Neural Network Acceleration on Mobile Devices

Dongyoung Kim
Machine learning team
Hyperconnect
Seoul, South Korea
dongyoung.kim@hpcnt.com

HYPERCONNECT

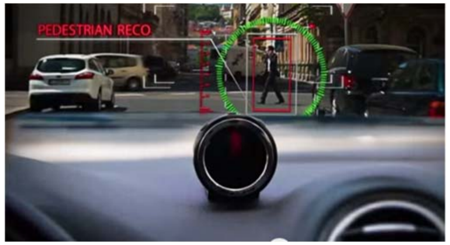
Agenda

- **Introduction**
- **Neural Processing Unit (NPU)**
  - NPU utilizing sparsity
    - Zero-aware neural network accelerator (ZeNA)
  - NPU utilizing reduced precision
    - Outlier quantization and Precision highway
- **Working as an AI System Architect in the Industry**
- **On-device Machine Learning**
  - Neural network acceleration on mobile CPU
  - Optimizing neural network for mobile devices
    - Temporal convolution for real-time keyword spotting on mobile devices

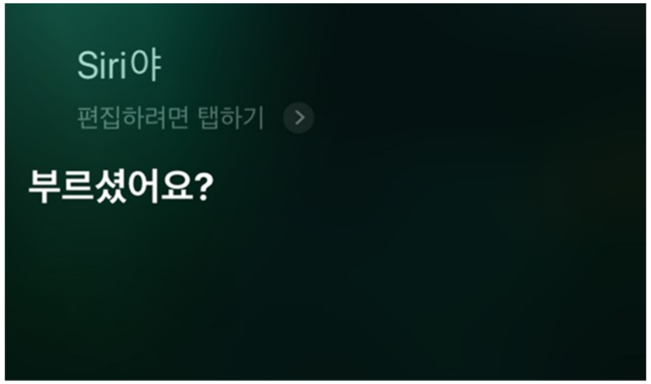
Introduction

HYPERCONNECT

Deep Neural Network (DNN)



Self driving car

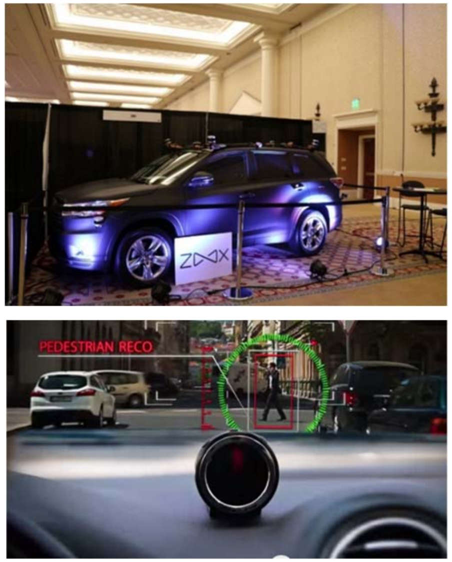


Voice recognition

Feed and ad

- Deep neural networks are ubiquitous in various applications.

DNN on Real-time Mobile Devices



Self driving car

Virtual Reality (VR)

Augmented Reality (AR)

- Especially, DNN shows reliable result on **real-time mobile devices**.

Challenges of DNN Applications on Mobile Devices

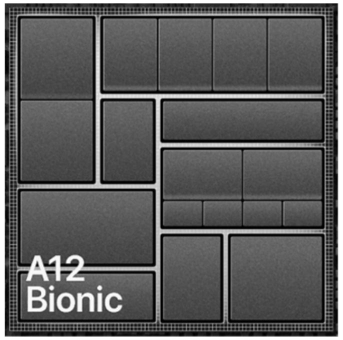
GPU server

Embedded systems

- DNN based applications perform well on large devices which have abundant resources but they are **unsuitable for mobile devices**.

Neural Processing Unit (NPU)

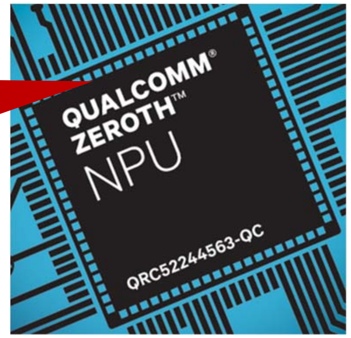
Neural Processing Unit (NPU)



A11 / A12 Bionic
(Apple)

Booming of NPU

Exynos 9820
(Samsung)



Snapdragon 855
(Qualcomm)

NPU Utilizing Sparsity

HYPERCONNECT

Sparsity

Layer	Zero Weight [%]	Zero Activation [%]
conv1	15.7	0
	62.1	50.9
	65.4	76.3
conv4	62.8	61.8
conv5	63.1	59.0

Pruning

ReLU

AlexNet

- Significant portion of **input values** in a convolutional layer is **zero**.
- Large number of **ineffectual computations can be skipped**.
- However, **it is difficult to utilize sparsity on CPU.**

Previous Works

Layer	Zero Weight [%]	Zero Activation [%
conv1	15.7	[illegible]
conv2	62.1	[illegible]
conv3	65.4	[illegible]
conv4	62.8	[illegible]
conv5	63.1	59.0

Energy ↓

Runtime ↓

Energy ↓

Runtime ↓

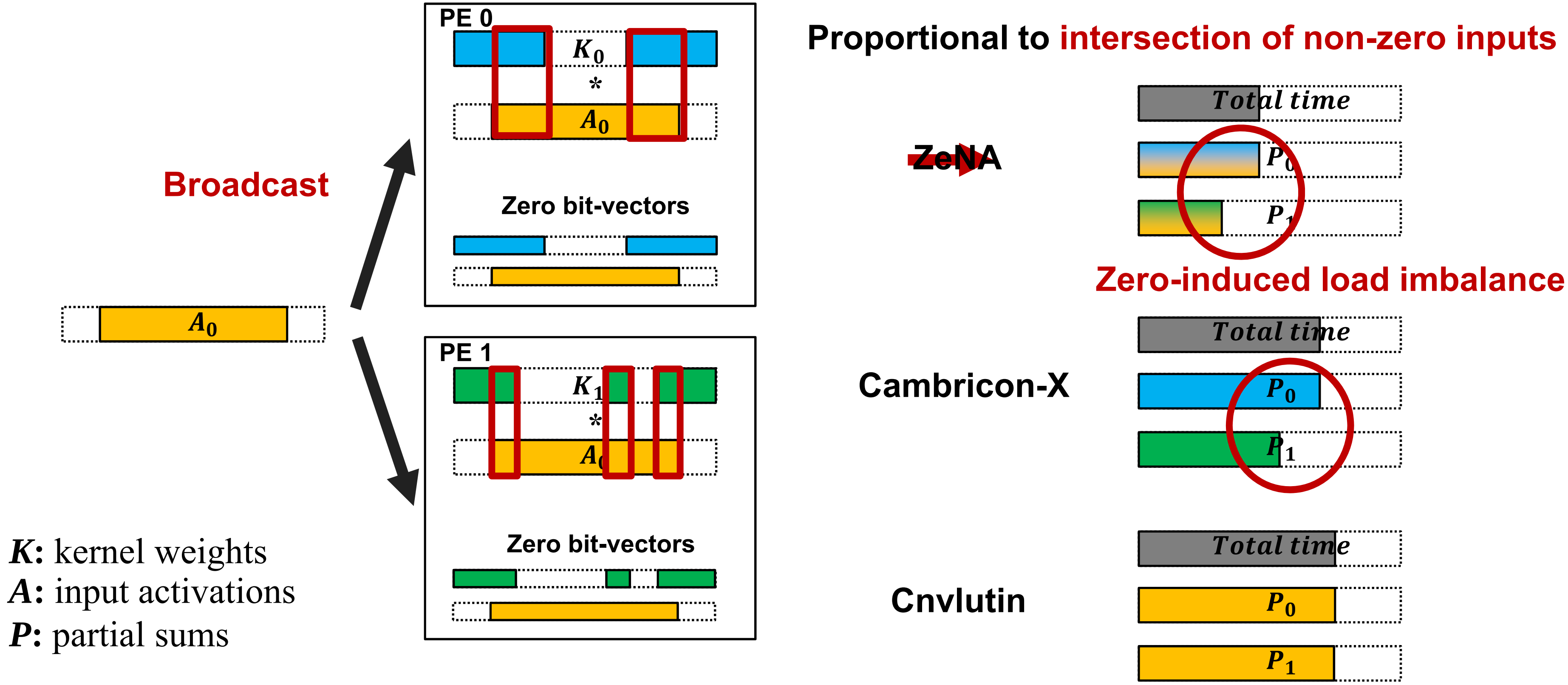
AlexNet

- **DaDianNao:** wide SIMD-like architecture.

Exploit zero values in both kernel weights and input activations

- **Cambricon-X:** utilizes zero weights for performance and energy.

Basic Idea of Zero-aware Neural Network Accelerator (ZeNA)



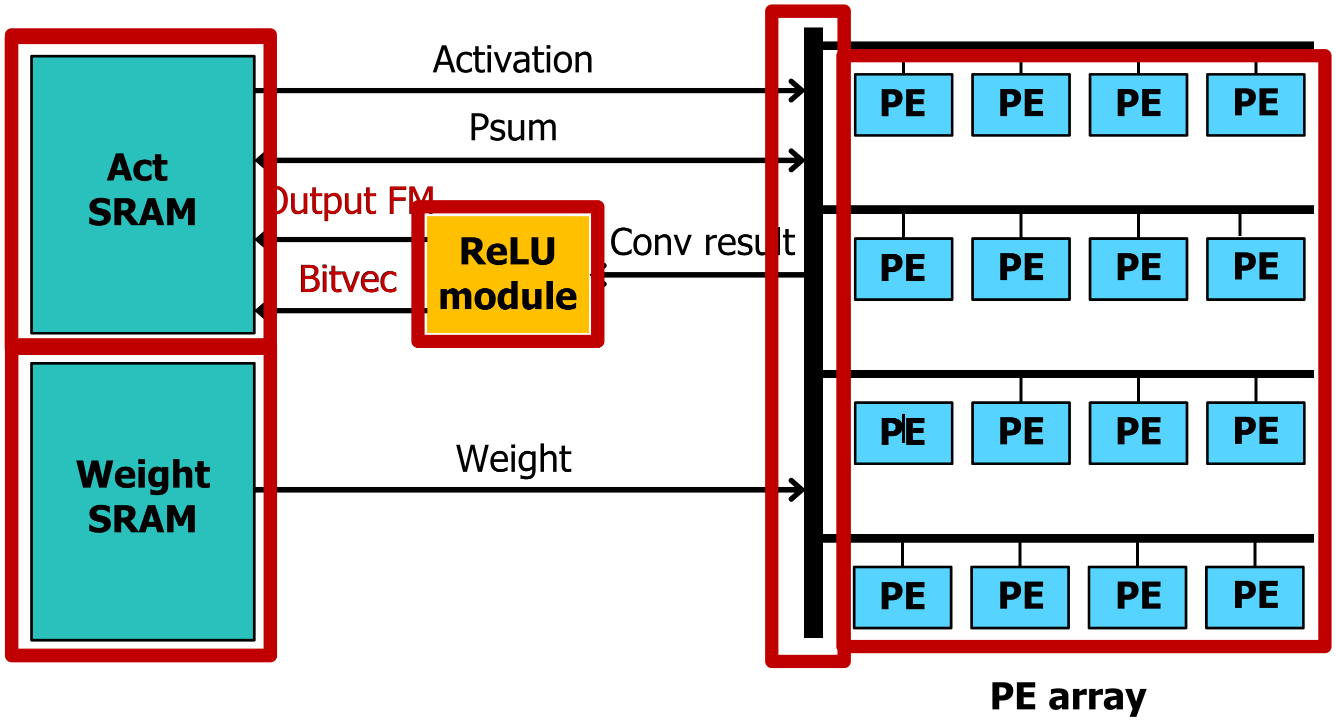
ZeNA Architecture Overview

Act SRAM stores
- Activations
- Output partial sums
- Output feature maps
- Non-zero bit-vectors

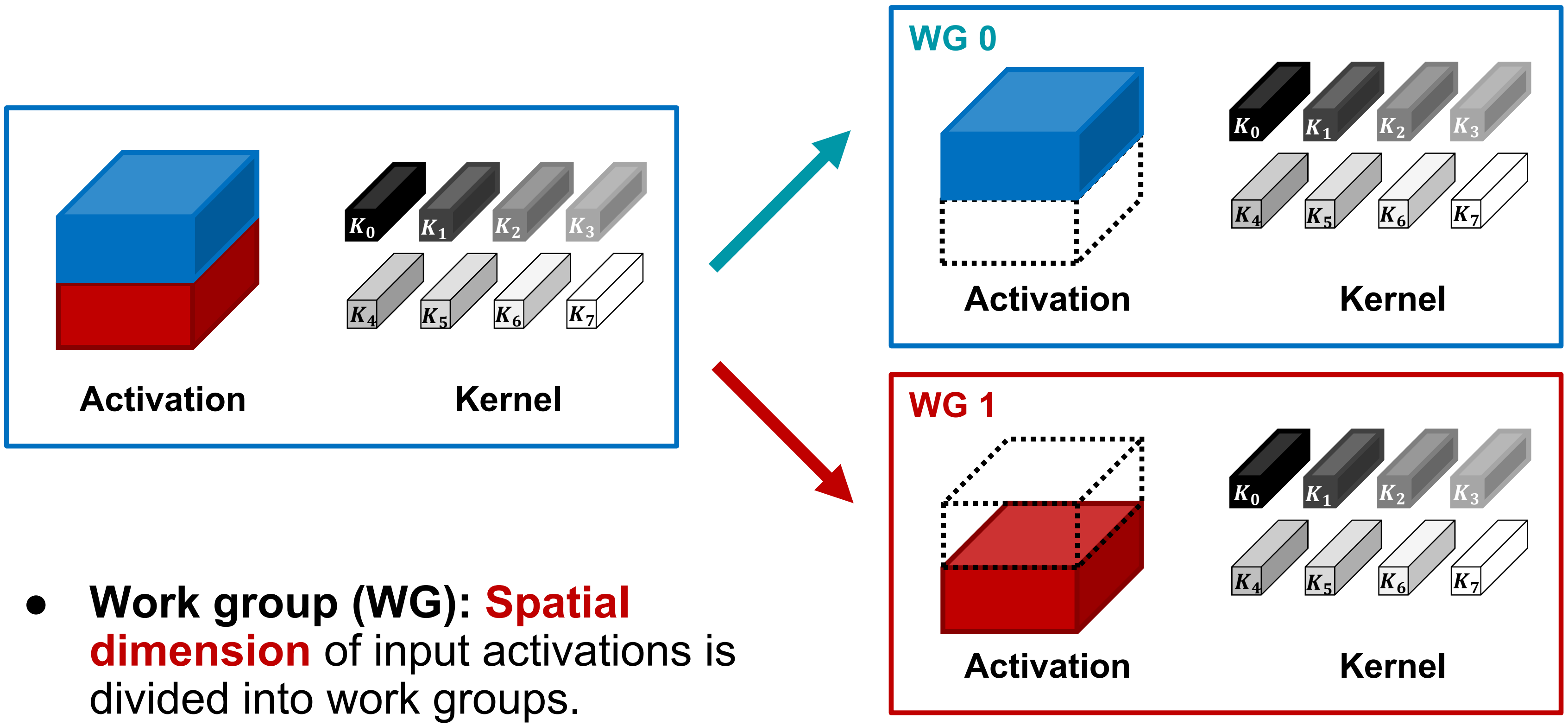
ReLU module generates
- Output feature maps
- Non-zero bit-vectors

Bus connects
- On-chip SRAM and PE array

Weight SRAM stores
- Kernel weights
- Non-zero bit-vectors

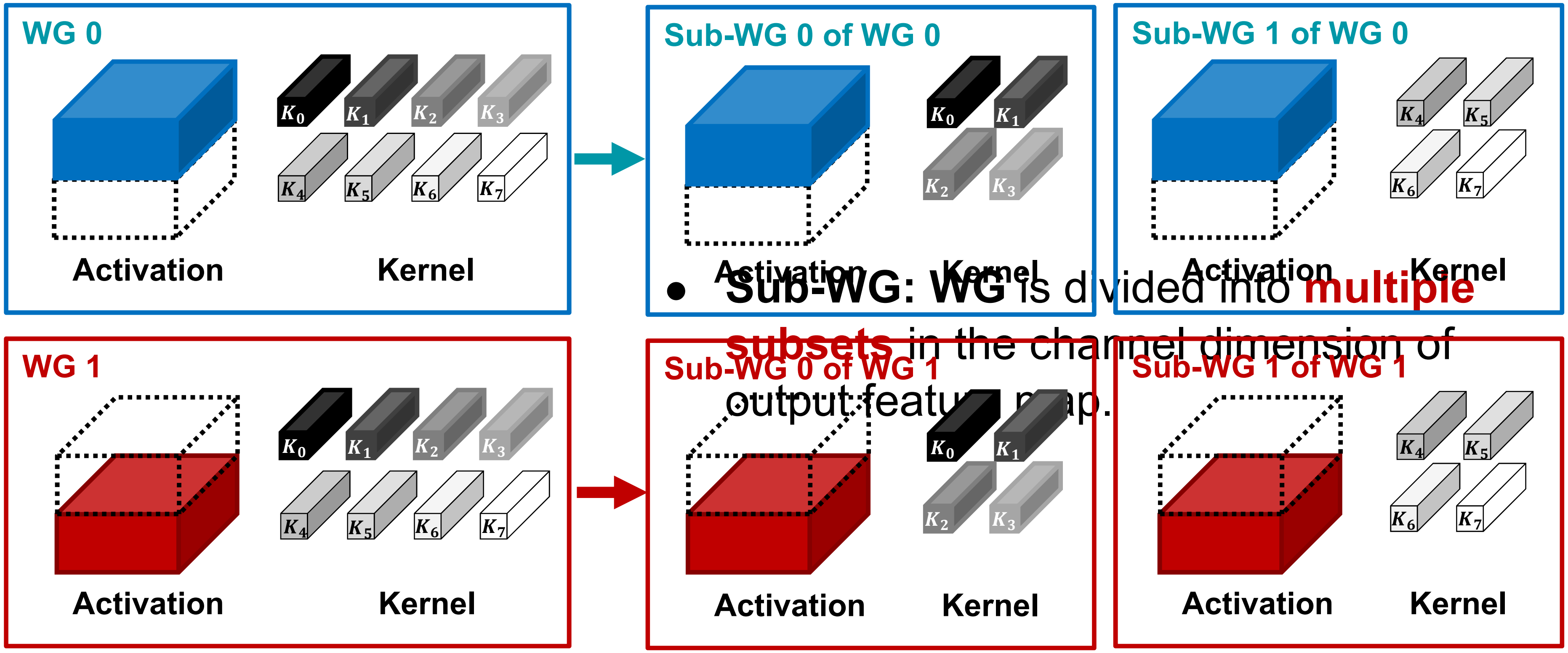


Work Group (WG)



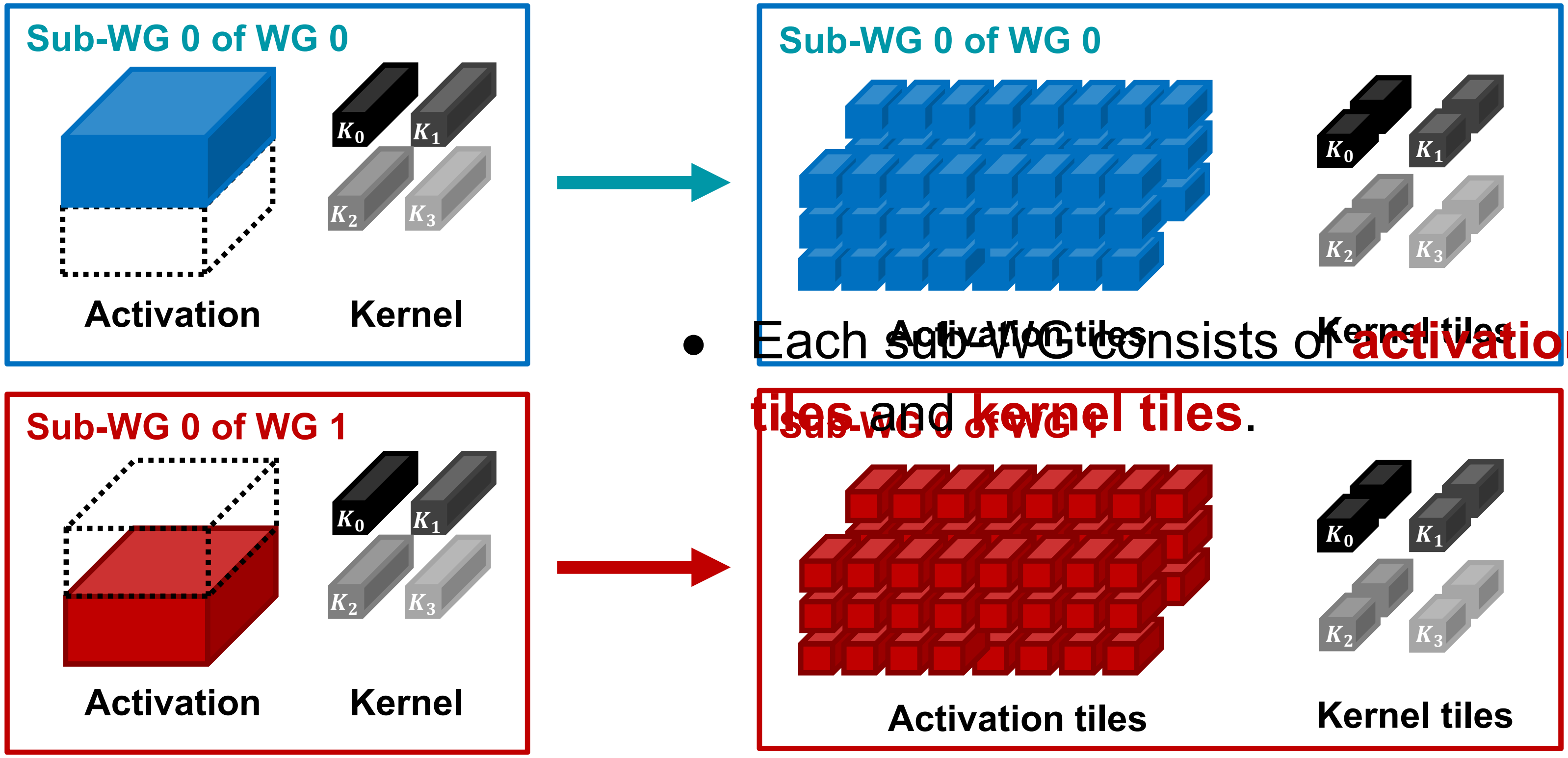
- **Work group (WG): Spatial dimension** of input activations is divided into work groups.

Work Group (WG)



- **Sub-WG: WG** is divided into **multiple subsets** in the channel dimension of output feature map.

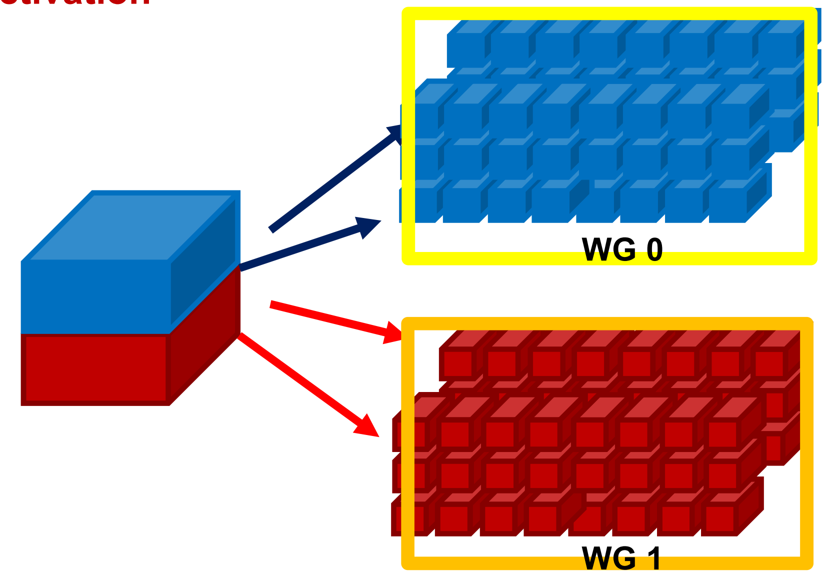
Work Group (WG)



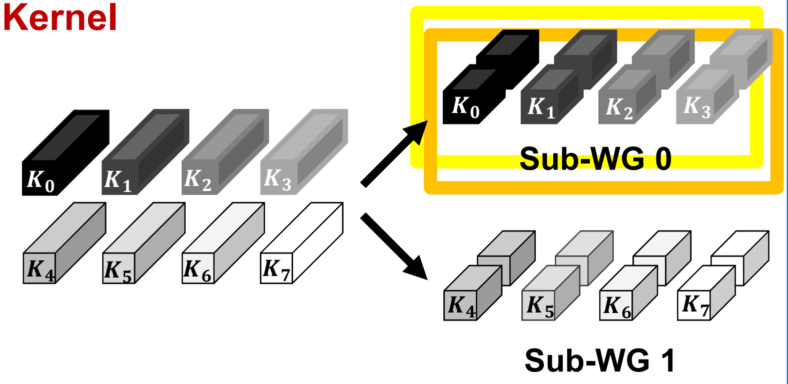
- Each sub-WG consists of **activation tiles** and **kernel tiles**.

Computation Procedure

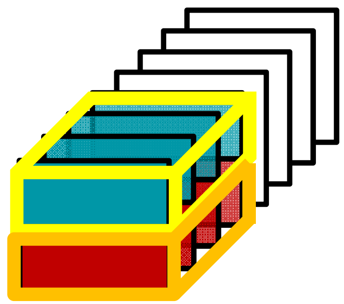
Activation



Kernel

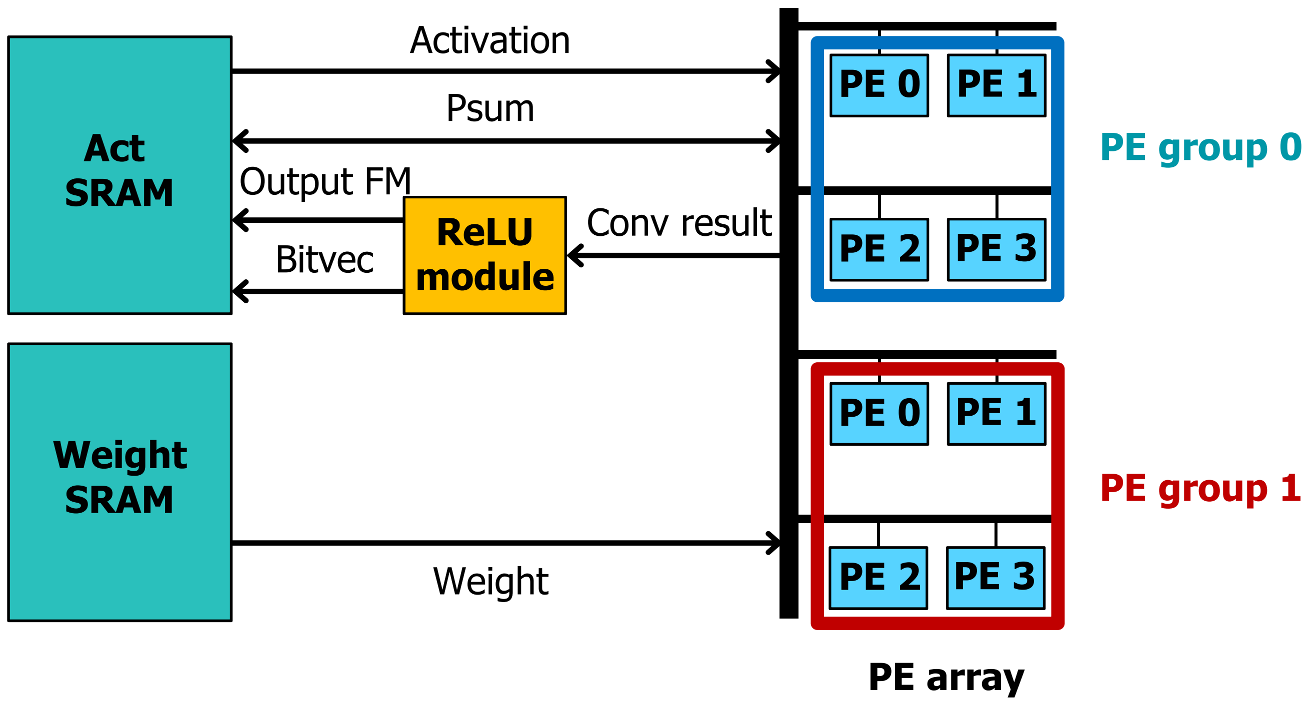


Output feature map

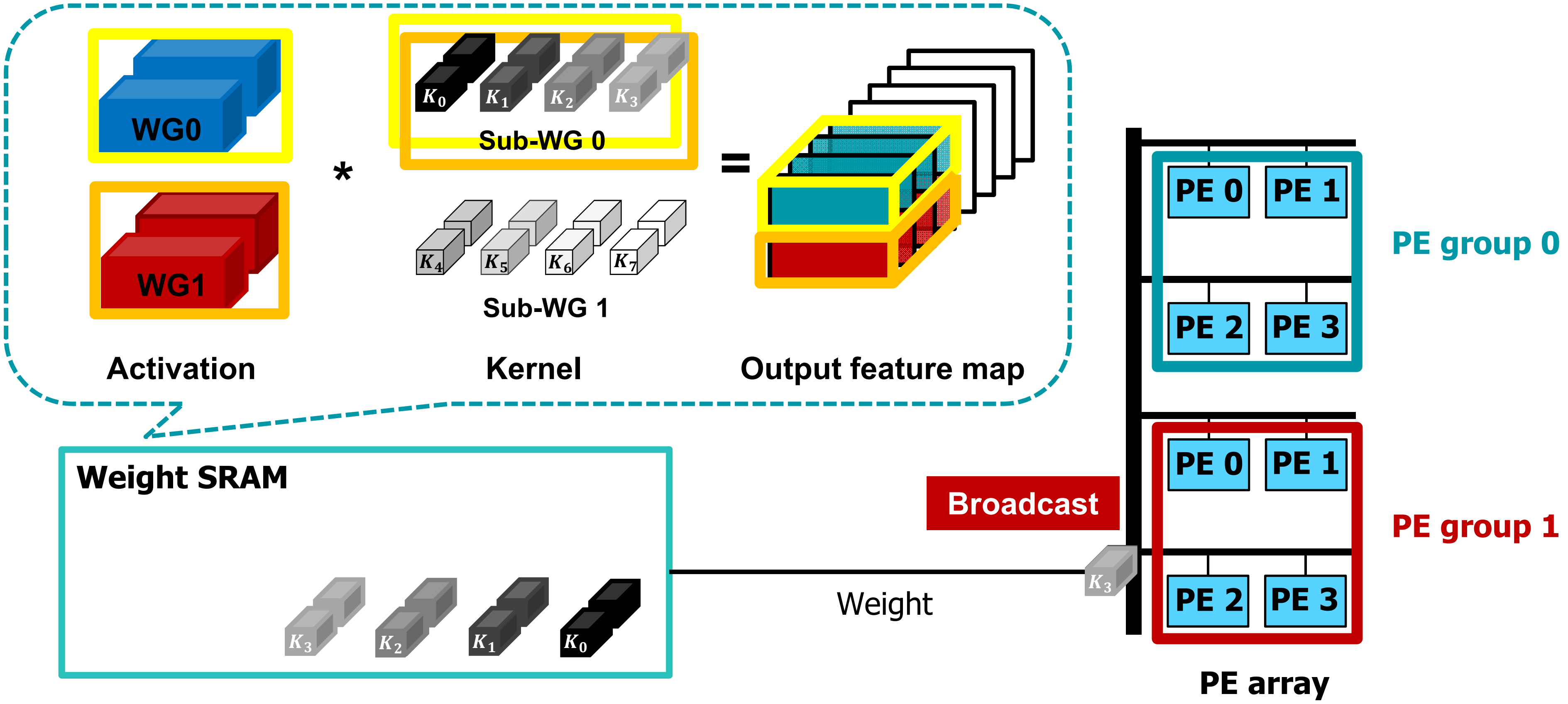


- Our accelerator performs convolution with activation and kernel tiles **iteratively** to compute output feature maps.

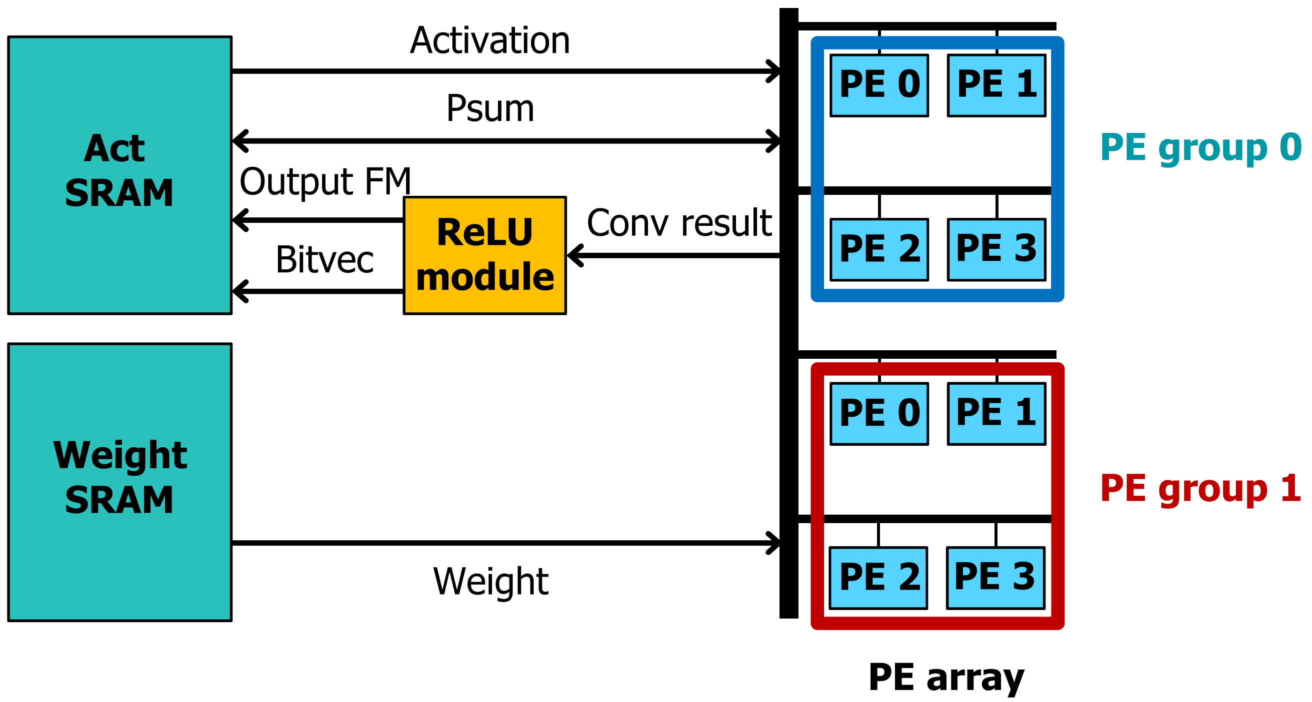
Data Flow and Computation: Kernel Broadcast



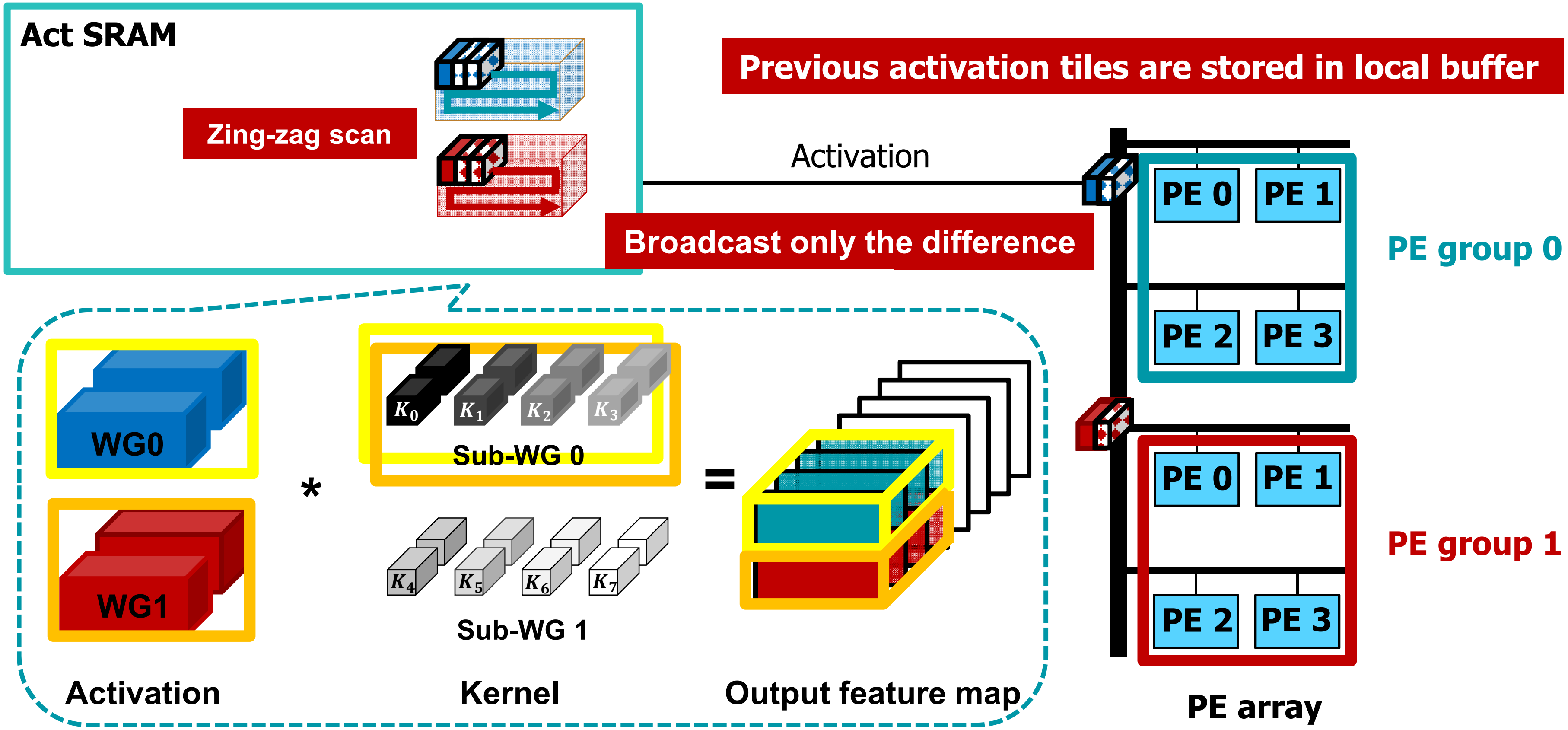
Data Flow and Computation: Kernel Broadcast



Data Flow and Computation: Activation Broadcast

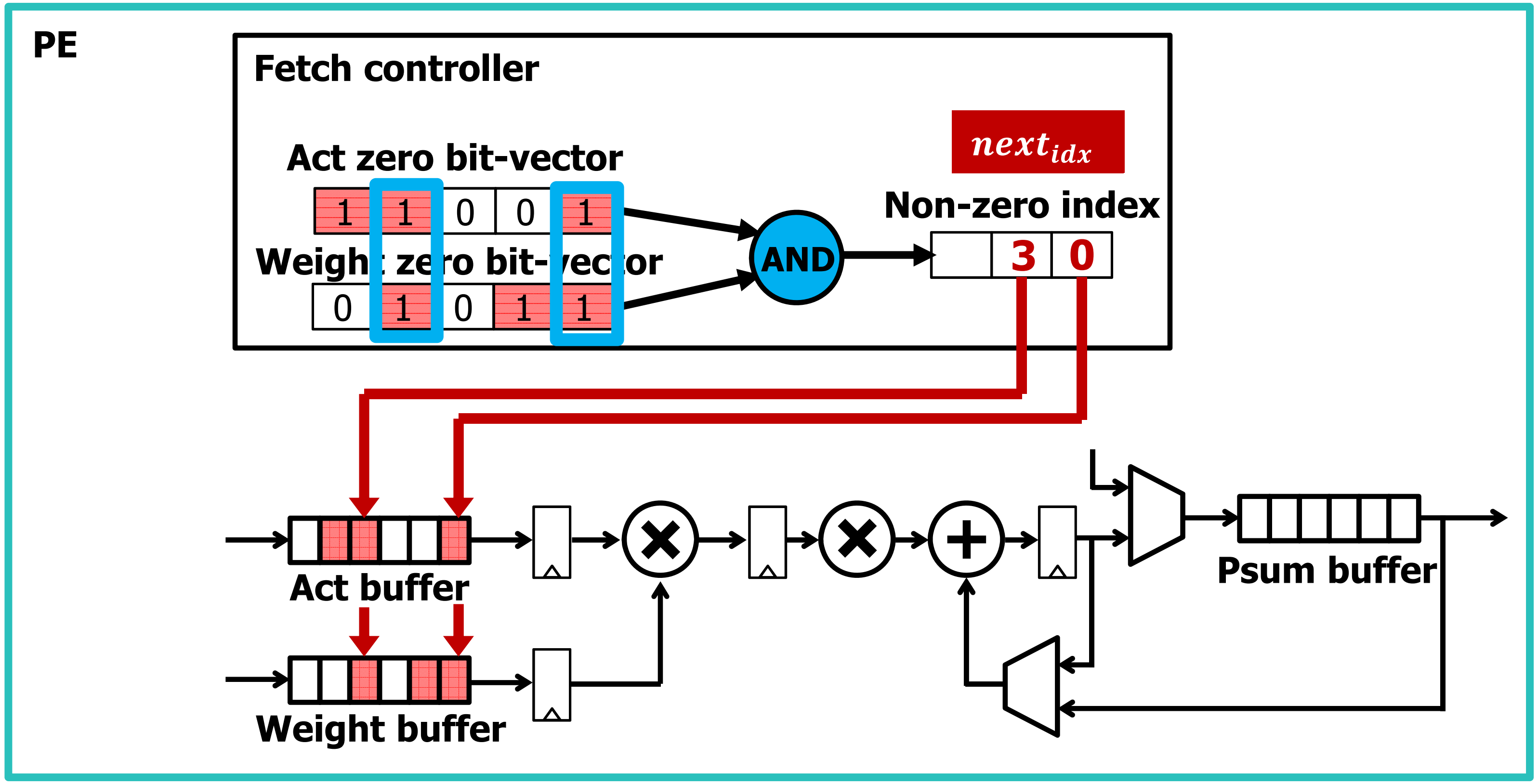


Data Flow and Computation: Activation Broadcast

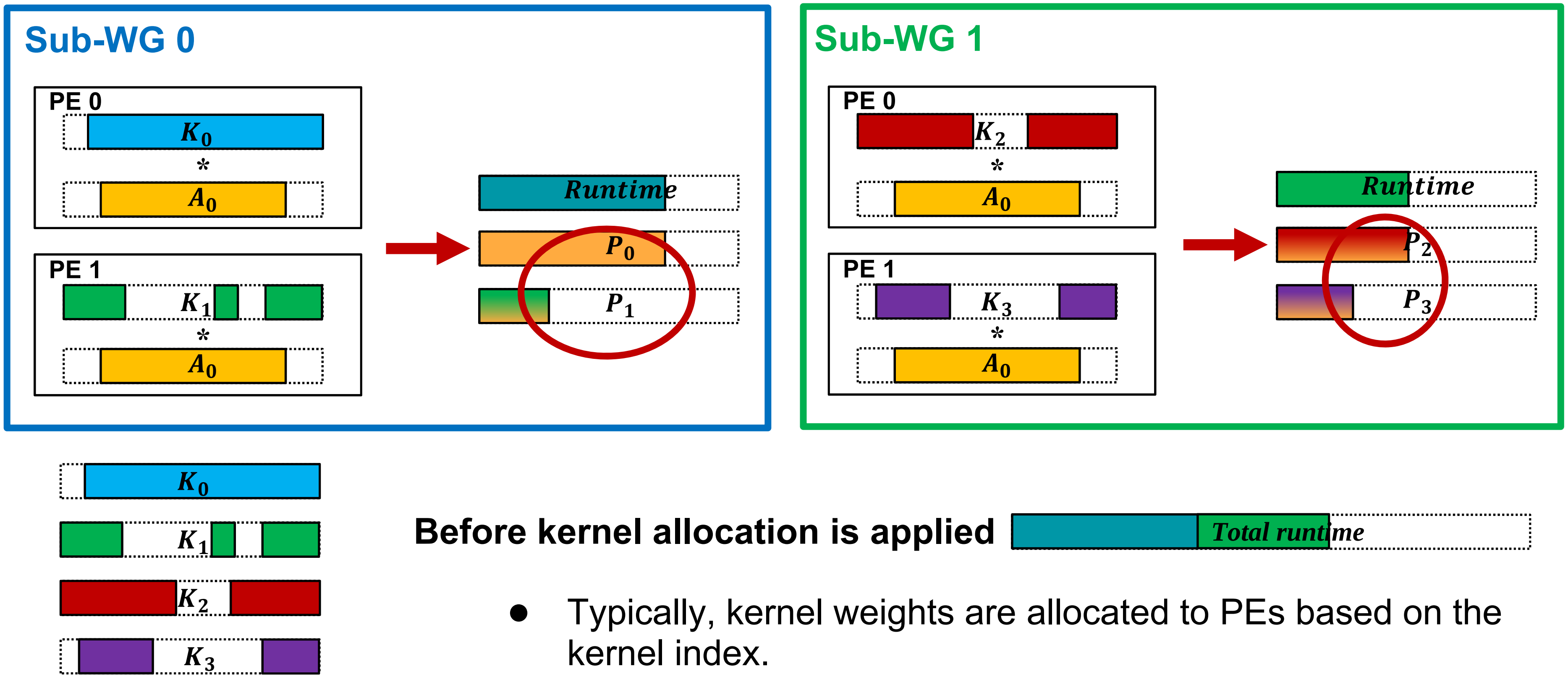


HYPERCONNECT

Zero-aware PE

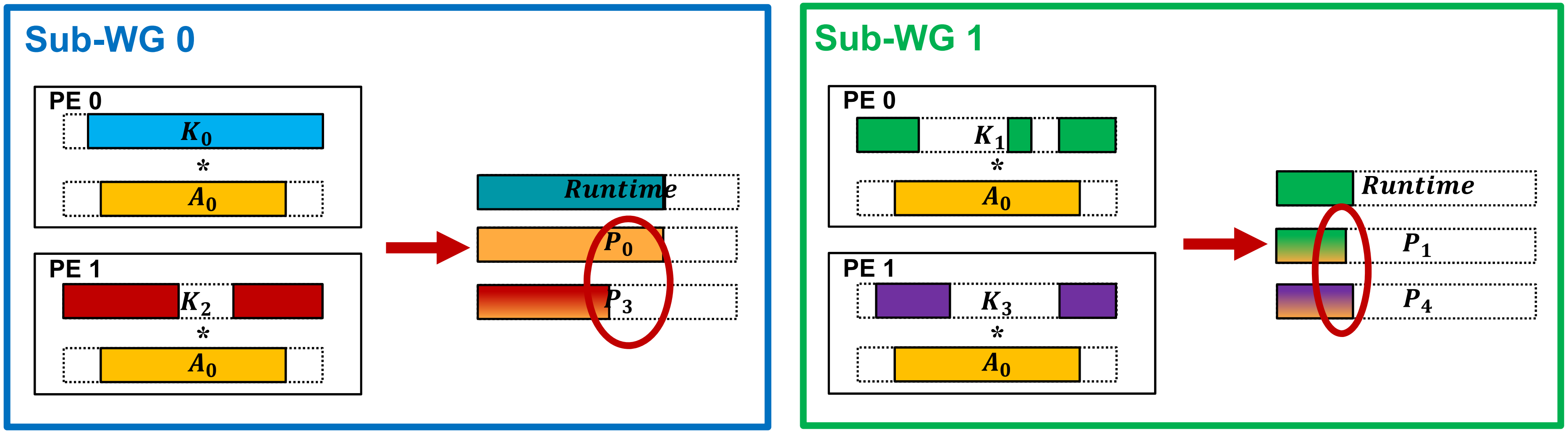


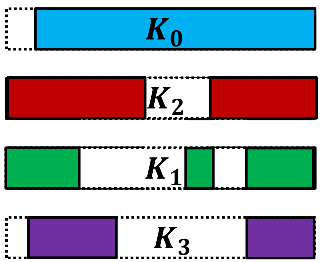
Zero-induced Load Imbalance



- Typically, kernel weights are allocated to PEs based on the kernel index.

Zero-aware Kernel Allocation



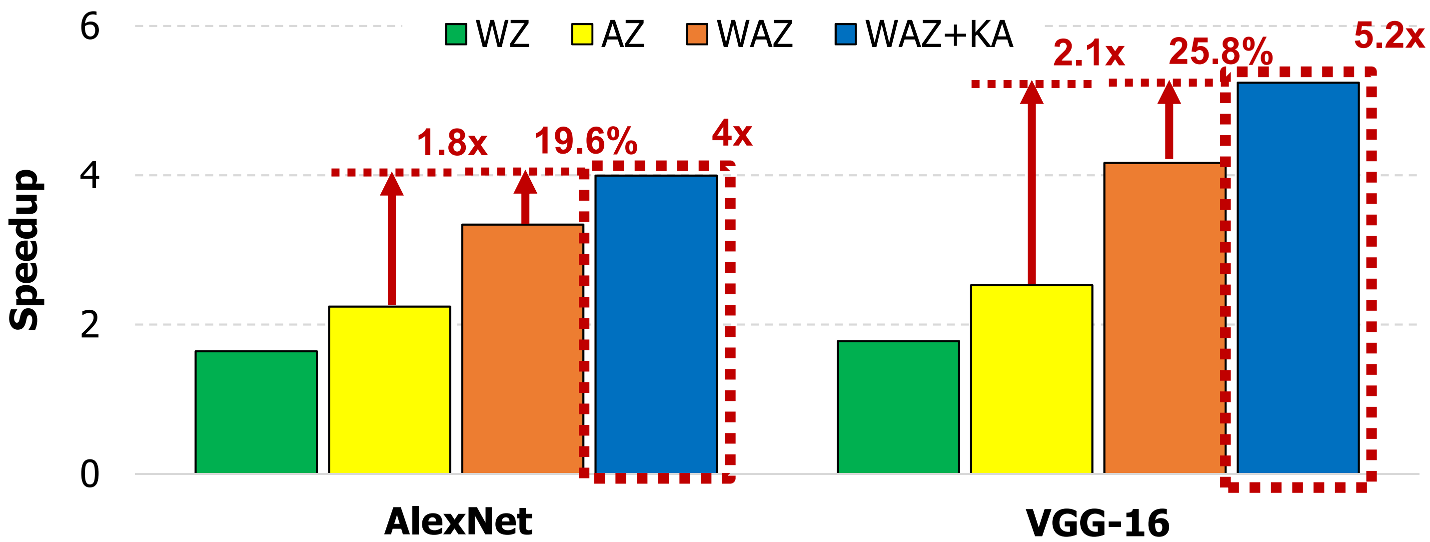


Before kernel allocation is applied

After kernel allocation is applied

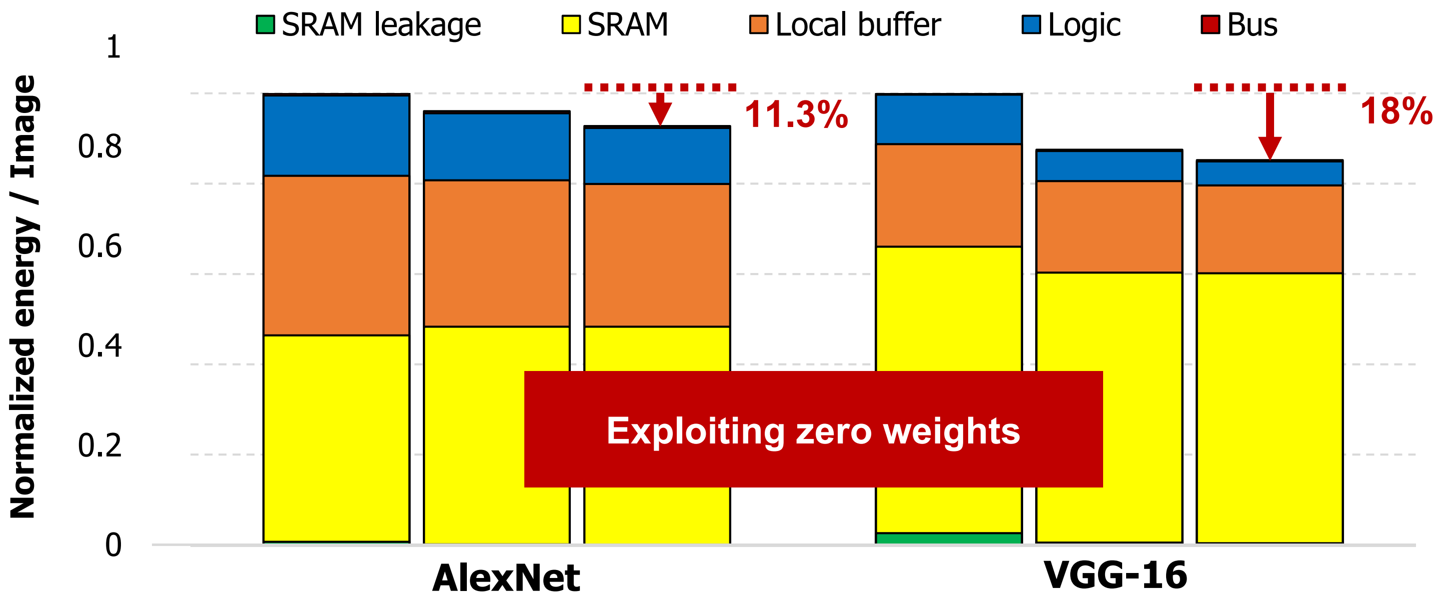
- First, we sort the sets of kernel weights **based on the number of non-zero weights** in the sets.
- First, allocate the sets of kernel weights to PEs in the sorted order.

Performance



- **4x** (AlexNet) and **5.2x** (VGG-16) speed up ***w.r.t. Eyeriss***.
- **1.8x** (AlexNet) and **2.1x** (VGG-16) speed up ***w.r.t. AZ (Cnvlutin)***.
- **19.6%** (AlexNet) and **25.8%** (VGG-16) speed up ***w.r.t. WAZ***.

Energy



- **11.3%** (AlexNet) and **18%** (VGG-16) energy reduction ***w.r.t. Eyeriss***.

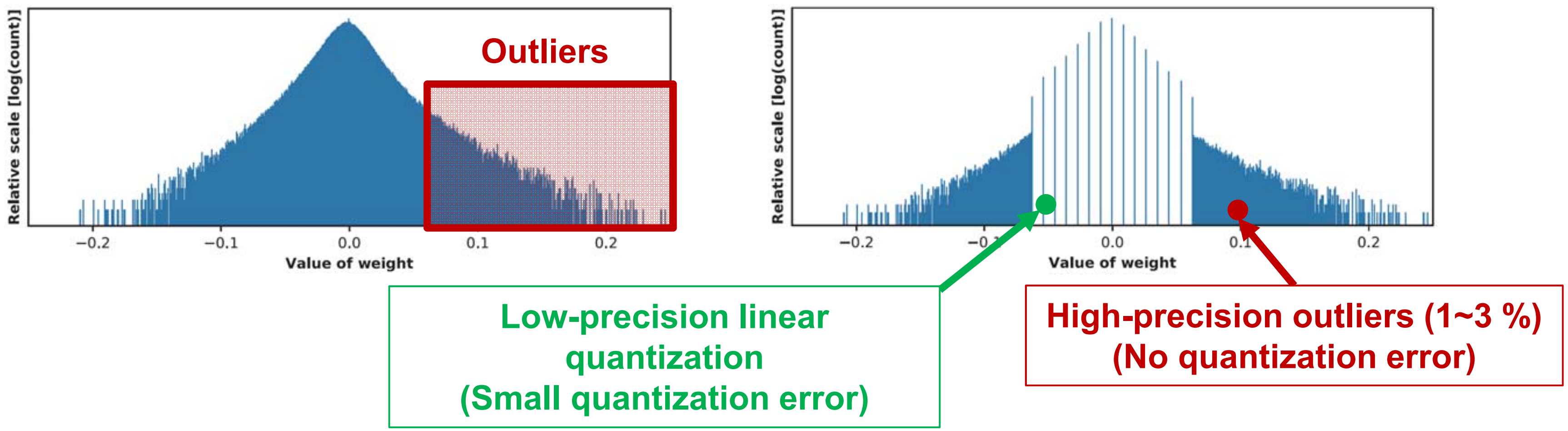
NPU Utilizing Reduced Precision

Reduced Precision

	FP32		INT8					
			Calibration using 5 batches		Calibration using 10 batches		Calibration using 50 batches	
NETWORK	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5
Resnet-50	73.23%	91.18%	73.03%	91.15%	73.02%	91.06%	73.10%	91.06%
Resnet-101	74.39%	91.78%	74.52%	91.64%	74.38%	91.70%	74.40%	91.73%
Resnet-152	74.78%	91.82%	74.62%	91.82%	74.66%	91.82%	74.70%	91.78%
VGG-19	68.41%	88.78%	68.42%	88.69%	68.42%	88.67%	68.38%	88.70%
Googlenet	68.57%	88.83%	68.21%	88.67%	68.10%	88.58%	68.12%	88.64%
Alexnet	57.08%	80.0[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
NETWORK	Top1	To[illegible]5	Diff Top1	Diff Top5	Diff Top1	Diff Top5	Diff Top1	Diff Top5
Resnet-50	73.23%	91.1[illegible]%	0.20%	0.03%	0.22%	0.13%	0.13%	0.12%
Resnet-101	74.39%	91.7[illegible]%	-0.13%	0.14%	0.01%	0.09%	-0.01%	0.06%
Resnet-152	74.78%	91.8[illegible]%	0.15%	0.01%	0.11%	0.01%	0.08%	0.05%
VGG-19	68.41%	88.7[illegible]%	-0.02%	0.09%	-0.01%	0.10%	0.03%	0.07%
Googlenet	68.57%	88.8[illegible]%	0.36%	0.16%	0.46%	0.25%	0.45%	0.19%
Alexnet	57.08%	80.0[illegible]%	0.08%	0.08%	0.08%	0.07%	0.03%	-0.01%

- Neural network shows **comparable accuracy** after **applying quantization**.
- Quantization method reduces **computation complexity** and **memory footprint**.

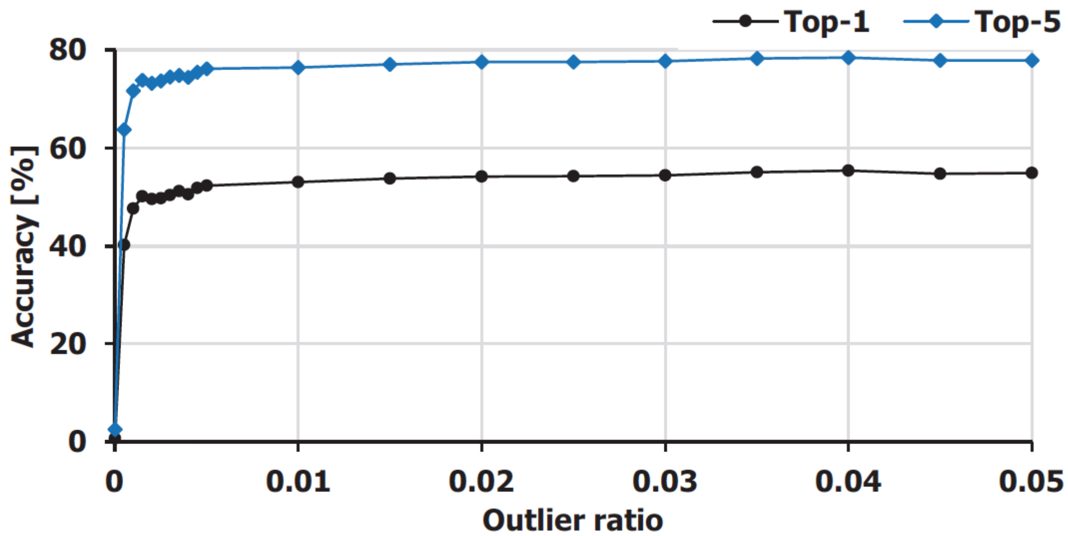
Outlier Quantization

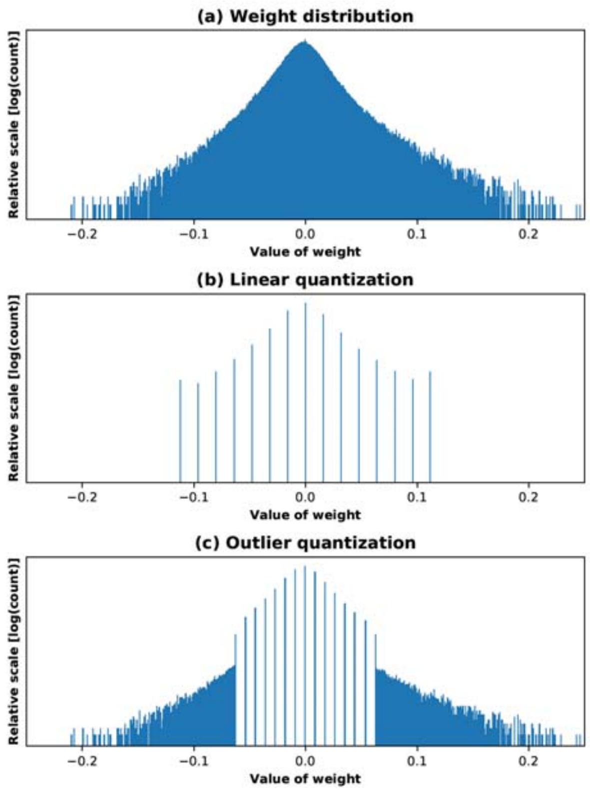


- **Outlier:** weight and activation having larger value than threshold.
- **Outlier incurs quantization error.**
- **Outlier-aware Quantization**
  - Keep **outliers** in **high-precision**.
  - Apply linear quantization to data except outliers.

Accuracy

- 4-bit quantization with **outliers of 0.5%** already gives good accuracy **without fine-tuning.**
- Outliers of 3.5% **lose only <1% accuracy**.





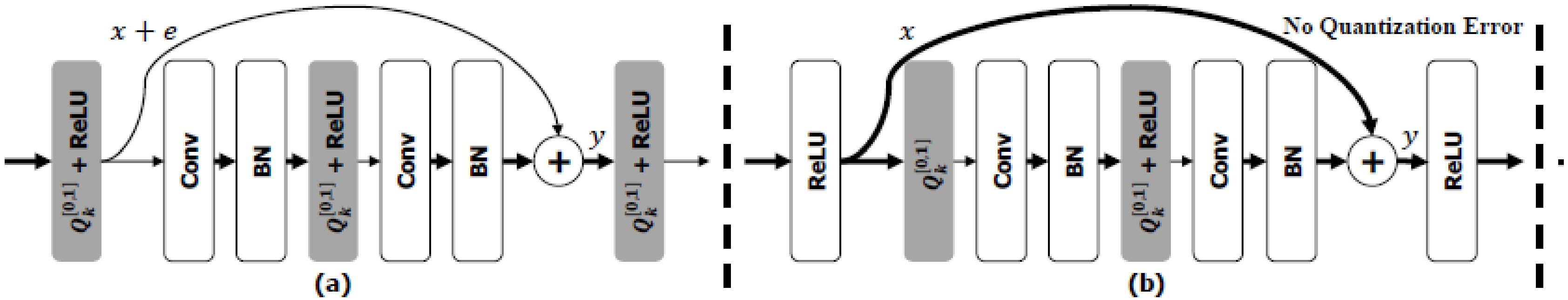
Precision Highway

Conventional approach

$$y = F(x+e) + x + e = F(x) + x + e_r + e$$

Precision highway

$$y = F(x+e) + x = F(x) + x + e_r$$



- **Precision Highway**
  - Keep **residual path** in **high-precision** (8-bit).
  - Apply low-bit linear quantization (2-bit) to data except residual path.

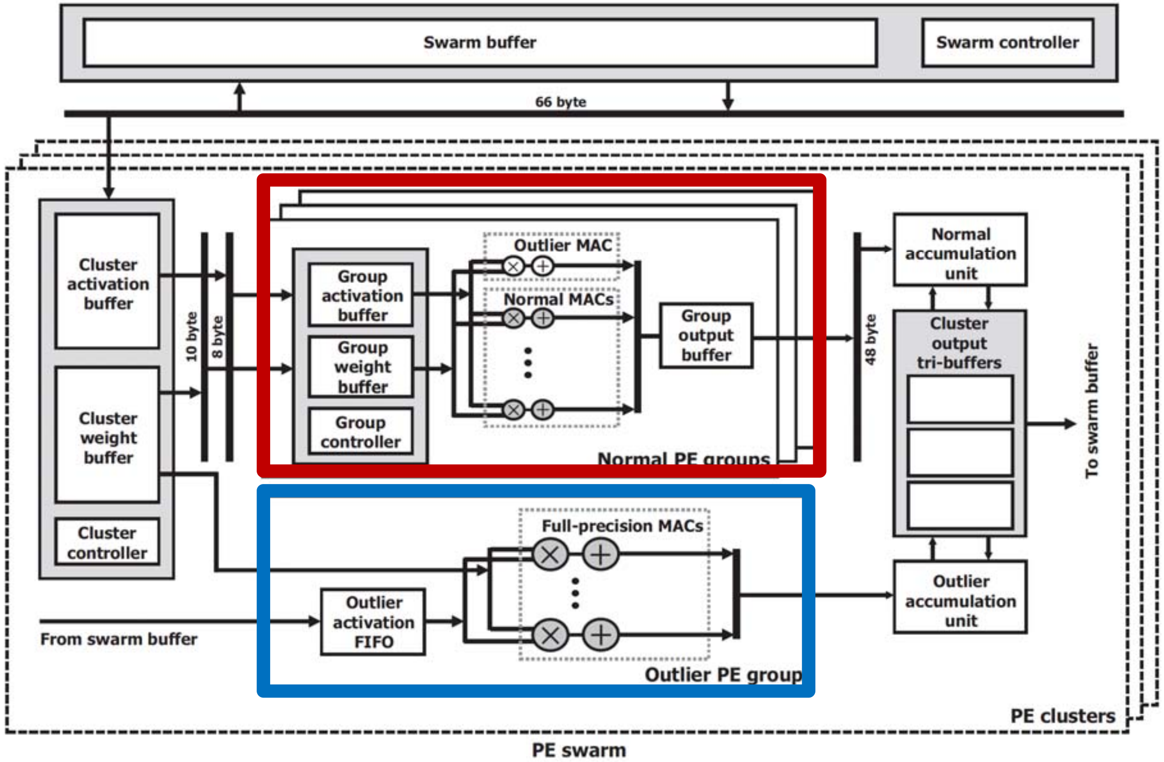
Precision Highway vs. Zhuang's (2-bit)

Laplace	Teacher	Highway	ResNet-18	ResNet-50
✓			61.66 / 84.28	70.50 / 89.84
✓	✓		62.66 / 85.00	71.70 / 90.39
✓		✓	65.83 / 86.71	72.99 / 91.19
✓	✓	✓	66.71 / 87.40	73.55 / 91.40
Full-precision			70.15 / 89.27	76.00 / 92.98
Zhuang's (ours)			60.06 / 83.34	69.04 / 89.14
Zhuang's (ours) + Teacher			61.21 / 84.36	70.48 / 89.83

Zhuang's: [Bohan Zhuang et al. Towards effective low-bitwidth convolutional neural networks. Computer Vision and Pattern Recognition(CVPR), 2018.]

- **Precision highway** shows **66.71%** and **73.55%** TOP-1 accuracy in ResNet-18 and ResNet-50, respectively.

NPU Supporting Mixed-precision Computation (OLAccel)



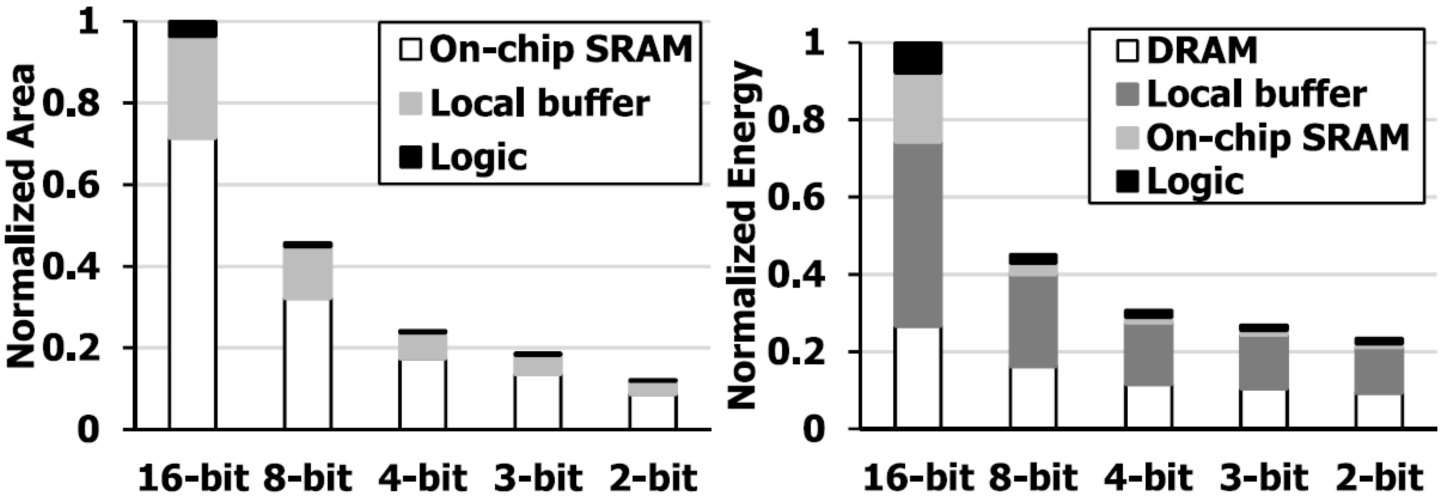
- **Mixed-precision computation**
  - **In case of outlier:**

    4-bit data (dense)
    + 16-bit activation/8-bit weight outliers (sparse)

  - **In case of precision highway:**

    2-bit data (dense)
    + 8-bit residual path (sparse)

Area and Energy Reduction



- **82.3 %** reduction in **chip area** (16-bit vs. 3-bit).
- **73.1 %** reduction in **energy consumption** (16-bit vs. 3-bit).

Working as an AI System Architect in the Industry

HYPERCONNECT

하이퍼커넥트

HYPERCONNECT

하이퍼커넥트는 WebRTC와 AI 기술을 바탕으로,
Video로 전 세계 사람들을 연결해 새로운 가치를 만들고 있습니다.

Social Discovery

[Chat]
Live chat Azar

[Entertainment]
Live streaming Hakuna

and more...

Real-time Video

WebRTC Video + Mobile Optimized AI

기술력

HYPERCONNECT

모바일 WebRTC를 최초로 상용화

- 전 세계 어느 국가, 어느 통신사, 어느 단말기에서도 안정적으로 영상 통화 가능
- 지구 반대편 남미의 통신사별 망 품질관리 모니터링 가능한 글로벌 인프라 확보

모바일에 최적화된 머신 러닝 기술

- 서버에 데이터를 보내지 않고 모바일에서 실시간으로 데이터를 처리하는 On-device AI
- 낮은 CPU 성능, 적은 메모리 환경에서도 빠르게 동작하는 딥러닝 가속 기술

오피스 라이프

HYPERCONNECT

Neural Network Acceleration on Mobile CPU

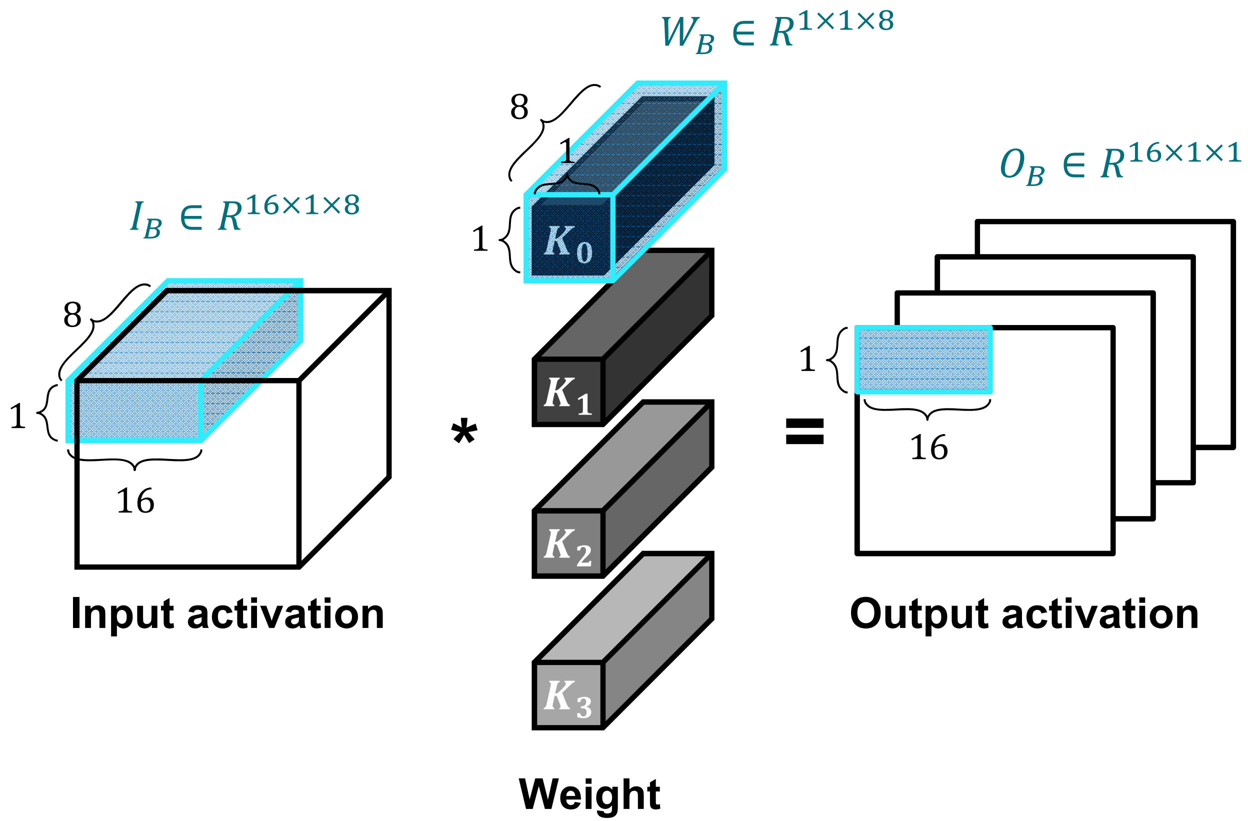
HYPERCONNECT

Running AI Applications Using Mobile CPU

- **AI applications are widely used** even on low-end mobile devices where **NPU is absent**.
- **CPU is required** for executing specific operations utilized in state-of-the-art neural networks.

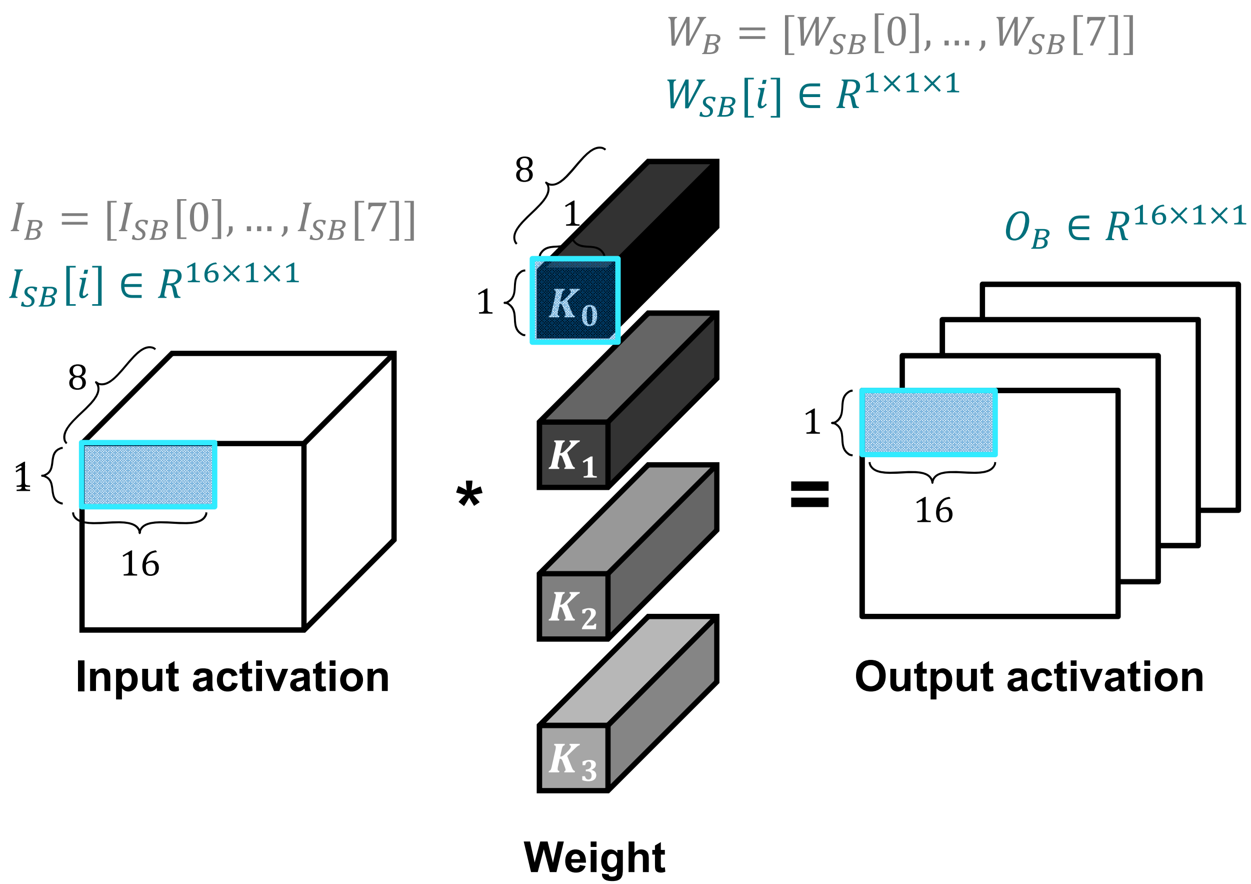
HYPERCONNECT

Example of 1X1 Convolution



HYPERCONNECT

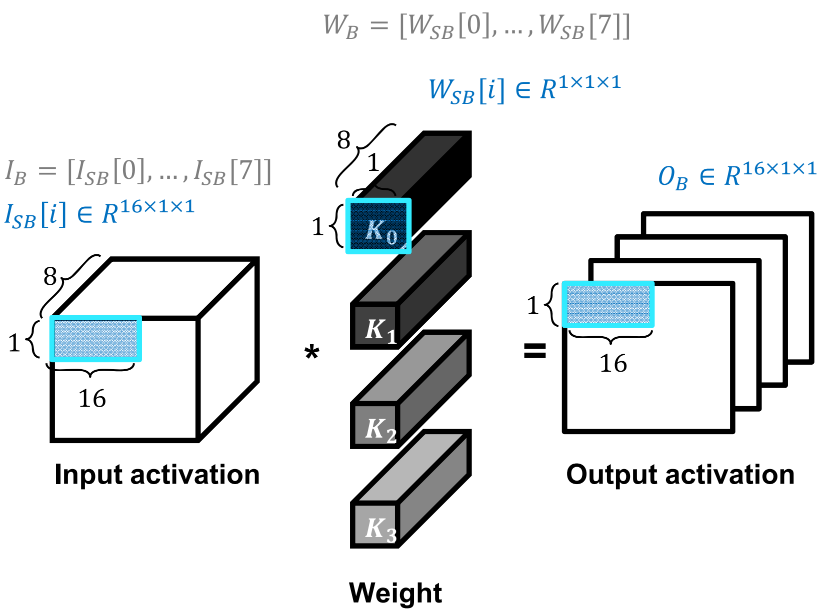
Example of 1X1 Convolution



HYPERCONNECT

1X1 Convolution on ARM CPU with 8-bit Quantization (w/o quality loss)

Example of 1X1 convolution



Pseudo code

\# assuming NCHW data format, int8 arithmetic operation and ARMv8 architecture

for i = 0 to 3

- **load output activation** $[O_B[4\times i], O_B[4\times i+1], O_B[4\times i+2], O_B[4\times i+3]]$ #int32x4

for i = 0 to 7

- **load input activation** $[I_{SB}[i][0], \ldots, I_{SB}[i][15]]$ #int8x16

  for j = 0 to 3

  - **MAC** $(I_{SB}[i][4\times j] \sim I_{SB}[i][4\times j+3], W_{SB\times4}[i], O_B[4\times j] \sim O_B[4\times j+3])$ #int32x4

for i = 0 to 3

- **store output activation** $[O_B[4\times i], O_B[4\times i+1], O_B[4\times i+2], O_B[4\times i+3]]$ #int32x4

1X1 Convolution on ARM CPU with 8-bit Quantization (w/o quality loss)

Pseudo code (float32; 72 cycle)

4 cycle

```
for i = 0 to 3
    ...ad output activation [O_B[4×i], O_B[4×i+1],
    ...[4×i+2], O_B[4×i+3]] #float32x4
```

64 cycle

```
for i = 0 to 7
    for j = 0 to 3
        • load input activation [I...
          j+1], I_SB[i][4×j+2], I_SB...
          #float32x4
        • MAC (I_SB[i][4×j] ~ I_SB[i][4×j+3], W_SB×4[i],
          O_B[4×j] ~ O_B[4×j+3]) #float32x4
```

4 cycle

```
for i = 0 to 3
    ...ore output activation [O_B[4×i], O_B[4×i+1],
    ...[4×i+2], O_B[4×i+3]] #float32x4
```

Pseudo code (int8; 48 cycle)

4 cycle

```
for i = 0 to 3
    • load output activation [O_B[4×i], O_B[4×i+1],
      O_B[4×i+2] , O_B[4×i+3]] #int32x4
```

```
for i = 0 to 7
    ...ut activation [I_SB[i][0], … , I_SB[i][15]]
    ...o 3
        ... (I_SB[i][4×j] ~ I_SB[i][4×j+3], W_SB×4[i],
        O_B[4×j] ~ O_B[4×j+3]) #int32x4
```

4 cycle

```
for i = 0 to 3
    store output activation [O_B[4×i], O_B[4×i+1],
    O_B[4×i+2] , O_B[4×i+3]] #int32x4
```

1.5x speed up

HYPERCONNECT

1X1 Convolution on ARM CPU with 8-bit Quantization (w/ quality loss)

Pseudo code (int8)

```
# assuming NCHW data format, int8 arithmetic operation and
ARMv8 architecture

for i = 0 to 3
    • load output activation [O_B[4×i], O_B[4×i+1], O_B[4×i+2] , O_B[4×i+3]] #int32x4

for i = 0 to 7
    • load input activation [I_SB[i][0], … , I_SB[i][15]] #int8x16
    for j = 0 to 3
        • MAC (I_SB[i][4×j] ~ I_SB[i][4×j+3], W_SBx4[i], O_B[4×j] ~ O_B[4×j+3]) #int32x4

for i = 0 to 3
    • store output activation [O_B[4×i], O_B[4×i+1], O_B[4×i+2] , O_B[4×i+3]] #int32x4
```

Pseudo code (int8 w/ quality loss)

```
# assuming NCHW data format, int8 arithmetic operation and
ARMv8 architecture

for i = 0 to 1
    • load output activation [O_B[8×i], O_B[8×i+1], … , O_B[8×i+7]] #int32x4

for i = 0 to 7
    • load input activation [I_SB[i][0], … , I_SB[i][15]] #int8x16
    • MAC (I_SB[i][0] ~ I_SB[i][7], W_SBx4[i], O_B[0] ~ O_B[7]) #int16x8
    • MAC (I_SB[i][8] ~ I_SB[i][15], W_SBx4[i], O_B[8] ~ O_B[15]) #int16x8

for i = 0 to 1
    • store output activation [O_B[8×i], O_B[8×i+1], … , O_B[8×i+7]] #int16x8
```

1X1 Convolution on ARM CPU with 8-bit Quantization (w/ quality loss)

Pseudo code (float32; 72 cycle)

4 cycle

```
for i = 0 to 3
    ...d output activation [O_B[4×i], O_B[4×i+1],
    ...[4×i+2], O_B[4×i+3]] #float32x4
```

64 cycle

```
for i = 0 to 7
    for j = 0 to 3
        • load input activation [I_SB...
          j+1], I_SB[i][4×j+2], I_SB[i][...
          #float32x4
        • MAC (I_SB[i][4×j] ~ I_SB[i][4×j+...
          O_B[4×j] ~ O_B[4×j+3]) #float32x4
```

4 cycle

```
for i = 0 to 3
    ...re output activation [O_B[4×i], O_B[4×i+1],
    ...[4×i+2], O_B[4×i+3]] #float32x4
```

Pseudo code (int8 w/ quality loss; 28 cycle)

2 cycle

```
for i = 0 to 1
    load output activation [O_B[8×i], O_B[8×i+1], … ,
    O_B[8×i+7]] #int16x8
```

```
for i = 0 to 7
    ... activation [I_SB[i][0], … , I_SB[i][15]]
    ...0] ~ I_SB[i][7], W_SB×4[i], O_B[0] ~ O_B[7])
    ...8] ~ I_SB[i][15], W_SB×4[i], O_B[8] ~ O_B[15])
    #int16x8
```

2 cycle

```
for i = 0 to 1
    store output activation [O_B[8×i], O_B[8×i+1], … ,
    O_B[8×i+7]] #int16x8
```

2.6x speed up

HYPERCONNECT

Optimizing Neural Network for Mobile Devices

Neural Network Optimization for Mobile Devices

AR glass (Google, Microsoft)

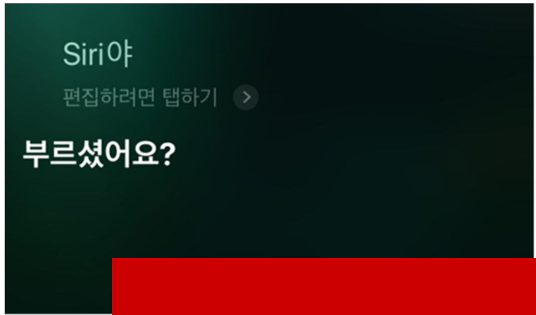
Robot (Amazon)

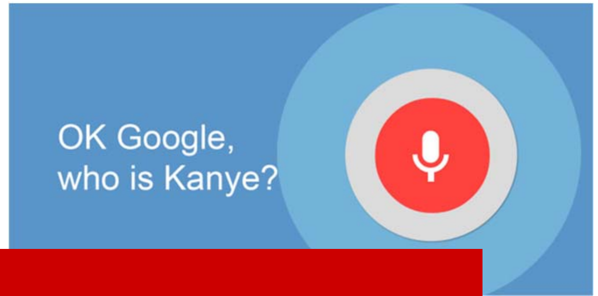
Self driving car (Tesla, Waymo, Uber)

IoT devices (Naver, Google, Amazon, Apple, ...)

- IT industries are interested in **IoT**, **AR/VR**, **robotics** and **self driving car**.
- They require **tremendous computations**.
- **Neural network optimization for mobile devices is required**.

Keyword Spotting (KWS)





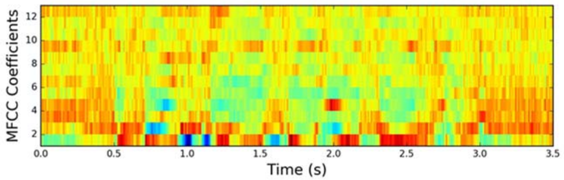
Widely used for hands-free control of mobile applications

ogle”

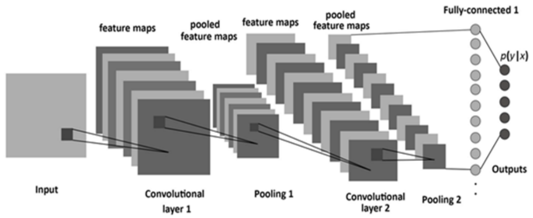
- **Keyword spotting (KWS)** deals with the **identification of predefined keywords in utterances**.
- Recognizing **wake-up word** (*“Hey Siri”* and *“OK Google”*) and distinguishing **common command** (*“yes”* or *“no”*).

CNN-based KWS

Utterance



2D input
(e.g., MFCC)



CNN

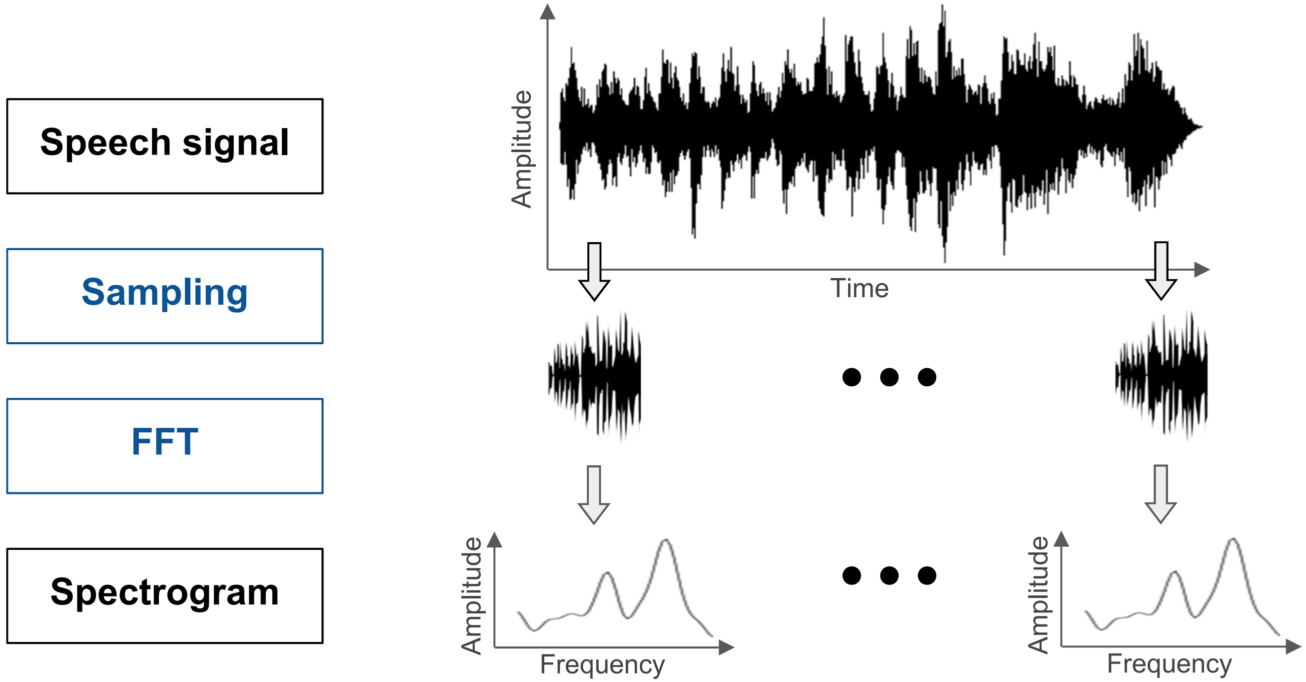
- **CNN-based KWS** studies show **remarkable accuracy**.
- Most of CNN-based KWS approaches **receive features as a 2D input** of a convolutional network.

Challenges of KWS on Real-time Mobile Devices

It is challenging to implement fast and accurate KWS on mobile devices with restricted hardware resources

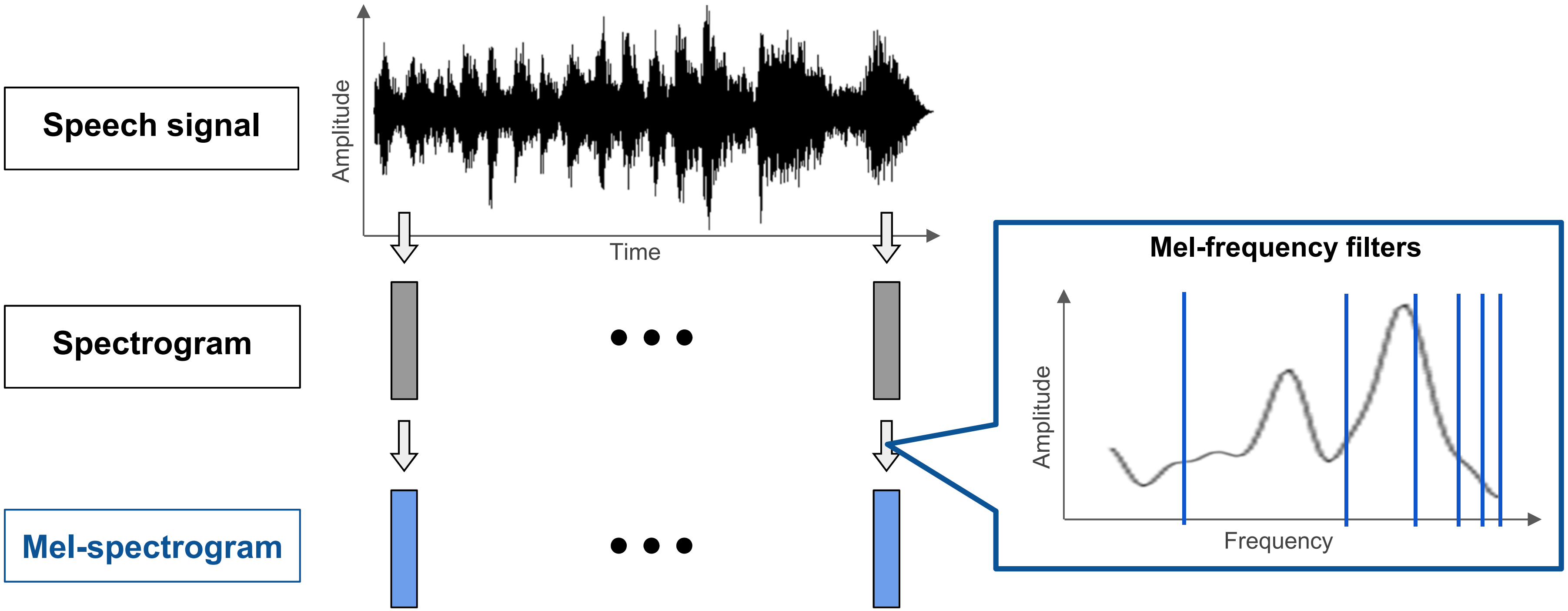
- Since use of **KWS** is commonly **concentrated on mobile devices**, the response of KWS should be both **fast** and **accurate**.

Preliminary: Spectrogram



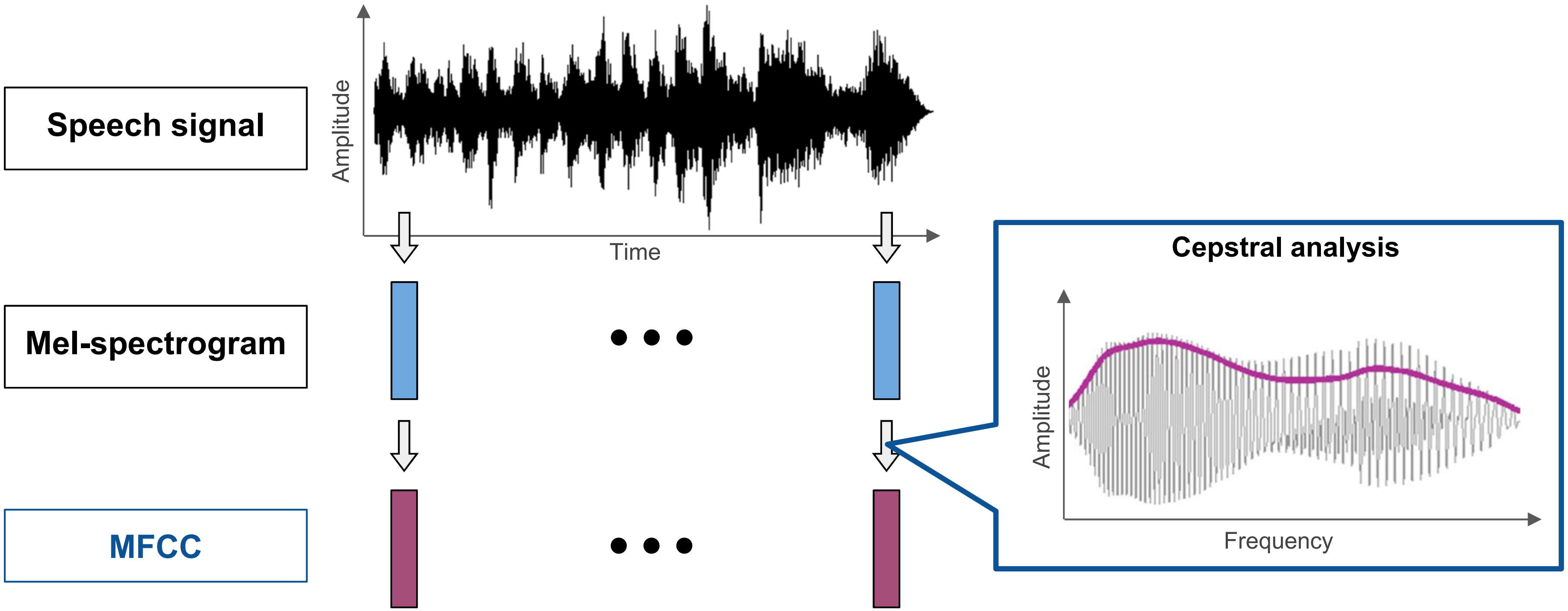
- **Time-frequency** representation of a speech signal is referred to as spectrogram.

Preliminary: Mel-frequency Filter



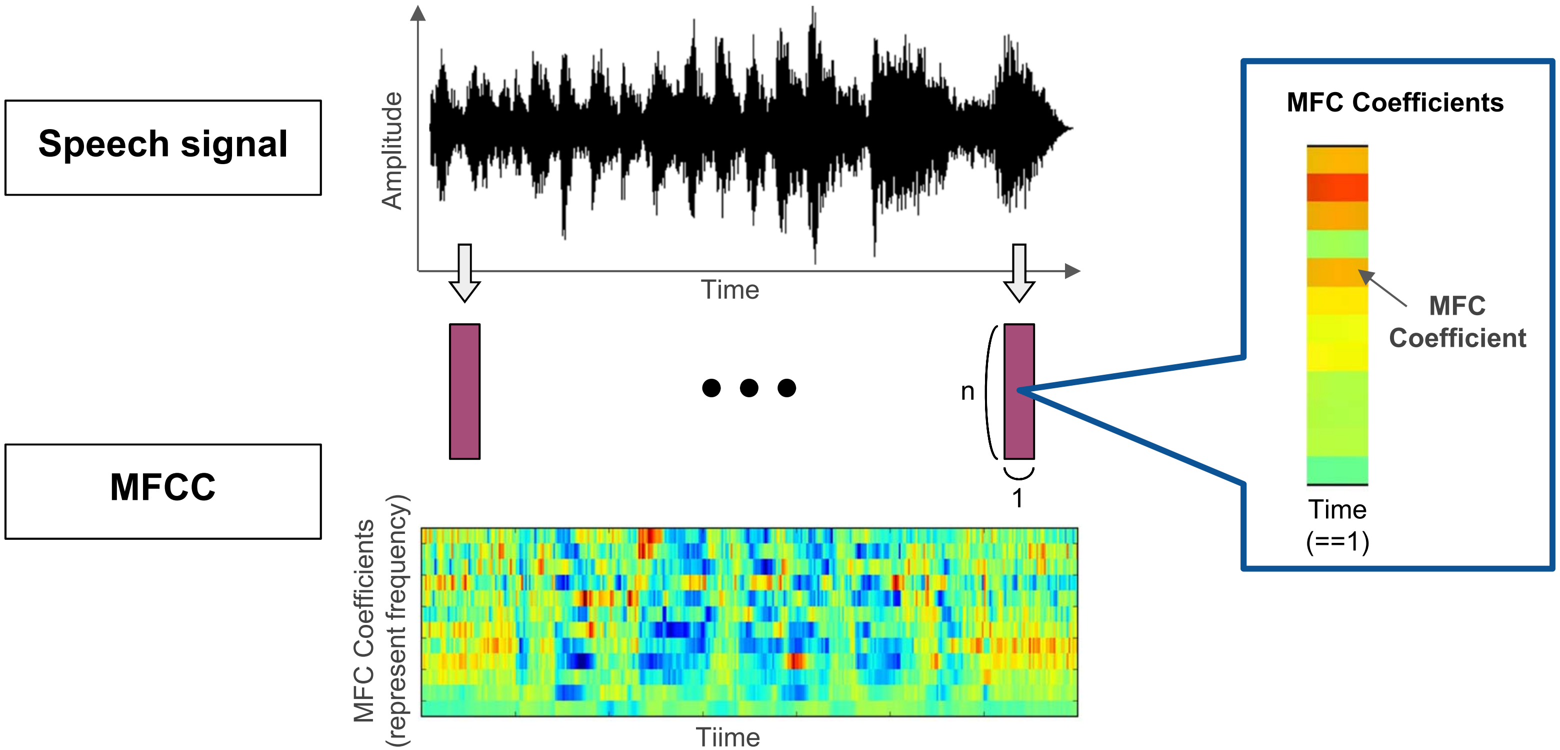
- It is observed that **human ears act as filter**.
- Mel-frequency filters are **non-uniformly spaced on the frequency axis**.

Preliminary: Mel-frequency Cepstral Coefficients (MFCC)



- **Cepstral coefficients** obtained from **Mel-spectrogram** are referred to as Mel-Frequency Cepstral Coefficients (MFCC).

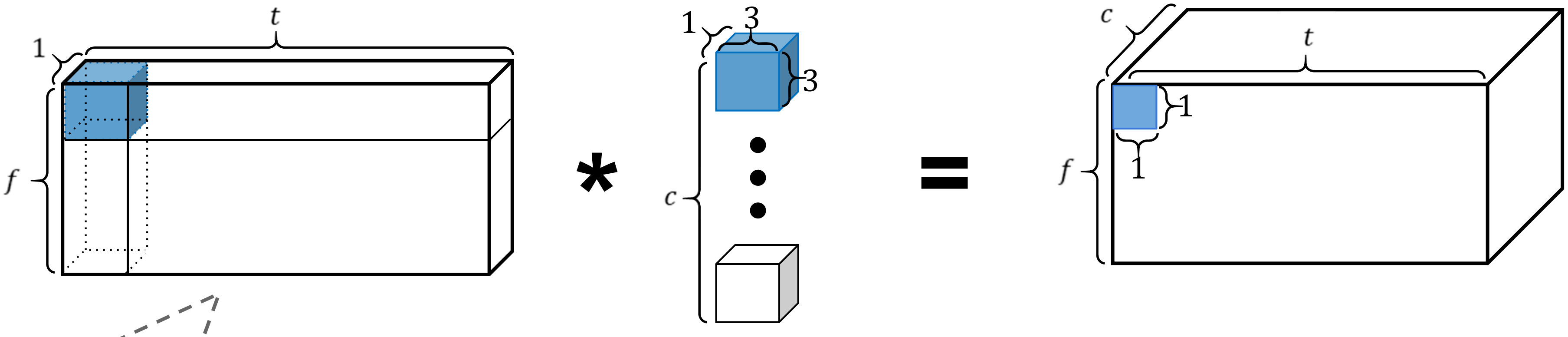
Preliminary: Mel-frequency Cepstral Coefficients (MFCC)

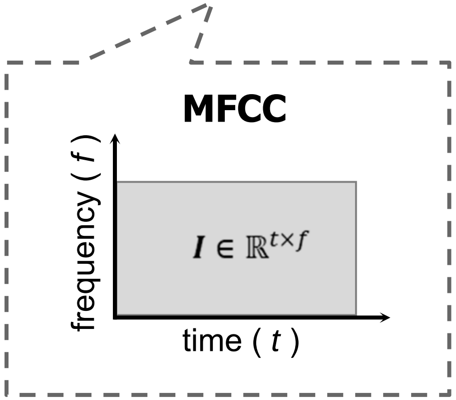


- **MFCC:** speech signal is represented as a sequence of cepstral vectors.

Conventional 2D Convolution for KWS

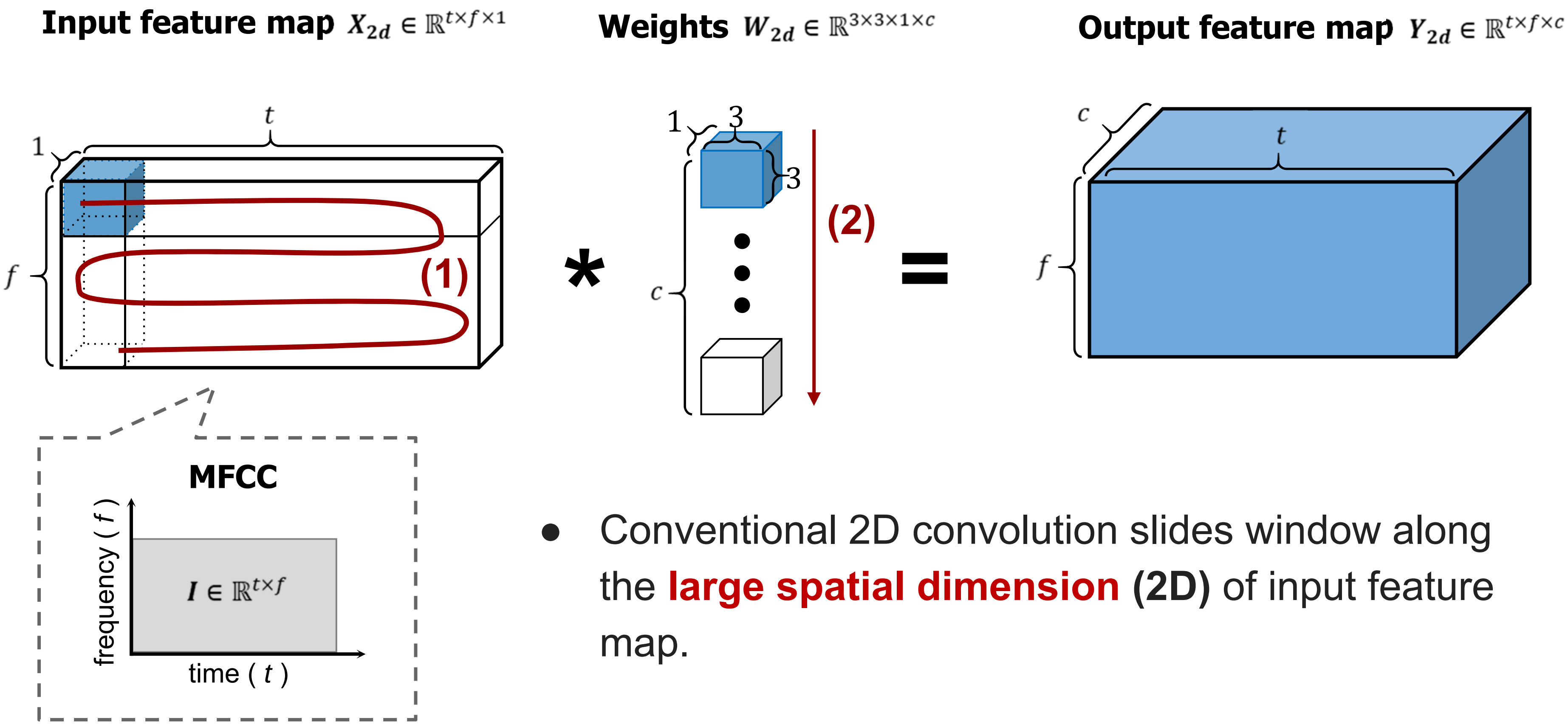
Input feature map $X_{2d} \in \mathbb{R}^{t \times f \times 1}$ **Weights** $W_{2d} \in \mathbb{R}^{3 \times 3 \times 1 \times c}$ **Output feature map** $Y_{2d} \in \mathbb{R}^{t \times f \times c}$





- Conventional 2D convolution for KWS utilizes **input tensor** $\mathbf{X} \in \mathbb{R}^{w \times h \times c}$ where w = t, h = f (or vice versa), and c = 1.

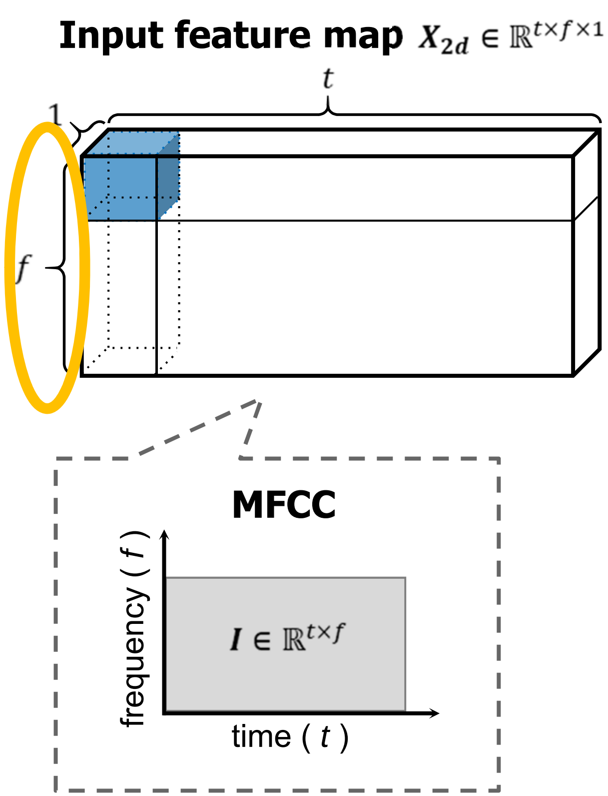
Conventional 2D Convolution for KWS



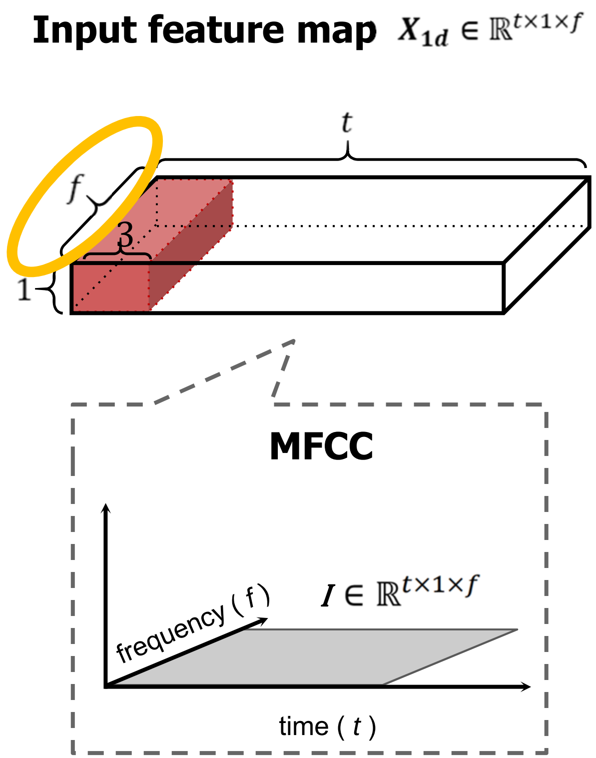
- Conventional 2D convolution slides window along the **large spatial dimension** **(2D)** of input feature map.

Proposed Temporal Convolution for KWS

Conventional approach

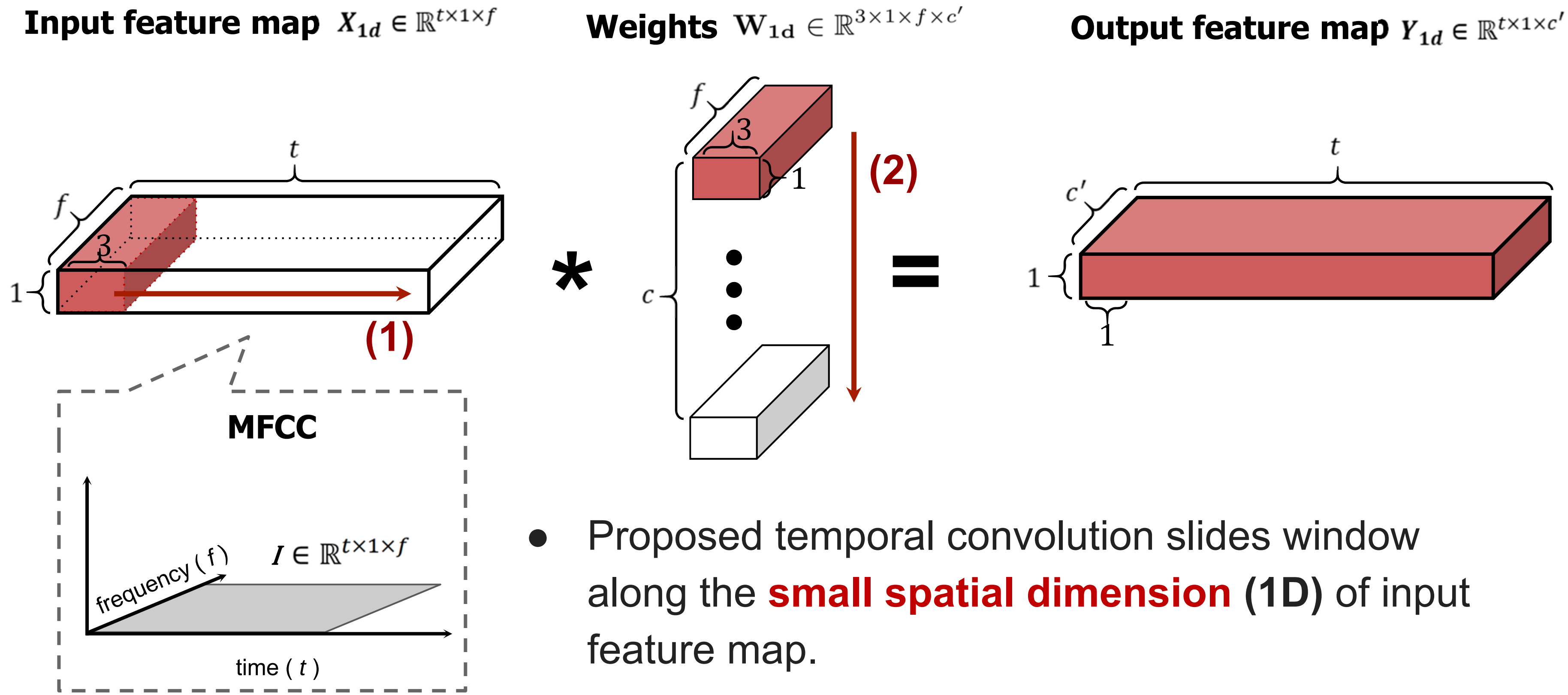


Proposed temporal convolution



HYPERCONNECT

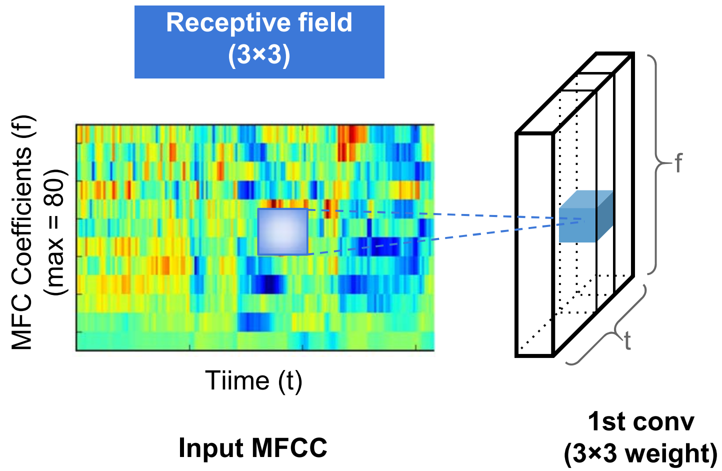
Proposed Temporal Convolution for KWS



- Proposed temporal convolution slides window along the **small spatial dimension** **(1D)** of input feature map.

Problem of Conventional 2D Convolution for KWS

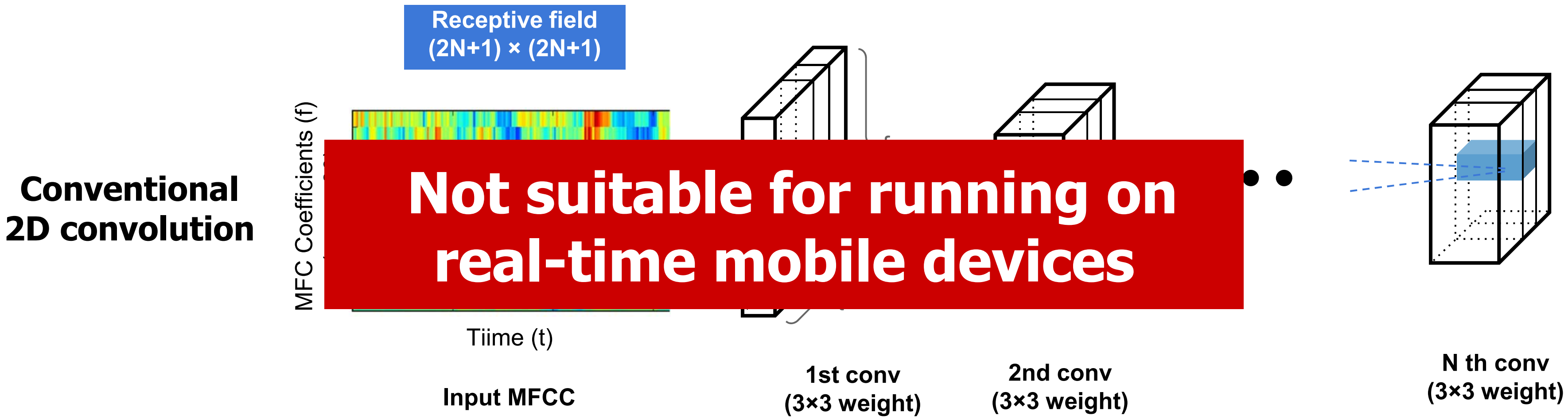
Conventional 2D convolution



- **Both low-and-high frequency data at the same time step includes informative features.**
- Since modern CNNs commonly **utilize small kernels**, it is **difficult to capture informative features from both low and high frequencies**.

HYPERCONNECT

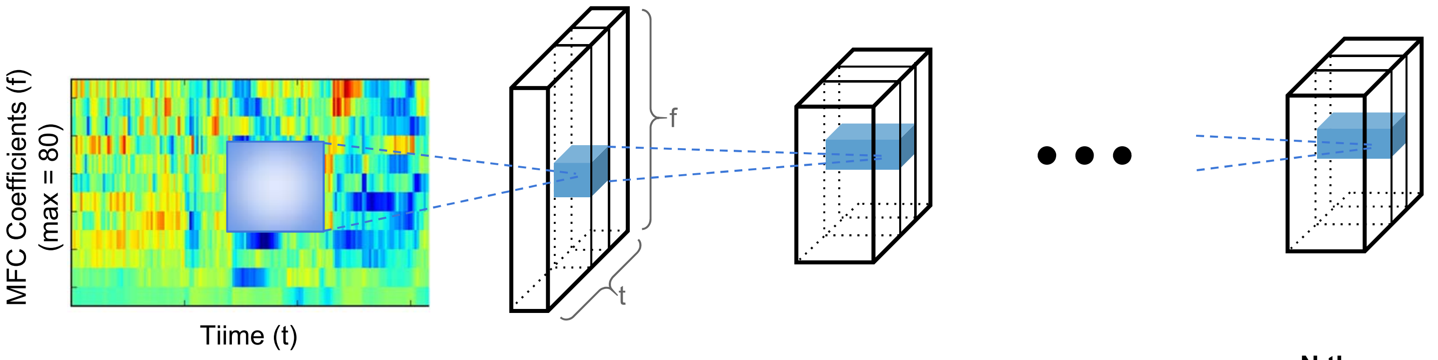
Small Receptive Field of Conventional 2D Convolution



- Assume that **N convolutional layers** of 3 × 3 weights with a stride of one exist, the **receptive field** of the network only grows up to **2N+ 1**.
- **Conventional 2D convolution requires a large number of operations to increase receptive field**.

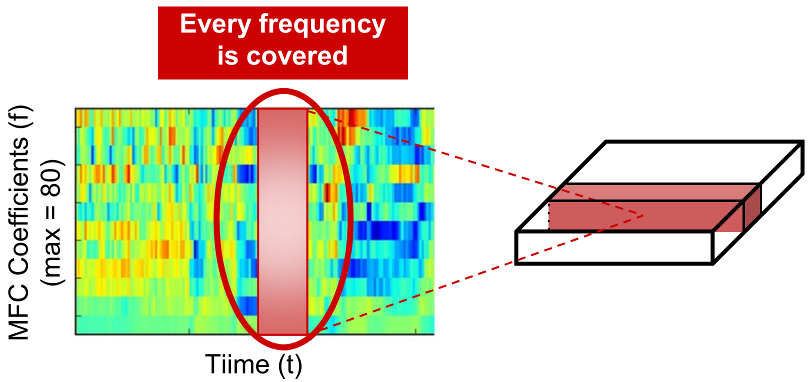
Large Receptive Field of Proposed Temporal Convolution

Conventional 2D convolution



Input MFCC | **1st conv (3×3 weight)** | **2nd conv (3×3 weight)** | **N th conv (3×3 weight)**

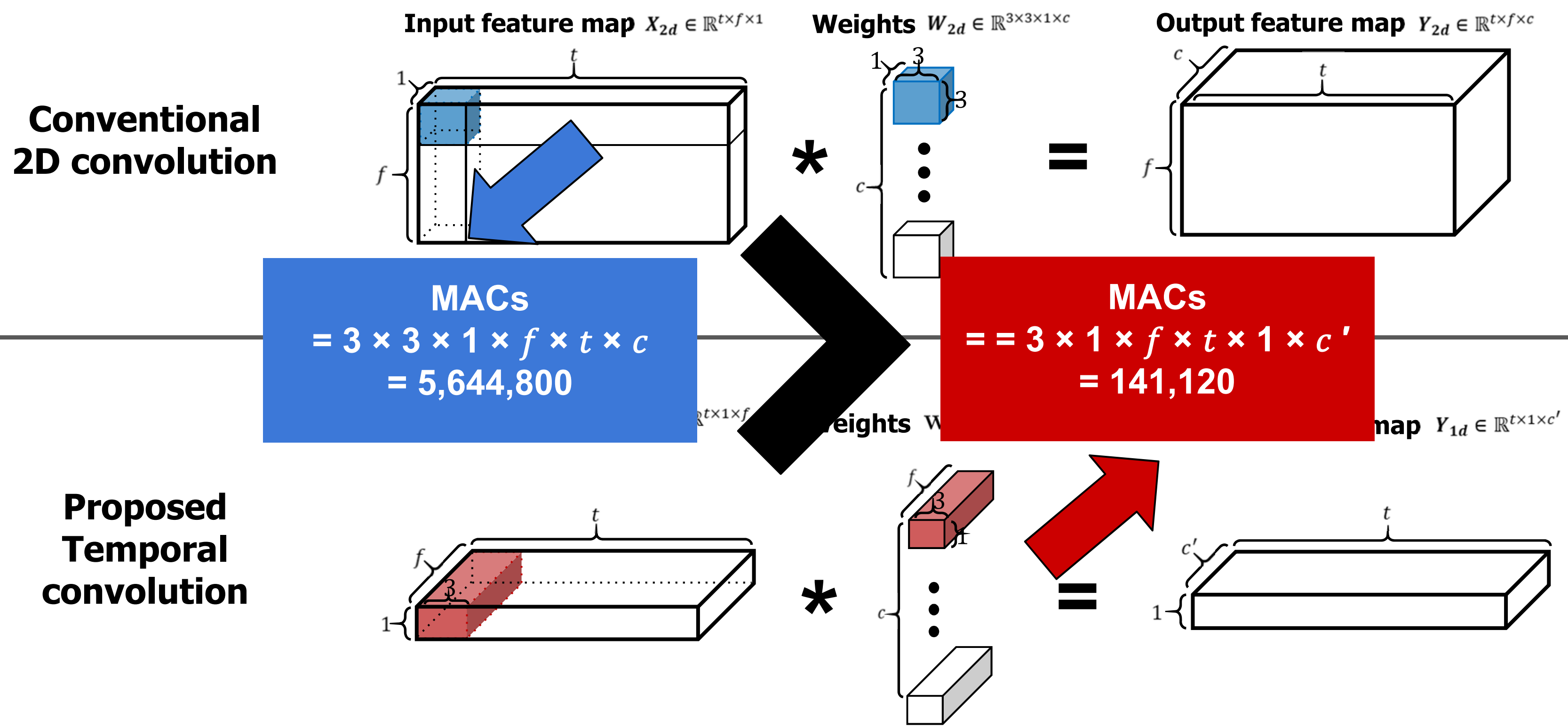
Proposed Temporal convolution



Input MFCC | **1st conv (3×1 weight)**

- In the proposed method, **all lower-level features always participate in forming the higher-level features** in the next layer.

Small Footprint and Low Computational Complexity



Note that both the parameters of a conventional 2D convolution and that of the temporal convolution have the same size in this example by setting t = 98, f = 40, c = 160, and c` = 12.

HYPERCONNECT

End-to-end Pipeline for Mobile Devices

- **End-to-end pipeline for Mobile Devices**
  - We release end-to-end pipeline codebase for **training**, **evaluating**, and **benchmarking** the baseline models and together with the proposed models.
- **Proposed codebase includes following components:**
  - **TensorFlow models**
  - Scripts to convert the models into the **TensorFlow Lite models** (mobile devices)
  - Pre-built **TensorFlow Lite Android benchmark tool** (mobile performance measurement)
- **Git repo**
  - https://github.com/hyperconnect/TC-ResNet

HYPERCONNECT

Accuracy and Inference Time

Model	Acc. (%)	Time (*ms*)	FLOPs	Params
CNN-1	90.7*	32	76.1M	524K
CNN-2	84.6*	**1.2**	1.5M	148K
DS-CNN-S	94.4*	1.6	5.4M	24K
DS-CNN-M	94.9*	5.2	19.8M	140K
DS-CNN-L	95.4*	16.8	56.9M	420K
Res8-Narrow	90.1*	47	143.2M	20K
Res8	94.1*	174	795.3M	111K
Res15-Narrow	94.0*	107	348.7M	43K
Res15	**95.8***	424	1950.0M	239K
TC-ResNet8	96.1	**1.1**	3.0M	66K
TC-ResNet8-1.5	96.2	2.8	6.6M	145K
TC-ResNet14	96.2	2.5	6.1M	137K
TC-ResNet14-1.5	**96.6**	5.7	13.4M	305K

- **TC-ResNet8** improves **11.5%p** accuracy with comparable latency compared to **latency state-of-the-art** (**CNN-2**).

HYPERCONNECT

Accuracy and Inference Time

Model	Acc. (%)	Time (*ms*)	FLOPs	Params
CNN-1	90.7*	32	76.1M	524K
CNN-2	84.6*	**1.2**	1.5M	148K
DS-CNN-S	94.4*	1.6	5.4M	24K
DS-CNN-M	94.9*	5.2	19.8M	140K
DS-CNN-L	95.4*	16.8	56.9M	420K
Res8-Narrow	90.1*	47	143.2M	20K
Res8	94.1*	174	795.3M	111K
Res15-Narrow	94.0*	107	348.7M	43K
Res15	**95.8***	424	1950.0M	239K
TC-ResNet8	96.1	**1.1**	3.0M	66K
TC-ResNet8-1.5	96.2	2.8	6.6M	145K
TC-ResNet14	96.2	2.5	6.1M	137K
TC-ResNet14-1.5	**96.6**	5.7	13.4M	305K

- **TC-ResNet8** achieves **385x** speedup while improving **0.3%p** accuracy compared to **accuracy state-of-the-art** (**Res15**).

Conclusion

Neural Network Optimization in the Future

AR glass (Google, Microsoft)

Robot (Amazon)

Self driving car (Tesla, Waymo, Uber)

IoT devices
(Naver, Google, Amazon, Apple, …)

- **Edge devices will run more complex tasks in the future.**
- Neural network acceleration becomes more important in the future.

Thanks

HYPERCONNECT

References

- **ZeNA**

  - Dongyoung Kim, Soobeom Kim, Sungjoo Yoo, "FPGA Prototyping of Low-Precision Zero-Skipping Accelerator for Neural Networks," in *Rapid System Prototyping (RSP)*, Nov. 2018.
  - Dongyoung Kim, Junwhan Ahn, and Sungjoo Yoo, "ZeNA: Zero-Aware Neural Network Accelerator," *in IEEE Design & Test*, Aug. 2017.
  - Dongyoung Kim, Junwhan Ahn, and Sungjoo Yoo, "A Novel Zero Weight/Activation-Aware Hardware Architecture of Convolutional Neural Network," *in Proc. Design Automation and Test in Europe (DATE)*, Mar. 2017.

- **Outlier quantization and Precision highway**

  - Eunhyeok Park, Dongyoung Kim, Sungjoo Yoo, “Energy-Efficient Neural Network Accelerator based on Outlier-Aware Low Precision Computation," in *International Symposium on Computer Architecture (ISCA)*, June 2018.
  - Eunhyeok Park, Dongyoung Kim, Sungjoo Yoo, Peter Vajda, “Precision Highway for Ultra Low-Precision Quantization,” *arXiv preprint arXiv:1812.09818*, Dec. 2018.

- **Temporal convolution for real-time KWS**

  - Seungwoo Choi, Seokjun Seo, Beomjun Shin, Hyeongmin Byun, Martin Kersner, Beomsu Kim, Dongyoung Kim, Sungjoo Ha, “Temporal Convolution for Real-time Keyword Spotting on Mobile Devices,” *in Annual Conference of the International Speech Communication Associatio (INTERSPEECH),* Sep. 2019.