SkyNet: a Hardware-Efficient Method for Object Detection and Tracking on Embedded Systems

Xiaofan Zhang[1], Haoming Lu[1], Cong Hao[1], Jiachen Li[1], Bowen Cheng[1], Yuhong Li[1] , Kyle Rupnow[2], Jinjun Xiong[3,1], Thomas Huang[1], Honghui Shi[3,1], Wen-mei Hwu[1], Deming Chen[1,2]

[1] C^{3}SR, UIUC [2] Inspirit IoT, Inc [3] IBM Research

Outline:

1) **Background & Challenges**

   Edge AI is necessary but challenging

2) **Motivations**

   Two major issues prevent better AI quality on embedded systems

3) **The Proposed SkyNet Solution**

   A bottom-up approach for building hardware-efficient DNNs

4) **Demonstrations on Object Detection and Tracking Tasks**

   Double champions in an international system design competition

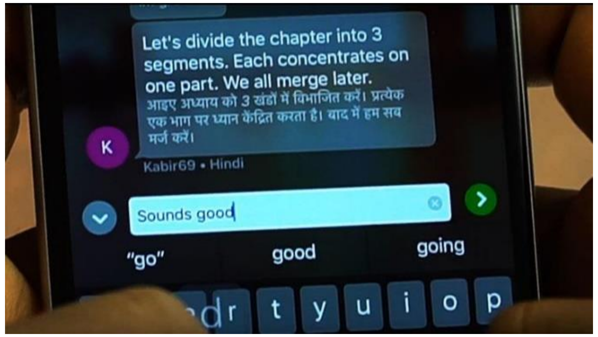
   Faster and better results than trackers with ResNet-50 backbone

5) **Conclusions**

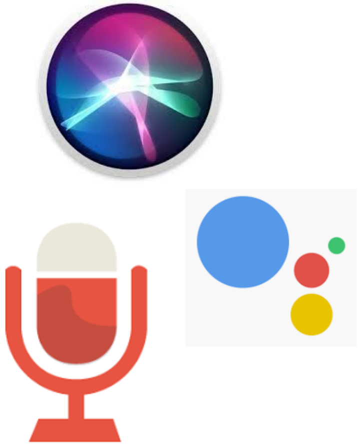
Cloud solutions for AI deployment

Major requirements:

- High throughput performance
- Short tail latency



Language Translation



Voice-activated assistant

Recommendations

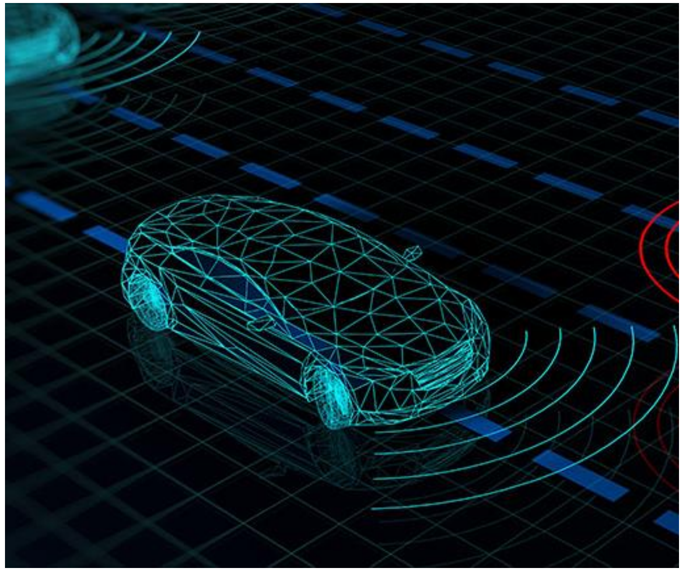
Video analysis

Why still need Edge solutions?

Communication

Privacy

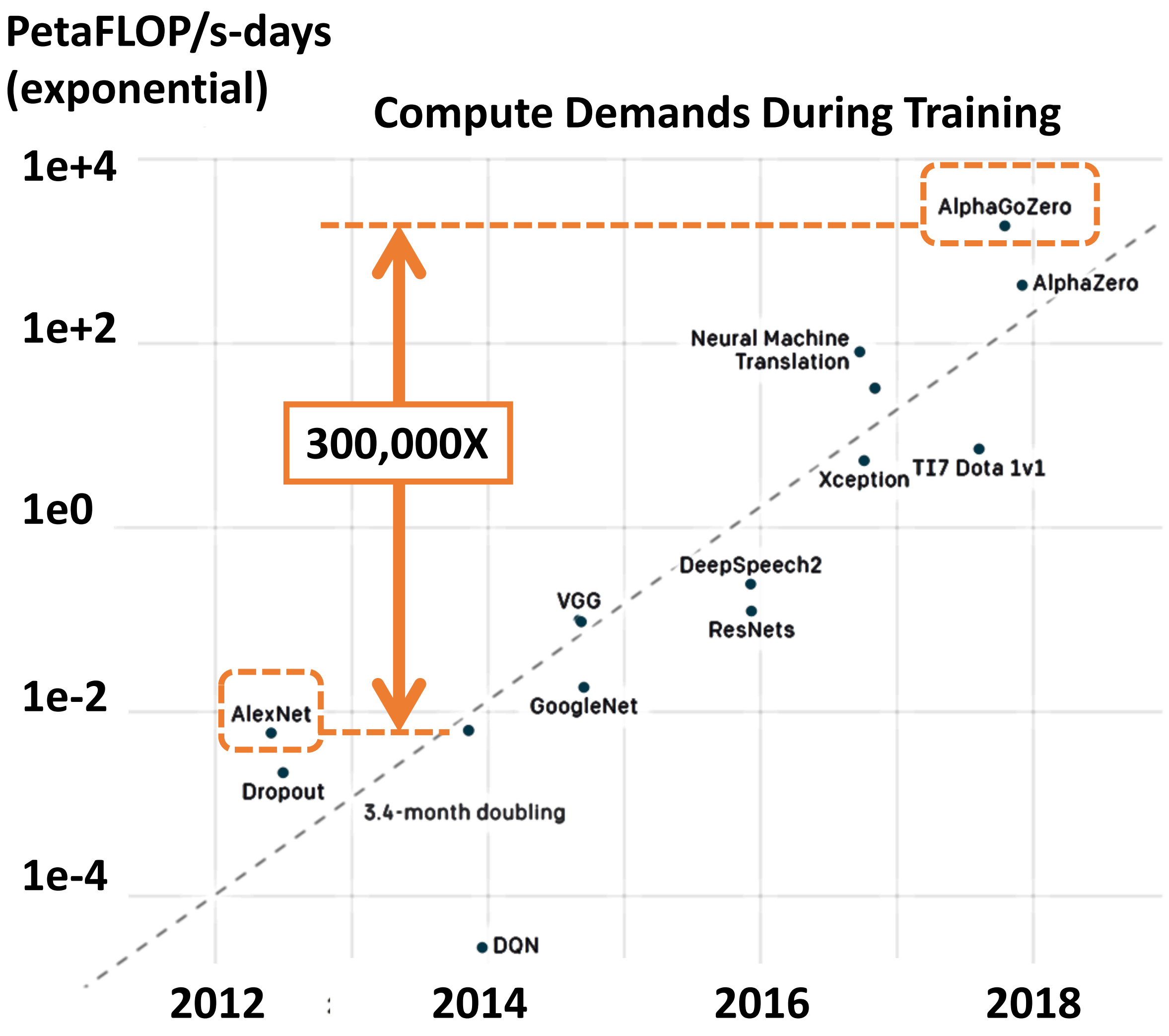
Latency



Demanding AI applications cause great challenges for Edge solutions.

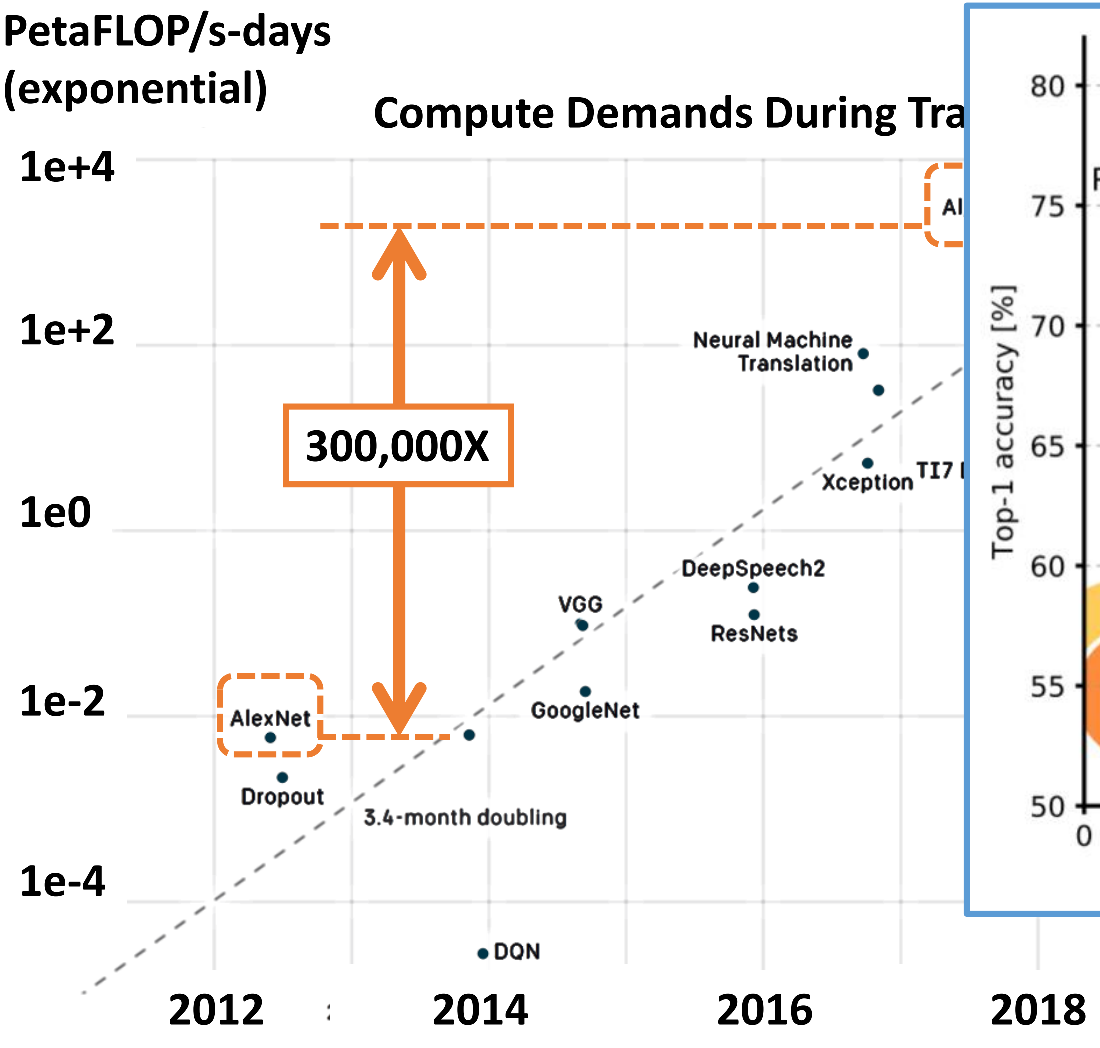
We summarize three major challenges

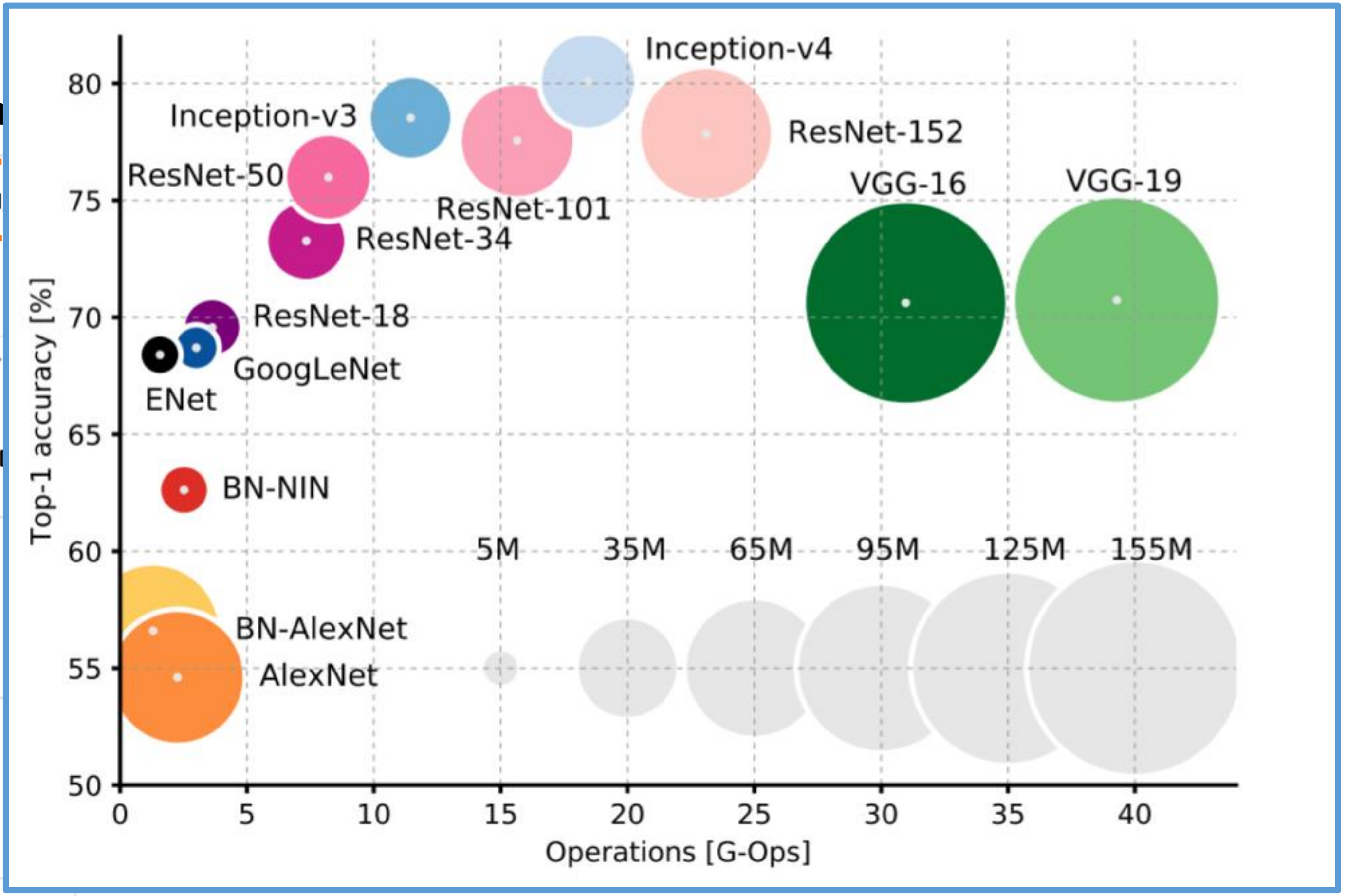
Edge AI Challenge #1 Huge compute demands



https://openai.com/blog/ai-and-compute/

Edge AI Challenge #1 Huge compute demands

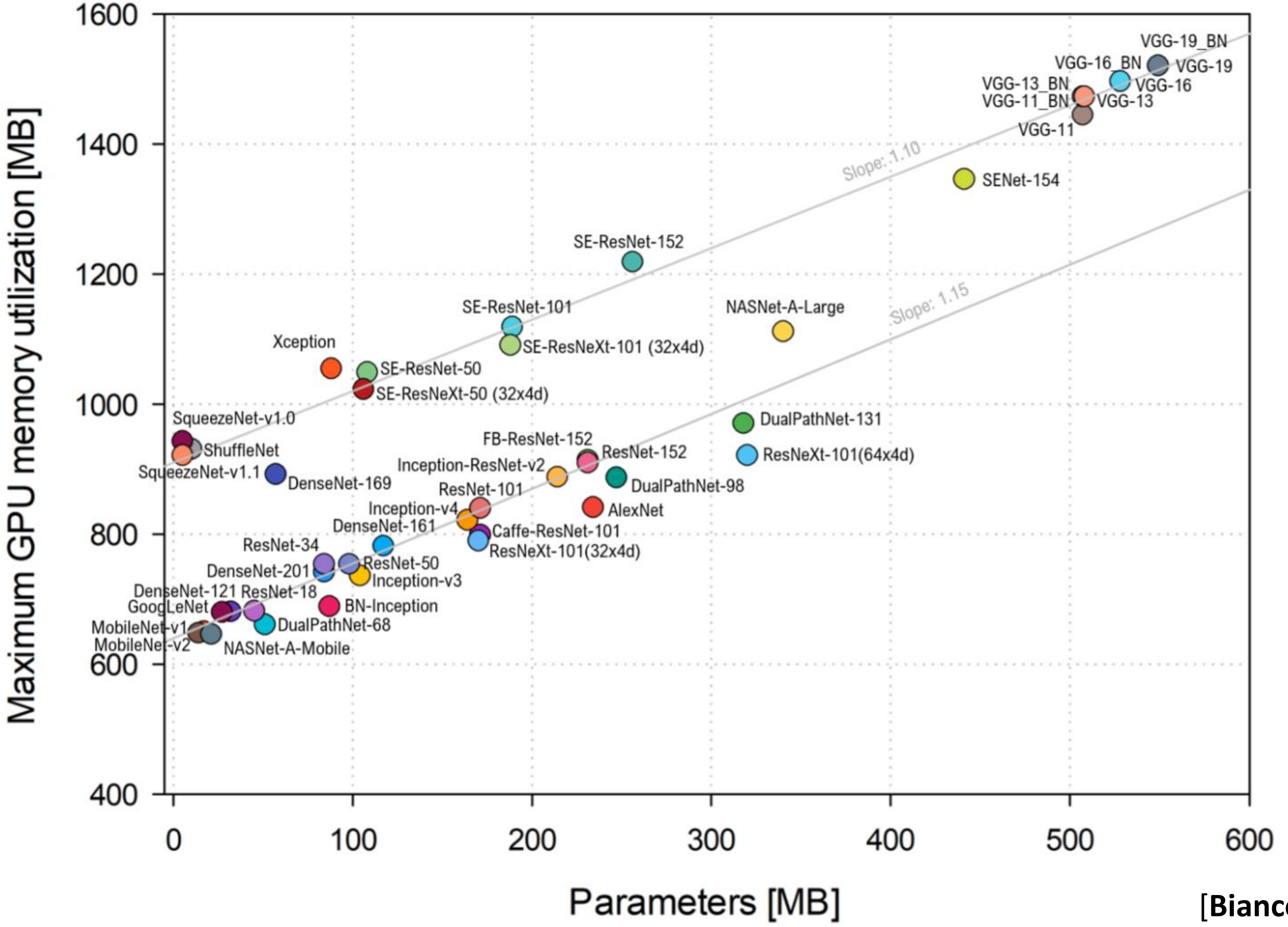




Compute Demands During Inference [**Canziani**, arXiv 2017]

https://openai.com/blog/ai-and-compute/

Edge AI Challenge #2 Massive memory footprint

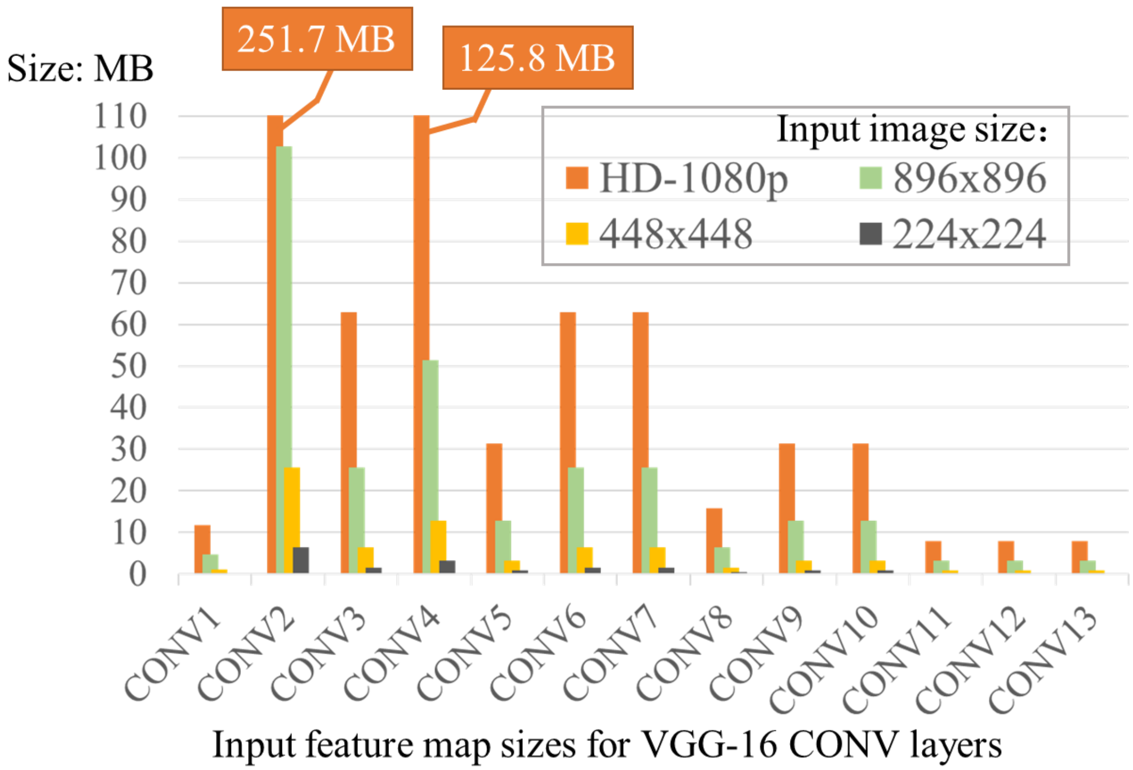


[**Bianco**, IEEE Access 2018]

Edge AI Challenge #2 Massive memory footprint

- HD inputs for real-life applications
  1) Larger memory space required for input feature maps
  2) Longer inference latency

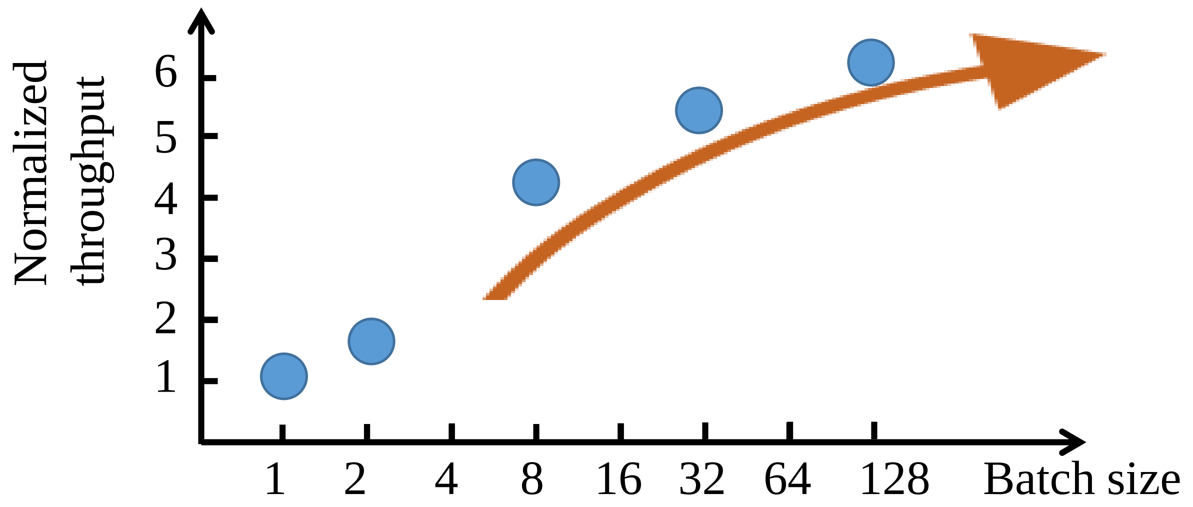
- Harder for edge-devices
  1) Small on-chip memory
  2) Limited external memory access bandwidth



Edge AI Challenge #3 Real-time requirement

- Video/audio streaming I/O

1) Need to deliver high throughput
   - 24FPS, 30FPS …



Edge AI Challenge #3 Real-time requirement

- Video/audio streaming I/O

1) Need to deliver high throughput
   - 24FPS, 30FPS …

2) Need to work for real-time
   - E.g., millisecond-scale response for self-driving cars, UAVs
   - Can’t wait for assembling frames into a batch

Outline:

1) Background & Challenges

   Edge AI is necessary but challenging

2) Motivations

   Two major issues prevent better AI quality on embedded systems

3) The Proposed SkyNet Solution

   A bottom-up approach for building hardware-efficient DNNs

4) Demonstrations on Object Detection and Tracking Tasks

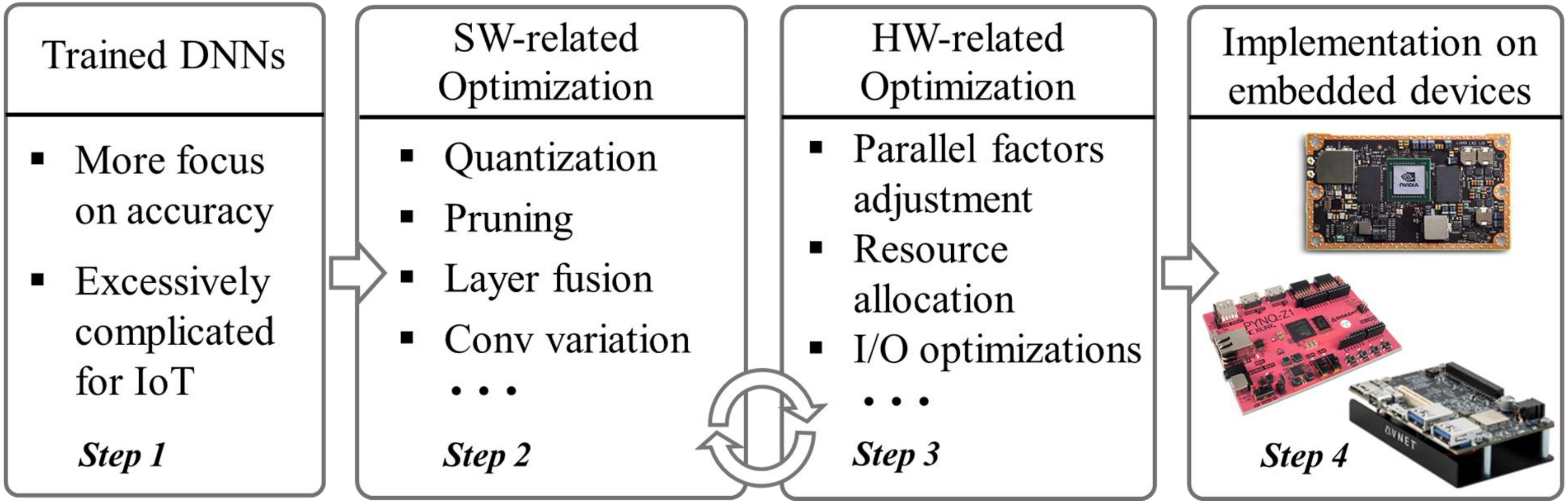
   Double champions in an international system design competition

   Faster and better results than trackers with ResNet-50 backbone

5) Conclusions

A Common flow to design DNNs for embedded systems

Various key metrics: Accuracy; Latency; Throughput; Energy/Power; Hardware cost, etc.



It is a top-down flow: form reference DNNs to optimized DNNs

Object detection design for embedded GPUs

- Target NVIDIA TX2 GPU

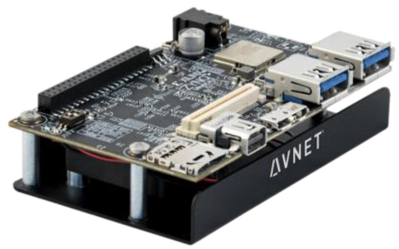
~665 GFLOPS @1300MHz

① Input resizing ② Pruning ③ Quantization ④ TensorRT
⑤ Multithreading

GPU-Track	Reference	Software Optimizations	Hardware Optimizations
’19 2nd Thinker	ShuffleNet + RetinaNet	①②③	⑤
’19 3rd DeepZS	Tiny YOLO	-	⑤
’18 1st ICT-CAS	Tiny YOLO	①②③④	-
’18 2nd DeepZ	Tiny YOLO	-	⑤
’18 3rd SDU-Legend	YOLOv2	①②③	⑤

[From the winning entries of DAC-SDC’18 and ’19]

Object detection design for embedded FPGAs



- Target Ultra96 FPGA

~144 GFLOPS @200MHz

① Input resizing ② Pruning ③ Quantization
⑤ CPU-FPGA task partition ⑥ double-pumped DSP ⑦ pipeline ⑧ clock gating

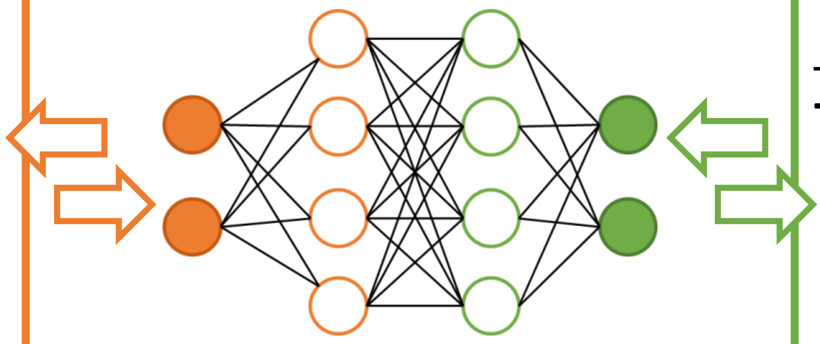
FPGA-Track	Reference	Software Optimizations	Hardware Optimizations
’19 2nd XJTU Tripler	ShuffleNetV2	②③	⑤⑥⑧
’19 3rd SystemsETHZ	SqueezeNet	①②③	⑦
’18 1st TGIIF	SSD	①②③	⑤⑥
’18 2nd SystemsETHZ	SqueezeNet	①②③	⑦
’18 3rd iSmart2	MobileNet	①②③	⑤⑦

[From the winning entries of DAC-SDC’18 and ’19]

Drawbacks of the top-down flow

1) Hard to balance the sensitivities of DNN designs on software and hardware metrics

<u>SW metrics:</u>
Accuracy;
Generalization;
Robustness;



<u>HW metrics:</u>
Throughput / latency;
Resource utilization;
Energy / power;

2) Difficult to select appropriate reference DNNs at the beginning

- Choose by experience
- Performance on published datasets

Outline:

1) Background & Challenges

    Edge AI is necessary but challenging

2) Motivations

    Two major issues prevent better AI quality on embedded systems

3) The Proposed SkyNet Solution

    A bottom-up approach for building hardware-efficient DNNs

4) Demonstrations on Object Detection and Tracking Tasks

    Double champions in an international system design competition

    Faster and better results than trackers with ResNet-50 backbone

5) Conclusions

The proposed flow

To overcome drawbacks, we propose a bottom-up DNN design flow:

- No reference DNNs; Start from scratch;
- Consider HW constraints; Reflect SW variations

It needs Bundle to cover both SW and HW perspectives

SW part:
DNN Models

determine

determine

HW part:
Embedded devices
which run the DNN

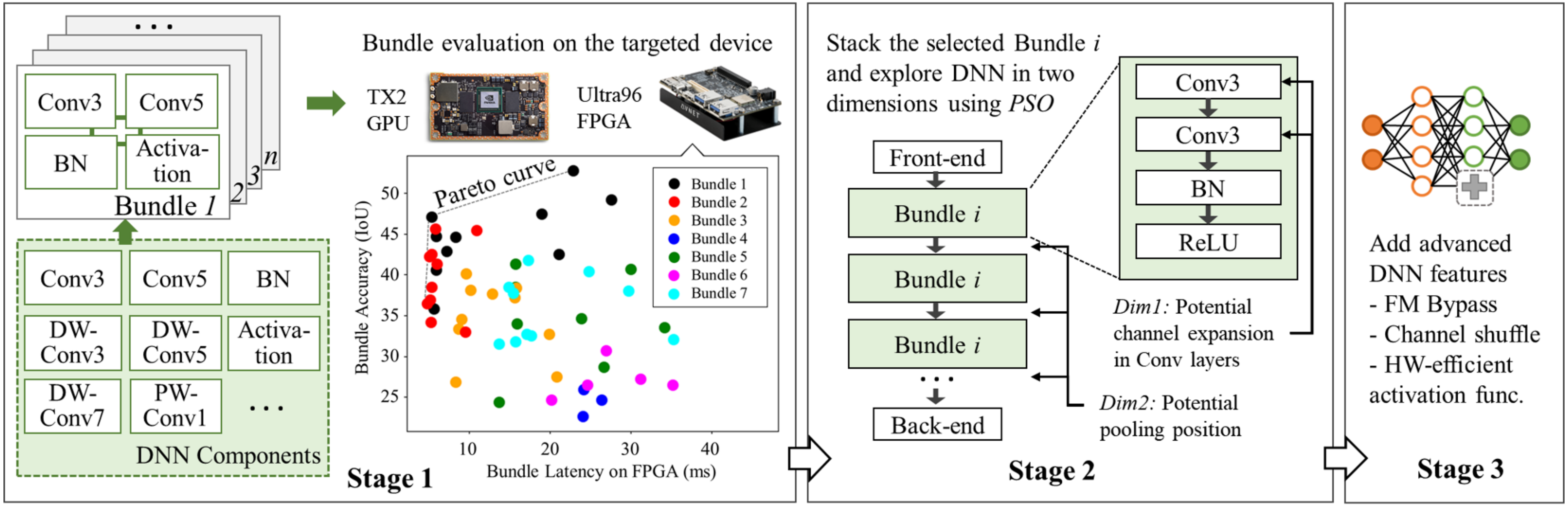
Perspectives
- SW : a set of sequential DNN layers (stack to build DNNs)
- HW: a set of IPs to be implemented on hardware

Bundle

The proposed flow [overview]

- It is a three-stage flow

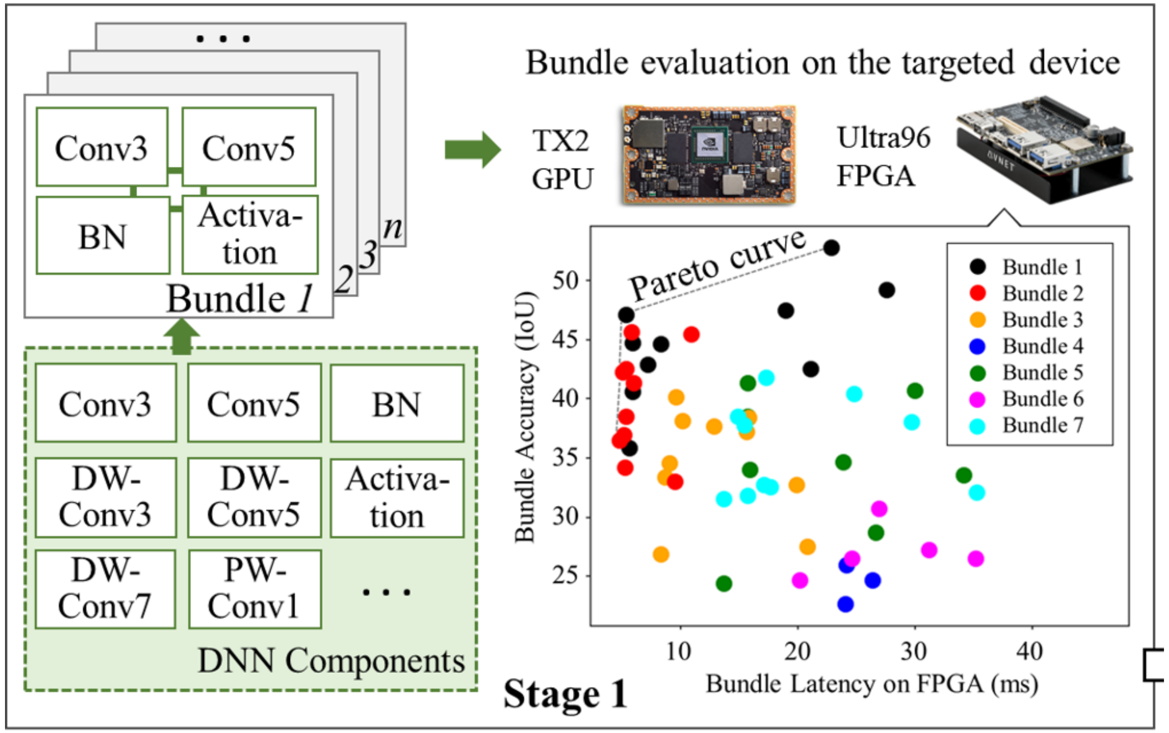
  Select Bundles -> Explore network architectures -> Add features



The proposed flow [stage 1]

➢ Start building DNNs from choosing the HW-aware Bundles

Goal: Let Bundles capture HW features and accuracy potentials

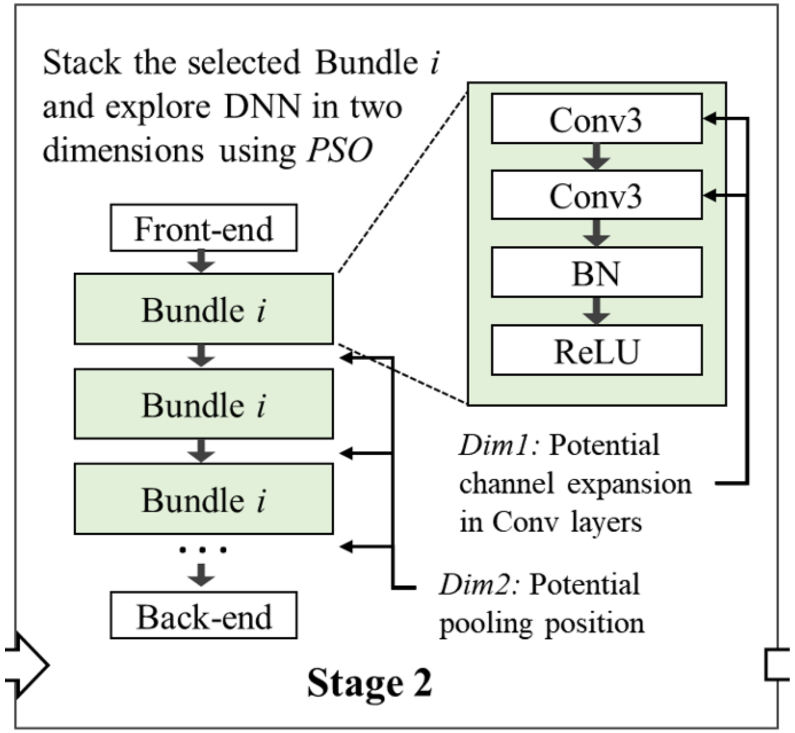


- Prepare DNN components
- Enumerate Bundles
- Evaluate Bundles (Latency-Accuracy)
- Select those in the Pareto curve

The proposed flow [stage 2]

➢ Start exploring DNN architecture to meet HW-SW metrics

Goal: Solve the multi-objective optimization problem

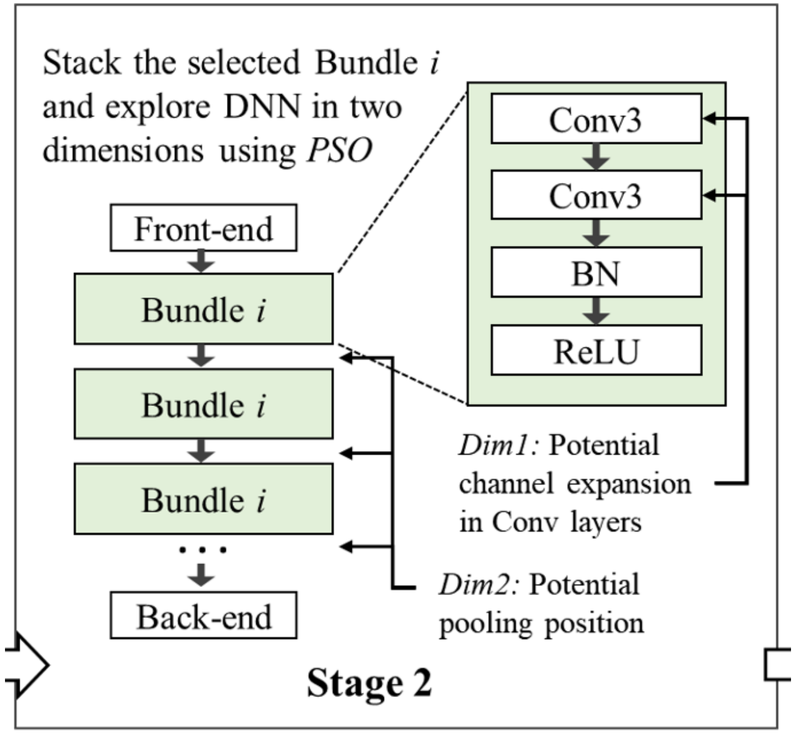


- Stack the selected Bundle
- Explore two hyper parameters using *PSO* (channel expansion factor & pooling spot)
- Evaluate DNN candidates (Latency-Accuracy)
- Select candidates in the Pareto curve

The proposed flow [stage 2] (cont.)

➢ Adopt a group-based ***PSO*** *(particle swarm optimization)*

- Multi-objective optimization: ***Latency-Accuracy***
- Group-based evolve: Candidates with the same **Bundle** are in the same group



Fitness Score:

$$Fit_j^i = \boxed{Acc_j^i} + \alpha \cdot (\boxed{Est(n_j^i)} - \boxed{Tar})$$

Candidate accuracy

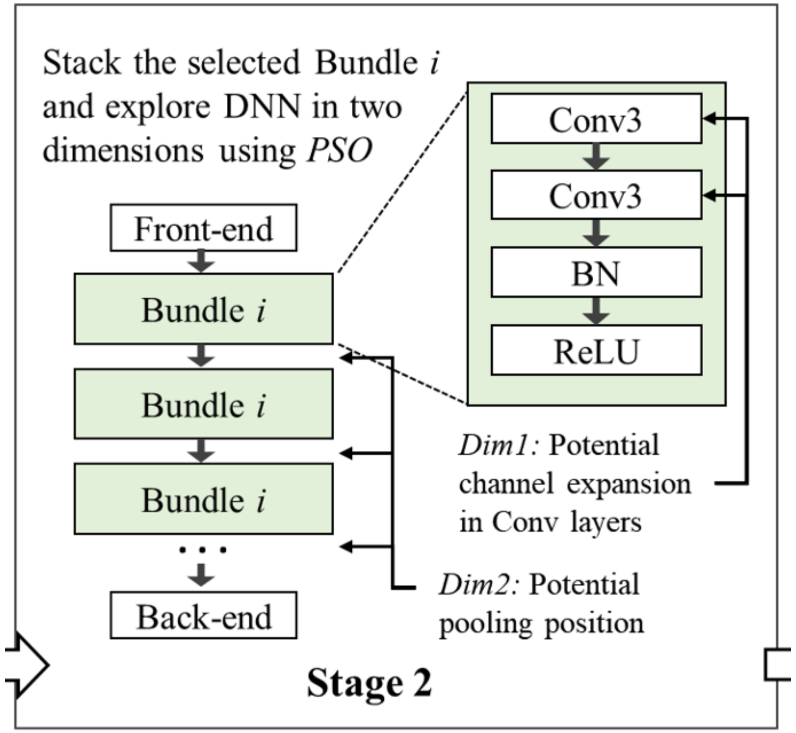
Candidate latency in hardware

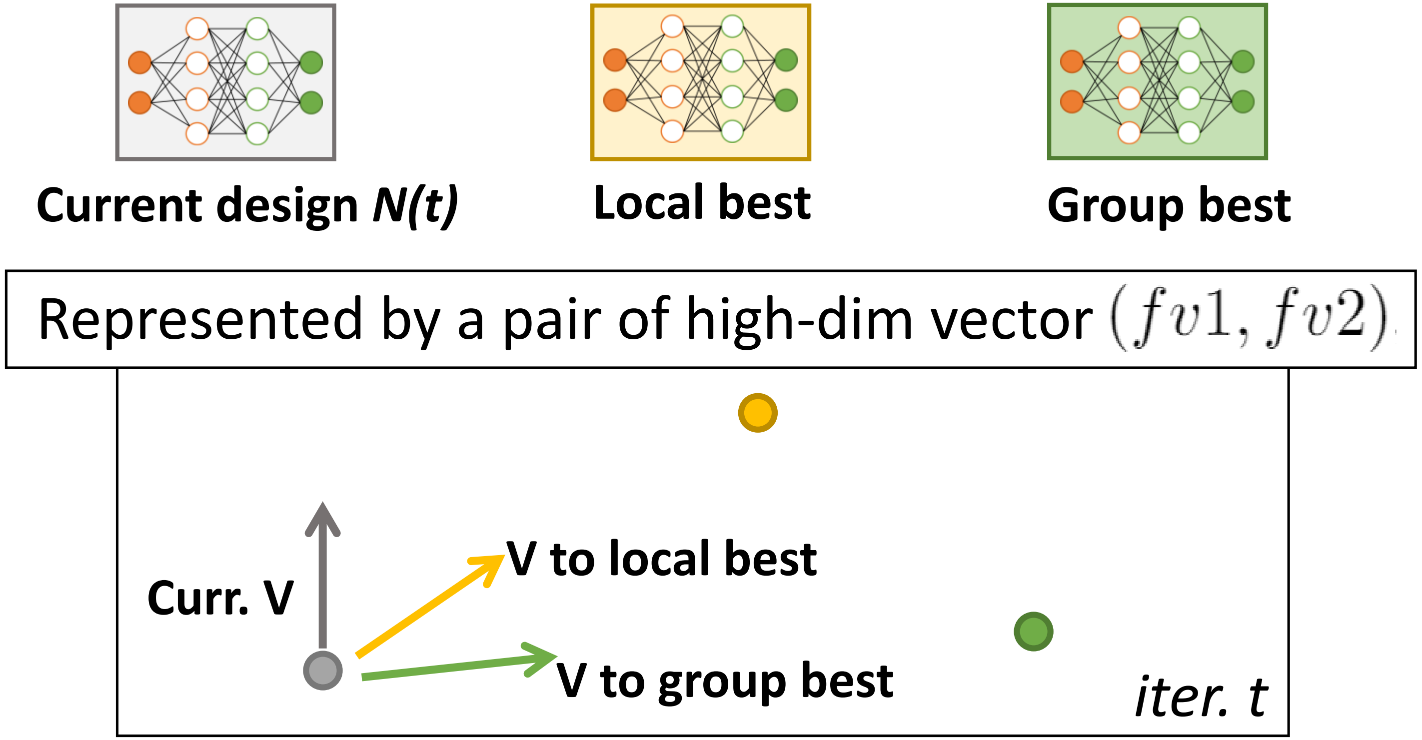
Targeted latency

α factor to balance accuracy and latency

The proposed flow [stage 2] (cont.)

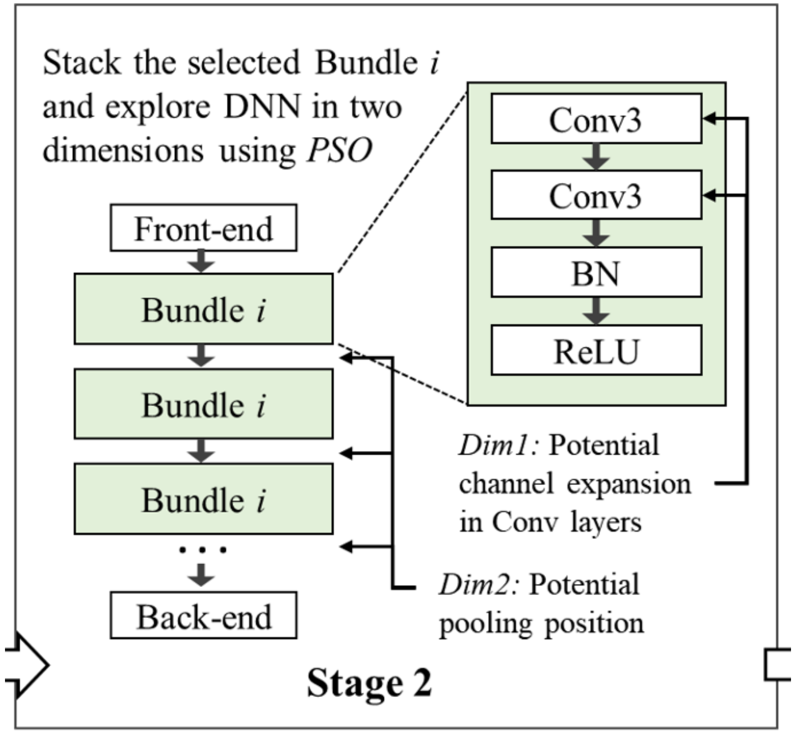
- Adopt a group-based ***PSO*** *(particle swarm optimization)*
- Multi-objective optimization: ***Latency-Accuracy***
- Group-based evolve: Candidates with the same **Bundle** are in the same group

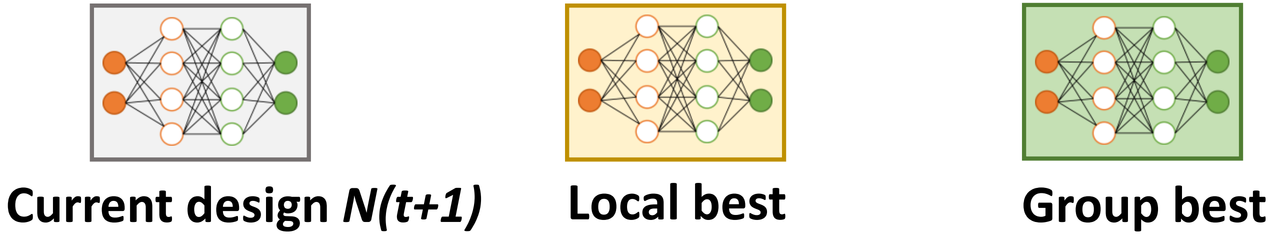




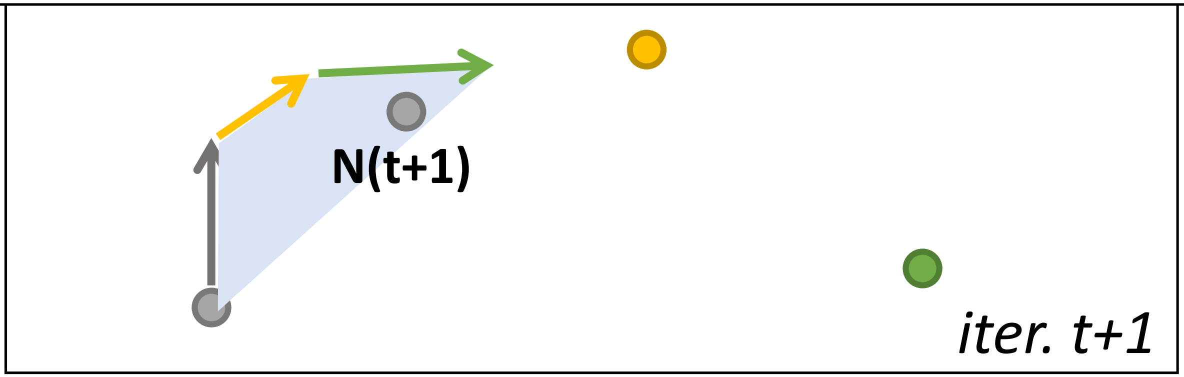
The proposed flow [stage 2] (cont.)

- ➢ Adopt a group-based ***PSO*** *(particle swarm optimization)*
- Multi-objective optimization: ***Latency-Accuracy***
- Group-based evolve: Candidates with the same **Bundle** are in the same group





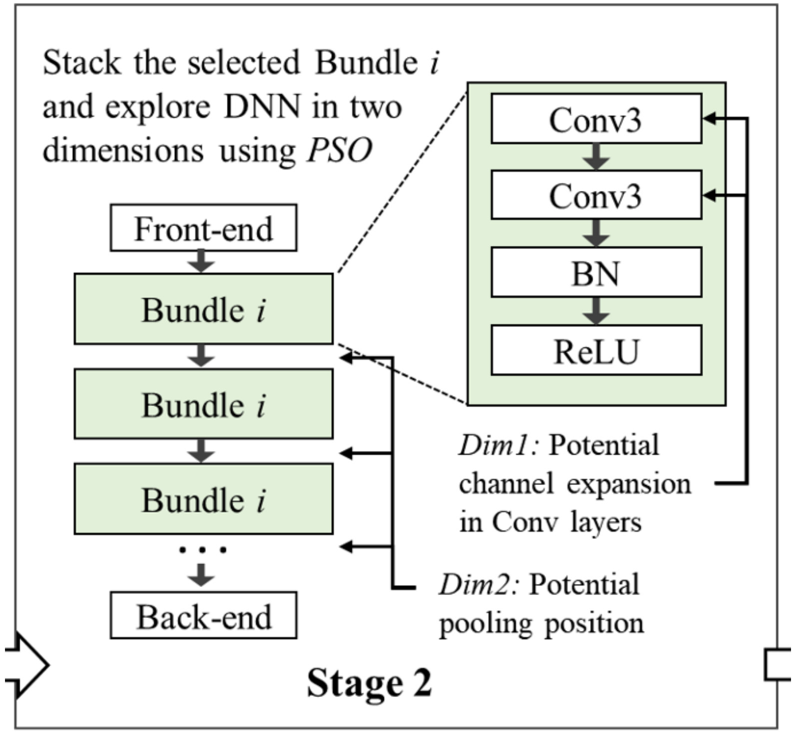
Represented by a pair of high-dim vector $(fv1, fv2)$

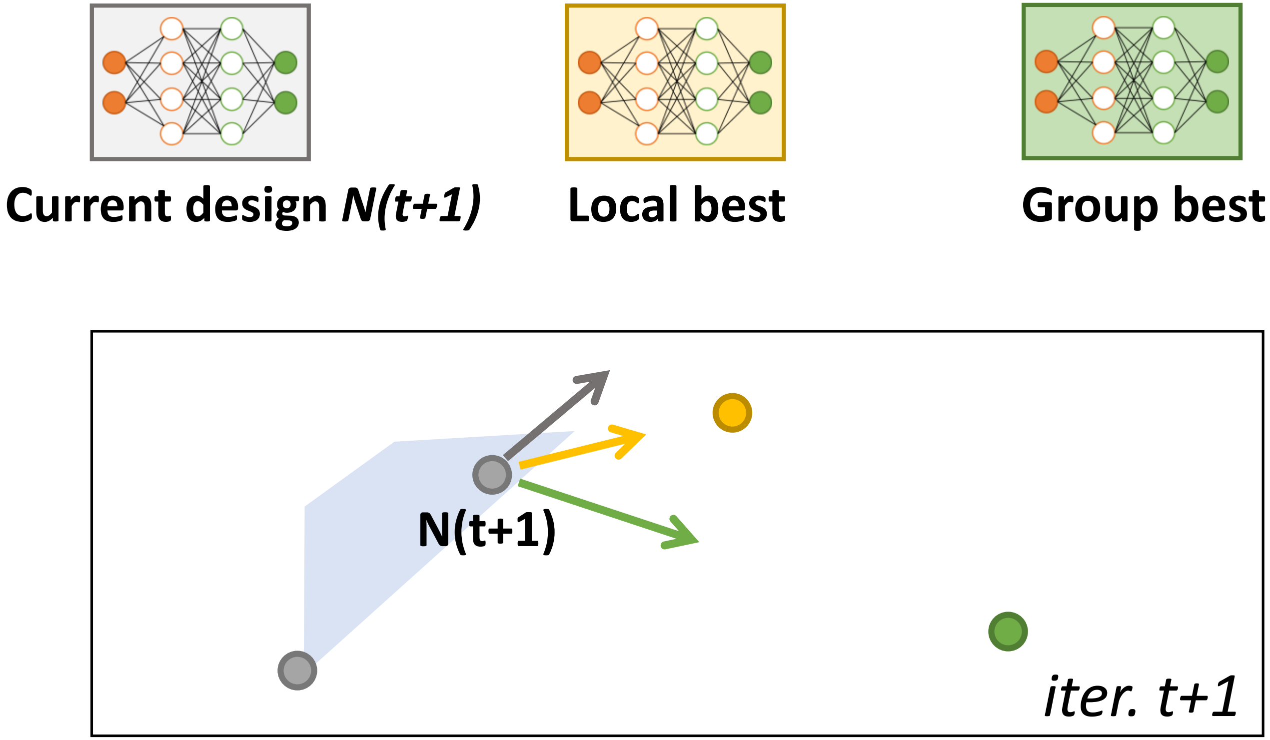


The proposed flow [stage 2] (cont.)

➢ Adopt a group-based ***PSO*** *(particle swarm optimization)*

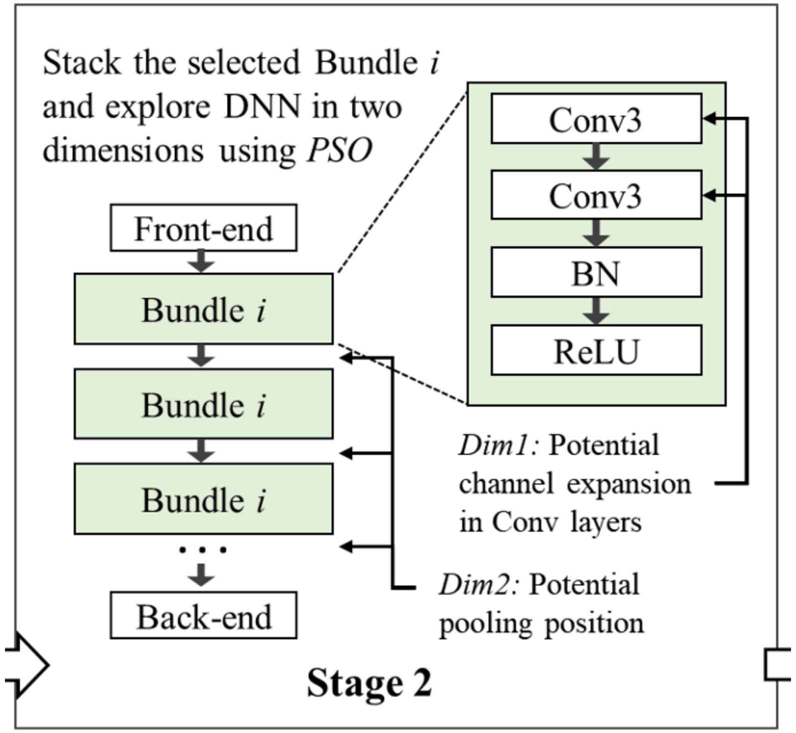
- Multi-objective optimization: ***Latency-Accuracy***
- Group-based evolve: Candidates with the same **Bundle** are in the same group

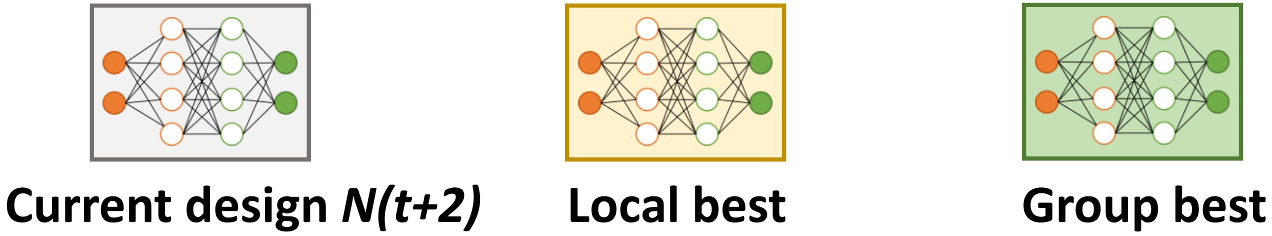


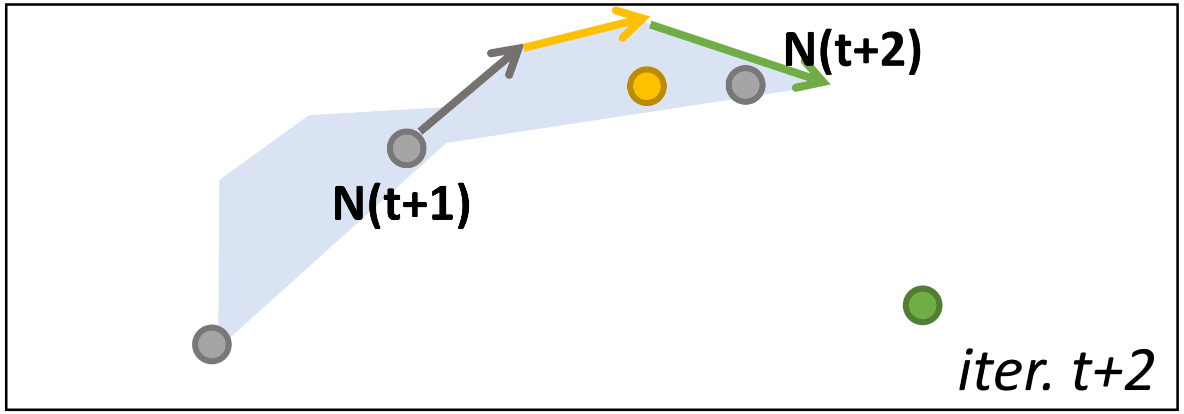


The proposed flow [stage 2] (cont.)

- Adopt a group-based ***PSO*** *(particle swarm optimization)*
  - Multi-objective optimization: ***Latency-Accuracy***
  - Group-based evolve: Candidates with the same **Bundle** are in the same group



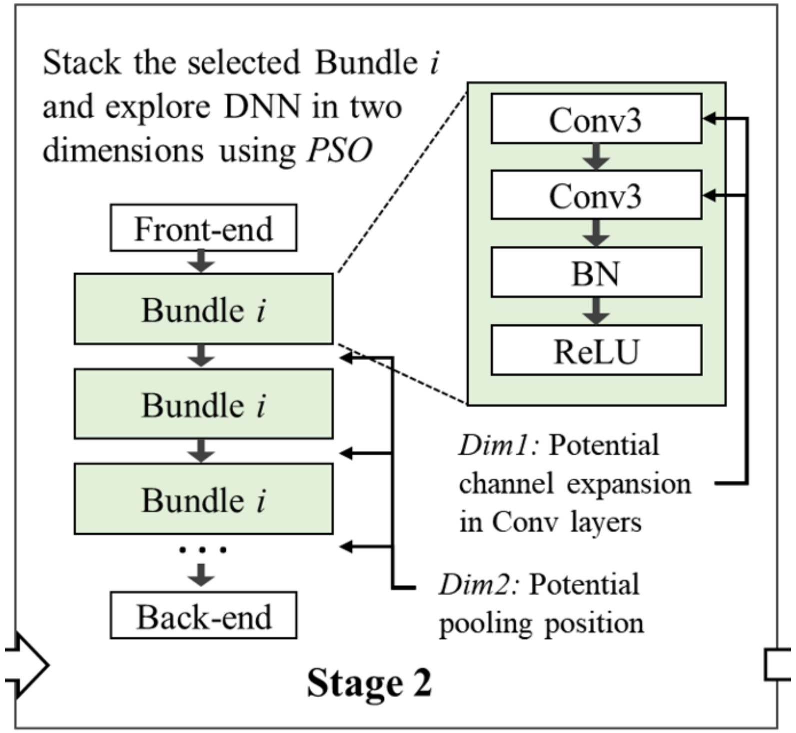


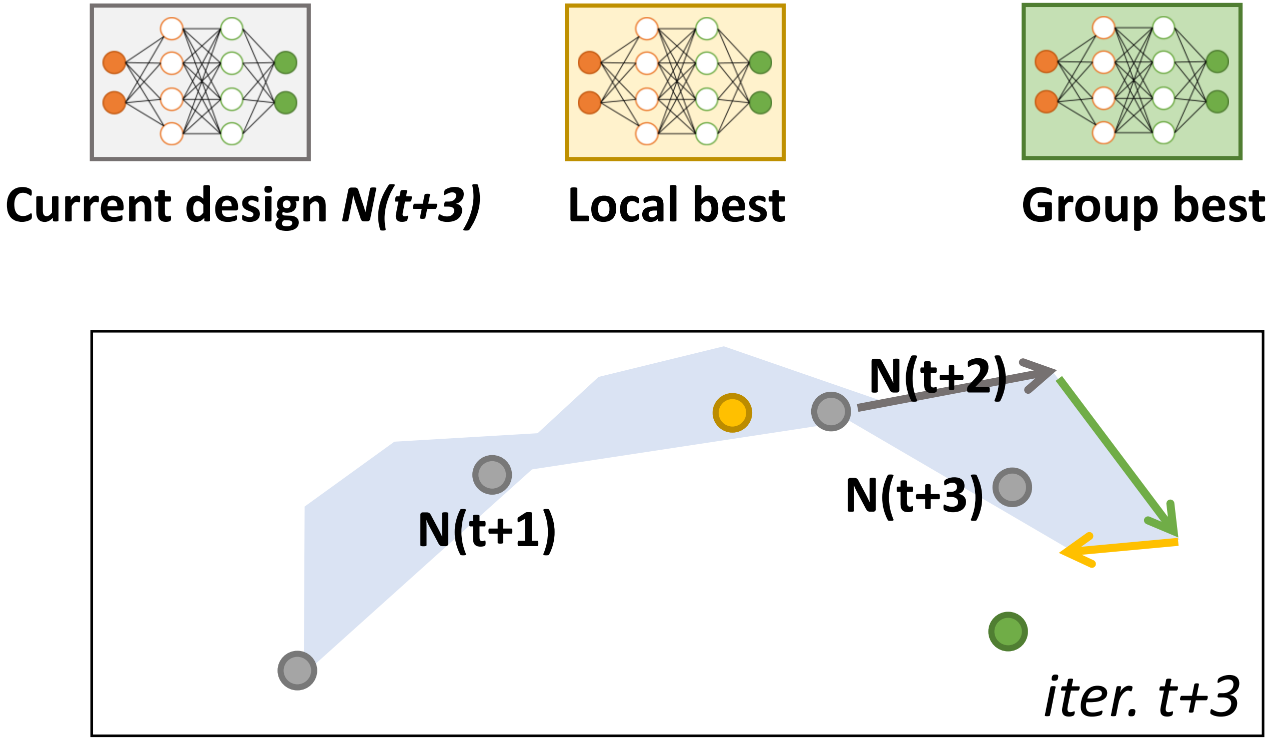


The proposed flow [stage 2] (cont.)

➢ Adopt a group-based ***PSO*** *(particle swarm optimization)*

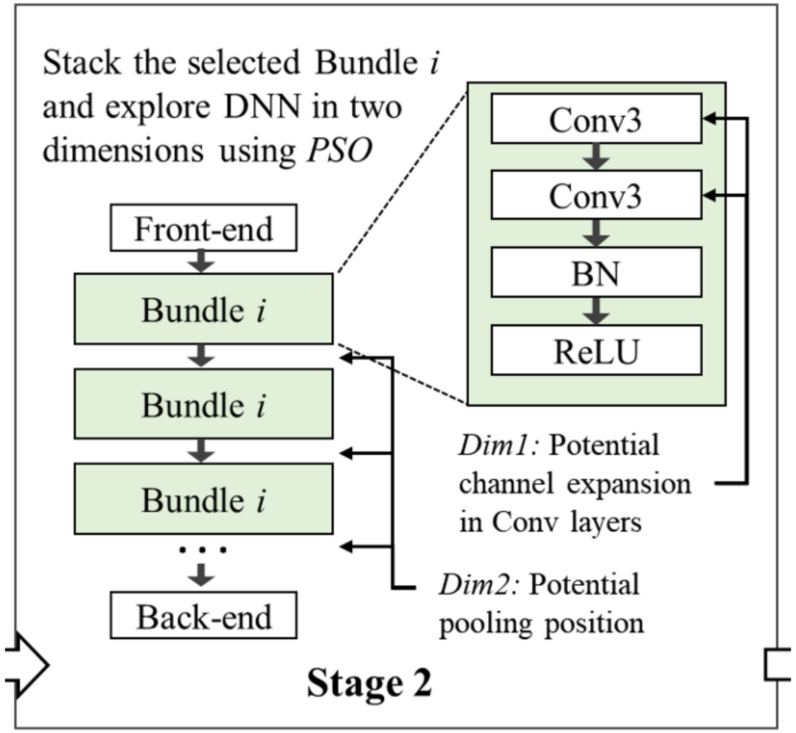
- Multi-objective optimization: ***Latency-Accuracy***
- Group-based evolve: Candidates with the same **Bundle** are in the same group

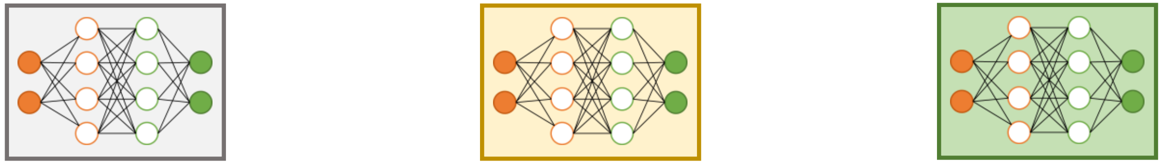




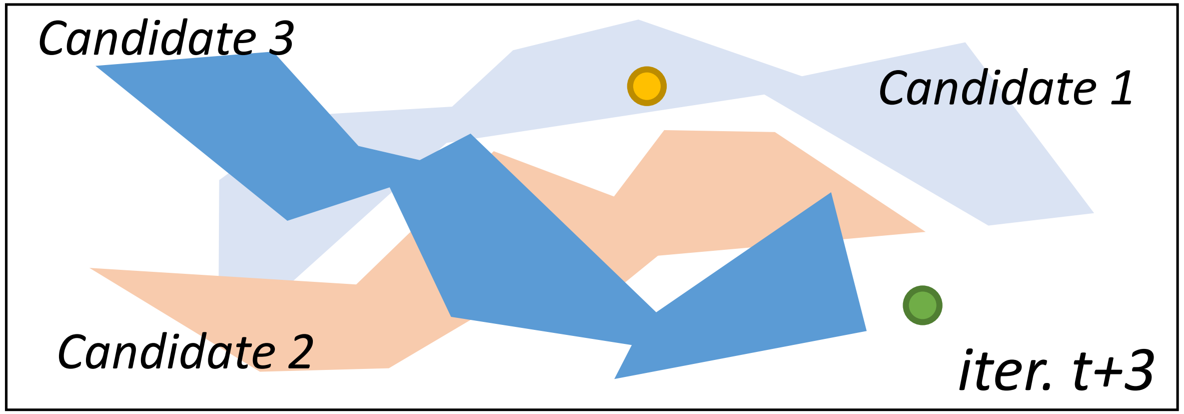
The proposed flow [stage 2] (cont.)

- Adopt a group-based ***PSO*** *(particle swarm optimization)*
- Multi-objective optimization: ***Latency-Accuracy***
- Group-based evolve: Candidates with the same **Bundle** are in the same group





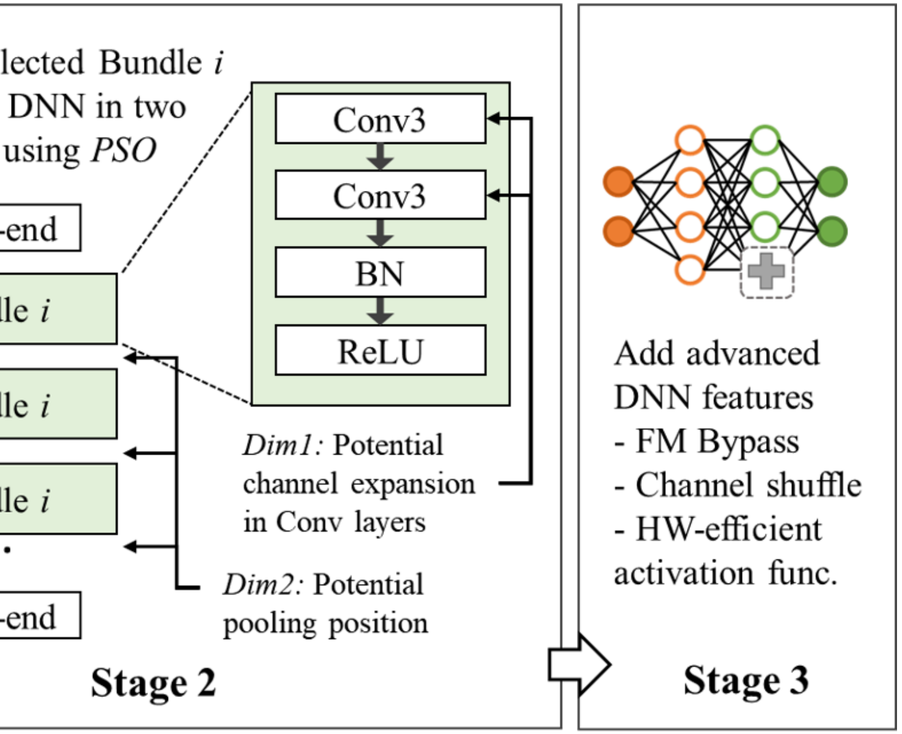
Current design *N(t+3)* **Local best** **Group best**



The proposed flow [stage 3]

➢ Add more features if HW constraints allow

Goal: better fit in the customized scenario



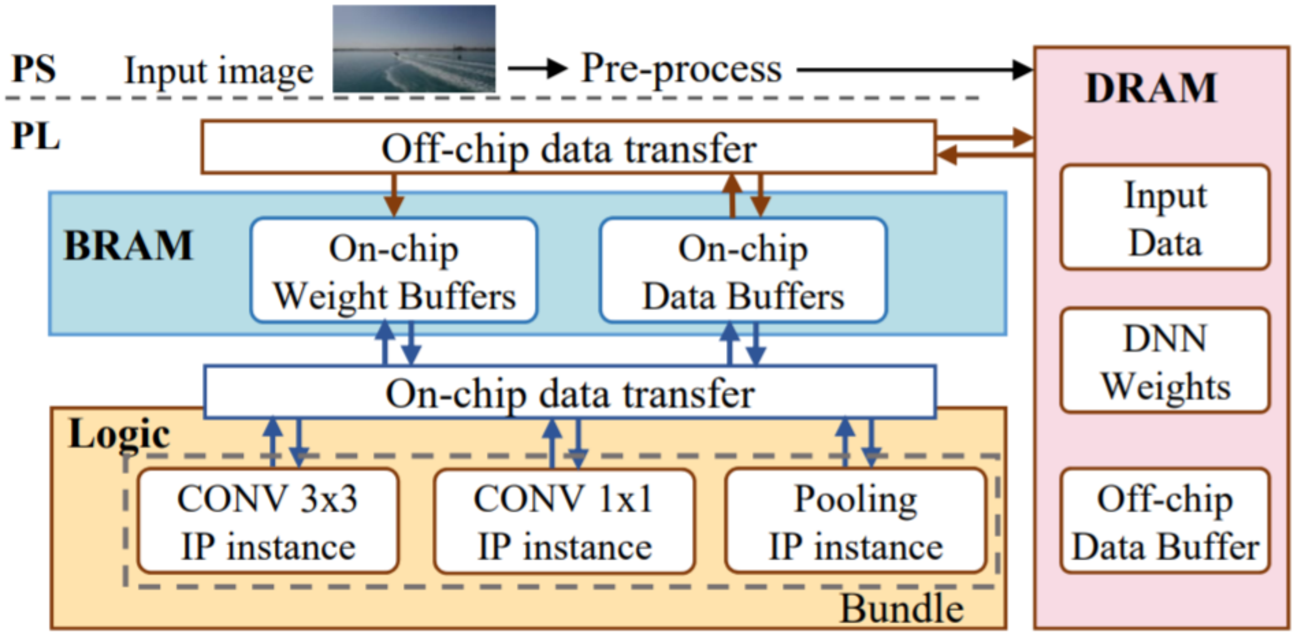
- For small object detection, we add feature map bypass
- Feature map reordering
- For better HW efficiency, we use ReLU6

The proposed flow [HW deployment]

- We start from a well-defined accelerator architecture

  Two-level memory hierarchy to fully utilize given memory resources

  IP-based scalable process engines to fully utilize computation resources

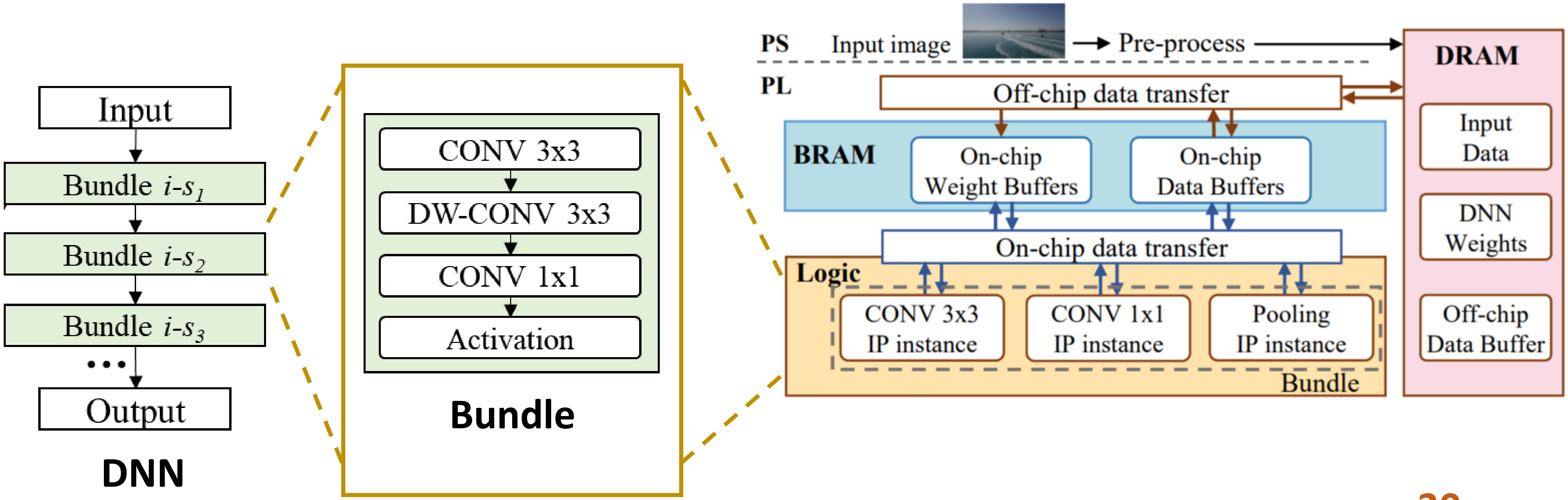


The proposed flow [HW deployment]

➢ We start from a well-defined accelerator architecture

Limit the DNN design space

Enable fast performance evaluation



Outline:

1) Background & Challenges

   Edge AI is necessary but challenging

2) Motivations

   Two major issues prevent better AI quality on embedded systems

3) The Proposed SkyNet Solution

   A bottom-up approach for building hardware-efficient DNNs

4) Demonstrations on Object Detection and Tracking Tasks

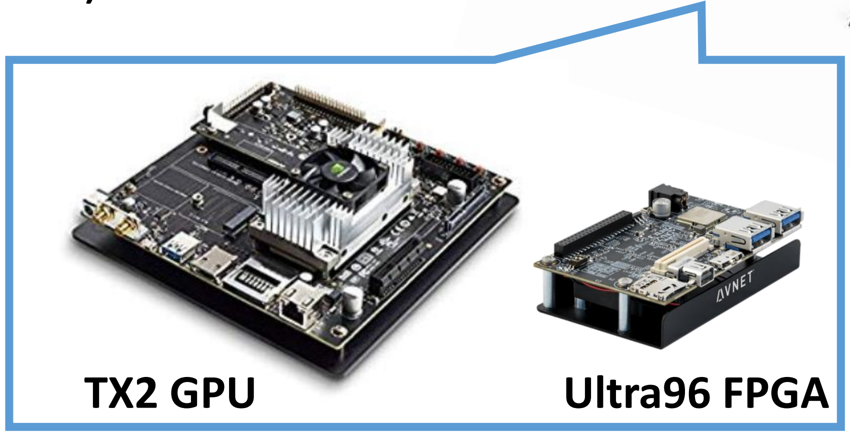
   Double champions in an international system design competition

   Faster and better results than trackers with ResNet-50 backbone

5) Conclusions

Demo #1: an object detection task for drones

- System Design Contest for ***low power object detection*** in the IEEE/ACM Design Automation Conference (DAC-SDC)



- DAC-SDC targets single object detection for real-life UAV applications

  Images contain 95 categories of targeted objects (most of them are small)

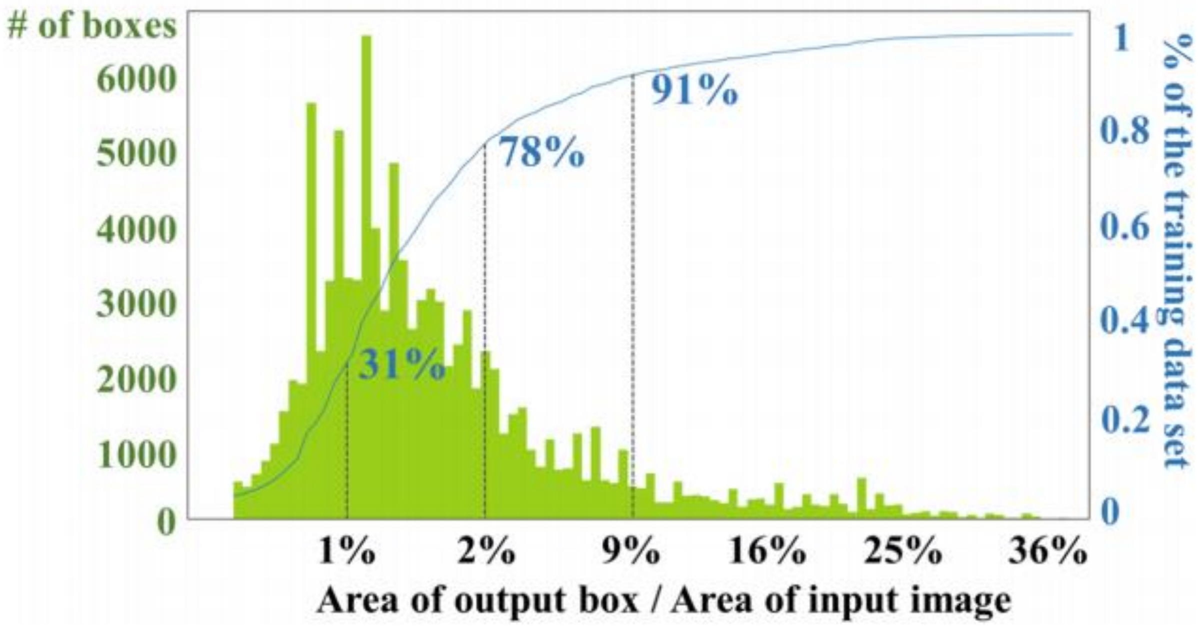
- Comprehensive evaluation: ***accuracy, throughput, and energy consumption***

$$TS_i = R_{IoU_i} \times (1 + ES_i)$$

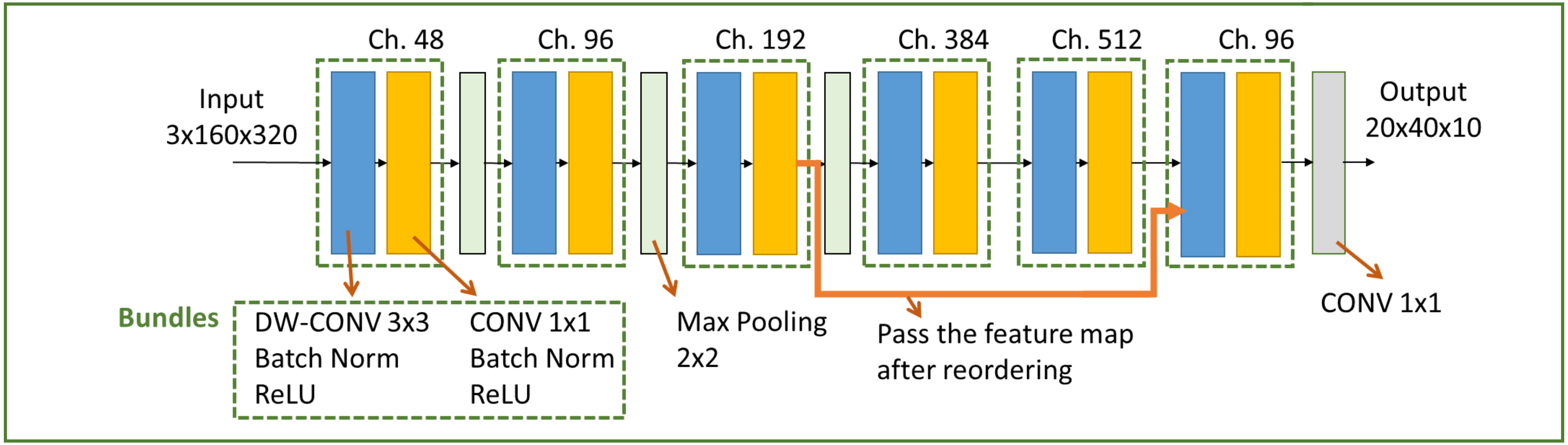
Demo #1: DAC-SDC dataset

- ➢ The distribution of target relative size compared to input image

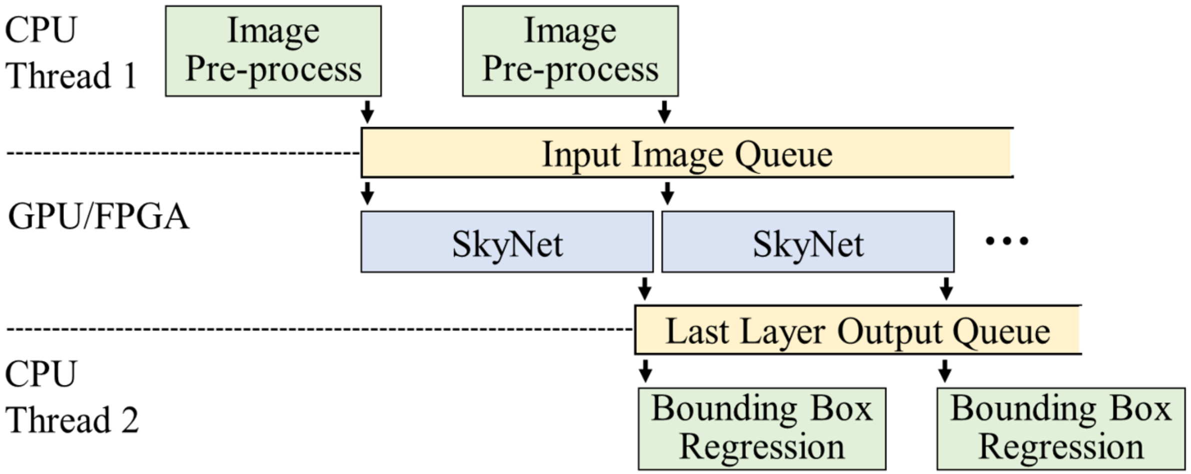
31% targets < 1% of the input size
91% targets < 9% of the input size



Demo #1: the proposed DNN architecture

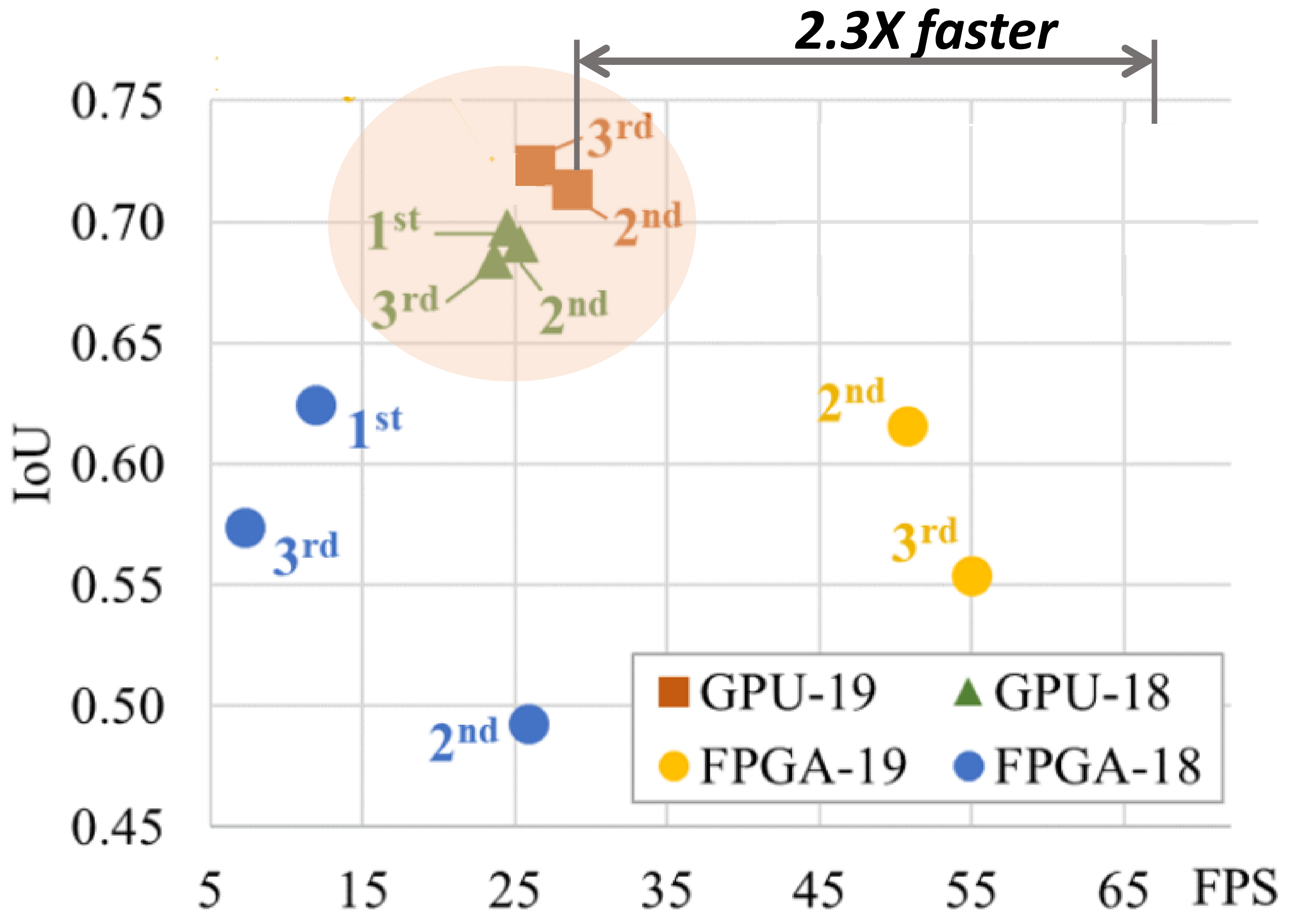


- 13 CONV with 0.4 million parameters
- For Embedded FPGA: Quantization, Batch, Tiling, Task partitioning
- For Embedded GPU: Task partitioning



Demo #1: Results from DAC-SDC [GPU]

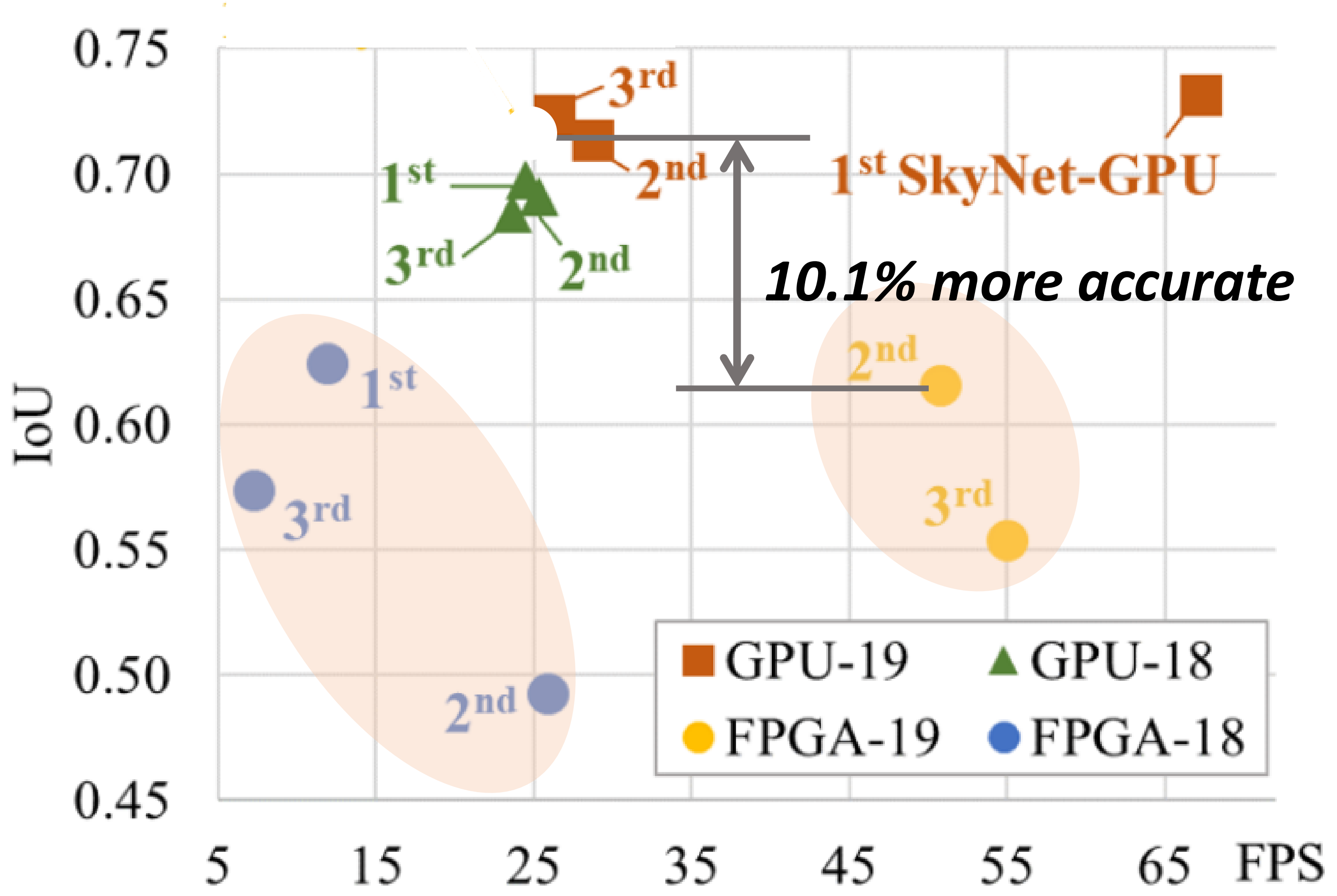
➢ Evaluated by 50k images in the official test set



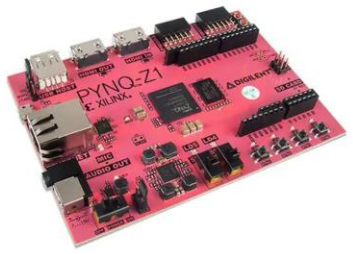
Designs using TX2 GPU

Demo #1: Results from DAC-SDC [FPGA]

➢ Evaluated by 50k images in the official test set



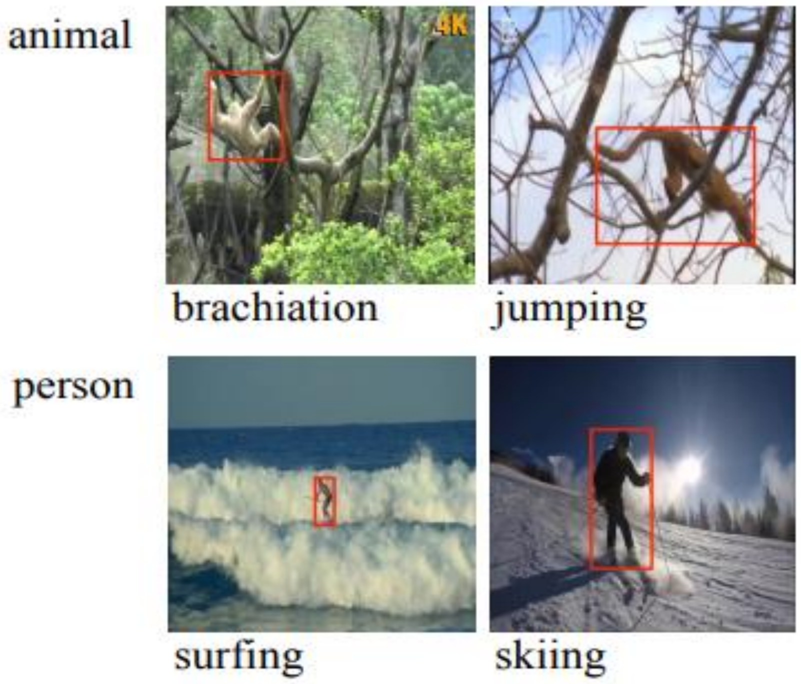
’19 Designs using Ultra96 FPGA



’18 Designs using Pynq-Z1 FPGA

Demo #2: generic object tracking in the wild

- We extend SkyNet to real-time tracking problems
- We use a large-scale high-diversity benchmark called ***Got-10K***
  - ***Large-scale:*** 10K video segments with 1.5 million labeled bounding boxes
  - **Generic:** 560+ classes and 80+ motion patterns (better coverage than others)



[From Got-10K]

Demo #2: Results from Got-10K

- Evaluated using two state-of-the-art trackers with single 1080Ti

SiamRPN++ with different backbones

Backbone	AO	$SR_{0.50}$	$SR_{0.75}$	FPS
AlexNet	0.354	0.385	0.101	52.36
ResNet-50	0.365	0.411	0.115	25.90
SkyNet	0.364	0.391	0.116	41.22

Similar AO, 1.6X faster vs. ResNet-50

SiamMask with different backbones

Backbone	AO	$SR_{0.50}$	$SR_{0.75}$	FPS
ResNet-50	0.380	0.439	0.153	17.44
SkyNet	0.390	0.442	0.158	30.15

Slightly better AO, 1.7X faster vs. ResNet-50

Outline:

1) Background & Challenges

   Edge AI is necessary but challenging

2) Motivations

   Two major issues prevent better AI quality on embedded systems

3) The Proposed SkyNet Solution

   A bottom-up approach for building hardware-efficient DNNs

4) Demonstrations on Object Detection and Tracking Tasks

   Double champions in an international system design competition

   Faster and better results than trackers with ResNet-50 backbone

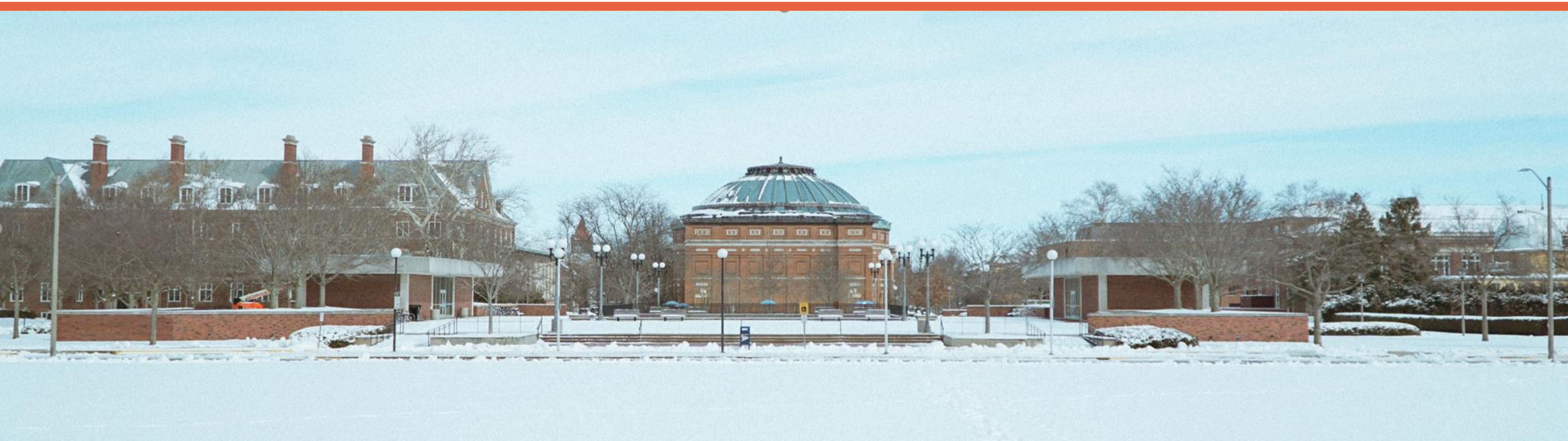
5) Conclusions

Conclusions

- We presented ***SkyNet*** & ***a hardware-efficient DNN design method***
  - a bottom-up DNN design flow for embedded systems
  - an effective way to capture realistic HW constraints
  - a solution to satisfy demanding HW and SW metrics

- ***SkyNet*** has been demonstrated by object detection and tracking tasks
  - Won the double champions in DAC-SDC
  - Achieved faster and better results than trackers using ResNet-50

- ***Scan for paper, slides, poster, code, & demo***
- ***Please come to Poster #11***



Thank you

MLSys