8GB RAM

32GB RAM

$200

② Model Compression

Accuracy Drop

Accuracy(%)

90.33 90.25 89.97 89.69 89.12 88.95 85.30 79.32 71.27

Parameters(M)

138 100 90 85 80 75 70 65 60

Over Compression
Existing Methods

Computing Offloading

DNN Partition

Part 1

Part 2

Edge Side

Transmit

Server Side

Attacks

Hacker

Existing Methods

8
Thinking

DNN Partition

Part 1 + Part 2

Edge Side

Server Side

Transmit

Keep Accuracy

① Transmission safety concern

② Transmission rate concern
Main Idea - Virtual Memory

Virtual Memory

Physical Memory + Page File Space

Ideal case

Naïve swap

Average Total Latency (ms)

Doing Inference

 Swap

Block 1

Model Block Swap

Wait for swapping

Block 2

Block 3

Big latency

RAM

Data Swap

Disk

Enough Memory Case

Naïve swap

1000 2000 3000 4000 5000

5122

473

Average Total Latency (ms)
Design #1 - Unified Memory Allocator

```c
1: // In Copy.cu
2: // data_ptr pointed to existing CPU tensor.
3: void* src = iter.data_ptr(1);
4: // Original method needs to allocate GPU Memory
5: and copy data to it.
6: void* dst = iter.data_ptr(0);
7: // cudaMemcpyAsync(dst, src, size, kind, stream);
8: void* dst = src;
9: cudaMemcpyAsync(dst, src, size, cudaMemcpyDefault);
10: cudaMemcpyAsync(dst, src, size, cudaMemcpyDefault);
11: return dst;
```
Design #2 - Weights restoration optimization
Challenge #2 - Inefficiency of Sequential Swap

(a) Unavoidable Swap latency of Sequential Inference

(b) Unavoidable Swap latency of Sequential Inference
Design #3 - Partition Module: Parallel Inference

\[ T_{\text{overlap}}(i) = T_{\text{swap\_out}}(i-1) + T_{\text{swap\_in}}(i+1) - T_{\text{inference}}(i) \]

Case 1: \[ T_{SO1} + T_{SI3} \leq T_{I2} \]

Case 2: \[ T_{SO1} + T_{SI3} > T_{I2} \]
Design #3 - Partition Module: Select Optimal Solution

\[ T_{\text{I/O}} \propto \text{Weight Size} \]

\[ T_{\text{restore}} \propto \text{param_depth} \]

\[ T_{\text{inference}} \propto \text{FLOPs} \]

\[ T_{\text{remove}} \propto \text{param_depth} \]
Design #3 - Partition Module: Select Optimal Solution

**Model Info Table**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size</th>
<th>Depth</th>
<th>FLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer1</td>
<td>0.38 MB</td>
<td>1</td>
<td>26.2 M</td>
</tr>
<tr>
<td>Layer2</td>
<td>1.49 MB</td>
<td>5</td>
<td>0.8 K</td>
</tr>
<tr>
<td>Layer3</td>
<td>1.12 MB</td>
<td>1</td>
<td>123.9 M</td>
</tr>
<tr>
<td>Layer4</td>
<td>5.93MB</td>
<td>5</td>
<td>4.2 K</td>
</tr>
<tr>
<td>Layer5</td>
<td>4.38MB</td>
<td>6</td>
<td>316.7 M</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Layer100</td>
<td>23.6 MB</td>
<td>1</td>
<td>30 K</td>
</tr>
<tr>
<td>Layer101</td>
<td>17.45 MB</td>
<td>1</td>
<td>5 K</td>
</tr>
</tbody>
</table>

**Overlap latency can be computed through the model info table**

**Decision Table**

<table>
<thead>
<tr>
<th>Partition Points</th>
<th>Maximum Memory</th>
<th>Predicted Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>exceed</td>
<td>null</td>
</tr>
<tr>
<td>1.3</td>
<td>exceed</td>
<td>null</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>30.66</td>
<td>105 MB</td>
<td>496 ms</td>
</tr>
<tr>
<td><strong>30.67</strong></td>
<td><strong>109 MB</strong></td>
<td><strong>488 ms</strong></td>
</tr>
<tr>
<td>98.100</td>
<td>exceed</td>
<td>null</td>
</tr>
<tr>
<td>99.100</td>
<td>exceed</td>
<td>null</td>
</tr>
</tbody>
</table>
Implementation

Proposed SwapNet Framework

1. Self-driving(4 tasks): lane detection, object detection, segmentation, traffic sign classification
2. Road-Side Unit (5 tasks): 2 object detection, 2 natural scenes classification and traffic light classification
3. UAV surveillance(2 tasks): fire source detection, wild animal recognition

Scenarios
Evaluation
Conclusion

• We introduce SwapNet, a middleware that logically executes large DNN models on a small memory budget. SwapNet partitions large DNN models into blocks for execution by swapping them between the memory and the external storage in order.

• Our main contribution is a transparent design that eliminates the substantial latency and memory overhead occurred during block swapping while remaining compatible with the DNN development tool chains for edge AI devices.

• Extensive evaluations show the promising performance gains of SwapNet in combination with parallel optimization for efficient execution.