UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

ES CAD CAD for Emerging Systems

Design of Reconfigurable Computing Systems for Smart IoT Applications

Deming Chen

Abel Bliss Professor of Engineering

Department of Electrical and Computer Engineering

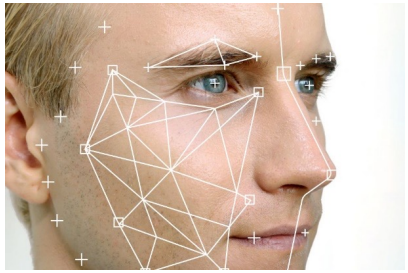
University of Illinois at Urbana-Champaign

IDEAL 2021

Deploying AI workloads in the cloud for IoT

Major requirements：

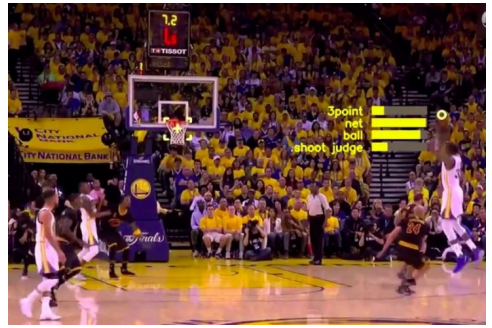
- High throughput performance
- Short tail latency
- High power efficiency



Facial recognition

Speech recognition, translation

Recommendations

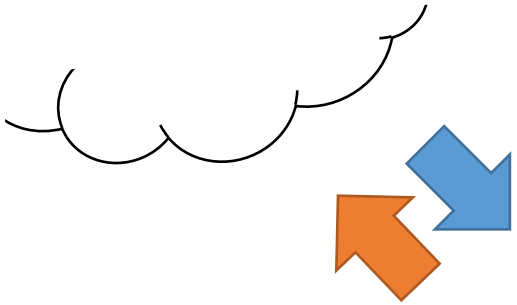


Auto-gen Sport Highlights

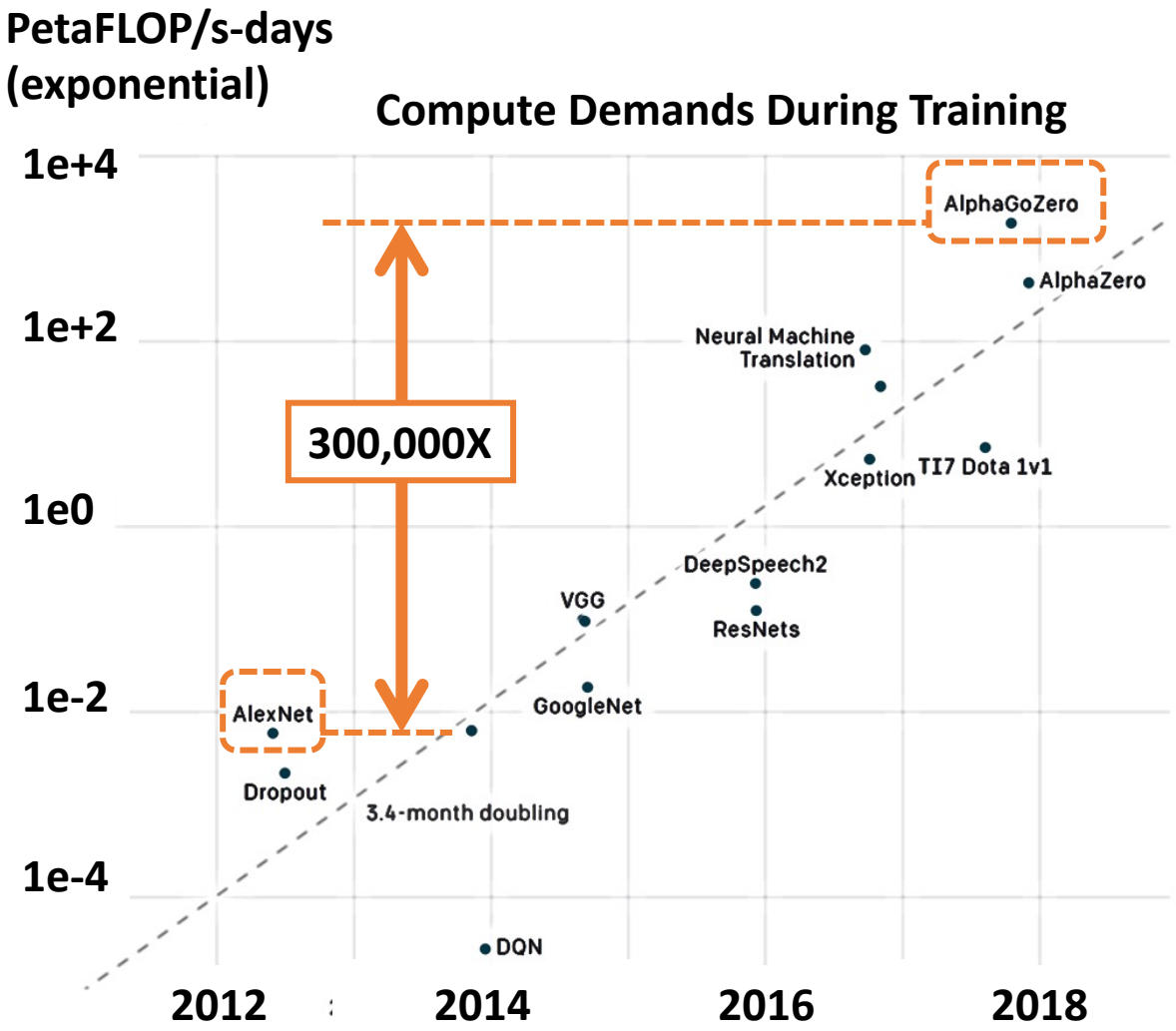
Deploying AI workloads at the edge for IoT

Major requirements：

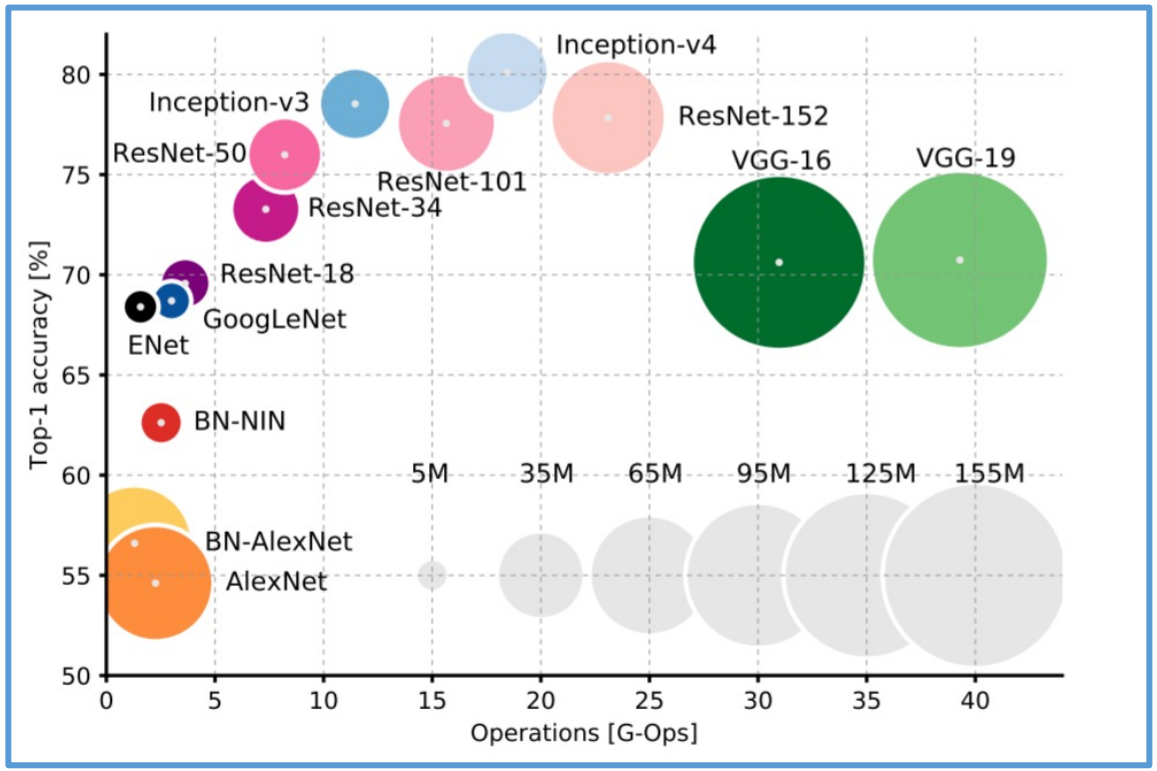
- Real-time ability
- Energy efficient design
- Area/form-factor constraint



AI Challenge #1: Huge Compute Demands

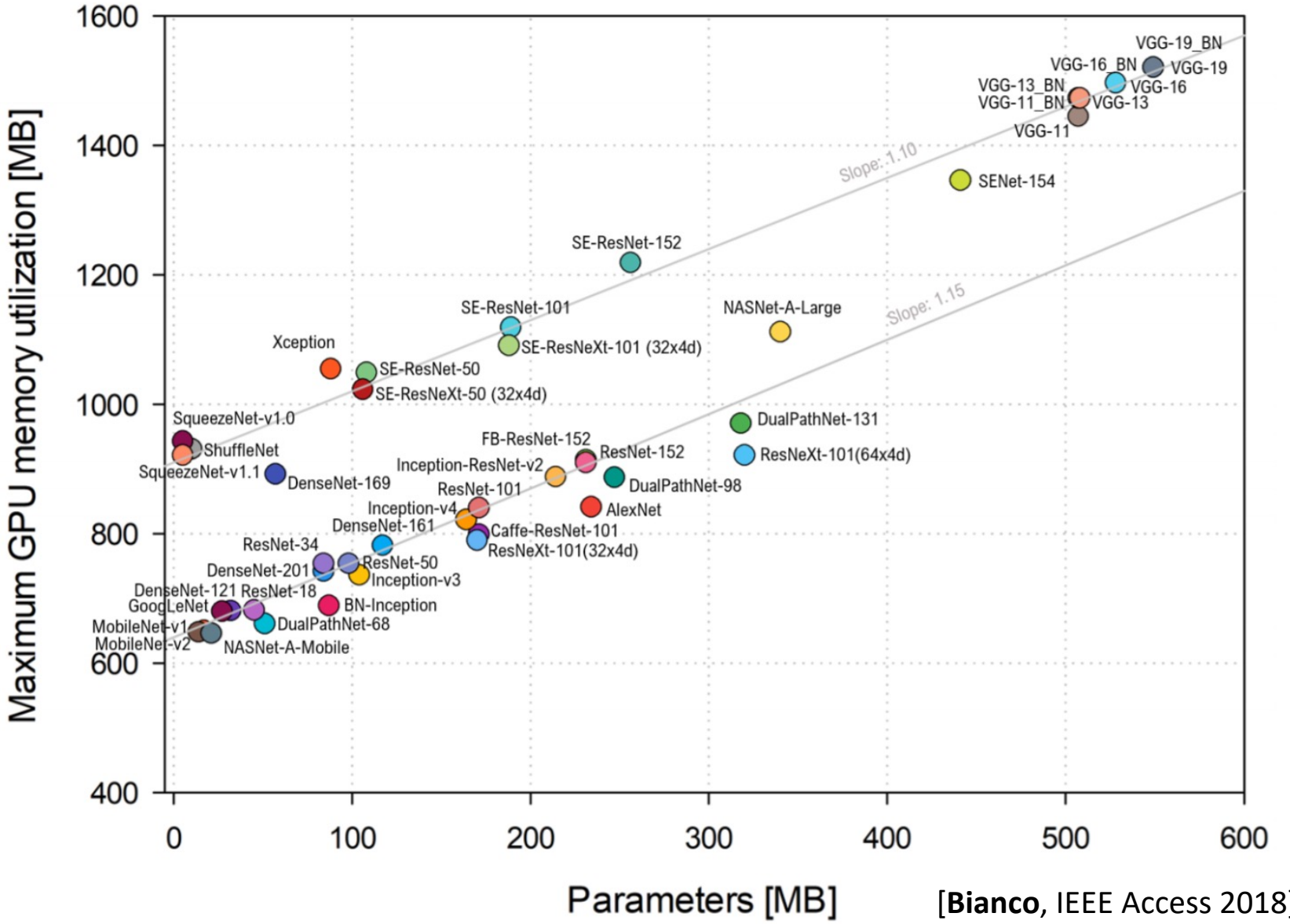


https://openai.com/blog/ai-and-compute/



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

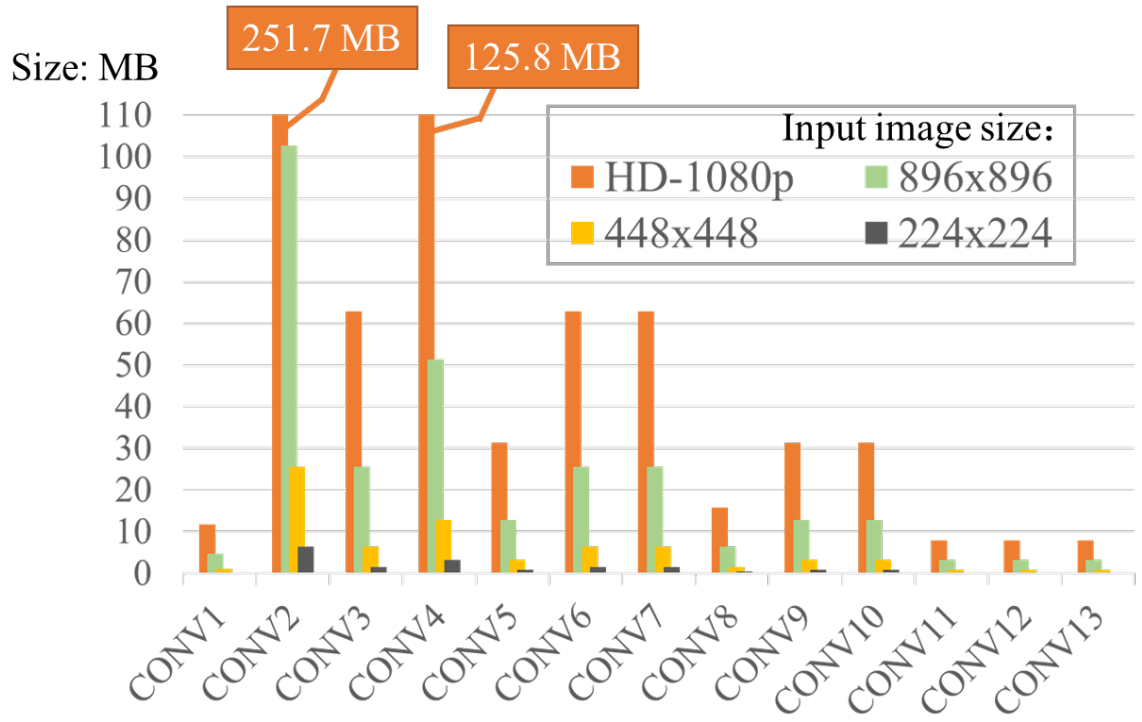
AI Challenge #2: Massive Memory Footprint



[**Bianco**, IEEE Access 2018]

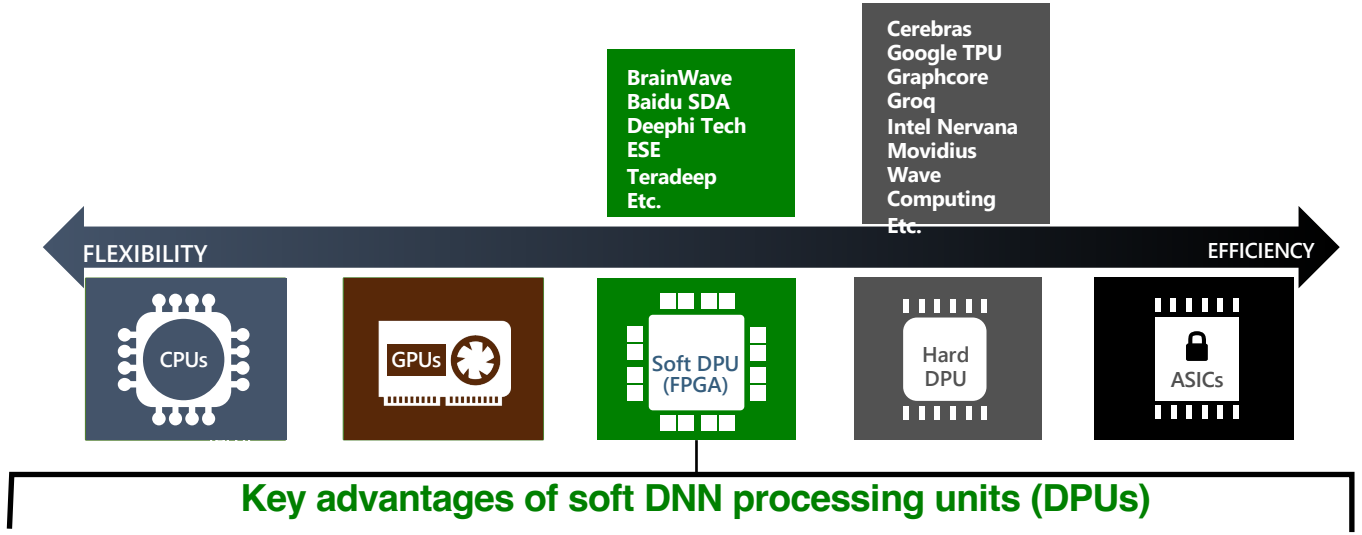
AI Challenge #2: Massive Memory Footprint

- 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



Input feature map sizes for VGG-16 CONV layers

Industry Landscape of DNN Acceleration



Performance

- Excellent inference performance at low batch sizes
- Ultra-low latency serving on modern DNNs
  - >10X lower than CPUs and GPUs
- Scale to many FPGAs in single DNN service

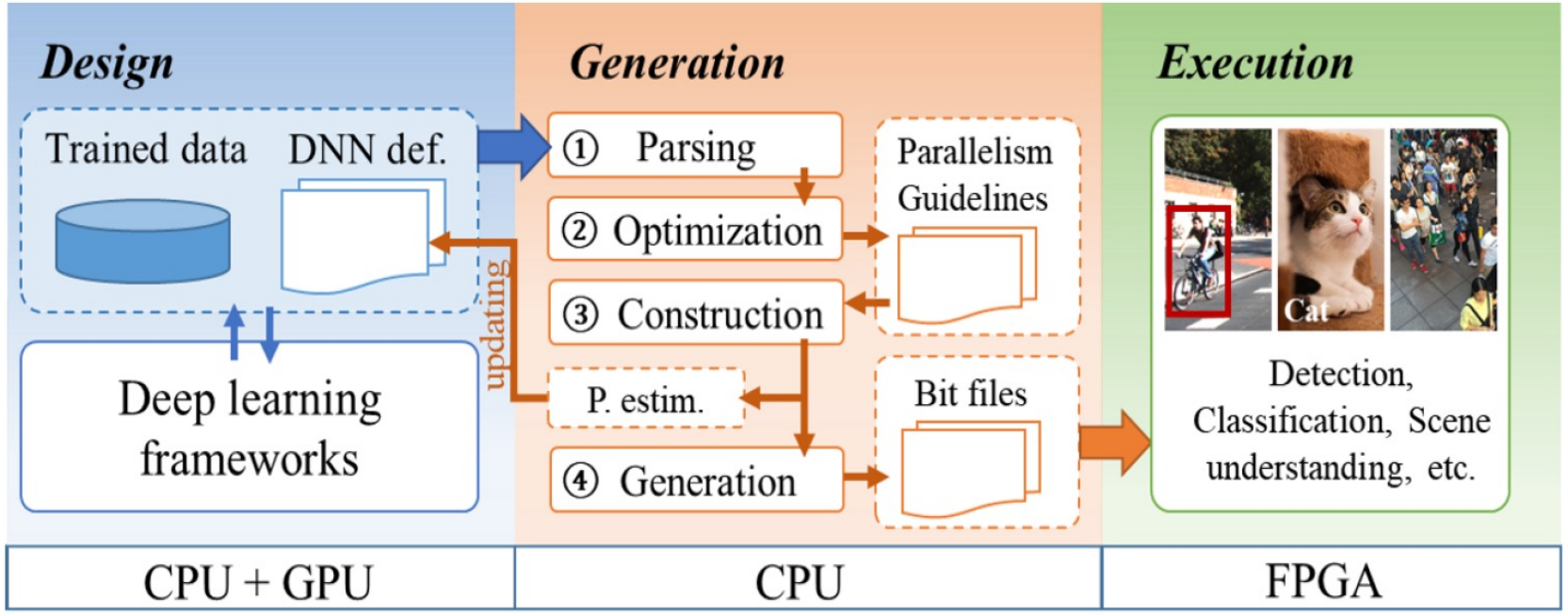
Flexibility

- FPGAs ideal for adapting to rapidly evolving ML
- CNNs, LSTMs, MLPs, reinforcement learning, feature extraction, decision trees, etc.
- Inference-optimized numerical precision
- Exploit sparsity, deep compression for larger, faster models

[source: Materials adapted from Microsoft BrainWave presentation at HotChips'2017]

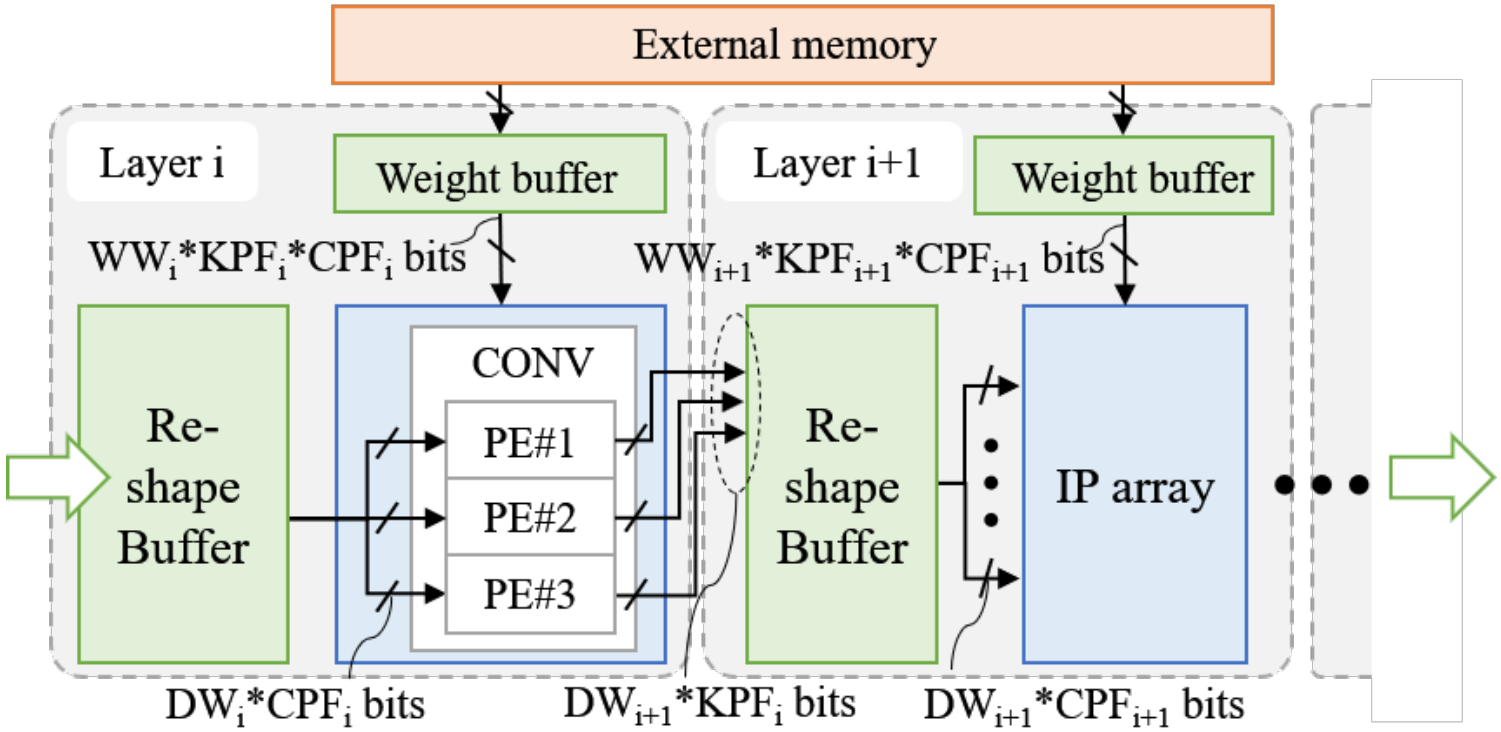
DNNBuilder – Building DNN from Ground UP

An end-to-end automation tool for mapping DNNs onto FPGAs for both edge and cloud computing



- DNN IPs as the building block
- Low initial latency
- Efficient use of FPGA on-chip memory
- Automatic on-chip resource allocation

PE Array Architecture



- **2-dim parallelism**

KPF - kernel parallel factor

CPF - channel parallel factor

- **Arbitrary quantization**

DW - bit-width for feature map

WW - bit-width for weight/bias

A “Cache” Architecture

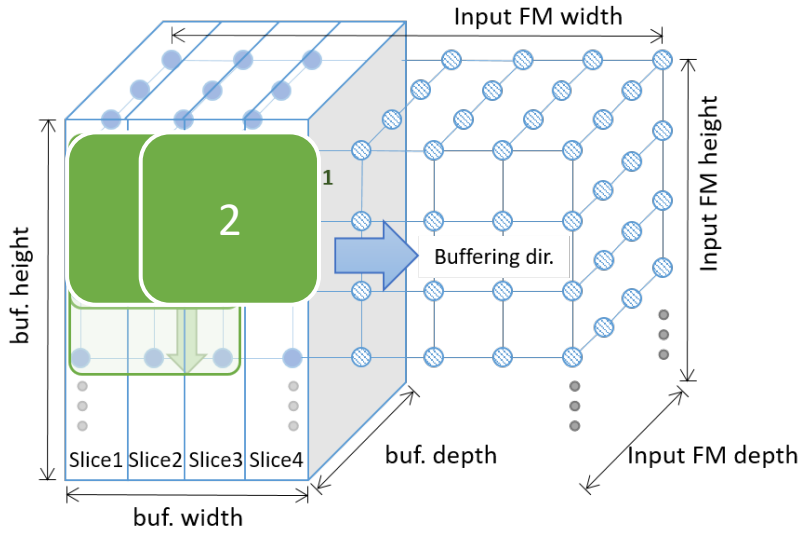
Column-based cache scheme

- **Save on-chip memory**
- **Enable layer overlapping**

For example:

Kernel size = 3

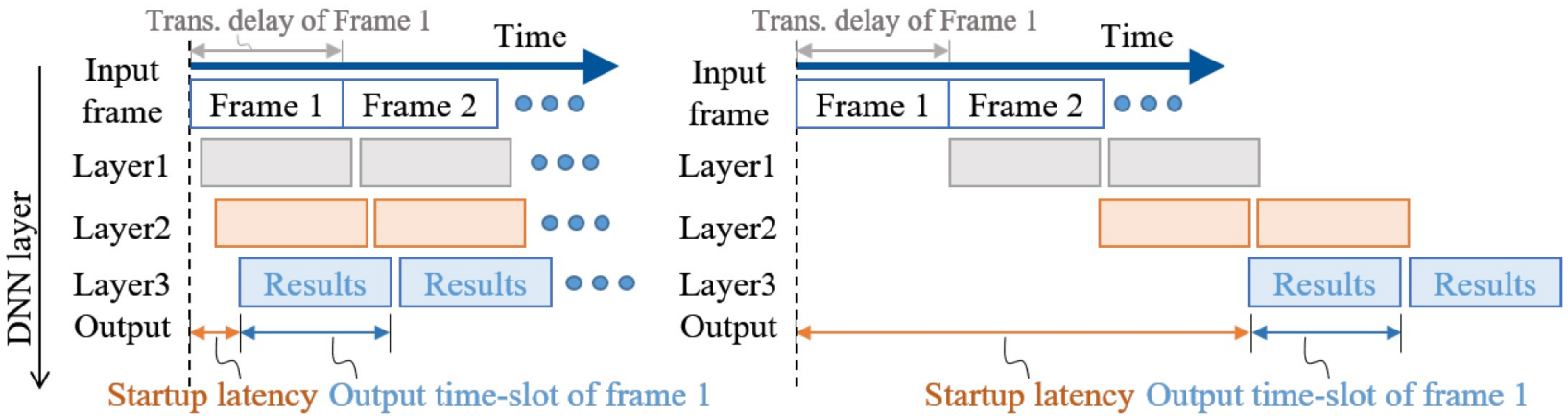
Stride = 1



4 slices cache on-chip instead of keeping the whole feature maps

Special Pipeline Architecture

A fine-grained layer-based pipelined architecture



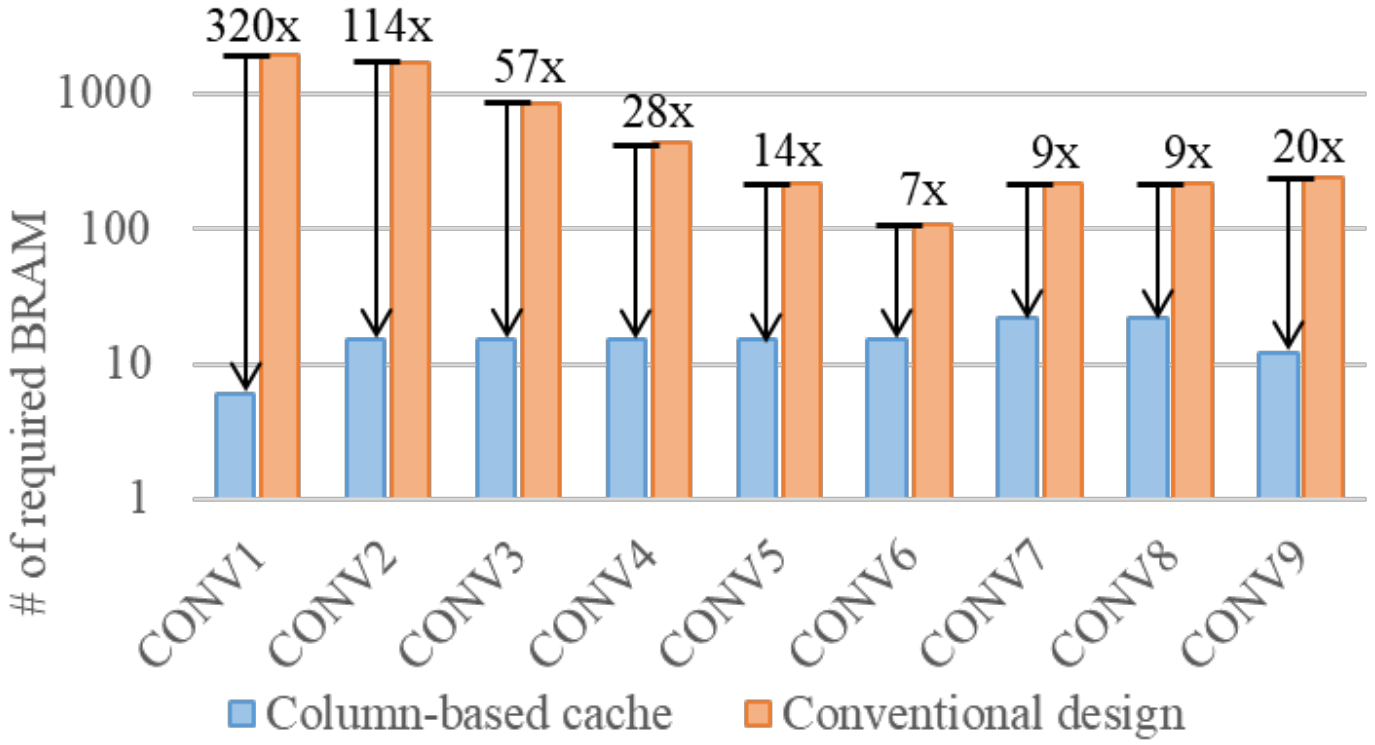
Proposed design

General design

- **Lower latency** vs. general pipeline structure
- **Higher throughput** vs. recurrent structure

Reduce 7.7x latency for running YOLO

BRAM Usage Reduction



BRAM usage reduction for storing feature maps: 320x ~ 7x

Reduce 43x BRAM usage on average for running YOLO

DNNBuilder Performance for Edge Devices

Reference	[1]	[2]	DNNBuilder
Categories	Edge-computing platforms		
FPGA chip	Zynq XC7Z045	Zynq XC7Z045	Zynq XC7Z045
Frequency	150 MHz	100 MHz	200MHz
Network	VGG	VGG	VGG
Precision	Fix16	Fix16	Fix16 (Fix8)
DSPs (used/total)	780/900	824/900	680/900
DSP Efficiency	44.0%	69.6%	96.2%
Performance (GOPS)	137	230	262 (524)
Power Efficiency (GOPS/W)	14.2	24.4	36.4 (72.8)

Zynq XC7Z045

- **LUT: 218,600**
- **FF: 437,200**
- **BRAM: 545**
- **DSP: 900**

Peaking at 524 GOPS

DNNBuilder Performance for Cloud Computing

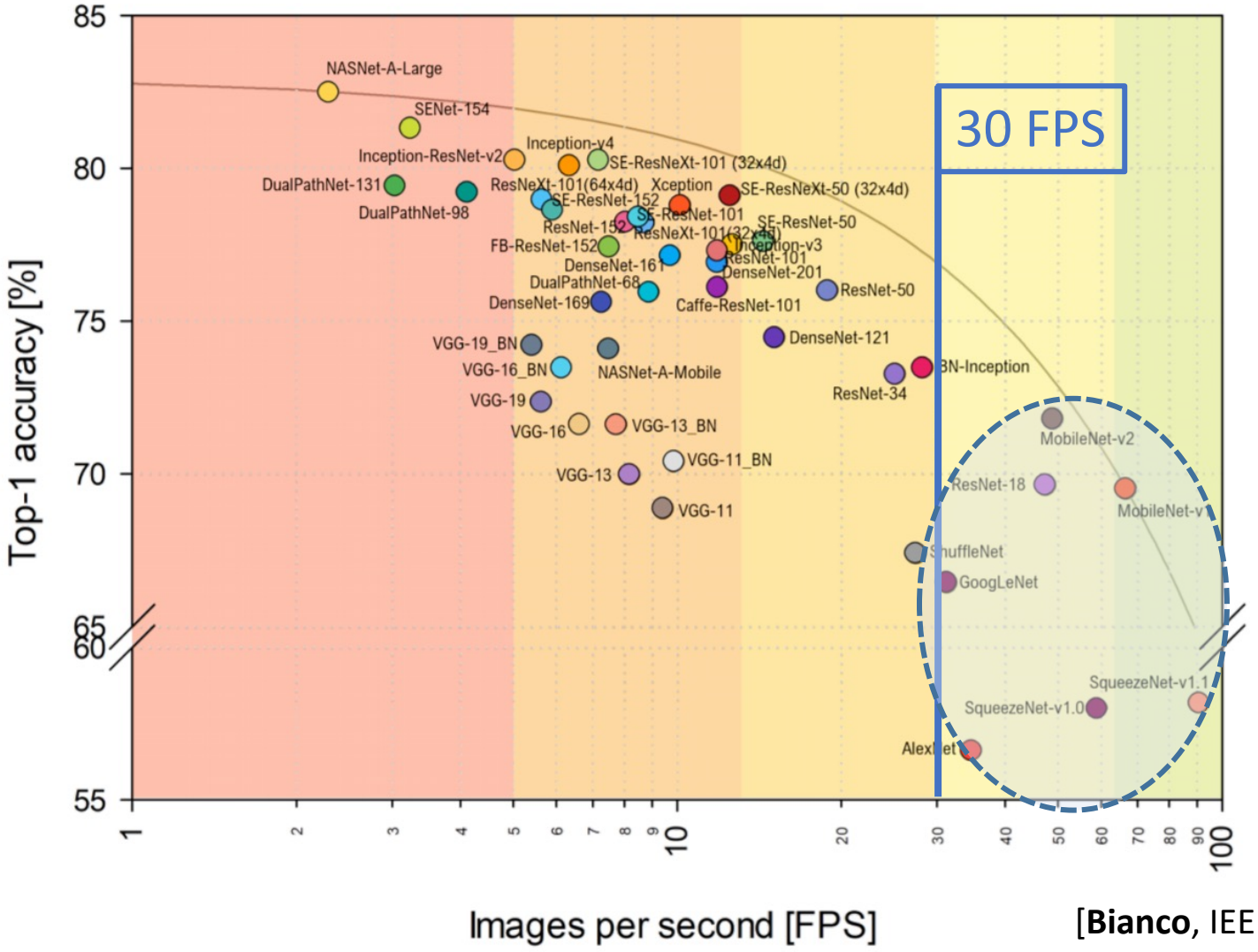
Reference	[5]	[6]	[10]	DNNBuilder
Categories	Cloud-computing platforms			
FPGA chip	Arria10-1150	Arria10-1150	Stratix-V GXA7 + CPU	KU115
Frequency	303 MHz	385 MHz	200 MHz & 2~3 GHz(CPU)	235 MHz
Network	Alexnet	VGG	Alexnet	VGG
Precision	Float16	Fix16	Fix16 in FPGA	Fix16 (Fix8)
DSPs (used/total)	2952/3036	2756/3036	512/512 in FPGA	4318/5520
DSP Efficiency	77.3%	84.3%	-	99.1%
Performance (GOPS)	1382	1790	781	2011 (4022)
Power Efficiency (GOPS/W)	30.7	47.8	-	90.2 (180.4)

KU115

- **LUT: 663,360**
- **FF: 1,326,720**
- **BRAM: 2160**
- **DSP: 5520**

Peaking at 4022 GOPS

AI Challenge #3: Real-time Requirement



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 …

Image processed per second using NVIDIA TX1 (batch size =1)

[**Bianco**, IEEE Access 2018]

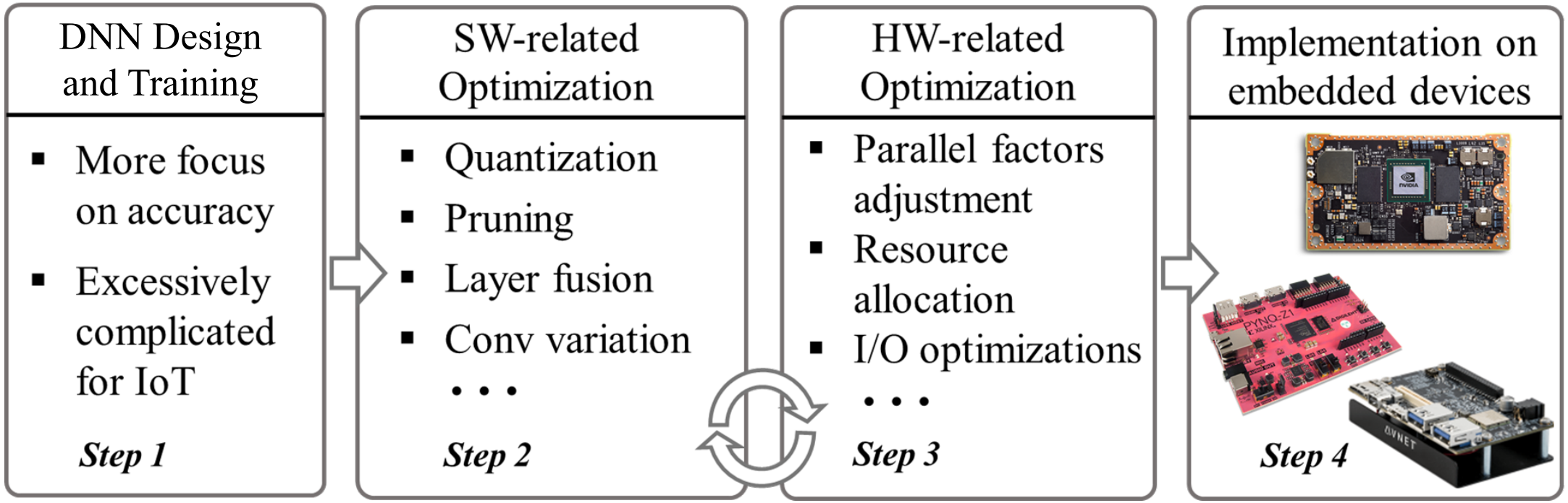
What Can Be an Effective Solution?

- Existing solutions
  - **Top-down** (independent) AI algorithm and accelerator design
  - Severe drawbacks for edge AI

- A new design methodology for Edge AI
  - **Algorithm and hardware (accelerator) co-design**

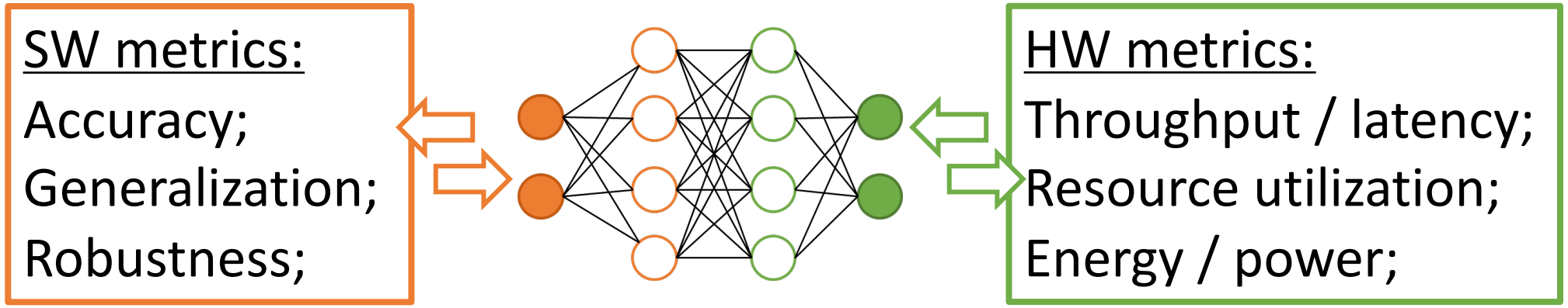
Top-down (independent) DNN Design and Deployment

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



Drawbacks of Top-down DNN Design and Deployment

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

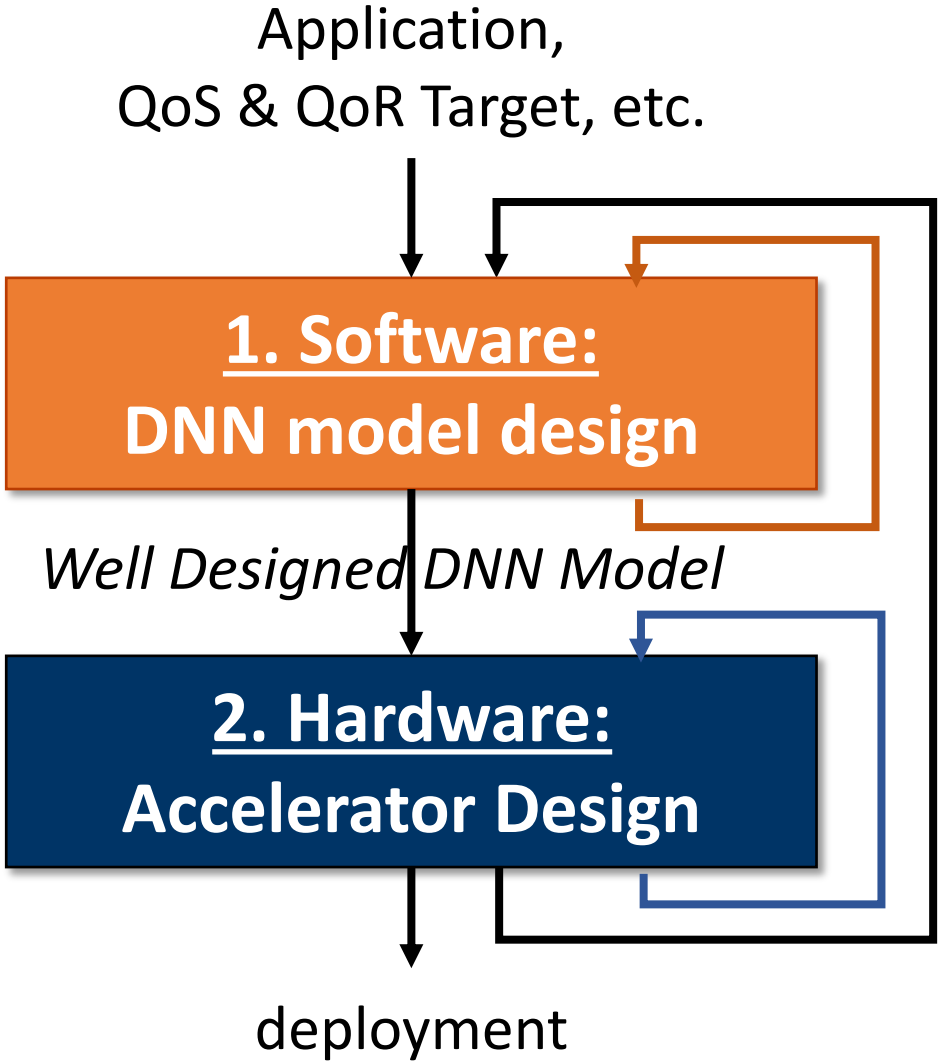


2) Difficult to select appropriate reference DNNs at the beginning

- Choose by experience
- Performance on published datasets

Drawbacks of Top-down DNN Design and Deployment

3) Long iterative procedure, tedious engineering efforts, especially for:
   - Resource limited edge devices
   - Strict performance requirement
     - E.g., faster than 35 FPS for real-time video processing



Sub-optimal (hardware-unfriendly) DNN models and accelerators on edge devices

DNN models and accelerators co-design!

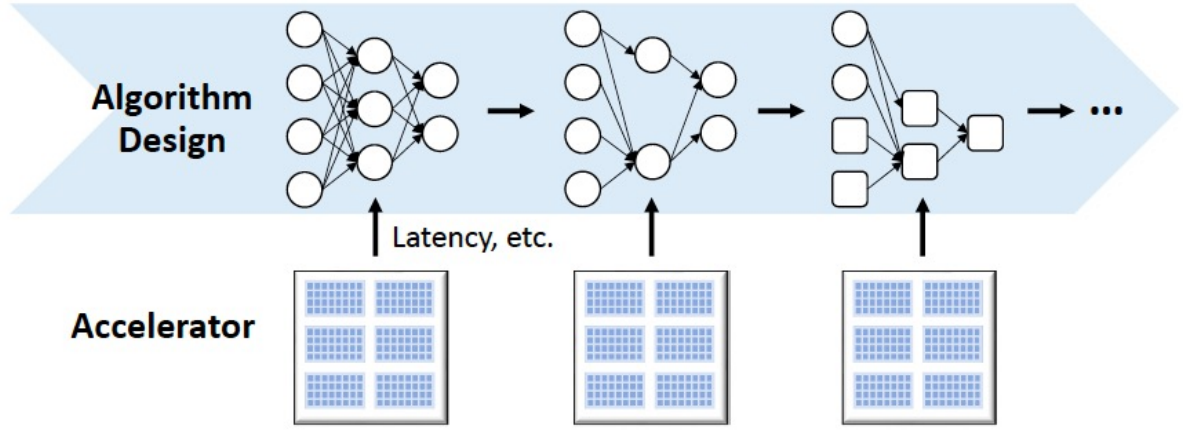
Simultaneous Algorithm / Accelerator Co-design Methodology

Conventional hardware-aware NAS

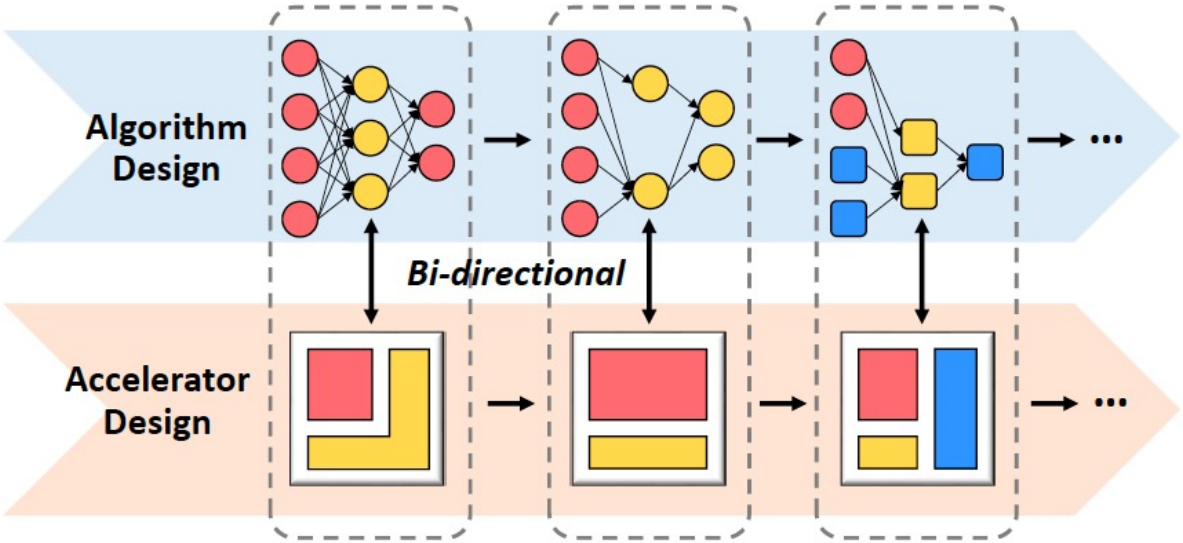
- The hardware accelerator design space is **fixed**

Simultaneous algorithm/accelerator co-design

- Both the algorithm and accelerator design spaces are **parameterized** and co-searched simultaneously.



(a) Hardware-aware NAS



(b) Simultaneous algorithm/accelerator co-design

Highlight of Our DNN and Accelerator Co-design Work

- Deep Neural Network Model and FPGA Accelerator Co-Design: Opportunities and Challenges,

  Cong Hao and Deming Chen. IEEE **ICSICT**, 2018. **← The very first work that discussed co-design**

- FPGA/DNN Co-Design: An Efficient Design Methodology for IoT Intelligence on the Edge,

  Cong Hao, et al., ACM/IEEE **DAC**, 2019

- NAIS: Neural Architecture and Implementation Search and its Applications in Autonomous Driving,

  Cong Hao, et al., ACM/IEEE **ICCAD**, 2019

- SkyNet: a hardware-efficient method for object detection and tracking on embedded systems,

  Xiaofan Zhang, et al., Machine Learning and Systems (**MLSys**), 2020

- EDD: Efficient Differentiable DNN Architecture and Implementation Co-search for Embedded AI Solutions

  Yuhong Li, et al., ACM/IEEE **DAC**, 2020

Highlight of Our DNN and Accelerator Co-design Work

- Deep Neural Network Model and FPGA Accelerator Co-Design: Opportunities and Challenges,

  Cong Hao and Deming Chen. IEEE **ICSICT**, 2018. **← The very first work that discussed co-design**

- FPGA/DNN Co-Design: An Efficient Design Methodology for IoT Intelligence on the Edge,

  Cong Hao, et al., ACM/IEEE **DAC**, 2019

- NAIS: Neural Architecture and Implementation Search and its Applications in Autonomous Driving,

  Cong Hao, et al., ACM/IEEE **ICCAD**, 2019

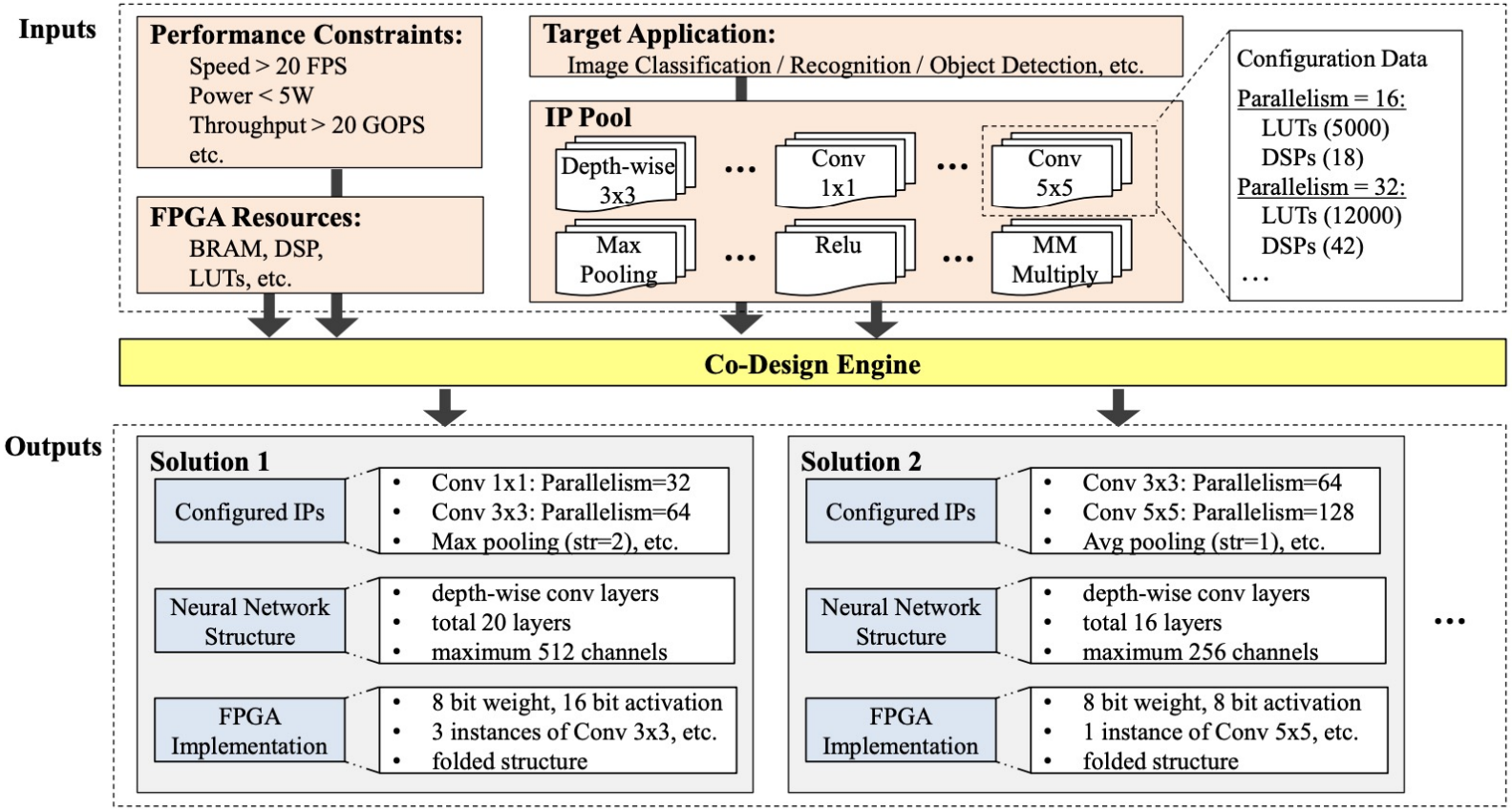
- SkyNet: a hardware-efficient method for object detection and tracking on embedded systems,

  Xiaofan Zhang, et al., Machine Learning and Systems (**MLSys**), 2020

- EDD: Efficient Differentiable DNN Architecture and Implementation Co-search for Embedded AI Solutions

  Yuhong Li, et al., ACM/IEEE **DAC**, 2020

Our Co-design Method Proposed in ICSICT 2018

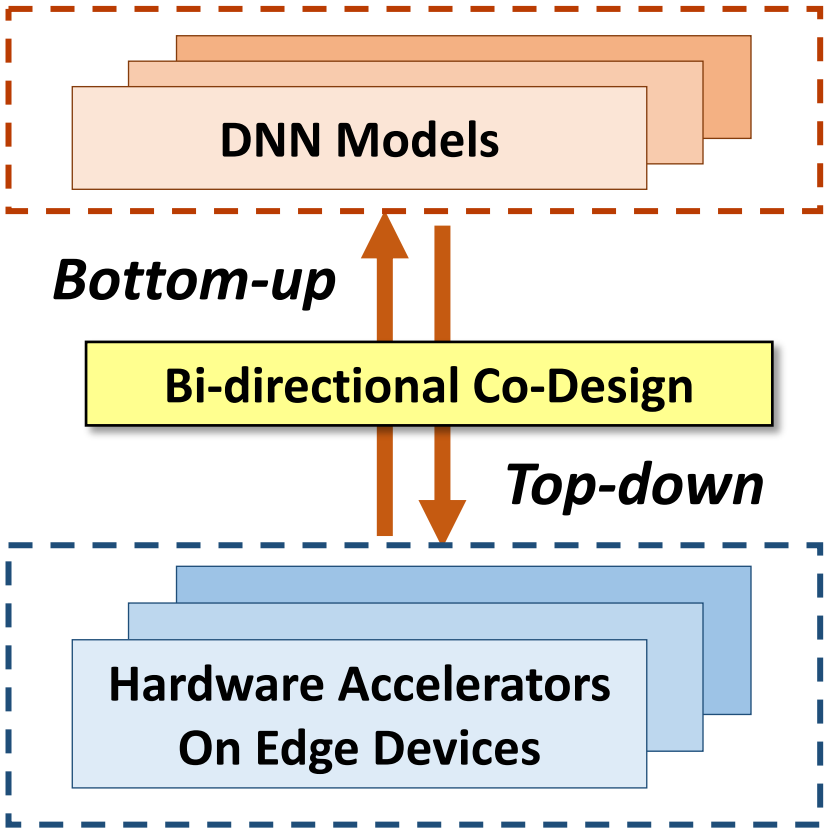


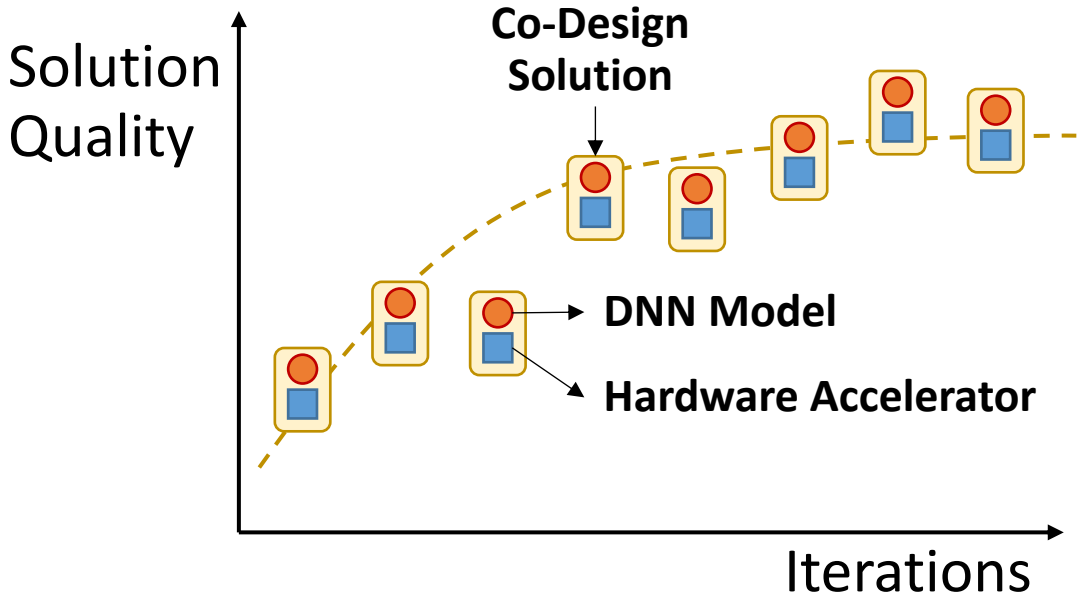
- The objective of the proposed co-design is: "*to automatically generate both DNN models and their corresponding implementations as pairs*" and "*shorten the DNN development time by avoiding tedious iterations between DNN model and its accelerator designs*"

[ICSICT 2018]

Co-design Idea Materialized in DAC 2019

- *Bottom-up*: architecture template guided DNN model search
- *Top-down*: evaluate DNN through architecture template mapping
- Bi-directional co-search for both DNN and FPGA accelerator



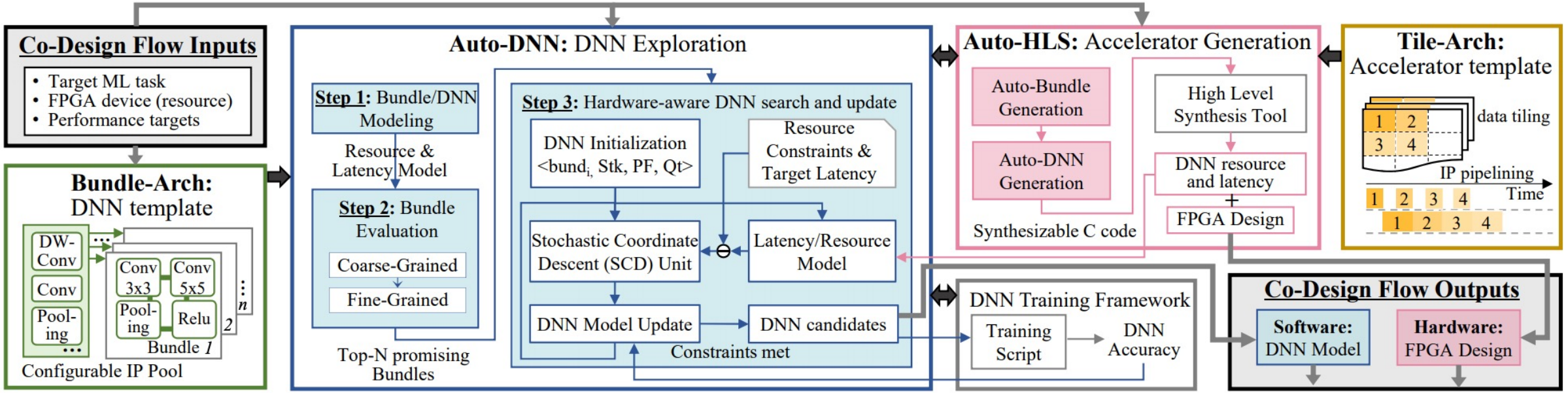


[DAC 2019]

The Co-design Framework in DAC 2019

Challenges

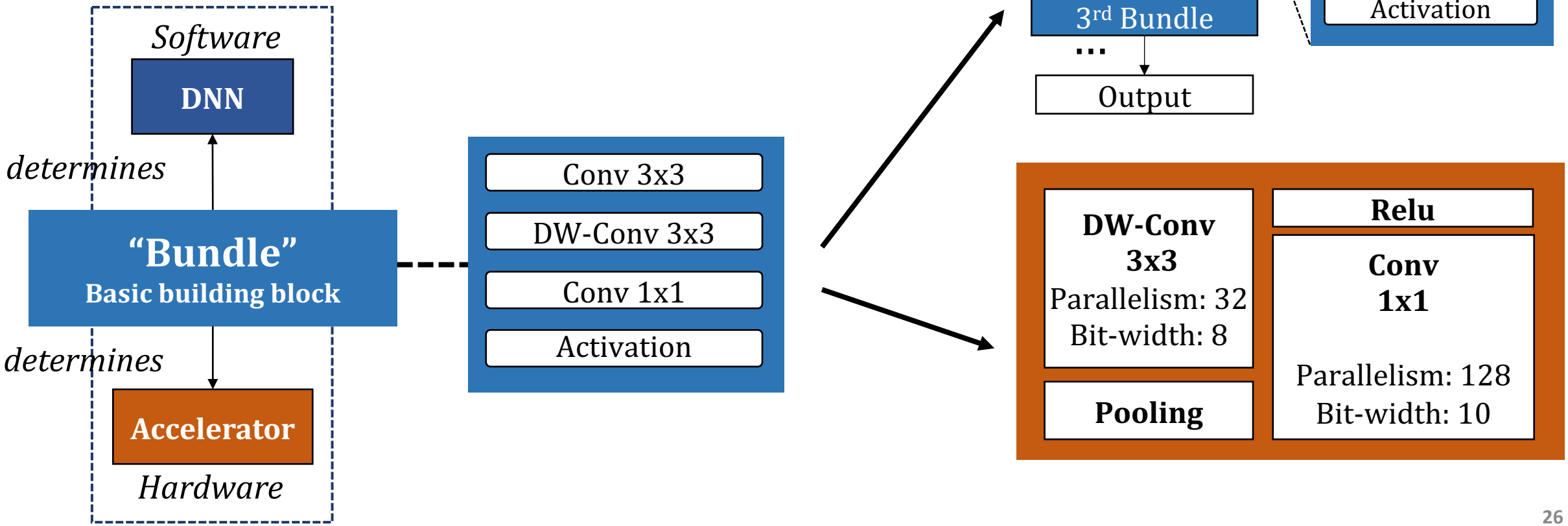
- High quality DNN design and its accelerator design are both challenging
- Co-design significantly expands design space
  - Difficult to converge
  - Requires novel search methodology



[DAC 2019]

Basic Building Blocks : Bundles

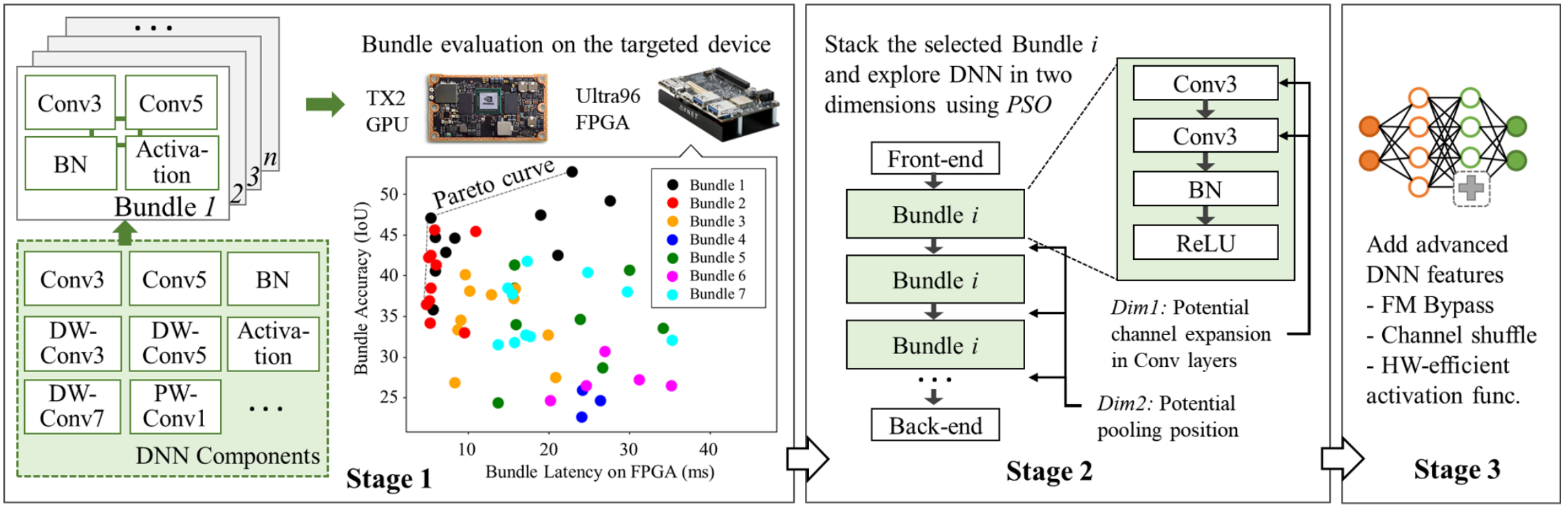
- **Bundle:** a set of sequential DNN layers as basic DNN building block
- Also represents a combination of IP instances used for DNN computation



Output of the Co-design: the SkyNet!

- **Three Stages:**

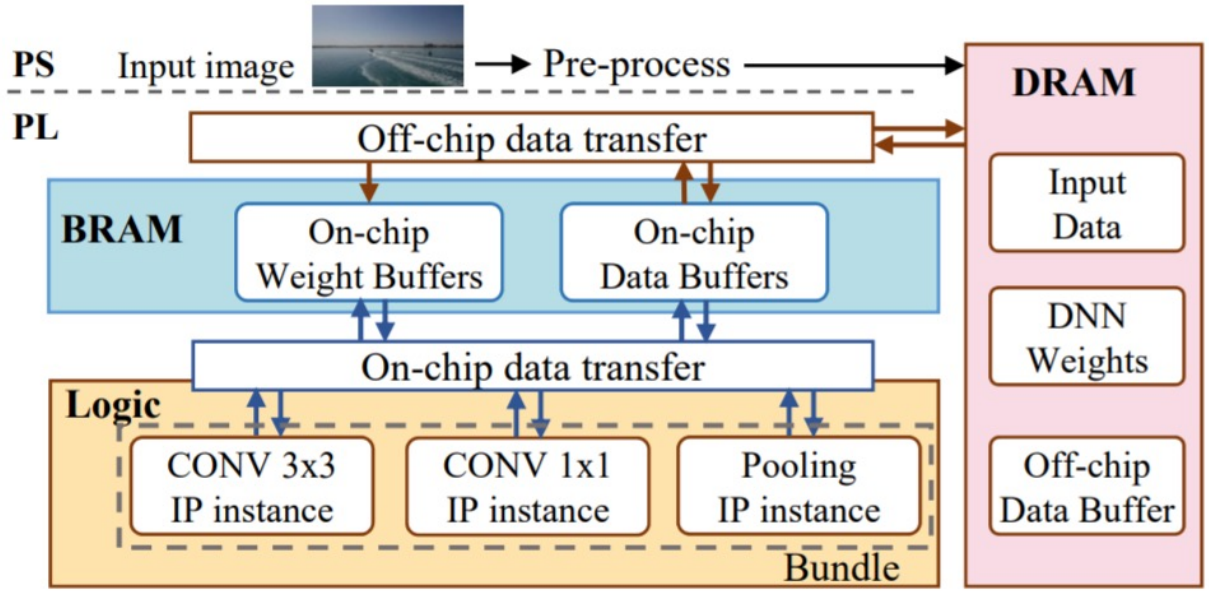
  ① Select **Basic Building Blocks** → ② Explore DNN and accelerator architectures based on **templates** → ③ Add features, fine-tuning and hardware deployment



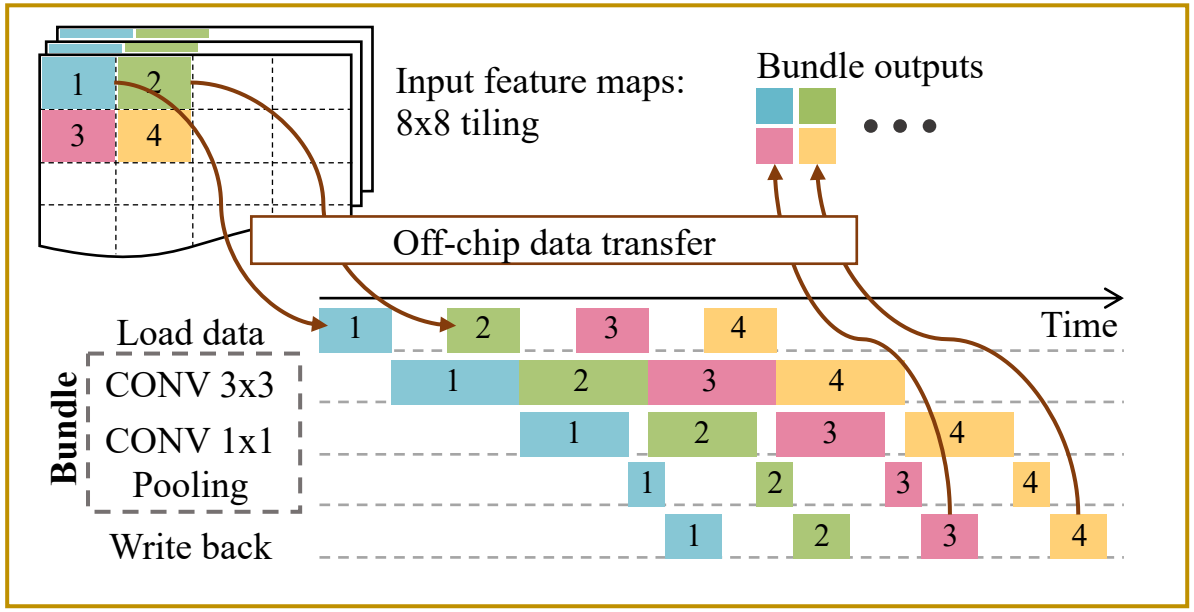
[DAC SDC 2019] & [MLSys 2020]

The HW Deployment

- Hardware (FPGA) accelerator using our proposed template



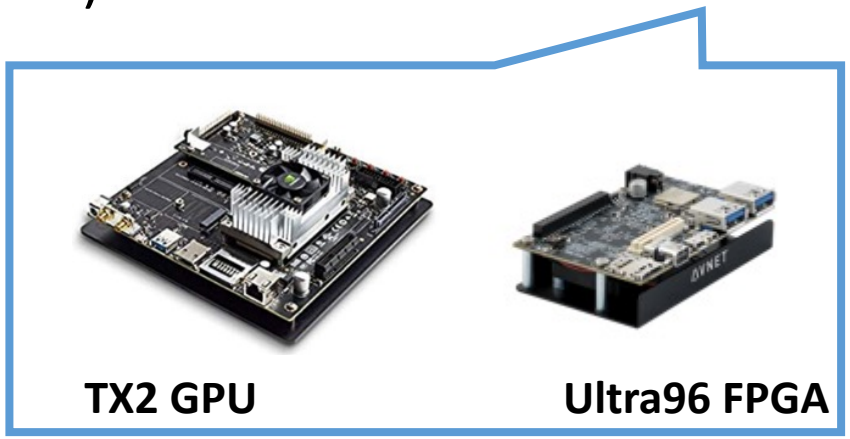
Overall system



Accelerator

Demo #1: Object Detection 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***

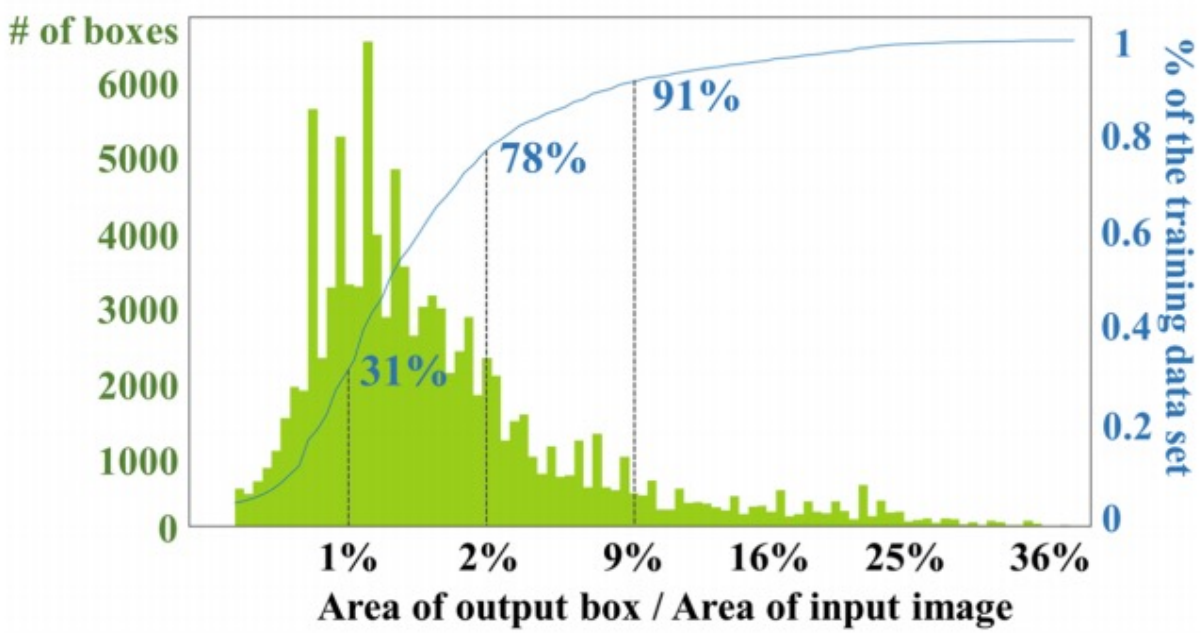
$$TS_i = R_{IoU_i} \times (1 + ES_i)$$

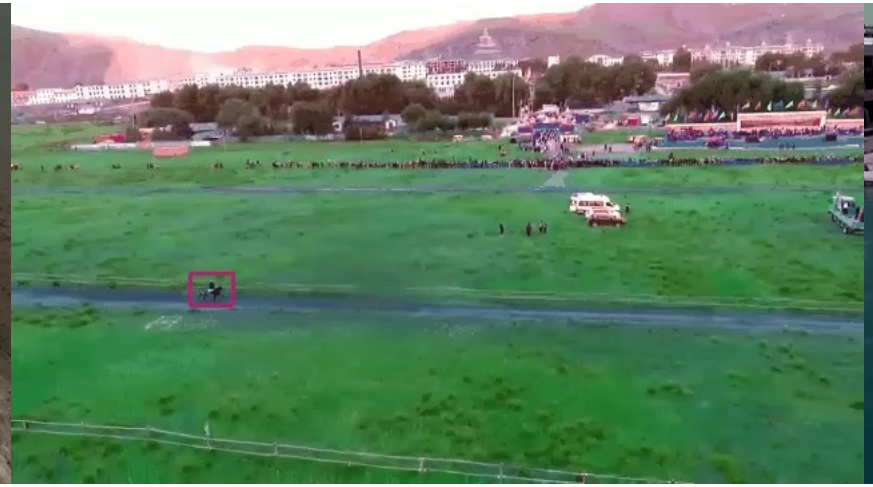
Demo #1: DAC-SDC Dataset

- The distribution of target relative size compared to input image

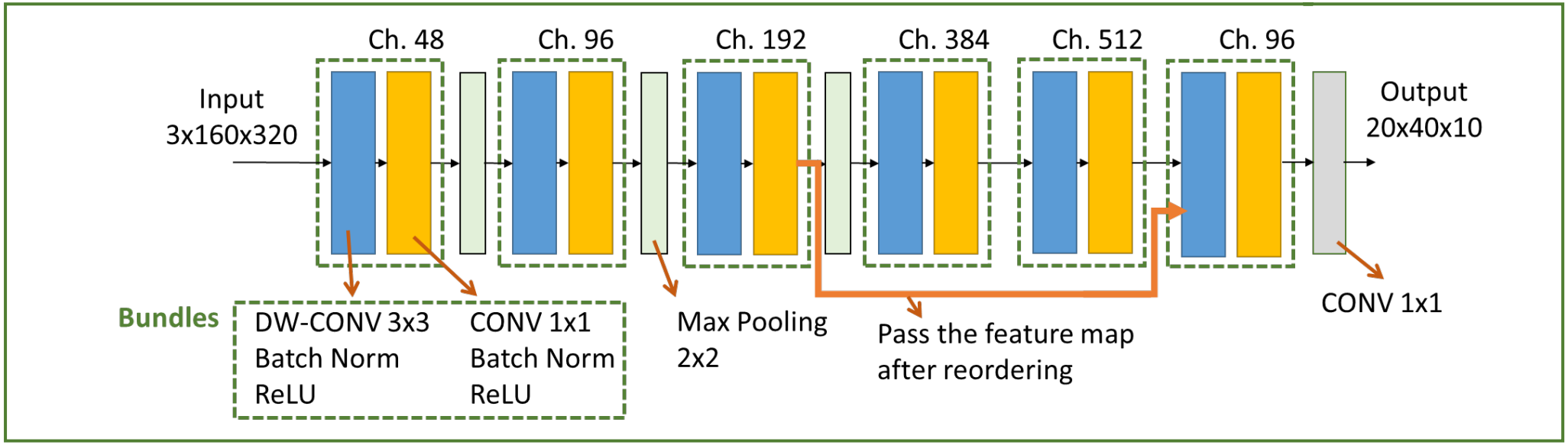
31% targets < 1% of the input size

91% targets < 9% of the input size

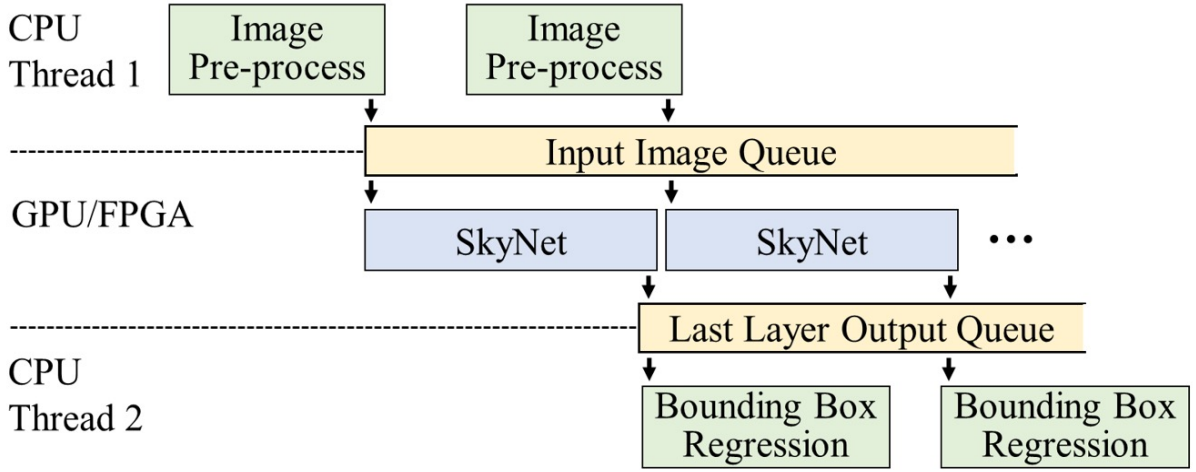




Demo #1: the SkyNet DNN Architecture

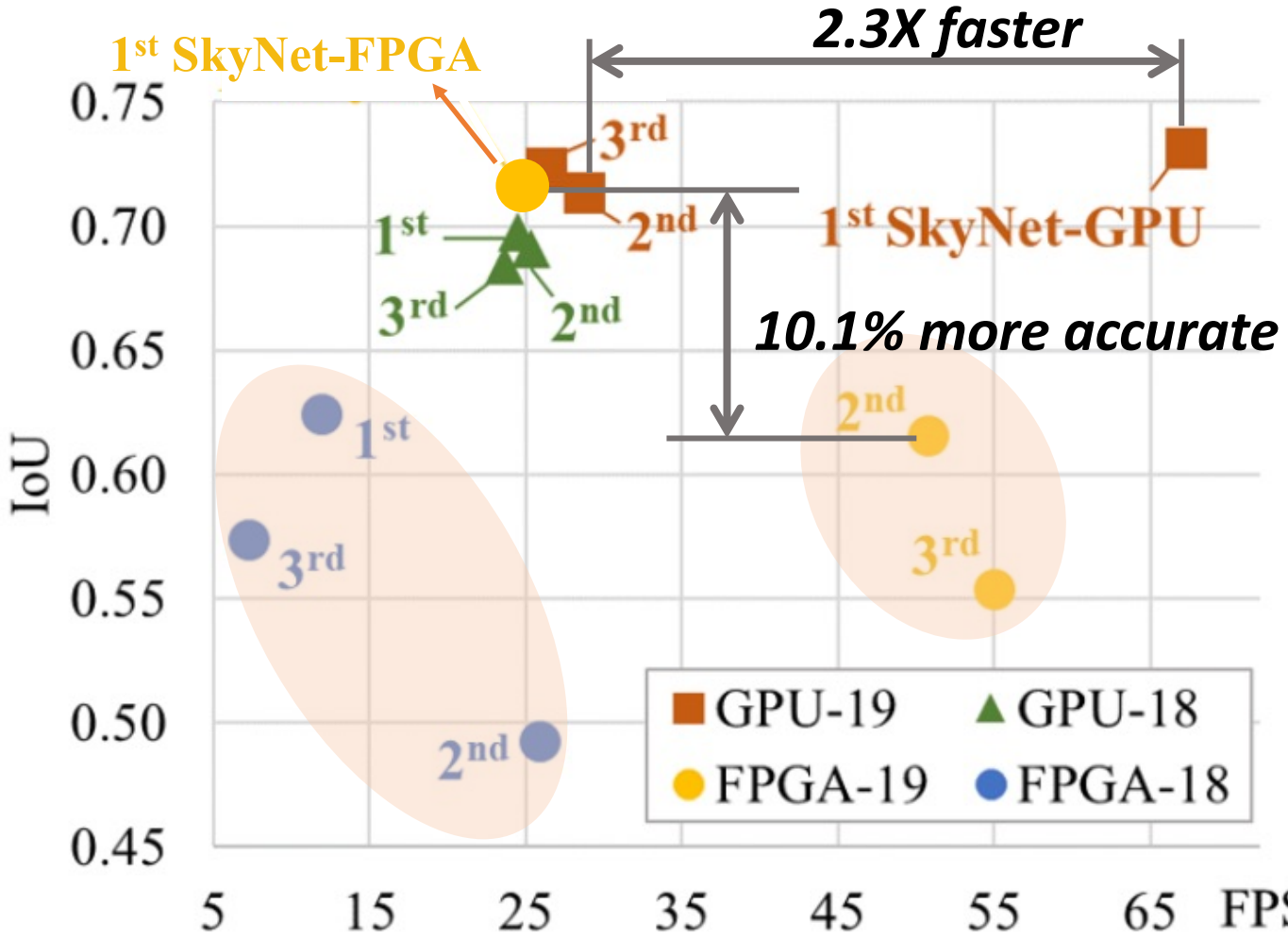


- 13 CONV with 0.4 million parameters
- For Embedded FPGA: Quantization, Batch, Tiling, Task partitioning
- For Embedded GPU: Task partitioning



Demo #1: SkyNet Results for DAC-SDC 2019

- Evaluated by 50k images in the official test set

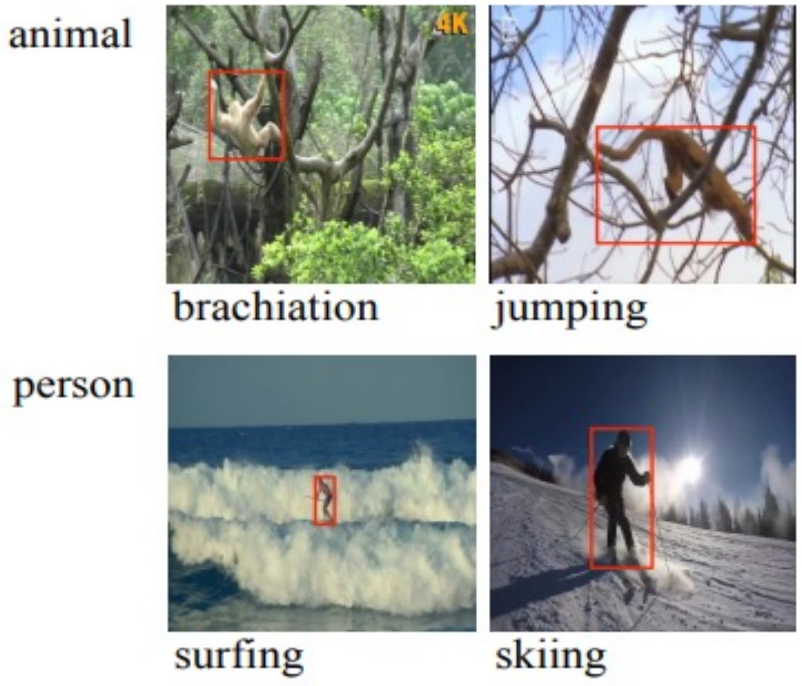


Designs using TX2 GPU

Designs using Ultra96 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

Conclusions and Future Work

- We presented ***DNNBuilder***
  - IP reuse and scheduling for design automation and on-chip memory usage reduction
  - Customized pipeline solution to improve throughput
- We presented ***SkyNet*** & ***a hardware-efficient DNN design method***
  - A bidirectional DNN/accelerator design flow for embedded systems
  - An effective way to capture realistic HW constraints
- **Co-design is an exciting research direction**
  - Co-design for distributed AI
  - Co-design for heterogeneous and large-scale AI
  - Co-design for emerging AI technologies

Thank you