TNPU: An Efficient Accelerator Architecture for Training Convolutional Neural Networks

Jiajun Li, Guihai Yan, Wenyan Lu, Shuhao Jiang, Shijun Gong, Jingya Wu, Junchao Yan, Xiaowei Li

2019/1/23

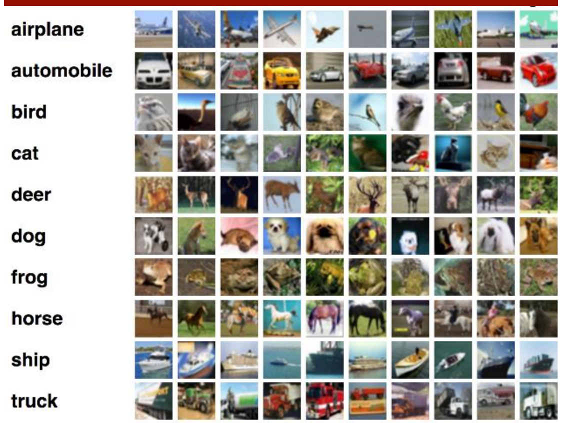
Architecture for Data Analytics and Processing Technology

计算机体系结构国家重点实验室
State Key Laboratory of Computer Architecture, ICT, CAS

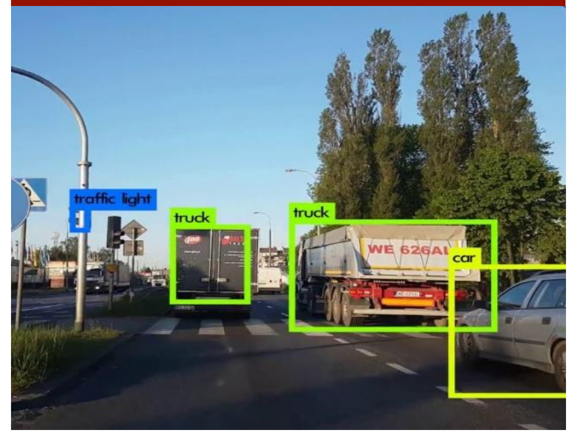
中国科学院计算技术研究所
INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES

Convolutional neural networks

Image classification

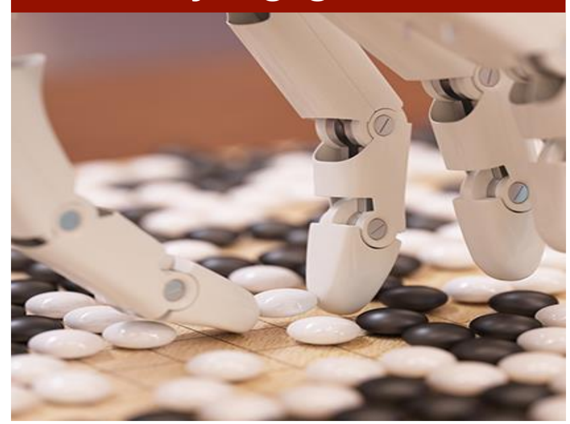


Object detection



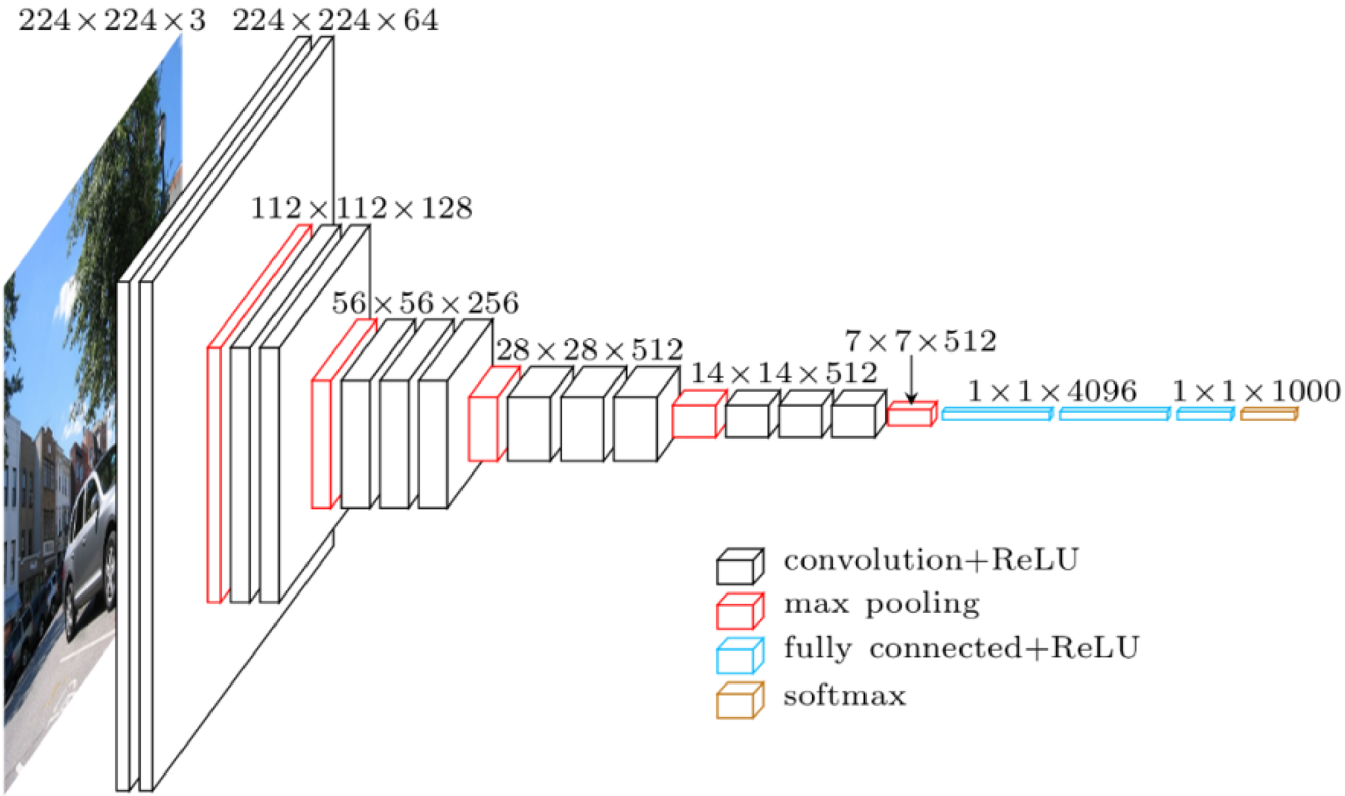
Speech recognition

Playing games



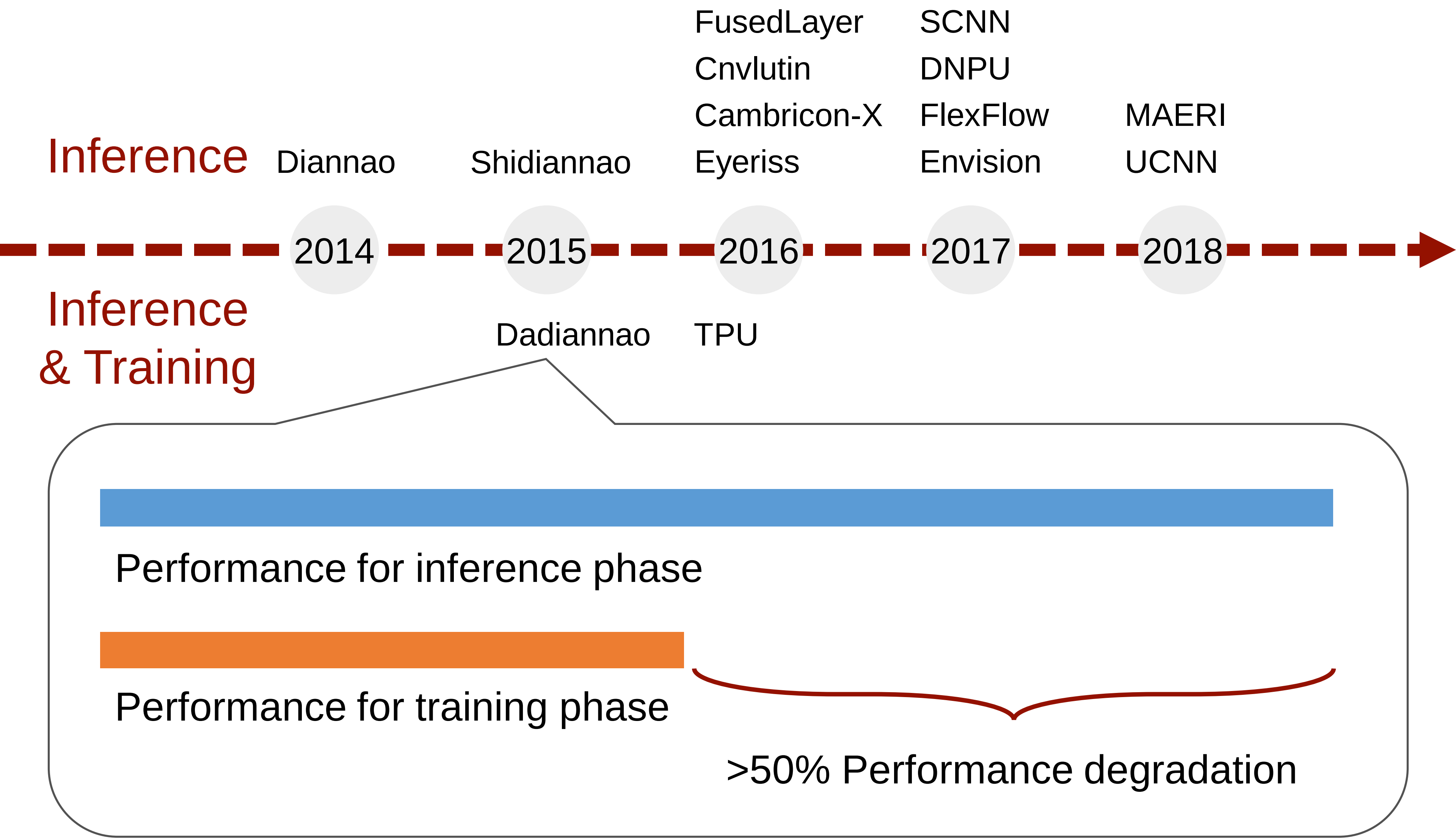
Intensive operations in CNNs

- Training AlexNet takes six days on two GPUs



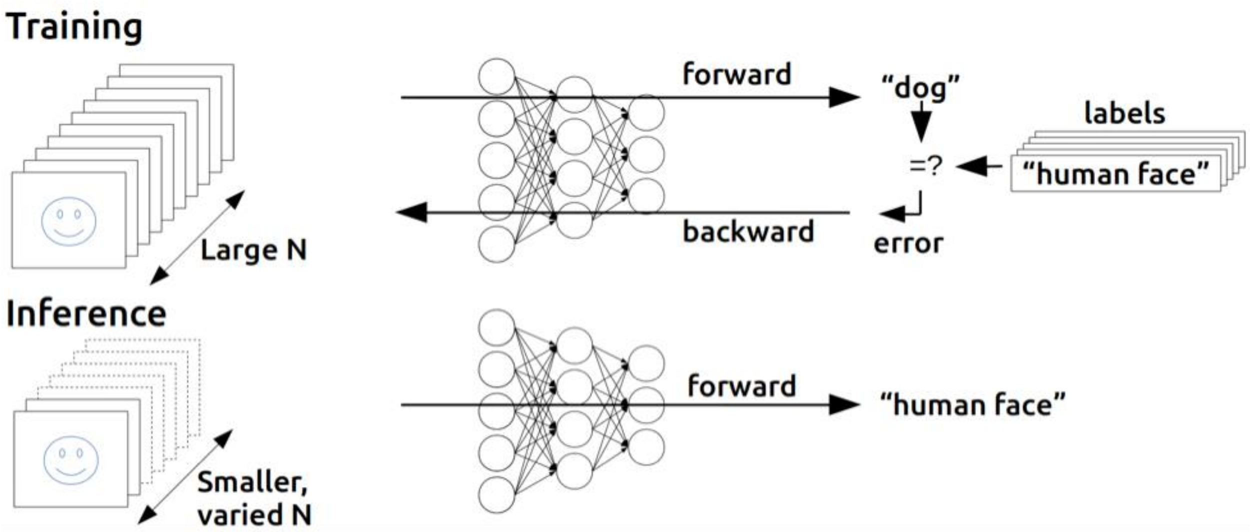
[1] Alex Krizhevsky et al. Imagenet classification with deep convolutional neural networks.

CNN accelerators



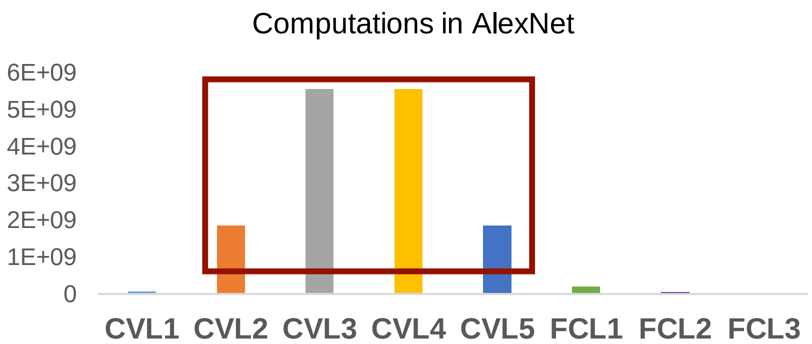
Inference vs. Training

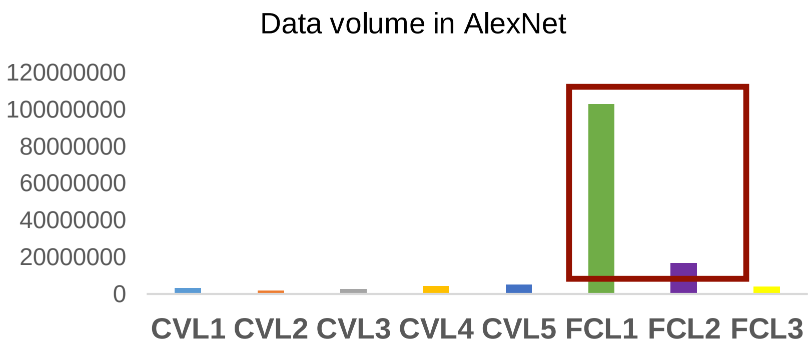
- New challenges in training
  - Optimization for fully-connected layers
  - Bidirectional data dependency
  - Wide-range of convolutional kernel sizes



New challenges in training (1/3)

- Comparison between CVLs and FCLs

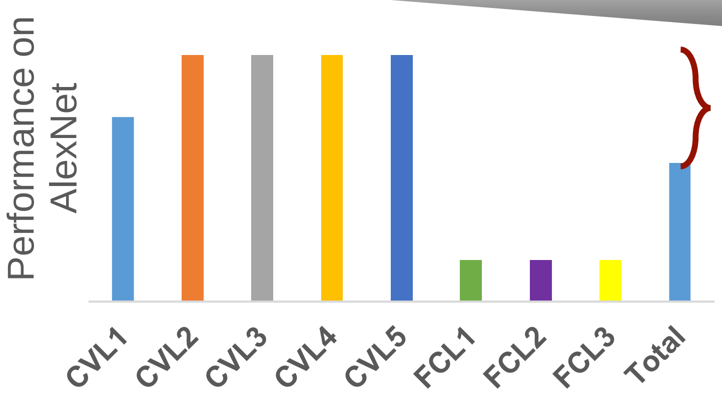




CVLs dominate computation volume

FCLs dominate data volume

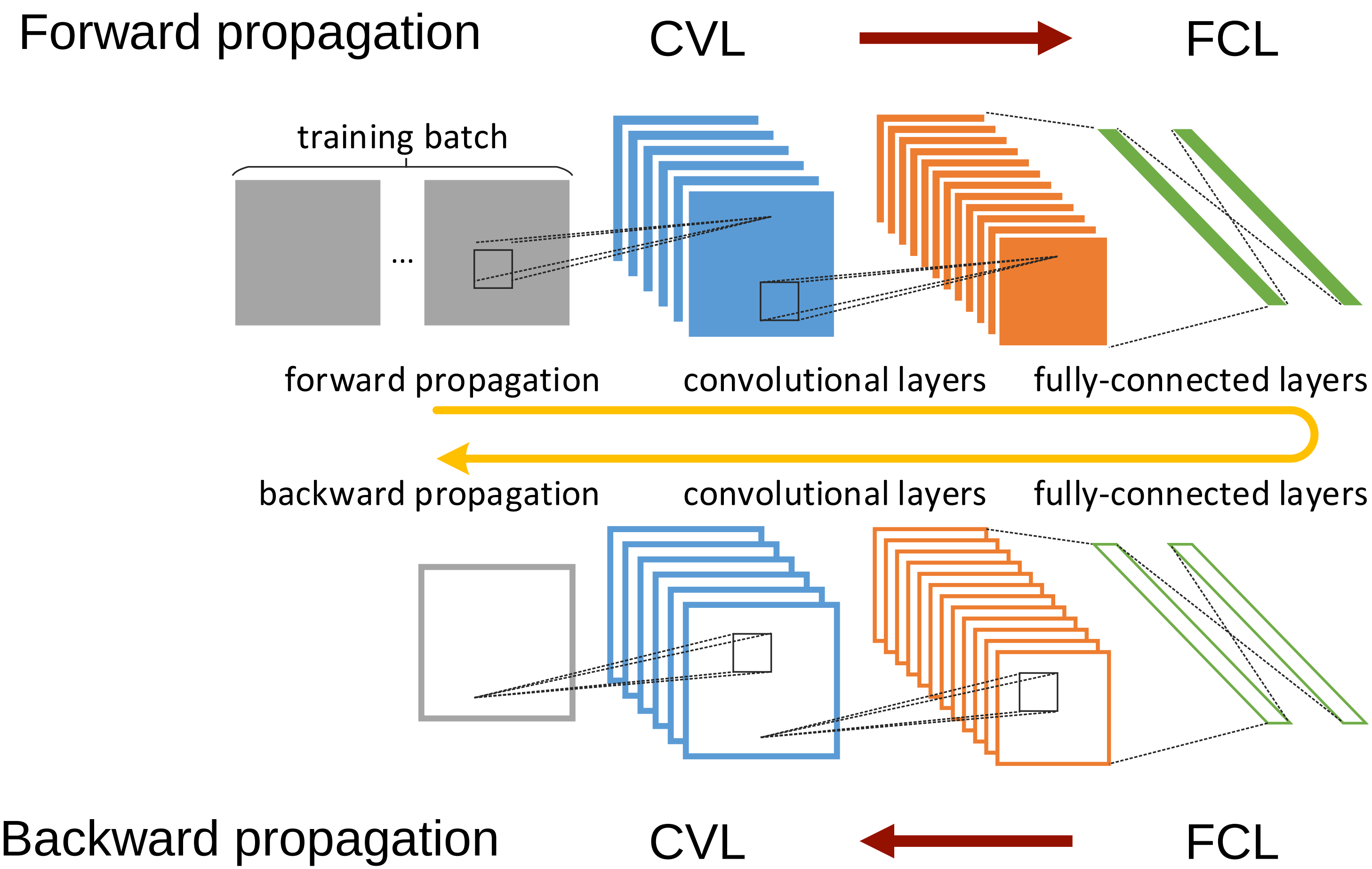
Cannot compress in training



Low performance on FCLs lead to significant performance degradation

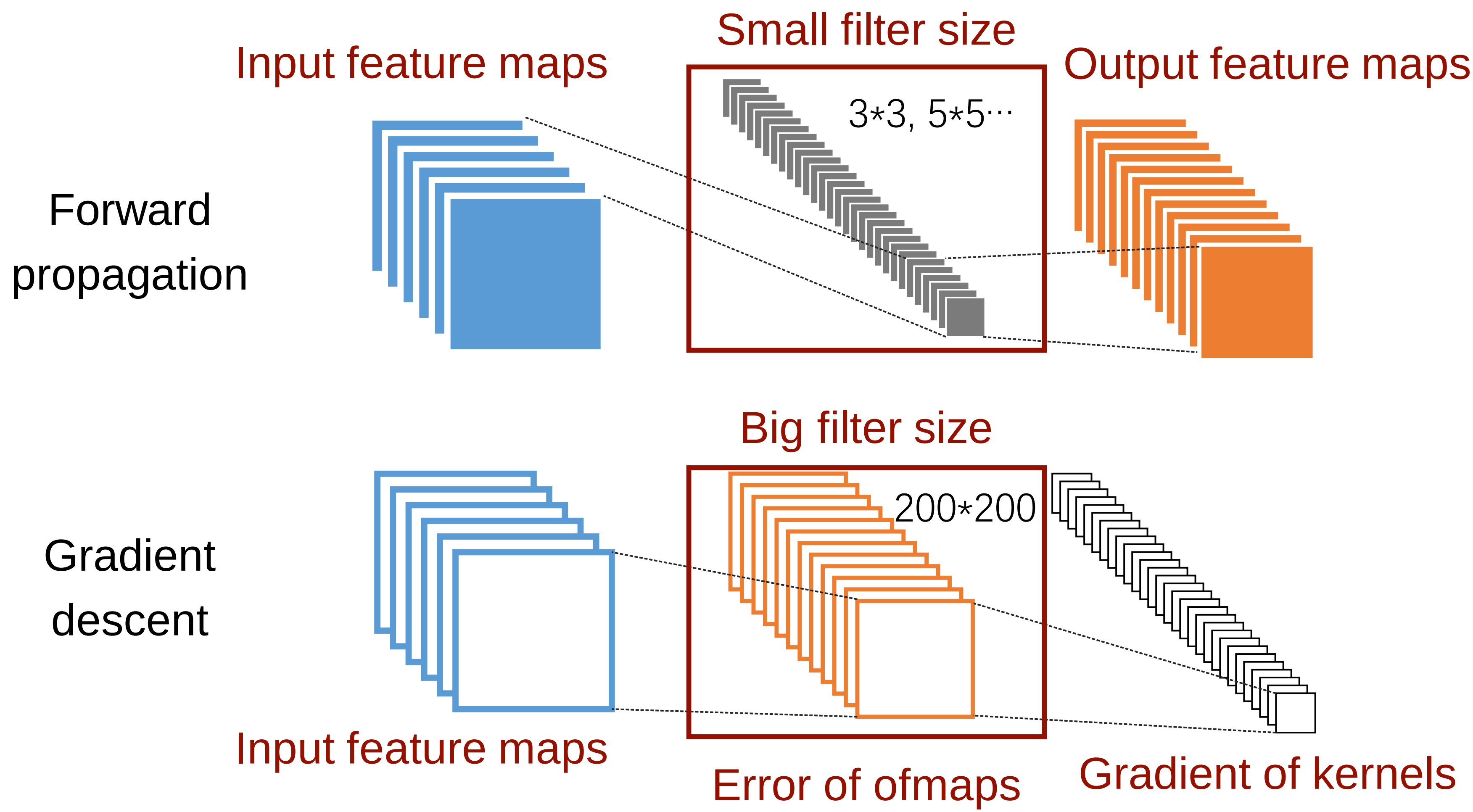
New challenges in training (2/3)

- Bidirectional data dependency



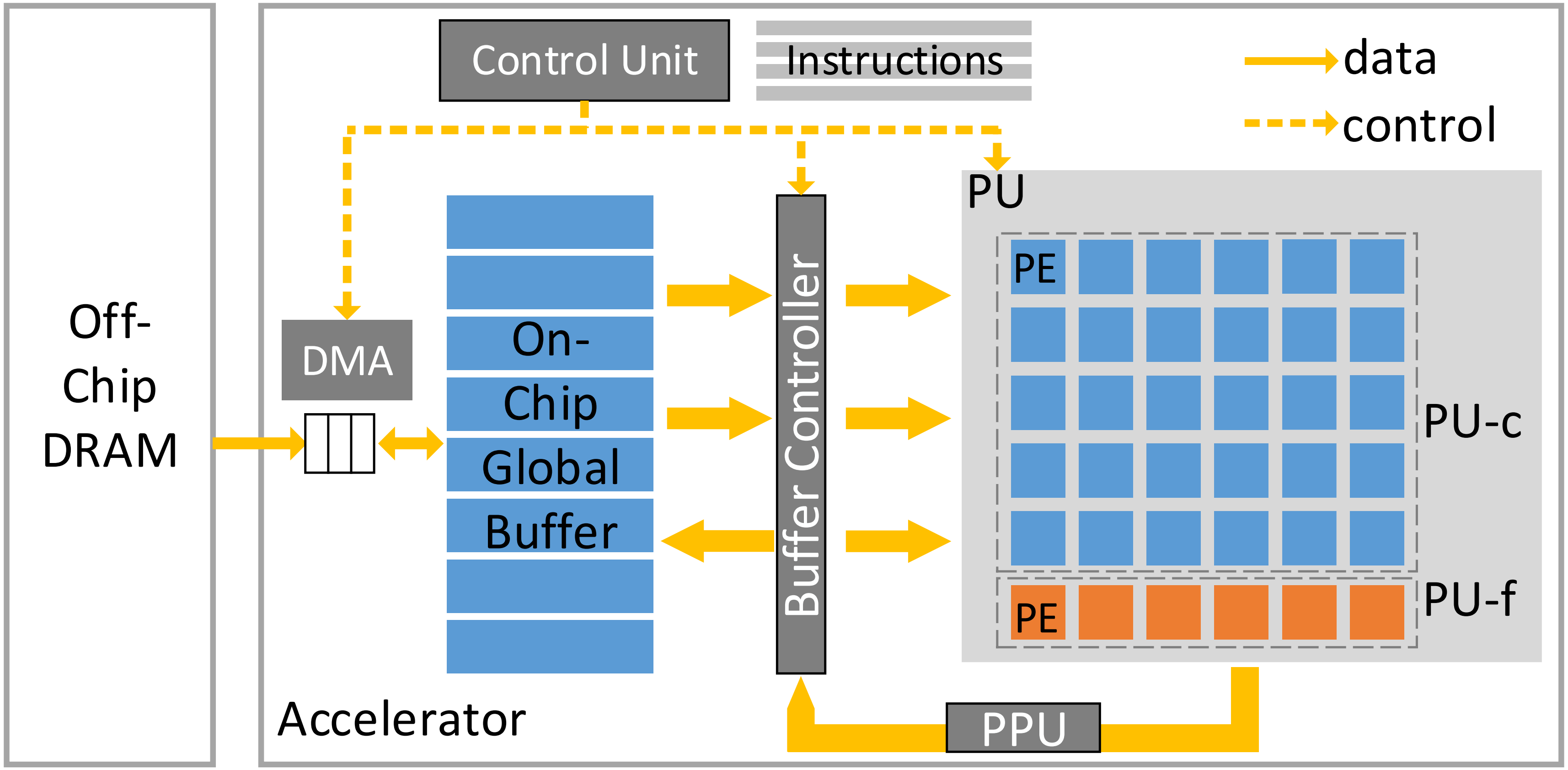
New challenges in training (3/3)

- Extremely large convolutional kernels



Our solution: TNPU

- **T**raining **N**eural Network **P**rocessing **U**nit



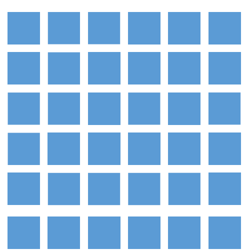
CVLs vs. FCLs

- CVLs: computation>> parameter
  - Computation hungry → High PE utilization
- FCLs: computation~=parameter
  - Memory bandwidth hungry → High BW utilization

CVLs vs. FCLs

- CVLs: computation>> parameter
  - Computation hungry → High PE utilization
- FCLs: computation~=parameter
  - Memory bandwidth hungry → High BW utilization

Processing Unit
(Monolithic
Architectures)



Processing
CVLs

PE utilization

high

BW utilization

low

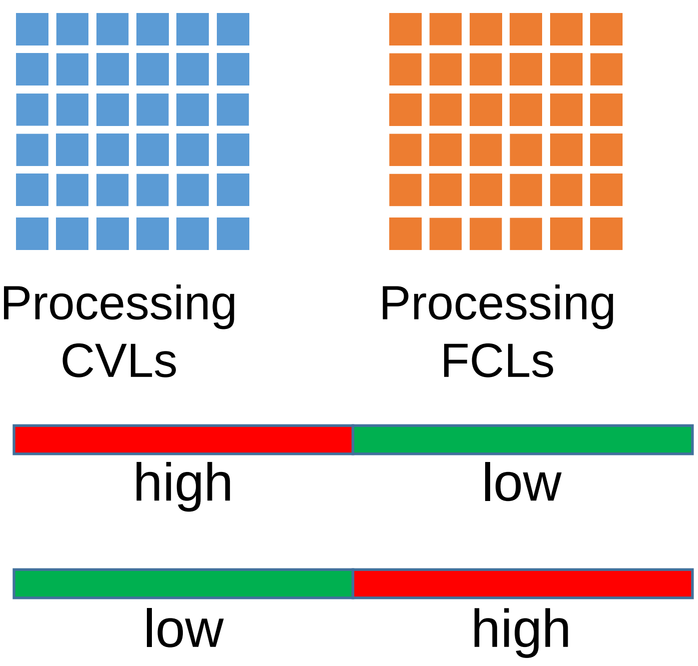
CVLs vs. FCLs

- CVLs: computation>> parameter
  - Computation hungry → High PE utilization
- FCLs: computation~=parameter
  - Memory bandwidth hungry → High BW utilization

Processing Unit (Monolithic Architectures)

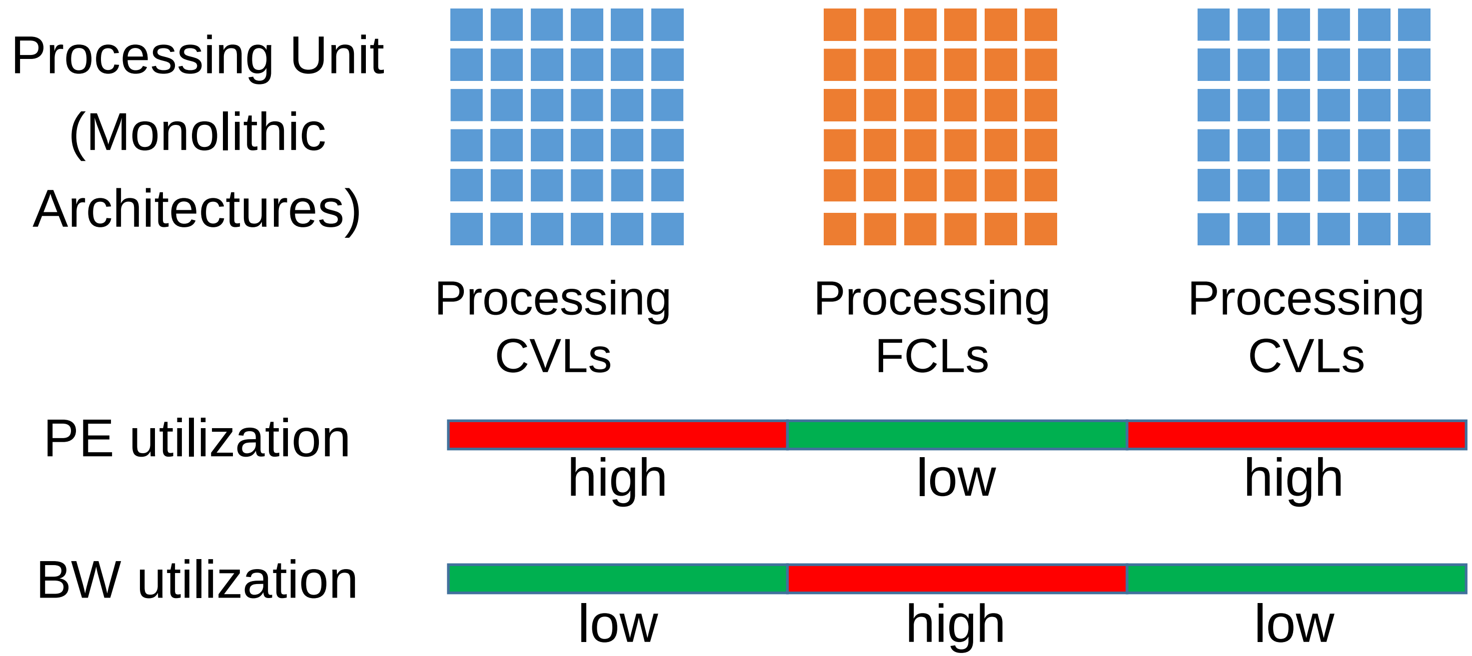
PE utilization

BW utilization



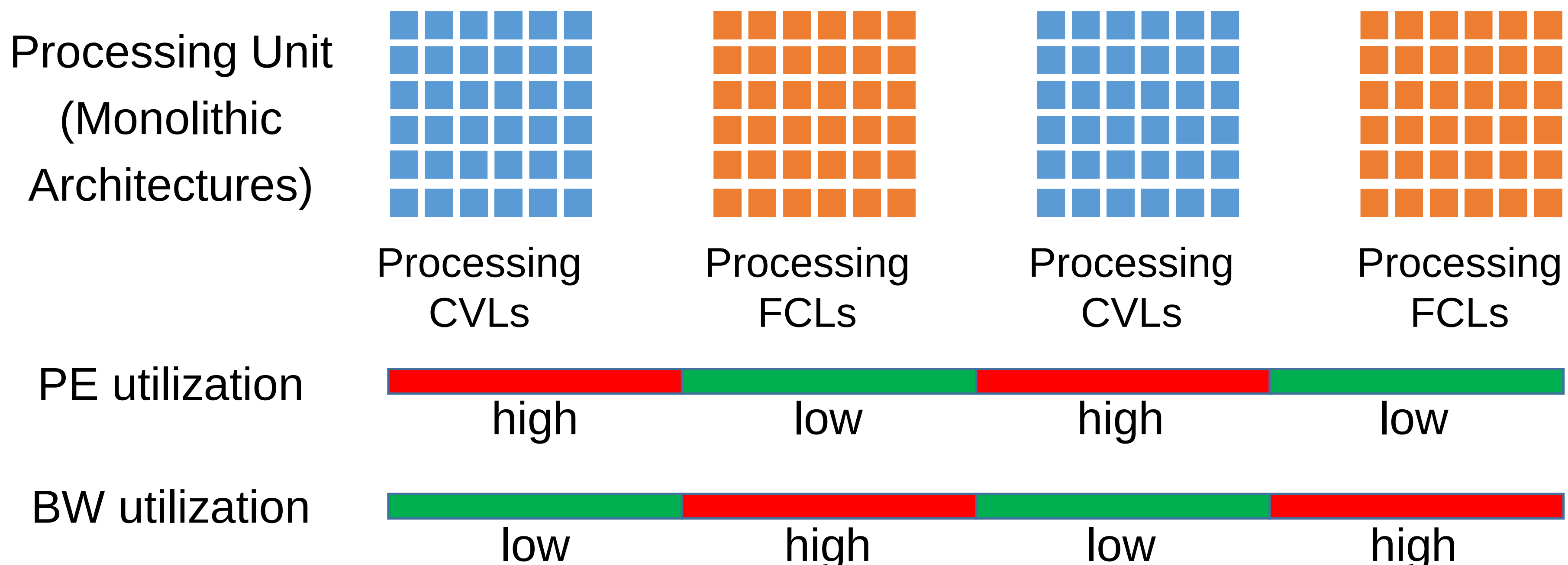
CVLs vs. FCLs

- CVLs: computation>> parameter
  - Computation hungry → High PE utilization
- FCLs: computation~=parameter
  - Memory bandwidth hungry → High BW utilization



CVLs vs. FCLs

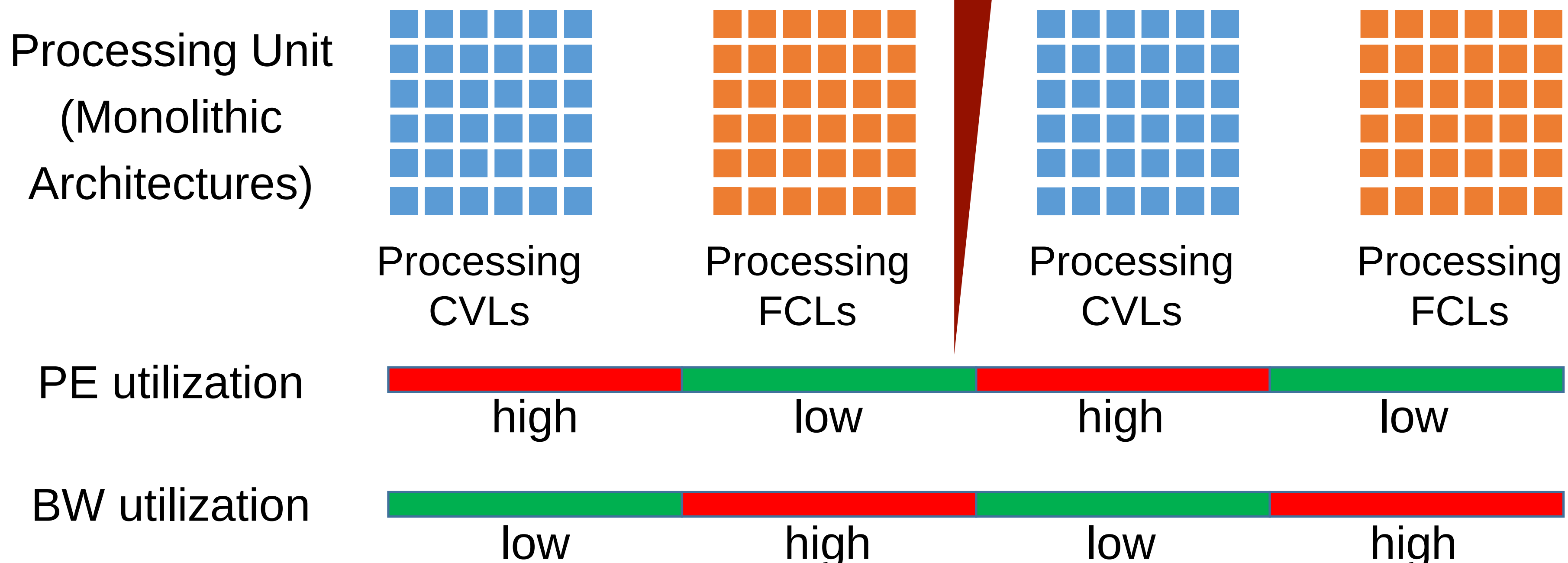
- CVLs: computation>> parameter
  - Computation hungry → High PE utilization
- FCLs: computation~=parameter
  - Memory bandwidth hungry → High BW utilization



CVLs vs. FCLs

- CVLs: computation>> parameter
  - Computation hungry
- FCLs: computation
  - Memory bandwidth h

Either PEs or memory bandwidth will be underutilized, thereby causing performance degradation



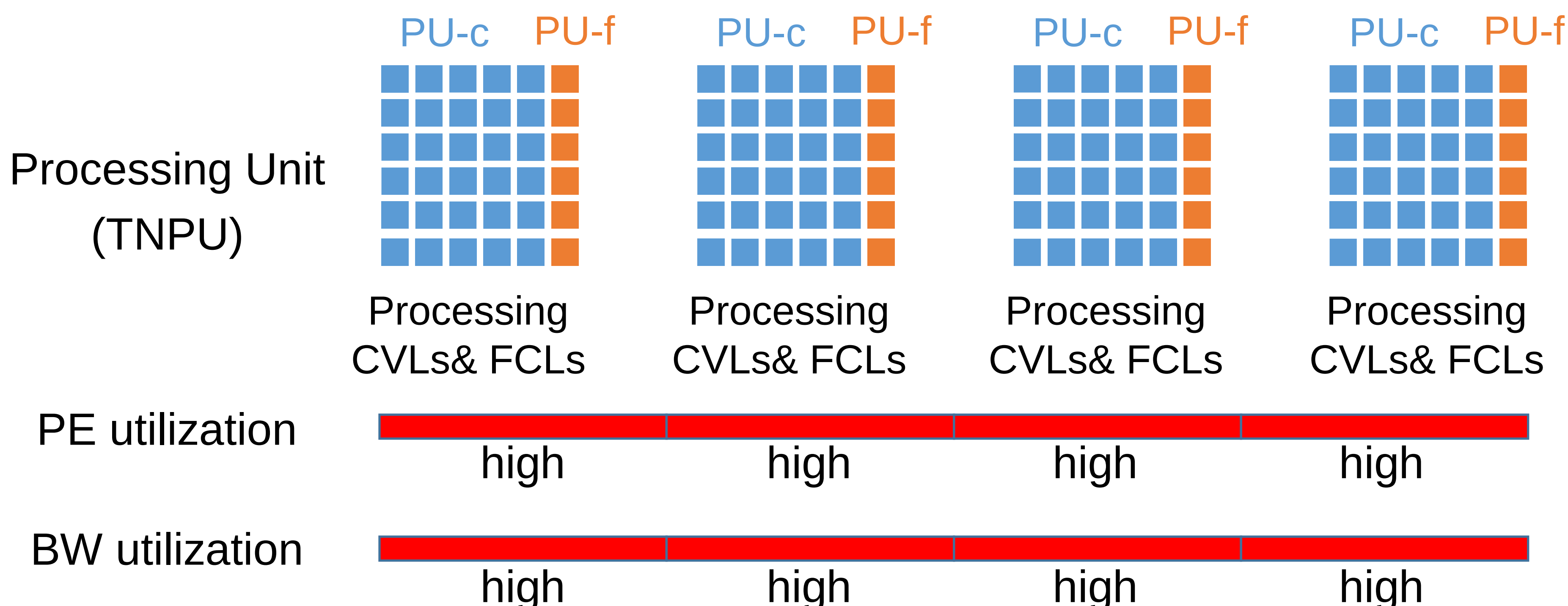
Complementary effects

- Resource requirement diversity
  - CVLs: more PE, less BW
  - FCLs: more BW, less PE

- Orchestrate CVLs and FCLs to be processed concurrently
  - Underutilized PEs in FCLs can be leveraged by CVLs
  - Underutilized BW in CVLs can be leveraged by FCLs

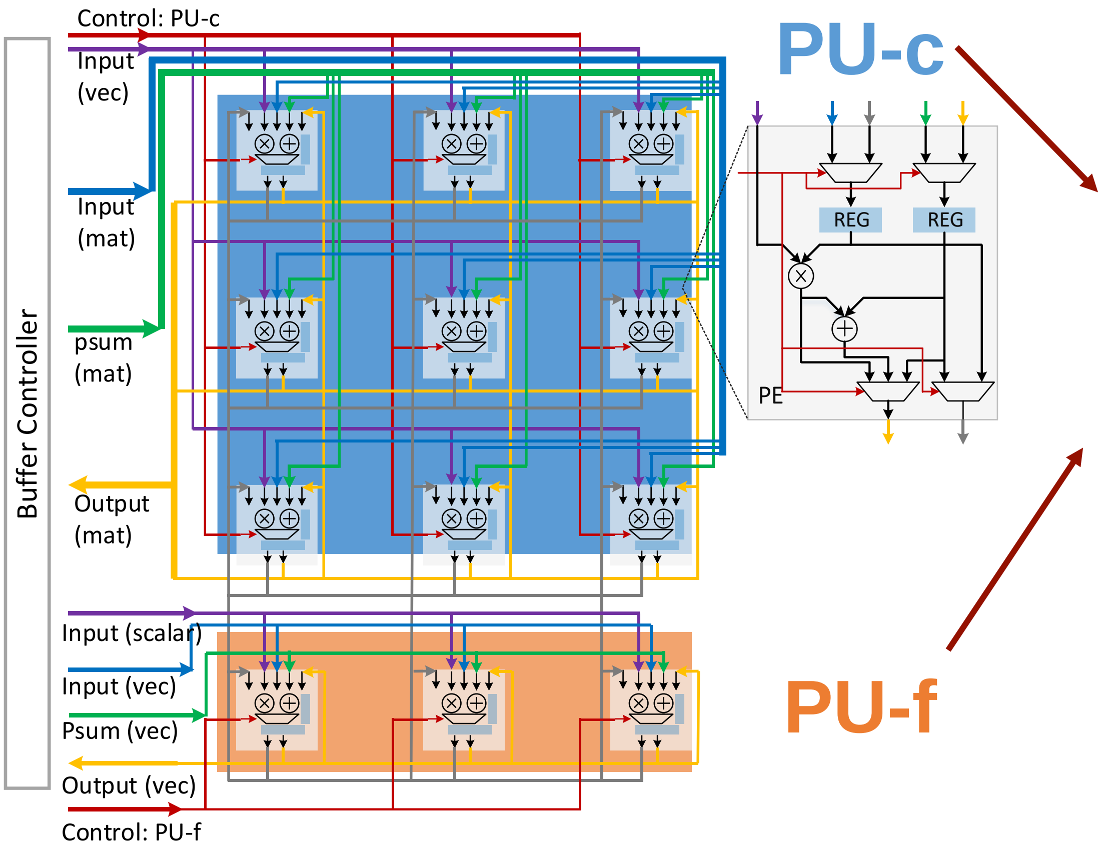
Asymmetric tandem architecture

- PU: Space division multiplexing
  - Exploits complementary effect of resource requirement between CVLs and FCLs to boost utilization



Asymmetric tandem architecture

- PE micro-architecture
  - Virtualize PU-c and PU-f from a 2D mesh PE array
  - PEs connect with each other via a NoC

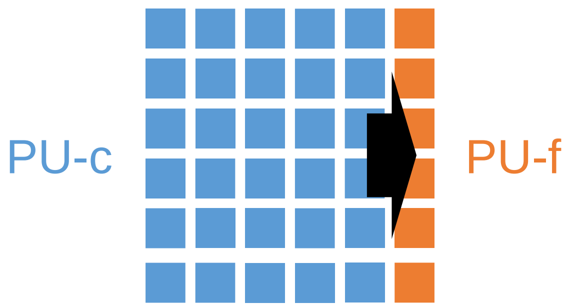


PU-c and PU-f work as two-stage pipeline because of data dependency between CVLs and FCLs

Bidirectional dataflow

- Data dependency: CVL→FCL→CVL

Forward propagation



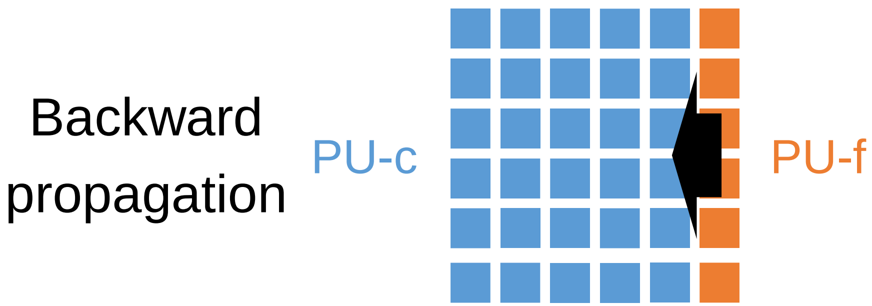
Data hazard
Structural hazard
Hazard free

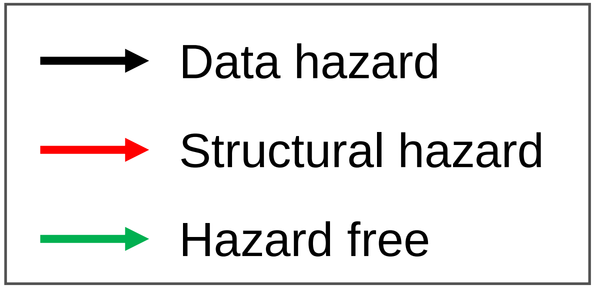
Cycle	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0-FP	**0-FP-C**	**0-FP-F**									
0-BP											
0-GD											
1-FP											
1-BP											
1-GD											
2-FP											
States	**Idle**	**Idle**									

FP-forward propagation, BP-backward propagation, GD-gradient descent

Bidirectional dataflow

- Data dependency: CVL→FCL→CVL





Cycle	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0-FP	0-FP-C	0-FP-F									
0-BP			0-BP-F	0-BP-C							
0-GD											
1-FP											
1-BP											
1-GD											
2-FP											
States	Idle	Idle	Idle	Idle							

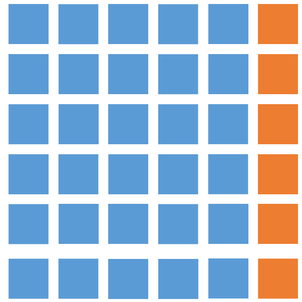
FP-forward propagation, BP-backward propagation, GD-gradient descent

Bidirectional dataflow

- Data dependency: CVL→FCL→CVL

Gradient descent

PU-c



PU-f

Data hazard

Structural hazard

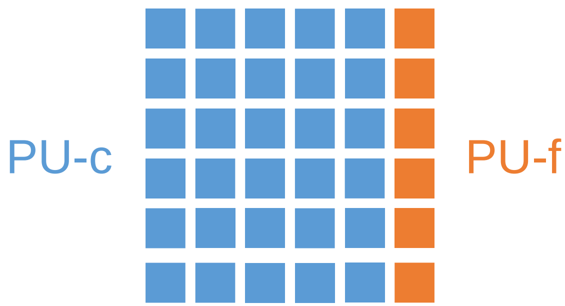
Hazard free

Cycle	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0-FP	0-FP-C	0-FP-F									
0-BP			0-BP-F	0-BP-C							
0-GD				0-GD-F	0-GD-C						
1-FP											
1-BP											
1-GD											
2-FP											
States	Idle	Idle	Idle	Busy	Idle						

FP-forward propagation, BP-backward propagation, GD-gradient descent

Bidirectional dataflow

- Data dependency: CVL→FCL→CVL



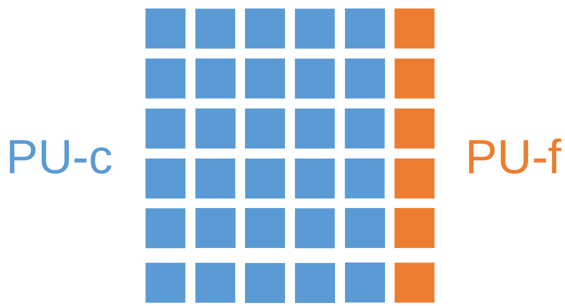
Data hazard
Structural hazard
Hazard free

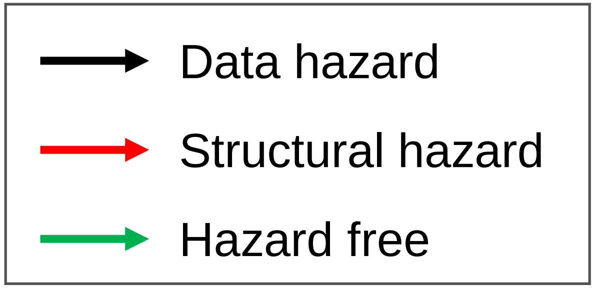
Cycle	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0-FP	0-FP-C	0-FP-F									
0-BP			0-BP-F	0-BP-C							
0-GD				0-GD-F	0-GD-C						
1-FP						1-FP-C	1-FP-F				
1-BP								1-BP-F	1-BP-C		
1-GD									1-GD-F	1-GD-C	
2-FP											2-FP-C
States	Idle	Idle	Idle	Busy	Idle	Idle	Idle	Idle	Busy	Idle	Idle

FP-forward propagation, BP-backward propagation, GD-gradient descent

Bidirectional dataflow

- Data dependency: CVL→FCL→CVL





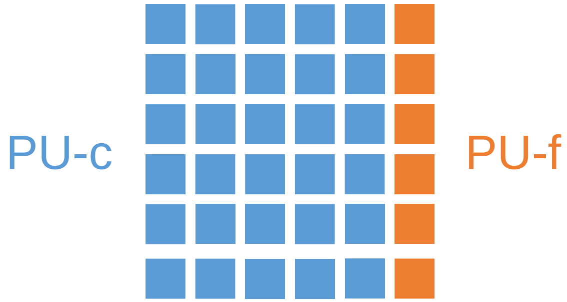
Cycle	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0-FP	0-FP-C	0-FP-F									
0-BP			0-BP-F	0-BP-C							
0-GD				0-GD-F	0-GD-C						
1-FP						1-FP-C	1-FP-F				
1-BP								1-BP-F	1-BP-C		
									1-GD-F	1-GD-C	
2-FP											2-FP-C
States	Idle	Idle	Idle	Busy	Idle	Idle	Idle	Idle	Busy	Idle	Idle

Many pipeline stalls if in-order executing

FP-forward propagation, BP-backward propagation, GD-gradient descent

Out-of-order scheduling

- Forwarding to eliminate hazard



Data hazard

Structural hazard

Hazard free

Cycle	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0-FP	0-FP-C	0-FP-F									
0-BP			0-BP-F	0-BP-C							
0-GD				0-GD-F	0-GD-C						
1-FP						1-FP-C	1-FP-F				
1-BP								1-BP-F	1-BP-C		
1-GD									1-GD-F	1-GD-C	
2-FP											2-FP-C
States	Idle	Idle	Idle	Busy	Idle	Idle	Idle	Idle	Busy	Idle	Idle

FP-forward propagation, BP-backward propagation, GD-gradient descent

Out-of-order scheduling

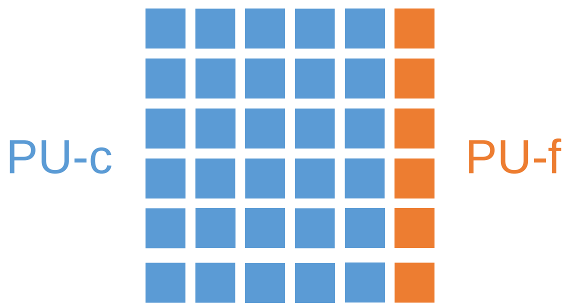
- Forwarding to eliminate hazard

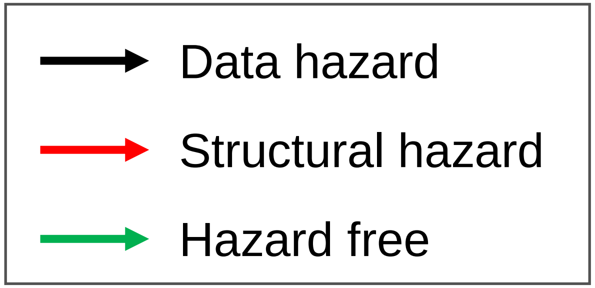
Cycle	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0-FP	0-FP-C	0-FP-F									
0-BP			0-BP-F	0-BP-C							
0-GD				0-GD-F	0-GD-C						
1-FP		1-FP-C					1-FP-F				
1-BP								1-BP-F	1-BP-C		
1-GD									1-GD-F	1-GD-C	
2-FP											2-FP-C
States	Idle	Busy	Idle	Busy	Idle		Idle	Idle	Busy	Idle	Idle

FP-forward propagation, BP-backward propagation, GD-gradient descent

Out-of-order scheduling

- Forwarding to eliminate hazard



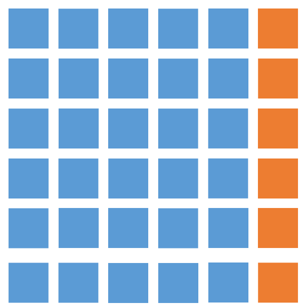


Cycle	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0-FP	0-FP-C	0-FP-F									
0-BP			0-BP-F	0-BP-C							
0-GD				0-GD-F	0-GD-C						
1-FP		1-FP-C					1-FP-F				
1-BP								1-BP-F	1-BP-C		
1-GD									1-GD-F	1-GD-C	
2-FP											2-FP-C
States	Idle	Busy	Idle	Busy	Idle		Idle	Idle	Busy	Idle	Idle

FP-forward propagation, BP-backward propagation, GD-gradient descent

Out-of-order scheduling

- Forwarding to eliminate hazard



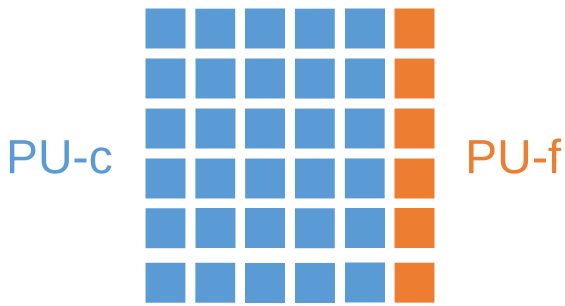
Data hazard
Structural hazard
Hazard free

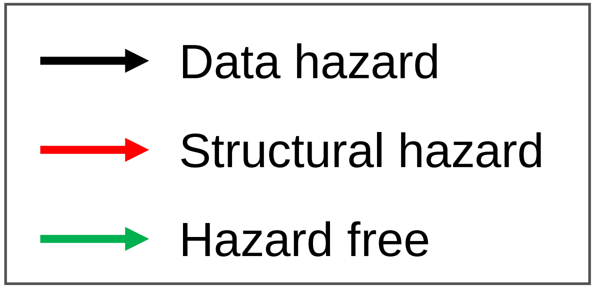
Cycle	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0-FP	0-FP-C	0-FP-F									
0-BP			0-BP-F	0-BP-C							
0-GD				0-GD-F	0-GD-C						
1-FP		1-FP-C			1-FP-F						
1-BP								1-BP-F	1-BP-C		
1-GD									1-GD-F	1-GD-C	
2-FP											2-FP-C
States	Idle	Busy	Idle	Busy	Busy			Idle	Busy	Idle	Idle

FP-forward propagation, BP-backward propagation, GD-gradient descent

Out-of-order scheduling

- Forwarding to eliminate hazard



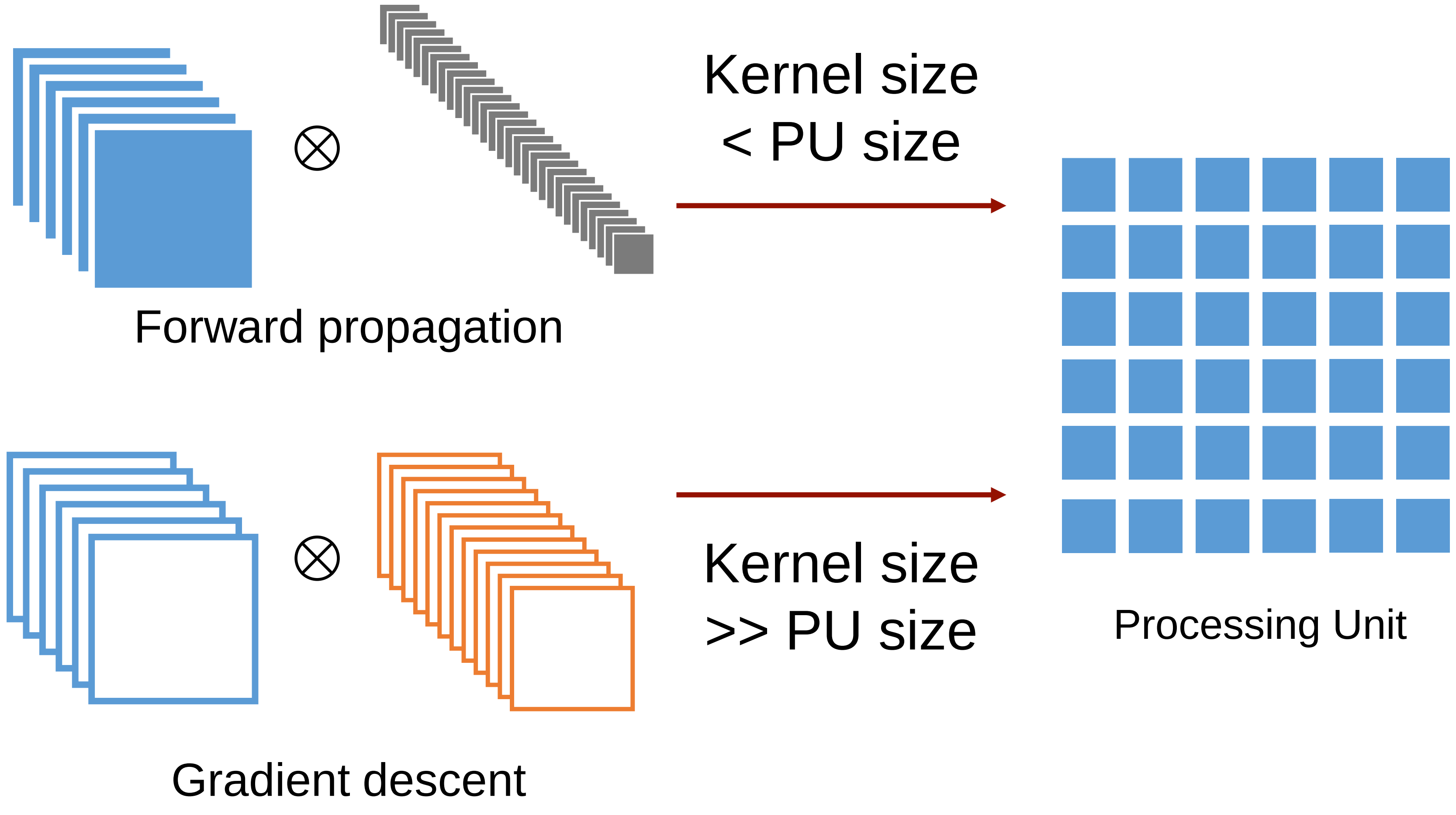


Cycle	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
0-FP	0-FP-C	0-FP-F									
0-BP			0-BP-F	0-BP-C							
0-GD				0-GD-F	0-GD-C						
1-FP		1-FP-C			1-FP-F	1-FP-C					
1-BP						1-BP-F	1-BP-C				
1-GD							1-GD-F	1-GD-C			
2-FP			2-FP-C								
States	Idle	Busy	Busy	Busy	Busy	Busy	Busy				

FP-forward propagation, BP-backward propagation, GD-gradient descent

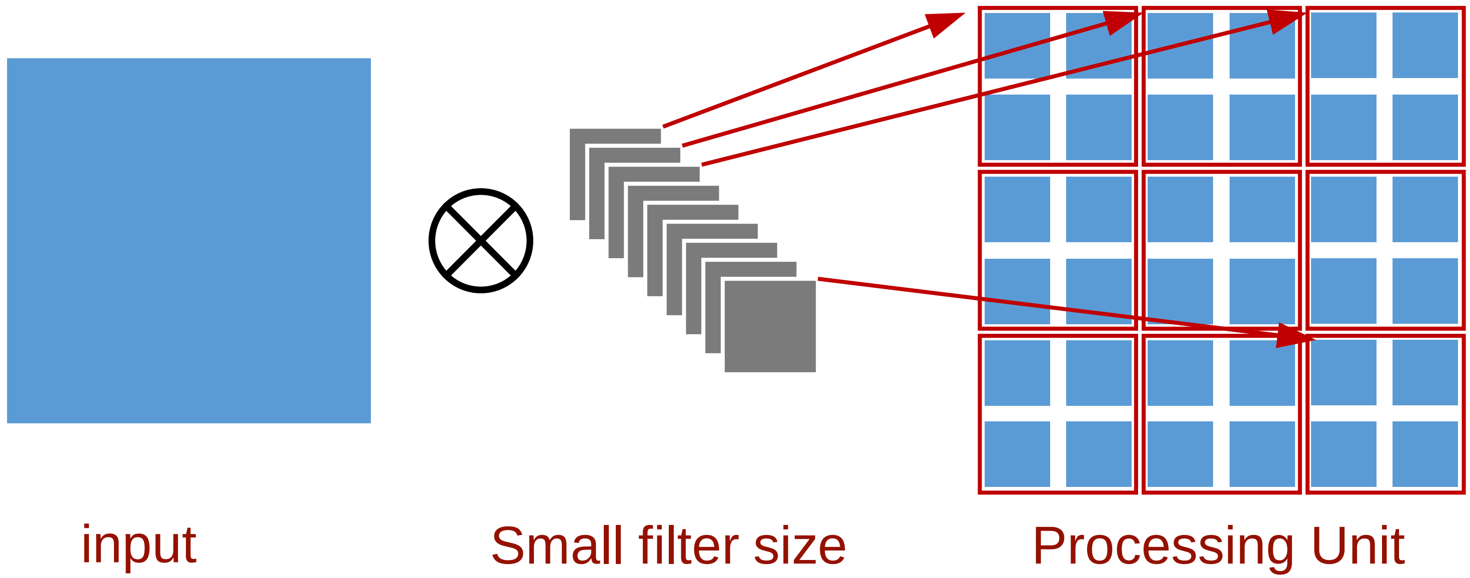
Wide-range of convolution kernels

- Flexible convolution allocation



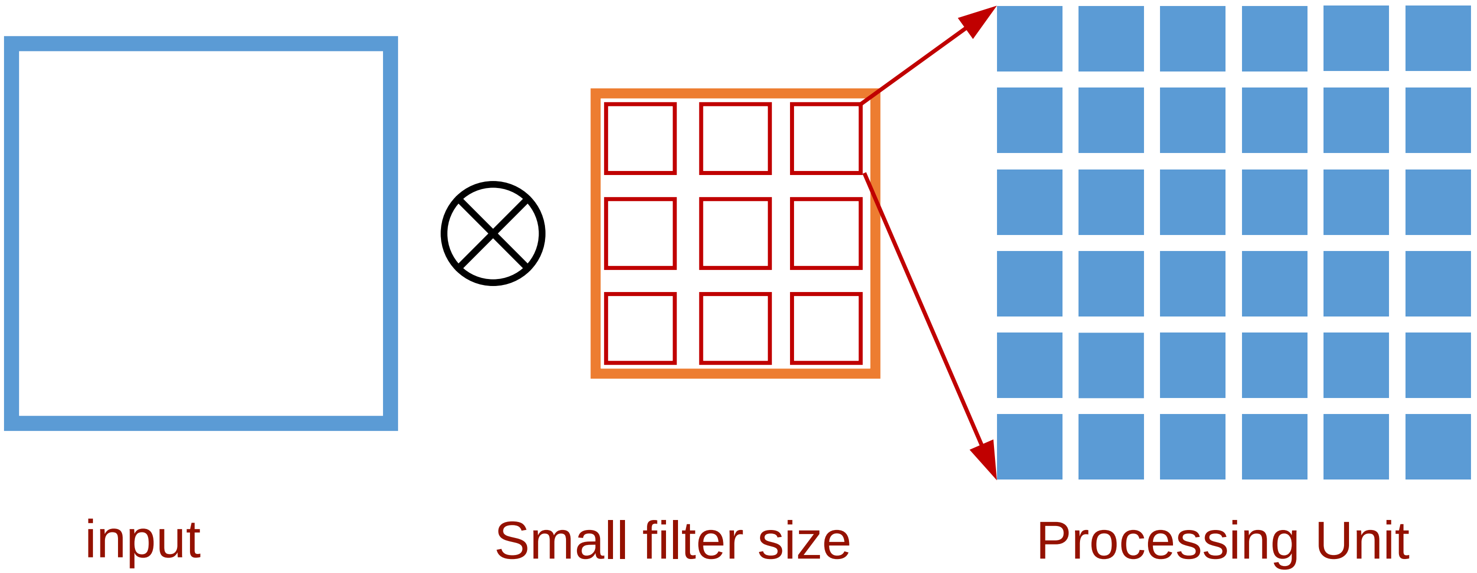
Flexible convolution allocation

- Kernel size< PU size: kernel aggregation
  - Aggregate multiple kernels to generate a big one, then map to Processing Unit



Flexible convolution allocation

- Kernel size> PU size: kernel partitioning
  - Partition large kernels into small sub-kernels [DATE'2018]



[1] Li J et al. Ccr: a concise convolution rule for sparse neural network accelerators..

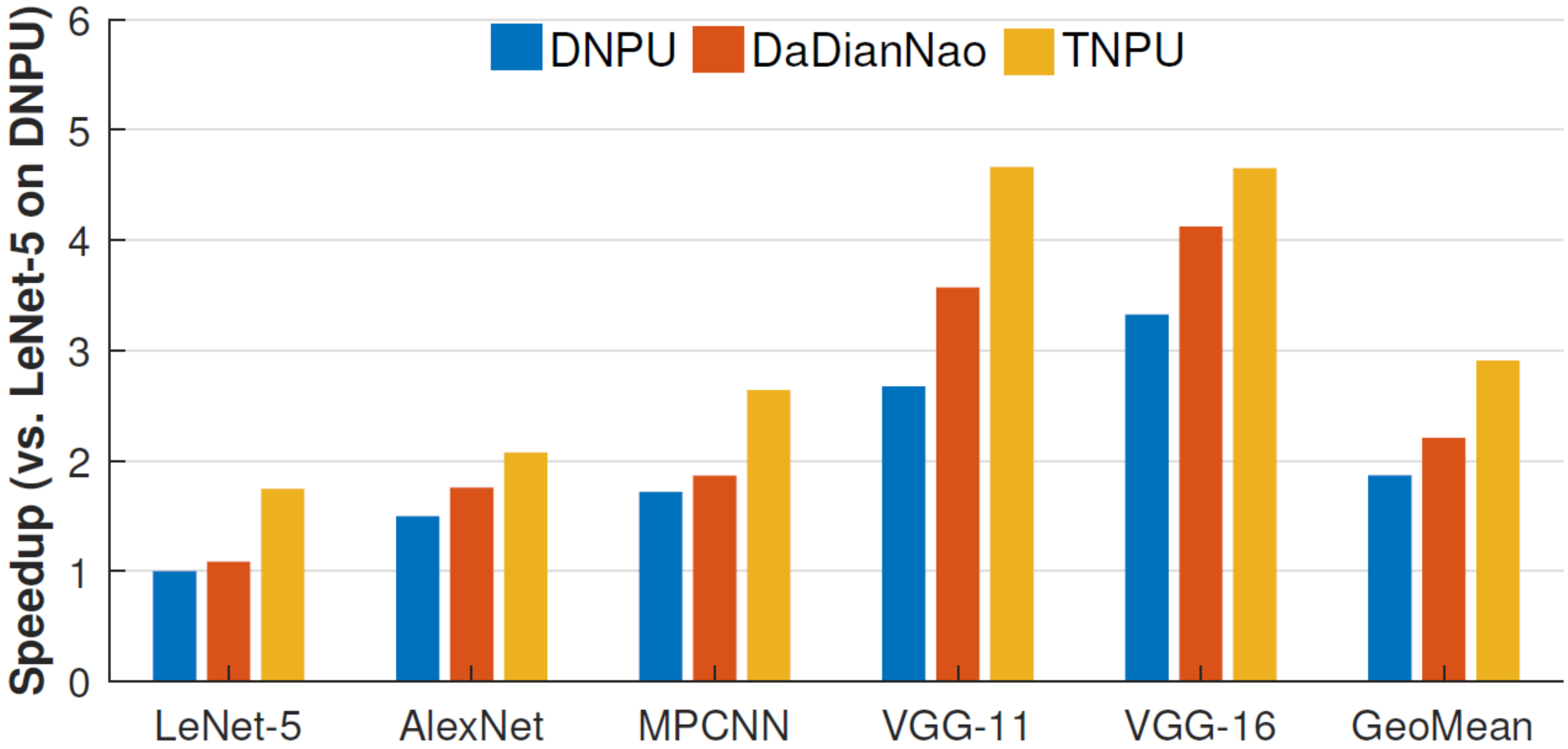
Experimental setup

- Evaluation platform
  - RTL-level cycle-accurate simulation for performance estimation
  - System-level energy estimation, based on synthesis and CACTI
- Platform configurations

	MACs	SRAM	Clock	DRAM bandwidth
TNPU	272	256KB	400MHz	17.1GB/s
Dadiannao-re	272	256KB	400MHz	17.1GB/s
DNPU-re	272	256KB	400MHz	17.1GB/s

Experimental results

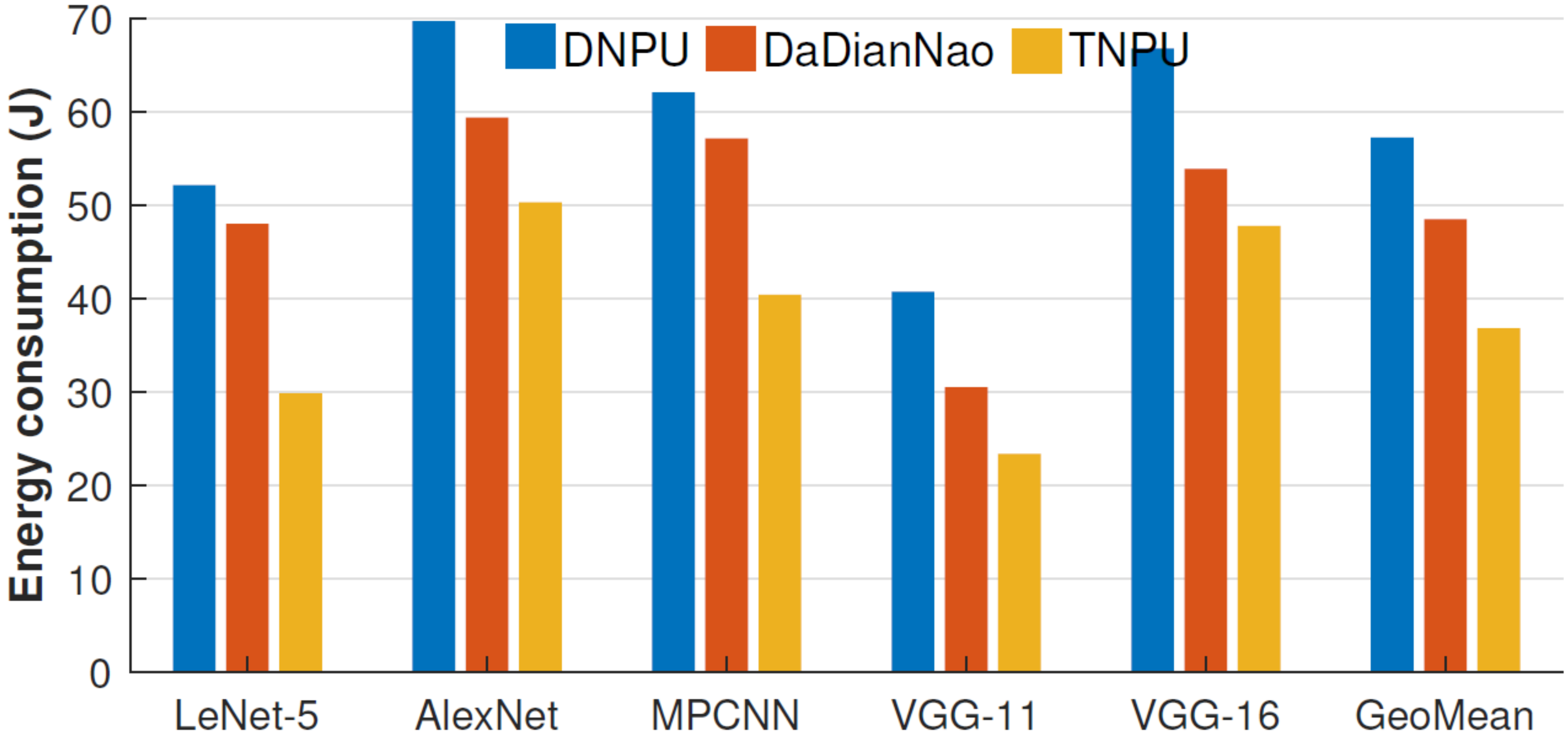
- Performance



55% performance improvement over DNPU
32% performance improvement over Dadiannao

Experimental results

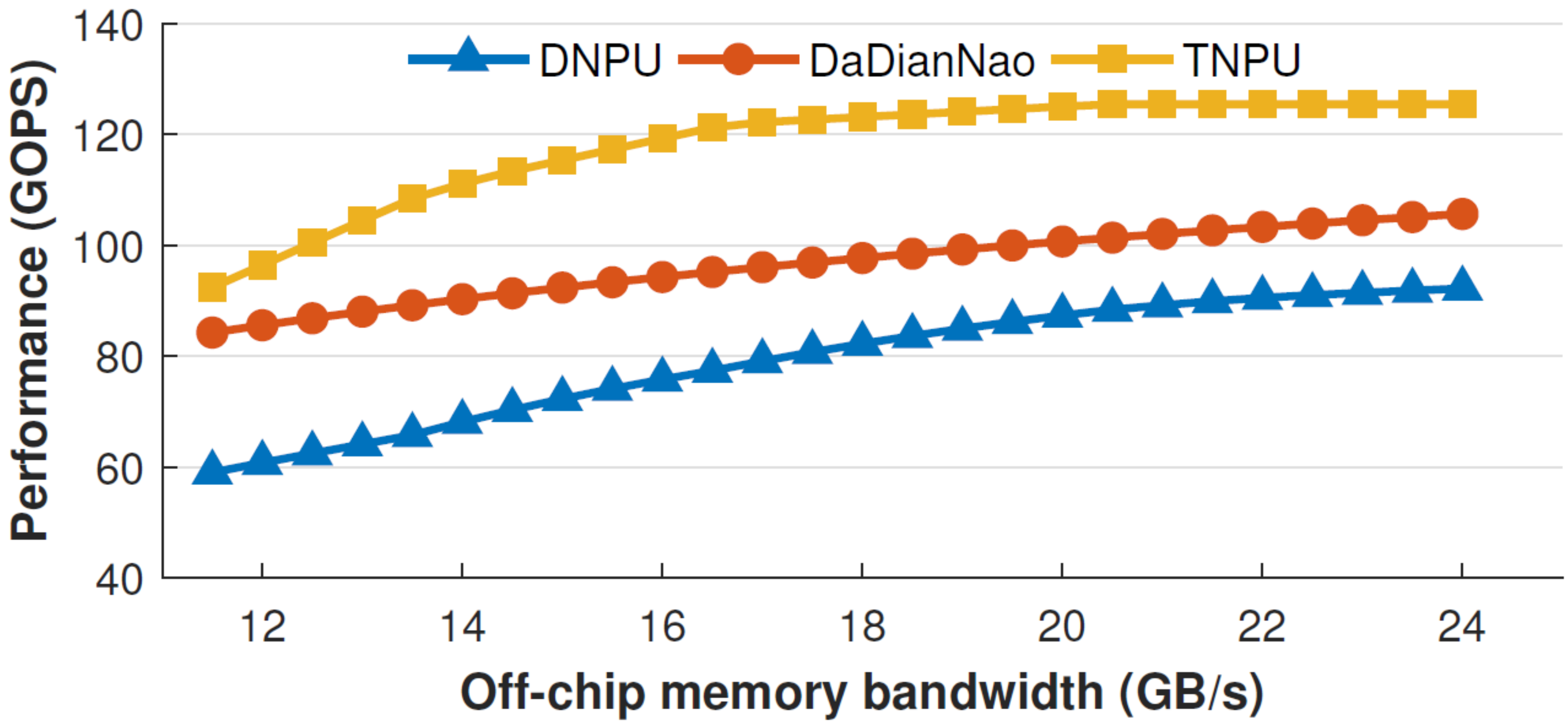
- Energy consumption



35% energy saving compared to DNPU
24% energy saving compared to Dadiannao

Experimental results

- Scalability to DRAM bandwidth



TNPU maintains a stable high performance and consistently outperforms DNPU and Dadiannao

Conclusion

- TNPU, an accelerator architecture for CNN training:
  - 55% and 32% performance improvement over DNPU and Dadiannao, respectively

- TNPU addresses the following challenges in CNN training:
  - Diversity between CVLs and FCLs
  - Bidirectional data dependency
  - Extremely large convolutional kernels

Thanks for your attention!

Email: lijiajun@ict.ac.cn