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

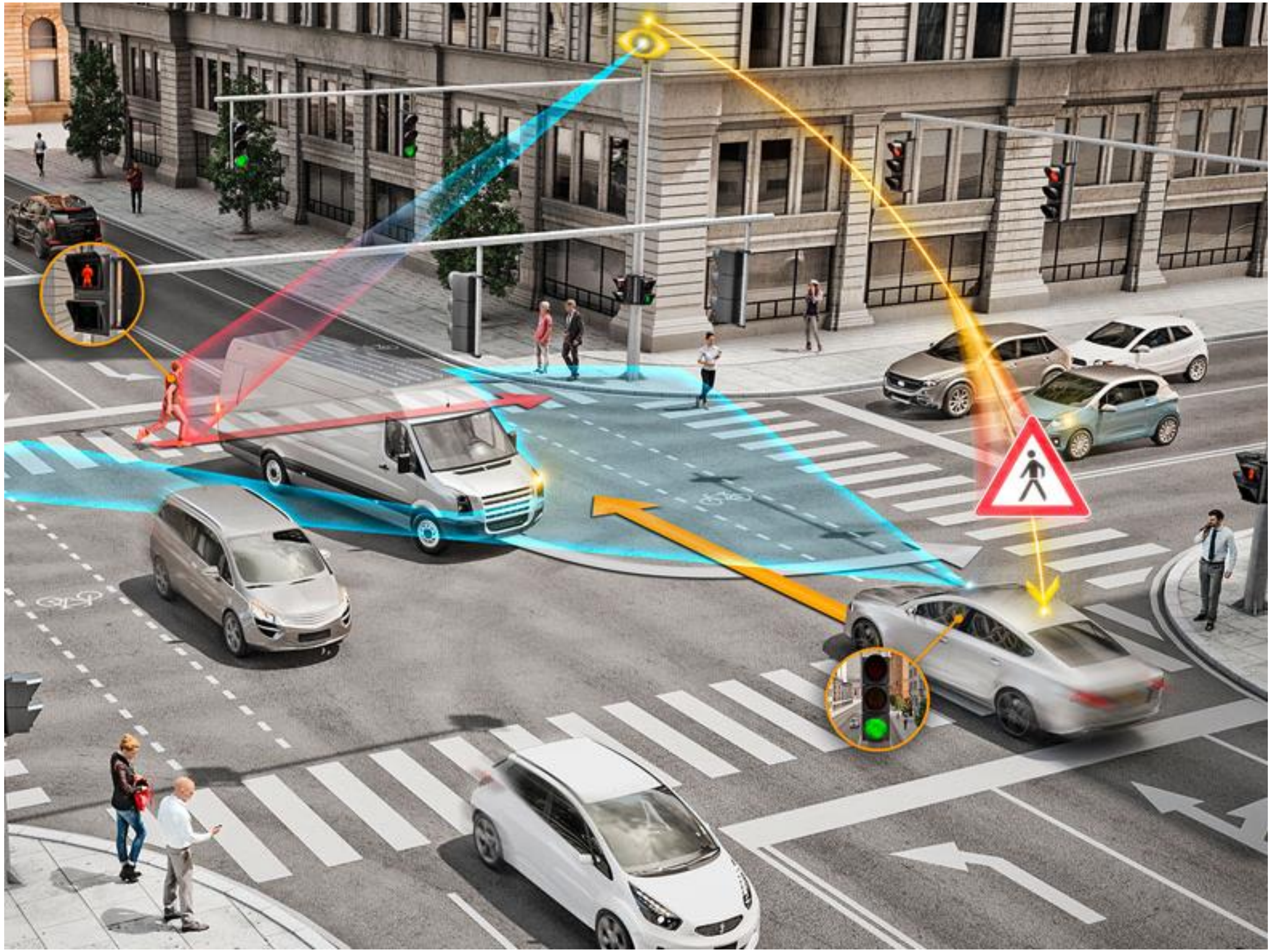
香港城市大學
City University of Hong Kong

Traffic condition in Hong Kong

Very **complex** road conditions

Very **high** traffic densities

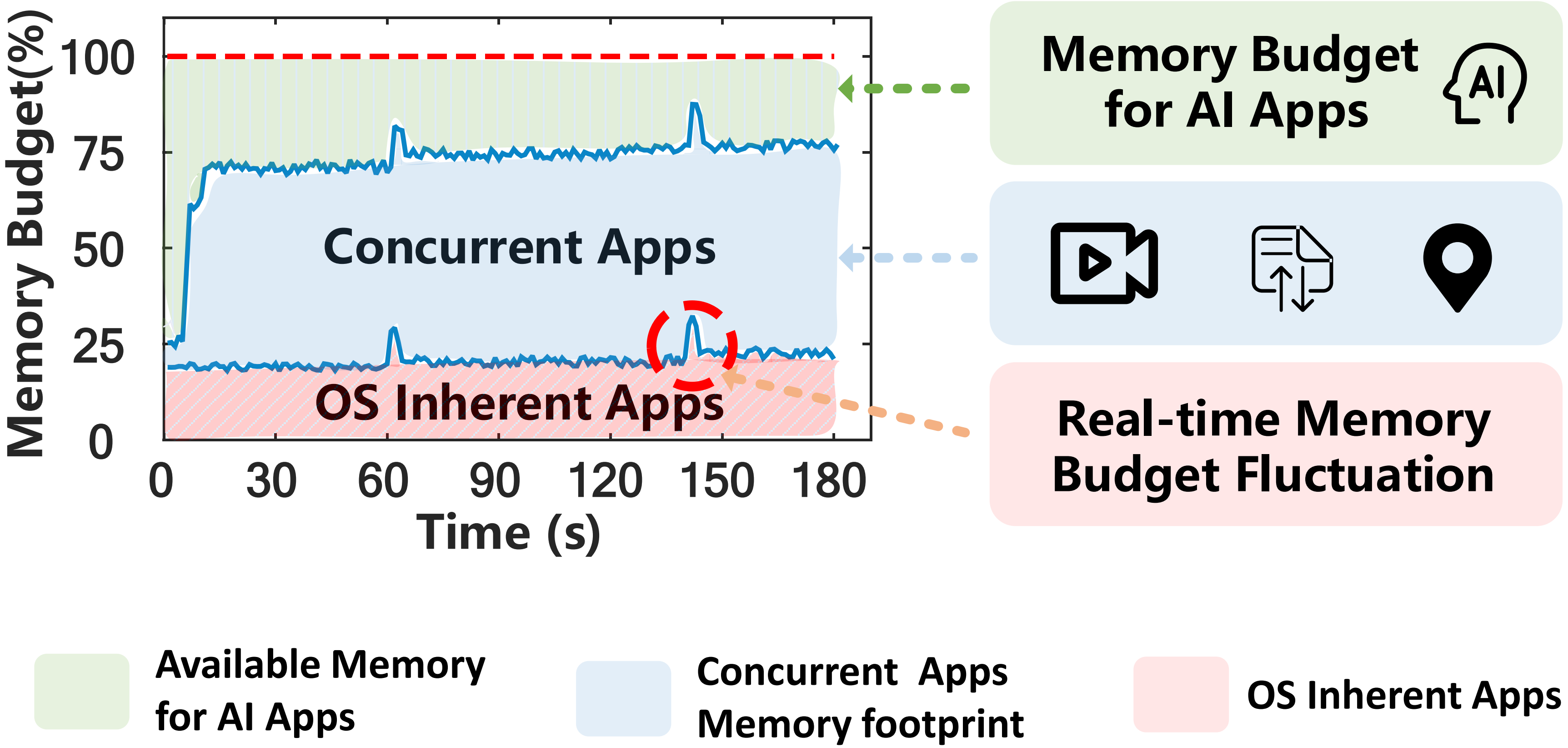
Vehicle-to-Everything (V2X)



Roadside Unit - Edge AI Device

NVIDIA Jetson Board

HUAWEI Atlas Board

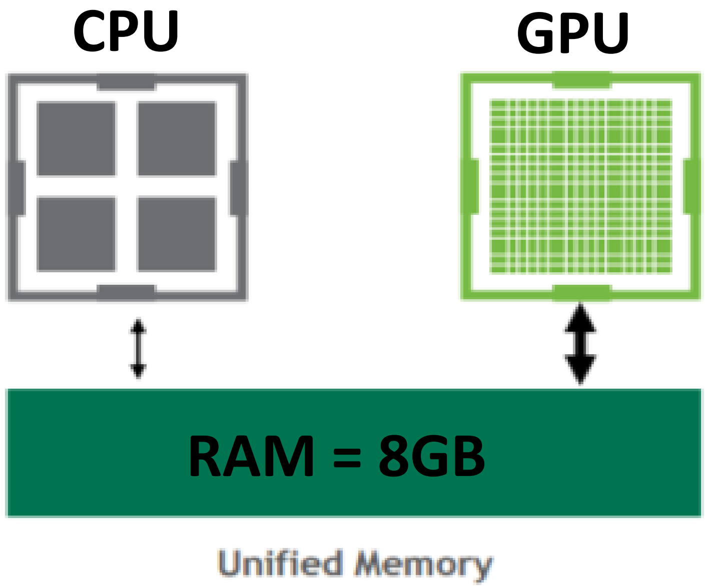


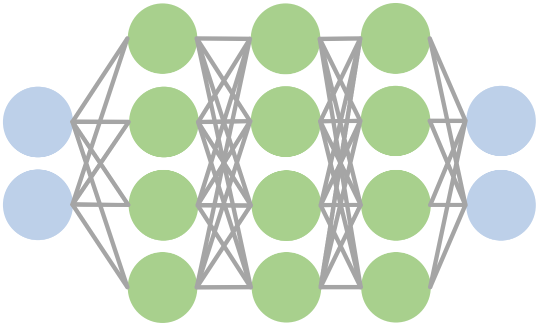
↓

Only Around 25% memory remain in NVIDIA Jetson Nano

Challenge #1 - Memory Scarcity Problem

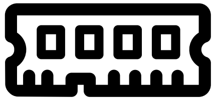
AI apps need to compete memory with other apps





Model Size:
> 2GB

>



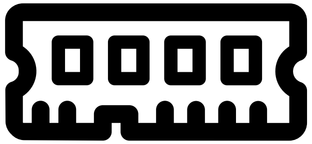
Available RAM:
< 2GB

Q How to run large model in small memory?

Existing Methods

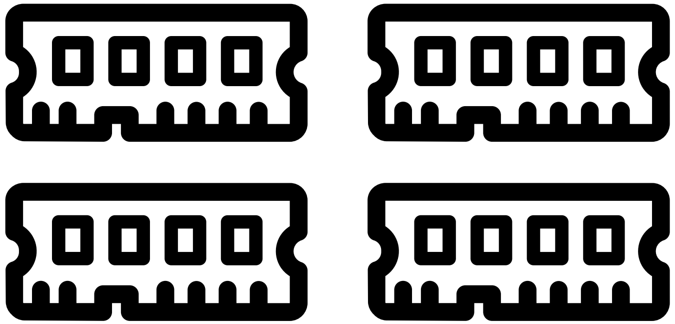
① Upgrade Memory

8GB RAM

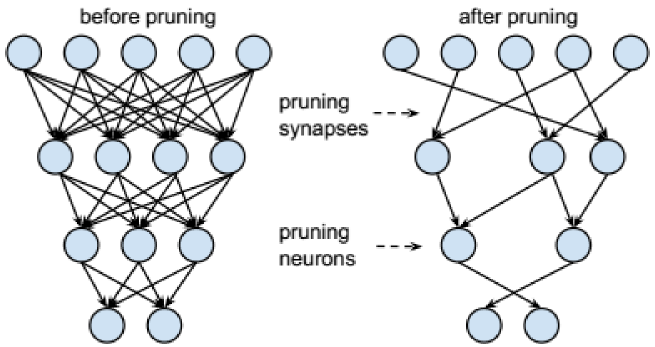


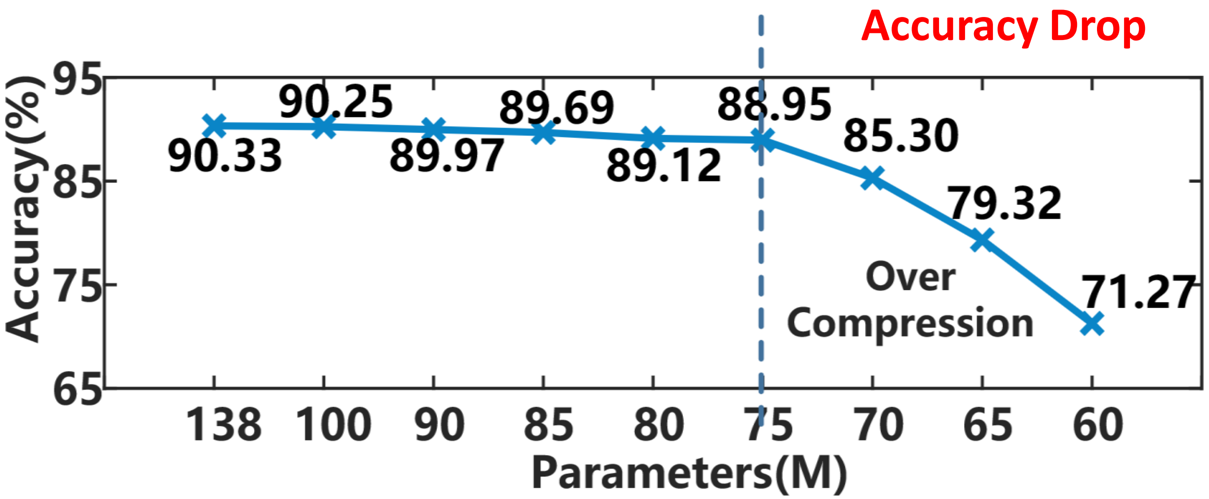
$200

32GB RAM

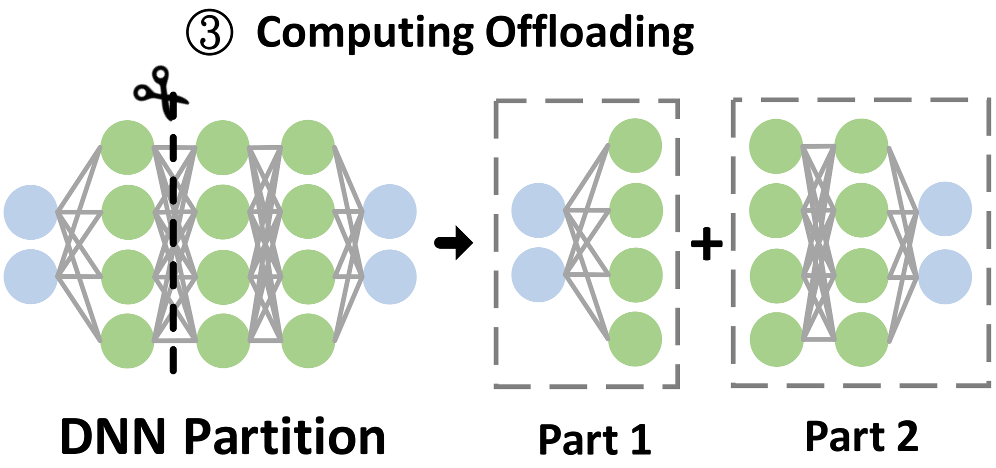


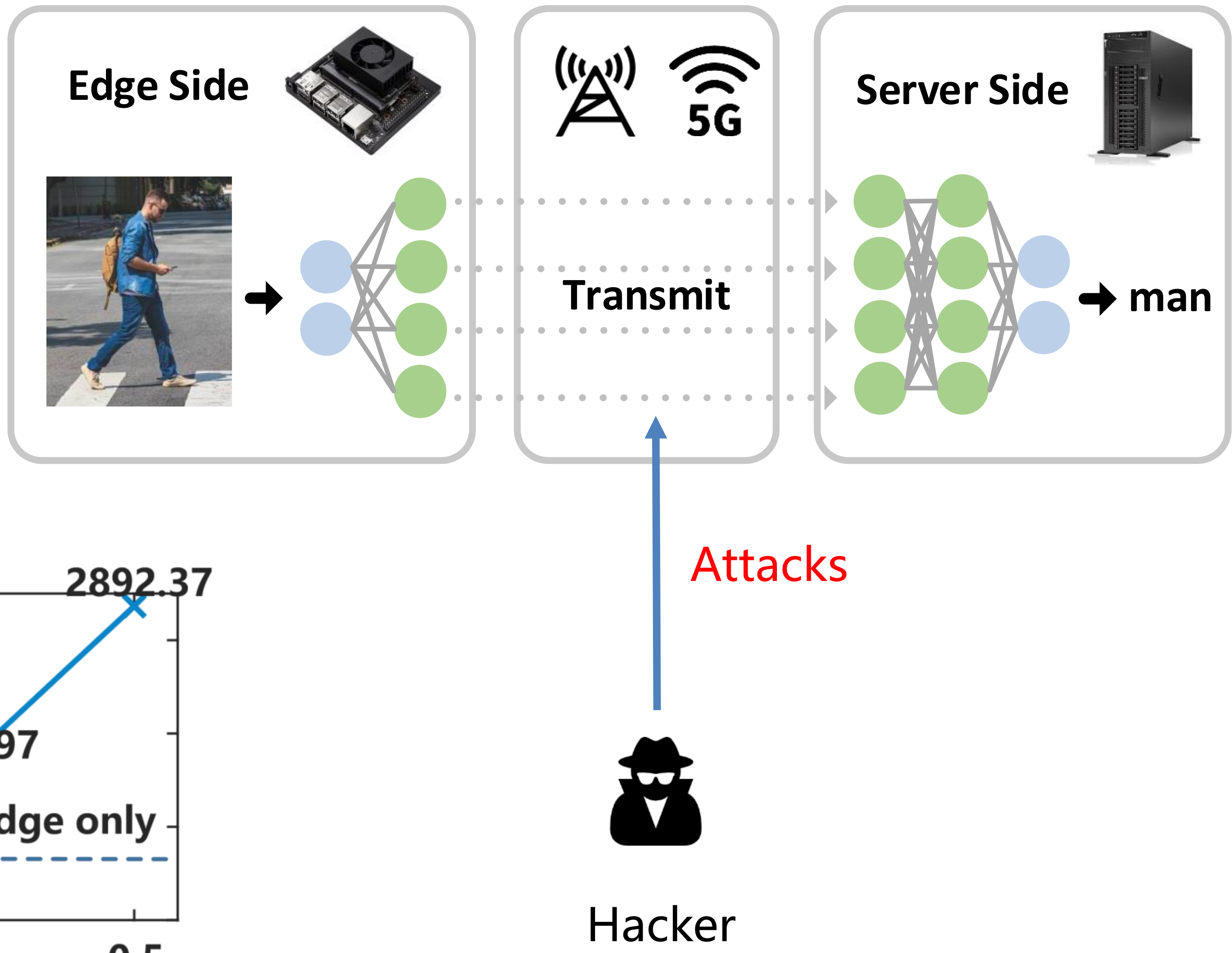
② Model Compression

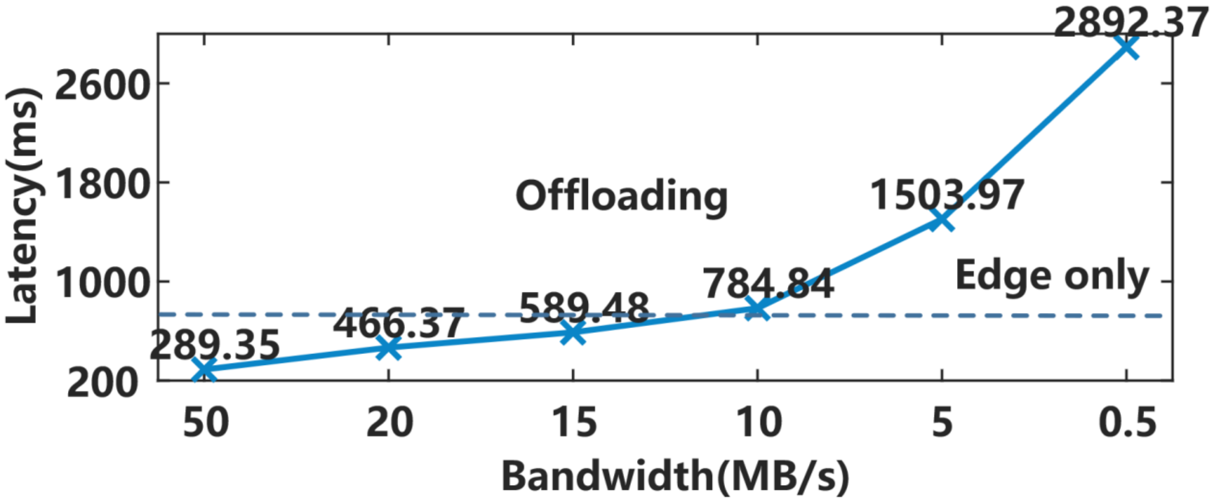




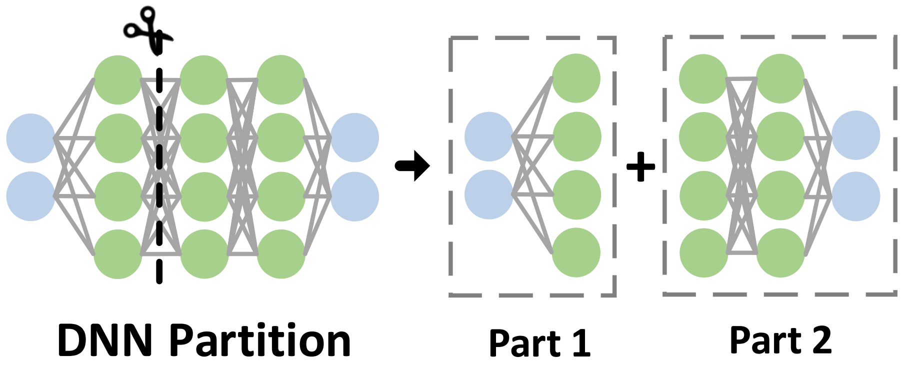
Existing Methods



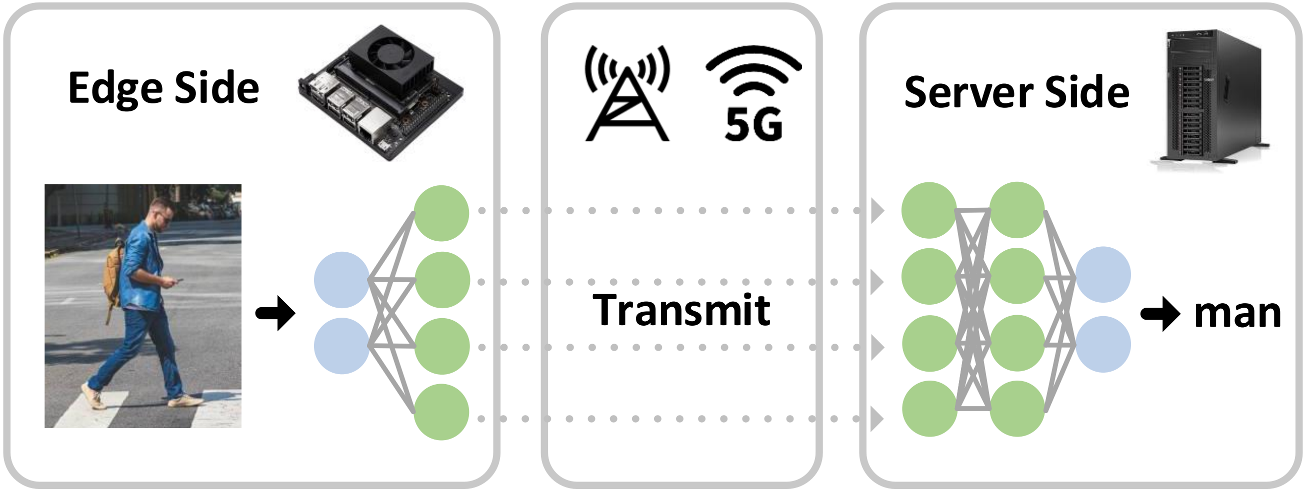


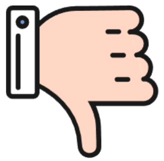


Thinking



Keep Accuracy

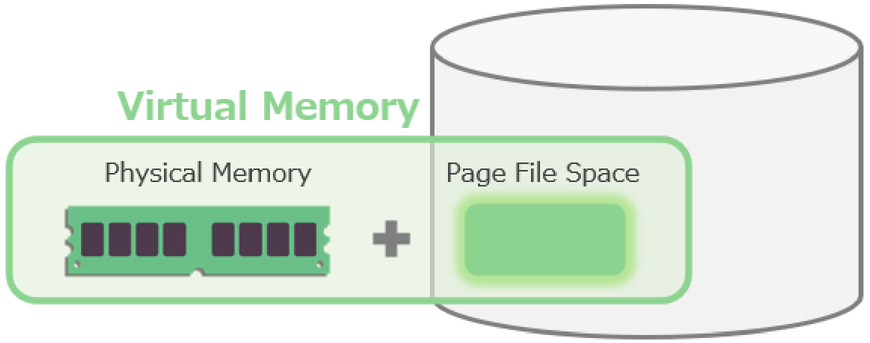


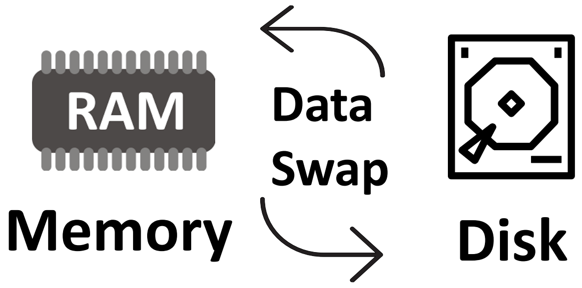


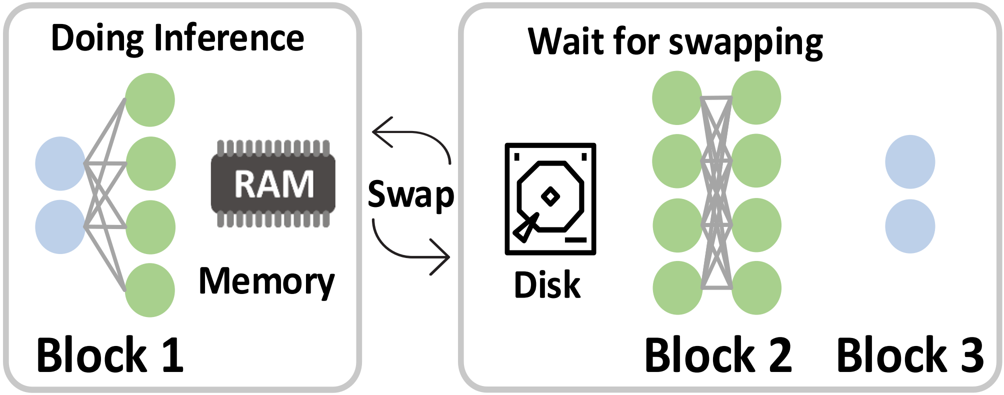
① Transmission safety concern

② Transmission rate concern

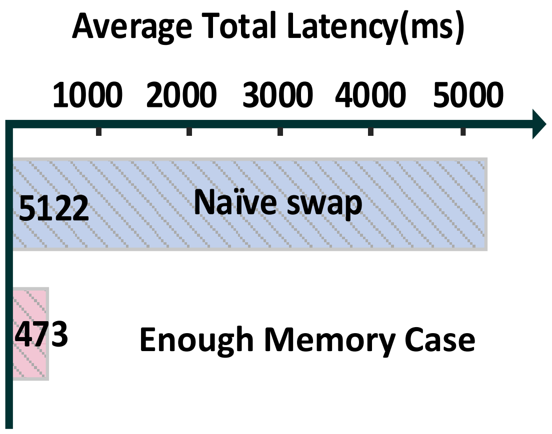
Main Idea - Virtual Memory





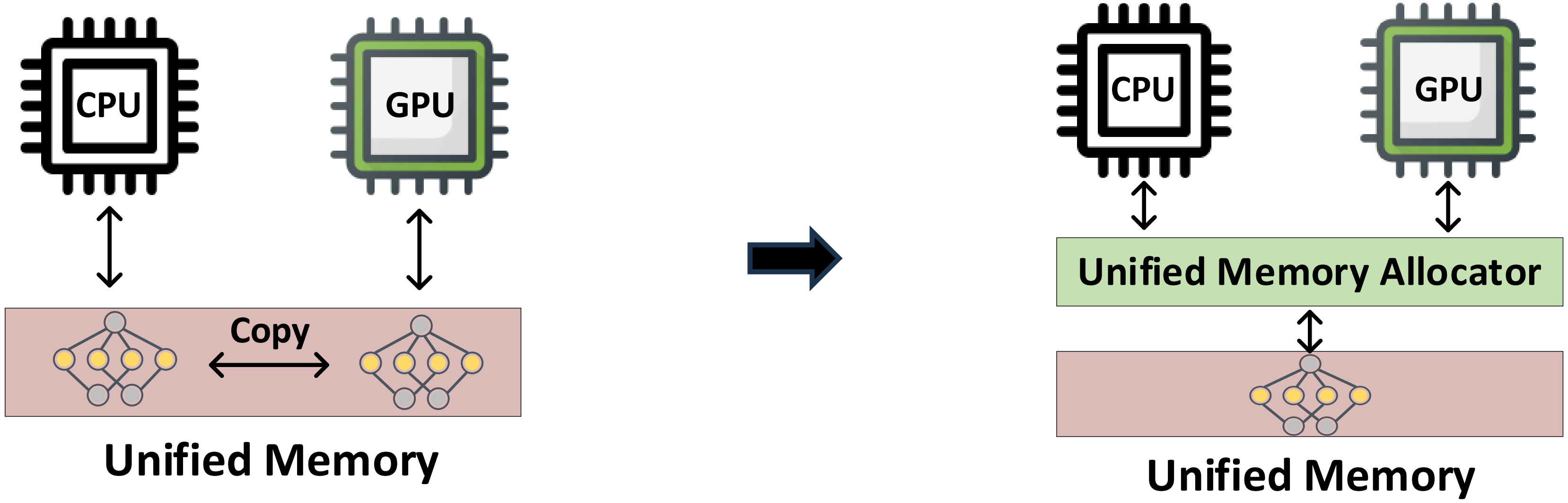


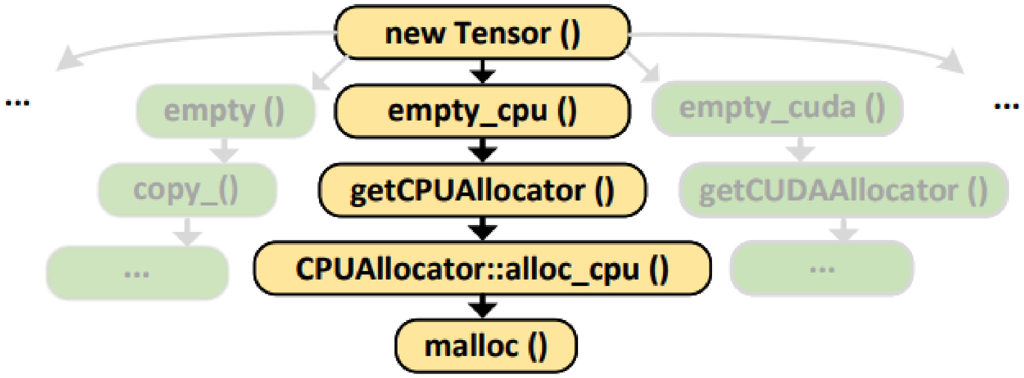
Model Block Swap



Big latency

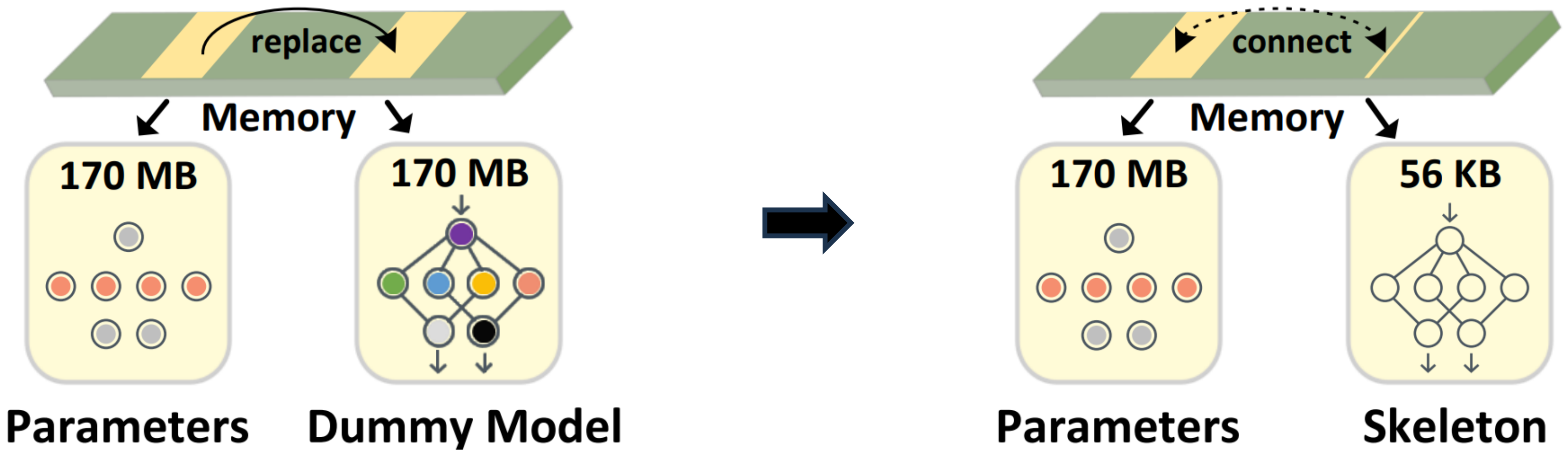
Design #1 - Unified Memory Allocator



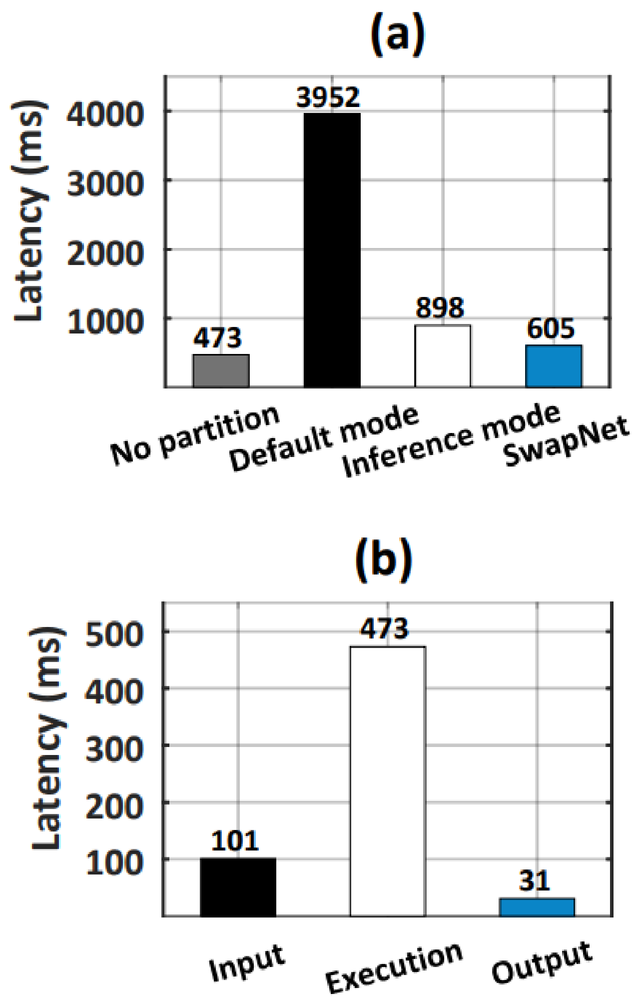


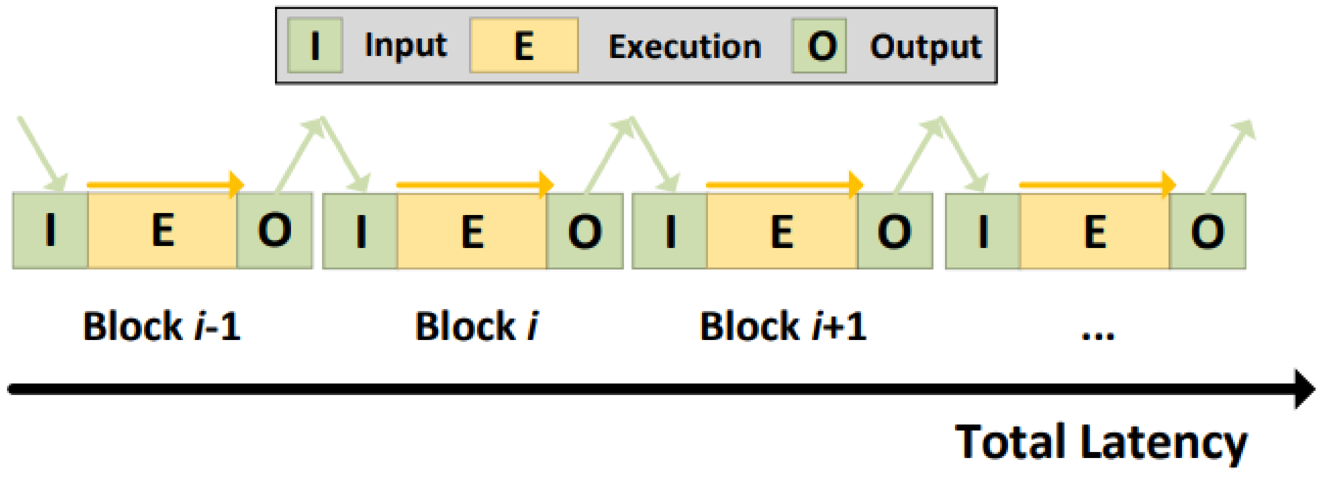

```
// In Copy.cu
// data_ptr pointed to existing CPU tensor.
   void* src = iter.data_ptr(1);
// Original method needs to allocate GPU Memory
   and copy data to it.
// void* dst = iter.data_ptr(0);
// cudaMemcpyAsync(dst, src, size, kind, stream);
   void* dst = src;
   cudaDeviceSynchronize();
   return dst;
```

Design #2 - Weights restoration optimization



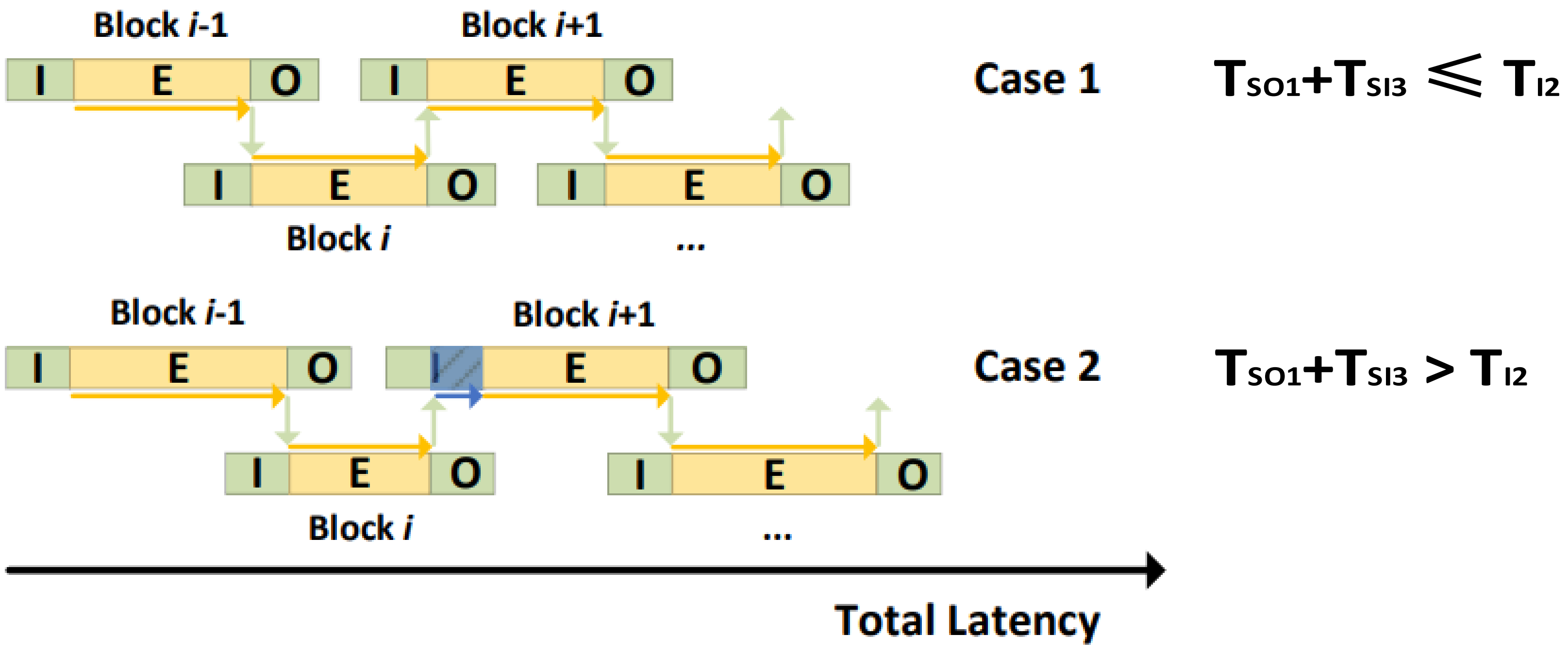
Challenge #2 - Inefficiency of Sequential Swap





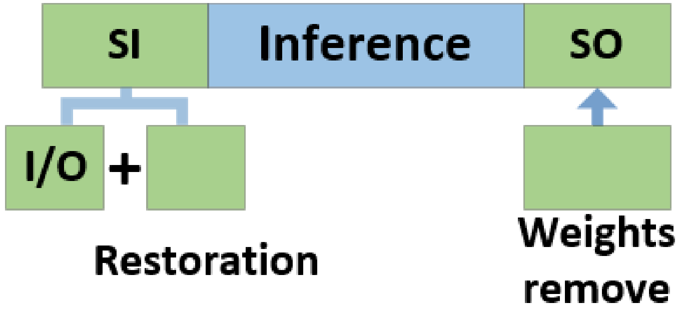
Unavoidable Swap latency of Sequential Inference

Design #3 - Partition Module: Parallel Inference

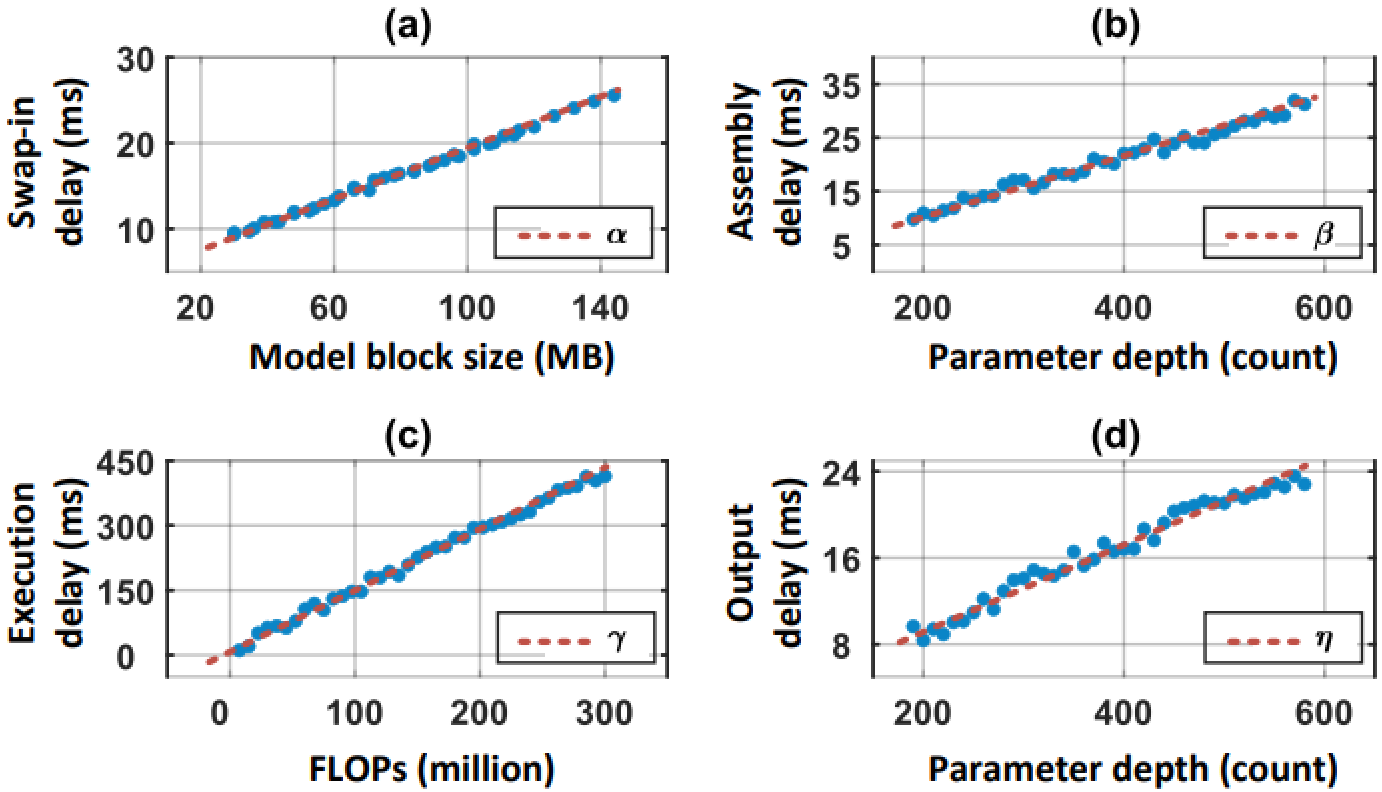


$T_{overlap(i)} = T_{swap\_out(i-1)} + T_{swap\_in(i+1)} - T_{inference(i)}$

Design #3 - Partition Module: Select Optimal Solution



$$\min \sum_{n=1}^{B} (\alpha * WS_n + (\beta + \gamma) * PD_n - \theta * FLOPs_n)$$



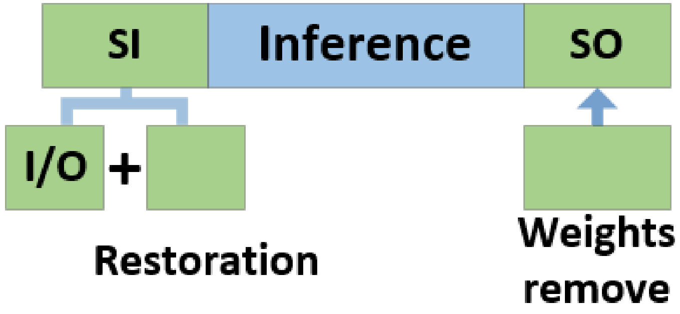
$T_{I/O} \propto Weight\_Size$

$T_{restore} \propto param\_depth$

$T_{inference} \propto FLOPs$

$T_{remove} \propto param\_depth$

Design #3 - Partition Module: Select Optimal Solution



$$\min \sum_{n=1}^{B} (\alpha * WS_n + (\beta + \gamma) * PD_n - \theta * FLOPs_n)$$

$T_{I/O} \propto Weight\_Size$

$T_{restore} \propto param\_depth$

$T_{inference} \propto FLOPs$

$T_{remove} \propto param\_depth$

Layer	Size	Depth	FLOPs
Layer1	0.38 MB	1	26.2 M
Layer2	1.49 MB	5	0.8 K
Layer3	1.12 MB	1	123.9 M
Layer4	5.93MB	5	4.2 K
Layer5	4.38MB	6	316.7 M
...	...	...	...
Layer100	23.6 MB	1	30 K
Layer101	17.45 MB	1	5 K

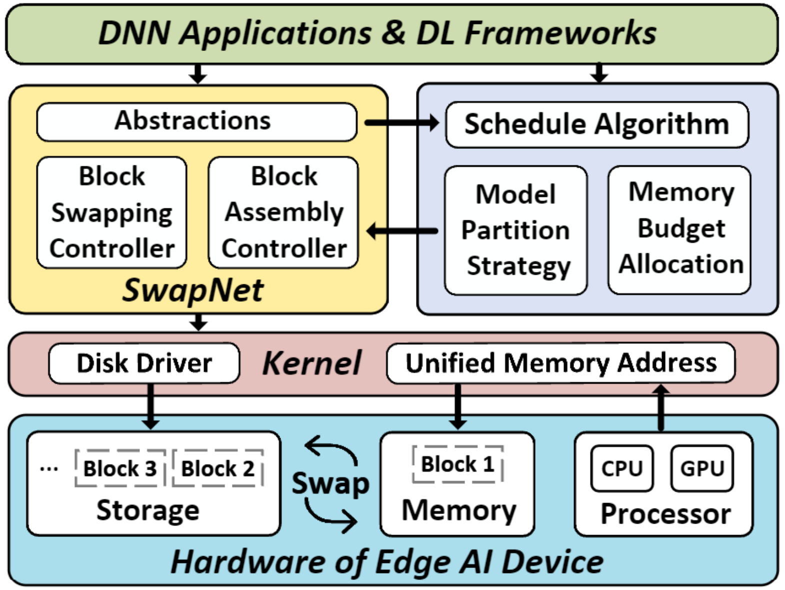
Model Info Table

Partition Points	Maximum Memory	Predicted Latency
1,2	exceed	null
1,3	exceed	null
...	...	...
30,66	105 MB	496 ms
30,67	109 MB	488 ms
...	...	...
98,100	exceed	null
99,100	exceed	null

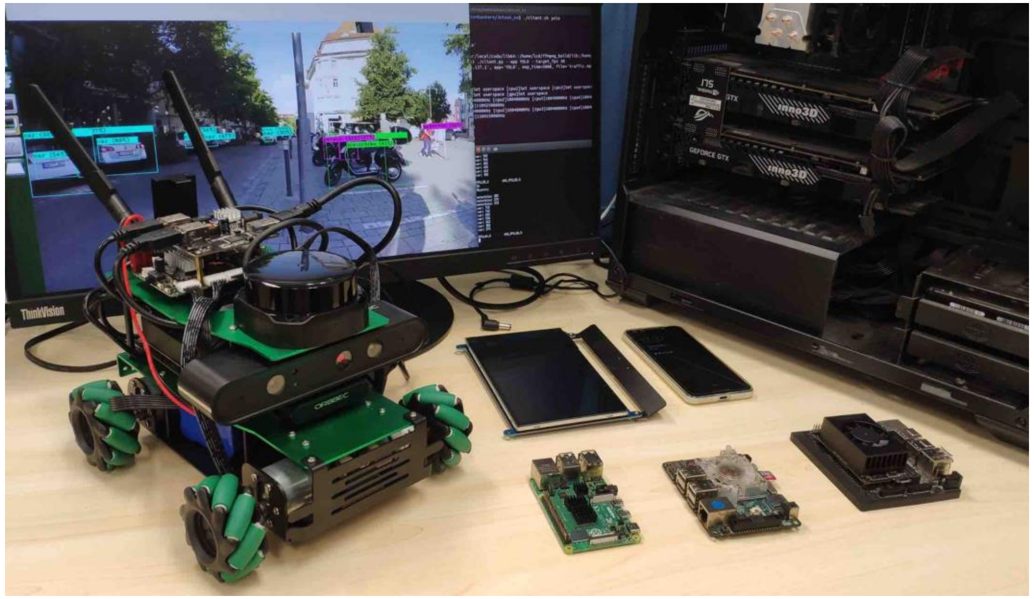
Decision Table

Overlap latency can be computed through the model info table

Implementation



Proposed SwapNet Framework

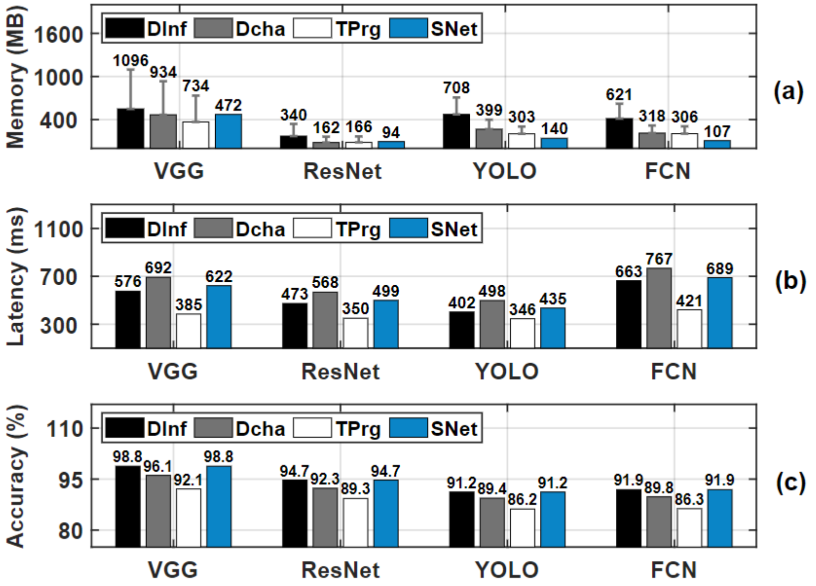


Test Bed

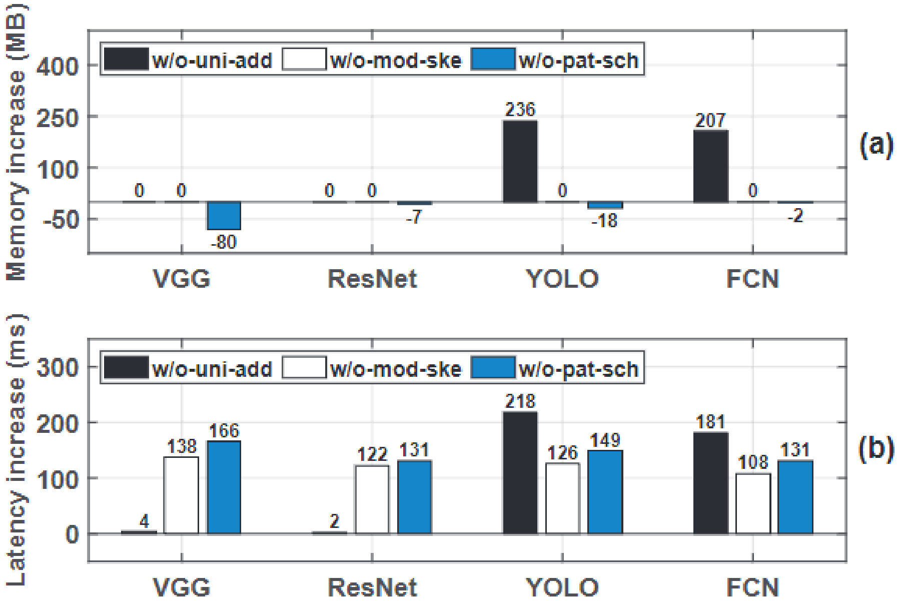
1. **Self-driving(4 tasks):** lane detection, object detection, segmentation, traffic sign classification
2. **Road-Side Unit (5 tasks):** 2 object detection, 2 natural scenes classification and traffic light classification
3. **UAV surveillance(2 tasks):** fire source detection, wild animal recognition

Scenarios

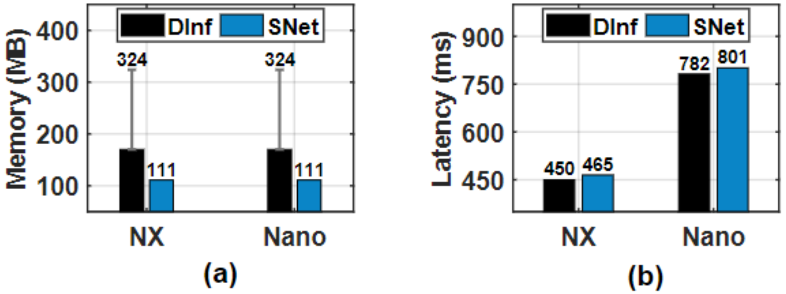
Evaluation



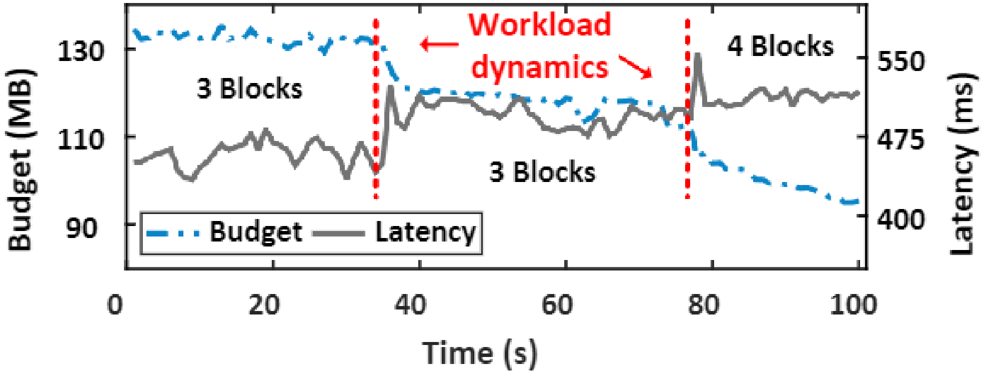
Memory (MB)
1600
1000
400
DInf
Dcha
TPrg
SNet
1096 934 734 472
340 162 166 94
708 399 303 140
621 318 306 107
VGG ResNet YOLO FCN
(a)
Latency (ms)
1100
700
300
576 692 385 622
473 568 350 499
402 498 346 435
663 767 421 689
(b)
Accuracy (%)
110
95
80
98.8 96.1 92.1 98.8
94.7 92.3 89.3 94.7
91.2 89.4 86.2 91.2
91.9 89.8 86.3 91.9
(c)



Memory increase (MB)
400
250
100
-50
w/o-uni-add
w/o-mod-ske
w/o-pat-sch
0 0 -80
0 0 -7
236 0 -18
207 0 -2
VGG ResNet YOLO FCN
(a)
Latency increase (ms)
300
200
100
0
4 138 166
2 122 131
218 126 149
181 108 131
(b)



Memory (MB)
400
300
200
100
DInf
SNet
324 111
324 111
NX
Nano
(a)
Latency (ms)
900
750
600
450
450 465
782 801
NX
Nano
(b)



Budget (MB)
130
110
90
Latency (ms)
550
475
400
Time (s)
0 20 40 60 80 100
3 Blocks
Workload dynamics
4 Blocks
3 Blocks
Budget
Latency

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.