SOGANG UNIVERSITY

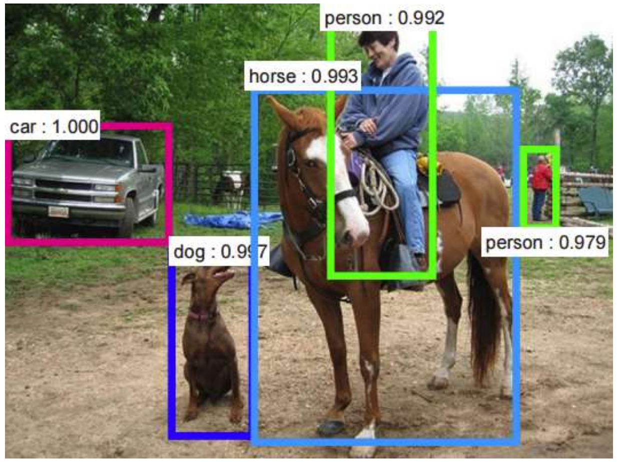
Optimizing FPGA-based Convolutional Neural Networks Accelerator for Image Super-Resolution

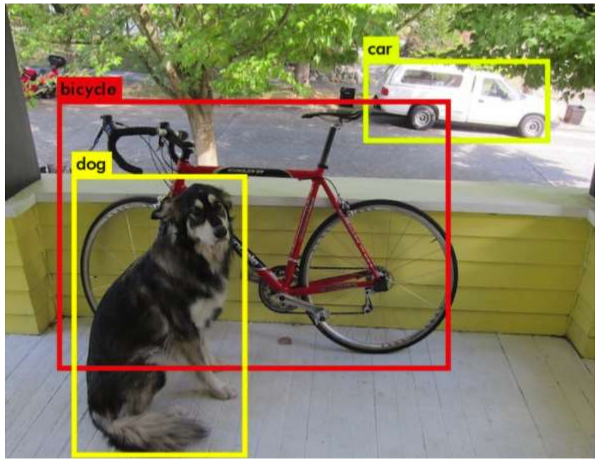
Jung-Woo Chang and Suk-Ju Kang
Sogang University, Seoul, South Korea

Outline

Introduction

- Deep Learning
  - Convolutional neural networks (CNN) can solve the problems faced by existing machine learning algorithms in such as object recognition and natural language processing, etc.

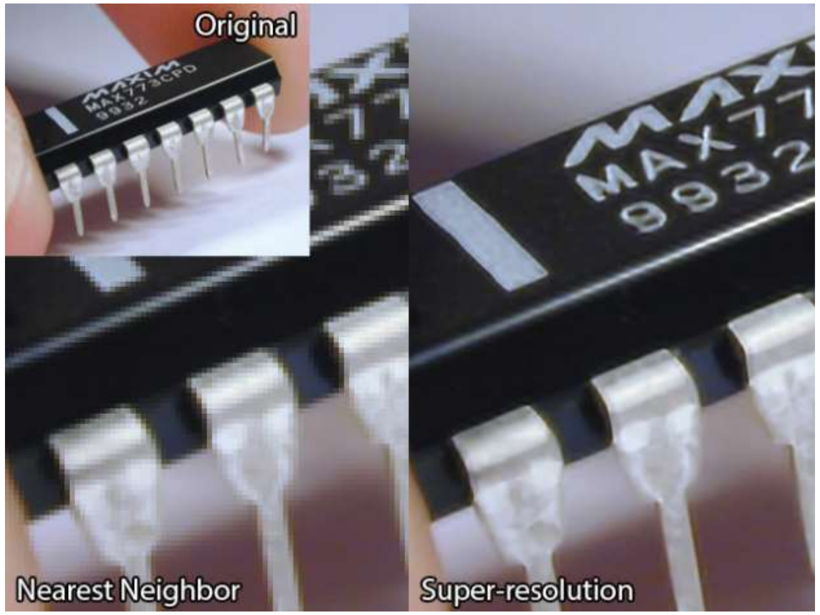




Two hockey players are fighting over the puck.

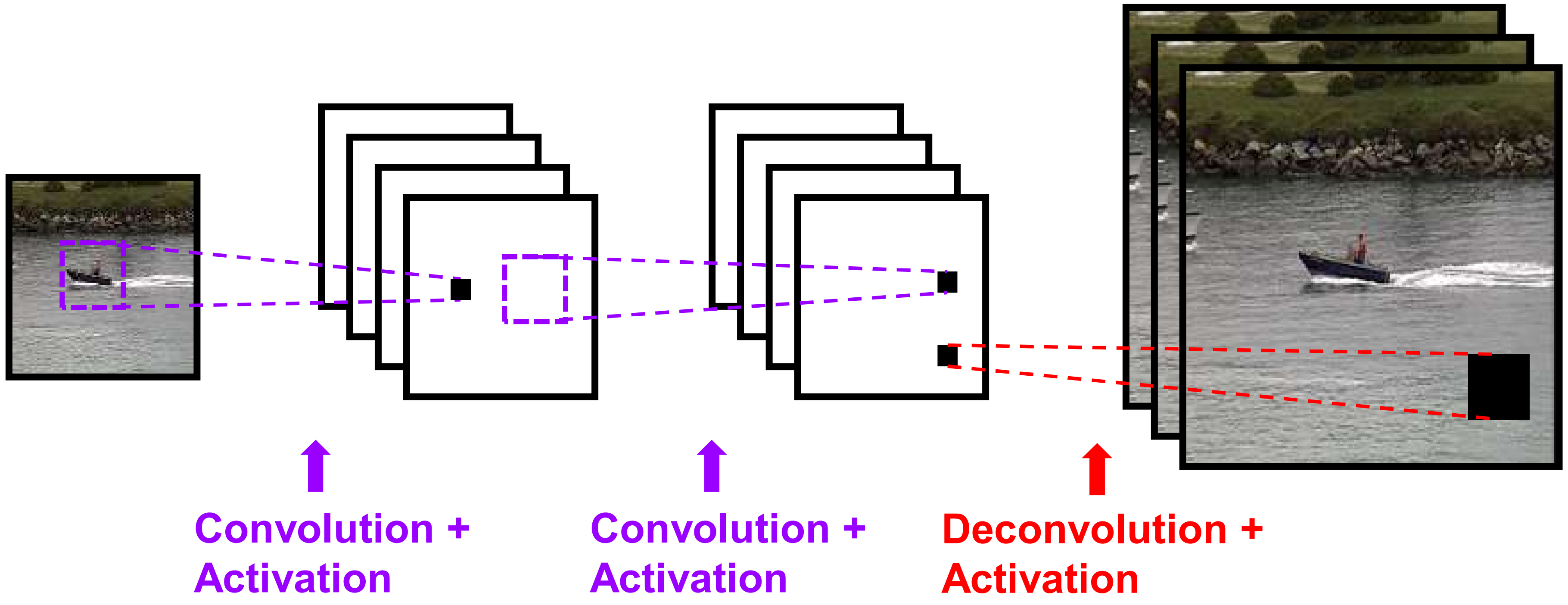
Introduction

- Deep Learning
  - ➢Recently, CNN have been used to improve image enhancement applications in such as image super-resolution (SR) and high dynamic range (HDR) imaging .



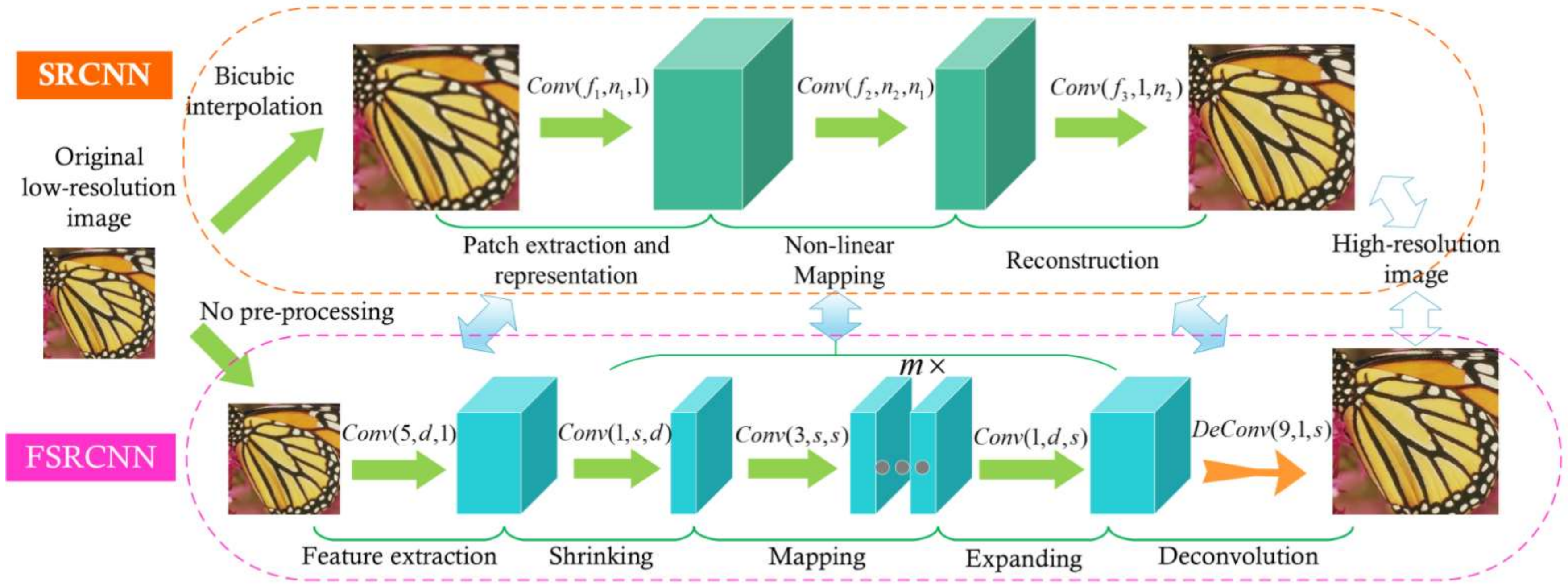
Introduction

- Deep Learning
  - Especially, deconvolutional neural networks (DCNN) are mainly used to reconstruct target images in image enhancement applications.



Introduction

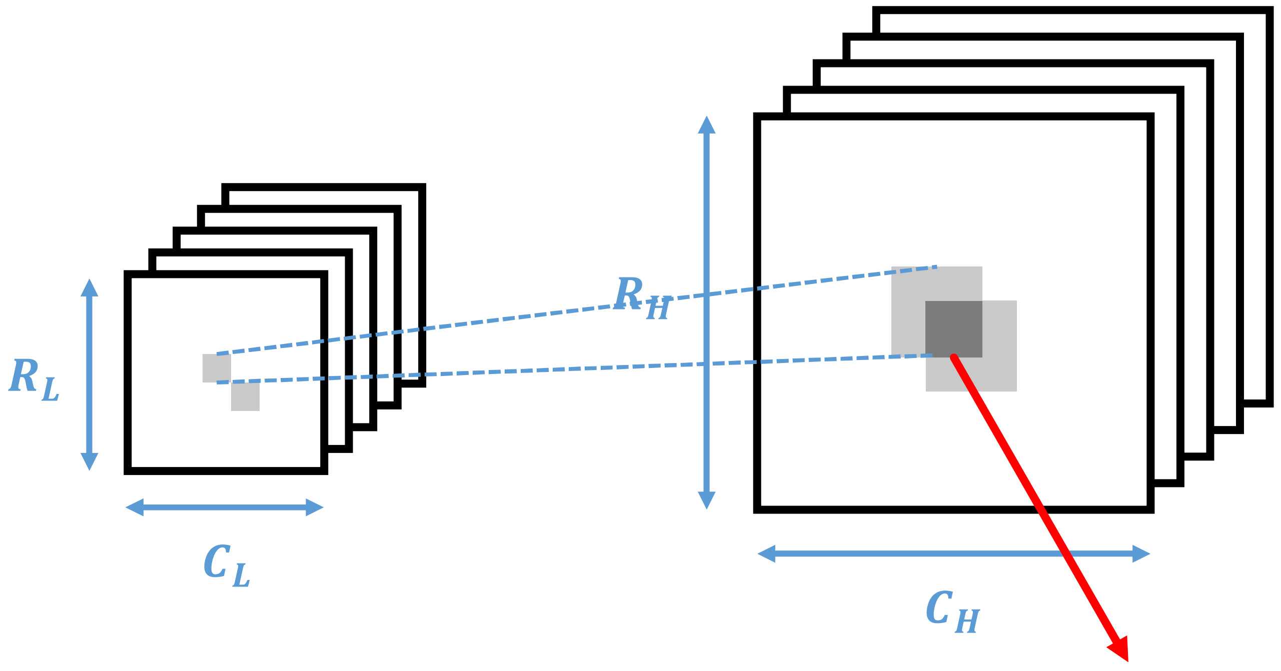
- Deep Learning
  - ➢FSRCNN [1]
    - Deep networks (7 CNN + 1 DCNN)
    - More cost effective than SRCNN, but it’s even better.



[1] C. Dong, et al. Accelerating the super-resolution convolutional neural network. In *Springer ECCV* 2016.

Introduction

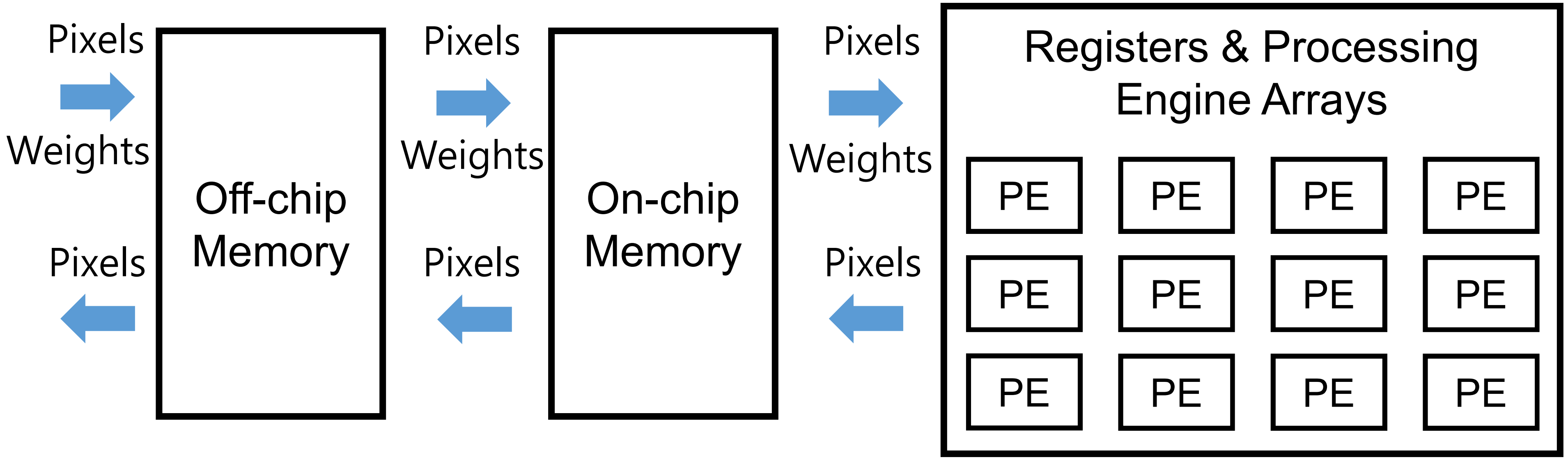
- Problems with DCNN
  1. **Overlapping sum problem**
  2. Loop dimensions



Need to sum these overlaps up

Introduction

- Problems with DCNN
  1. **Overlapping sum problem**
  2. Loop dimensions

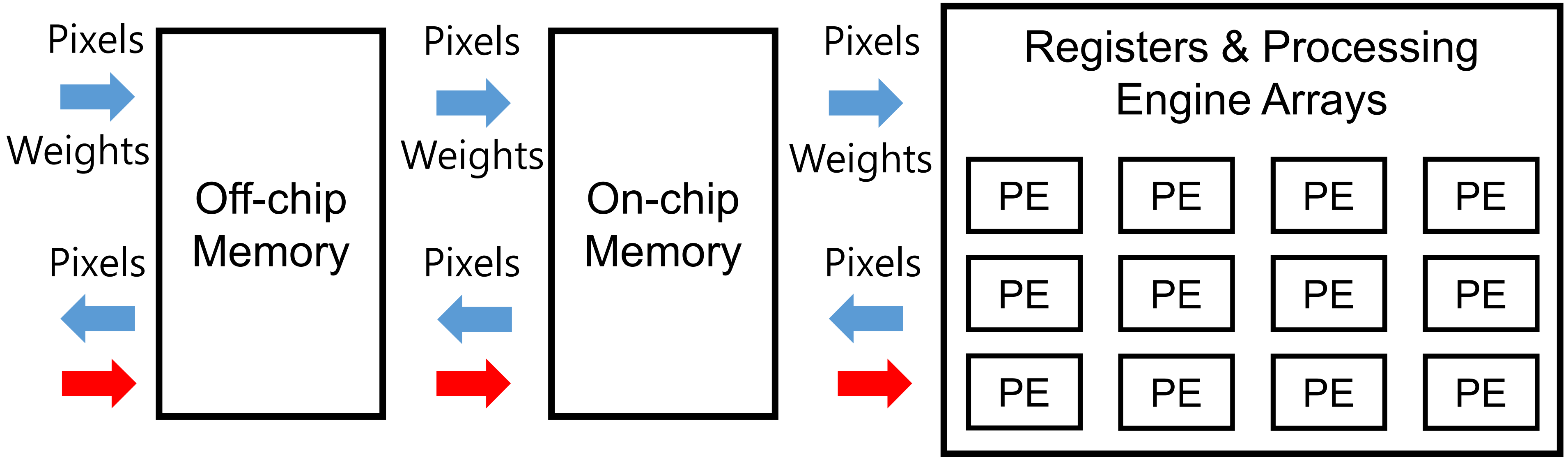


<General CNN Acceleration system>

Introduction

- Problems with DCNN
  1. **Overlapping sum problem**
  2. Loop dimensions

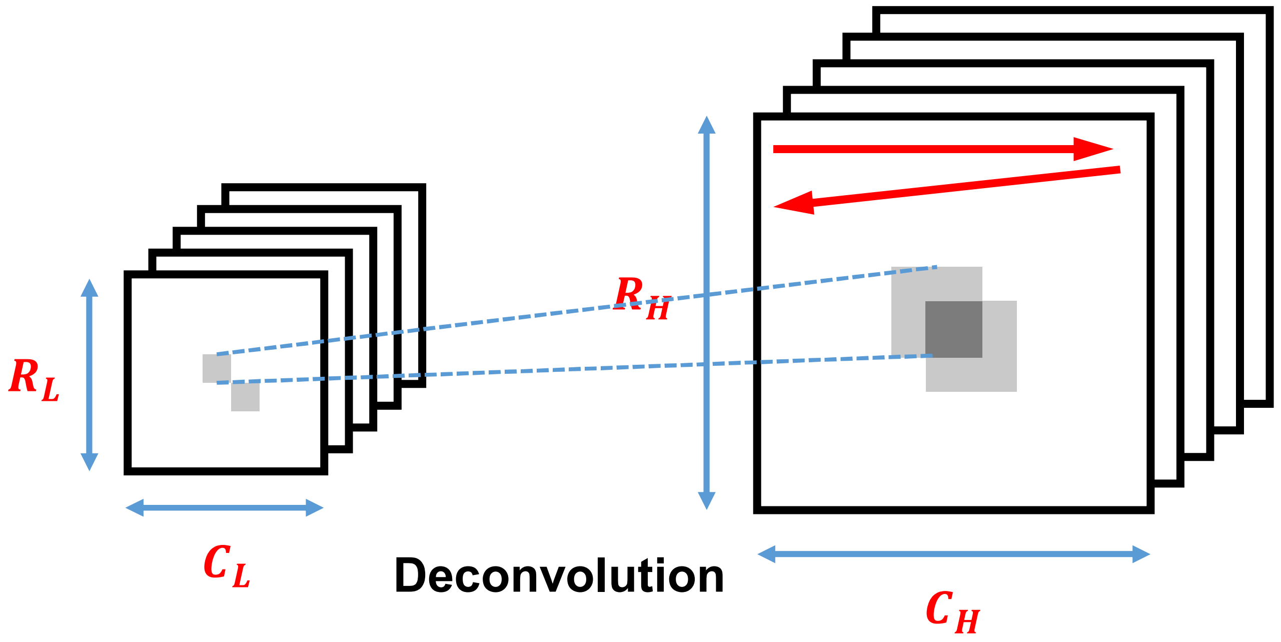
Increase latency, energy consumption and additional hardware resources !



<General CNN Acceleration system>

Introduction

- Problems with DCNN
  1. Overlapping sum problem
  2. **Loop dimensions**

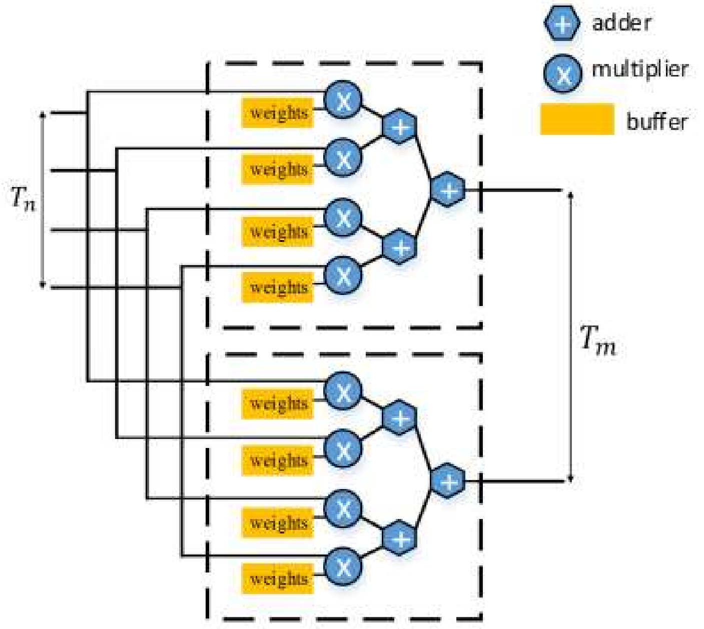


$$R_H = upscaling\_factor \times R_L$$
$$C_H = upscaling\_factor \times C_L$$

Background

- Loop Optimization Technique [2]
  - ➢Tile-parallel architecture
  - ➢Single convolutional layer processor (CLP) method
    - Resource underutilization problem

```
...
for ( ti_h = 0 ; ti_h < T_h ; ti_h ++)
 for ( ti_w = 0 ; ti_w < T_w ; ti_w ++)
#pragma HLS PIPELINE
  for ( t_m = 0 ; t_m < T_m ; t_m ++)
#pragma HLS UNROLL
   for ( t_n = 0 ; t_n < T_n ; t_n ++)
#pragma HLS UNROLL
     output_feature_map[ t_m ][ ti_h ][ ti_w ] +=
     Wc[ t_m ][ t_n ][ k_h ][ k_w ]×
     input_feature_map[ t_n ][ S × ti_h + k_h ][ S × i_w + k_w ]
...
```

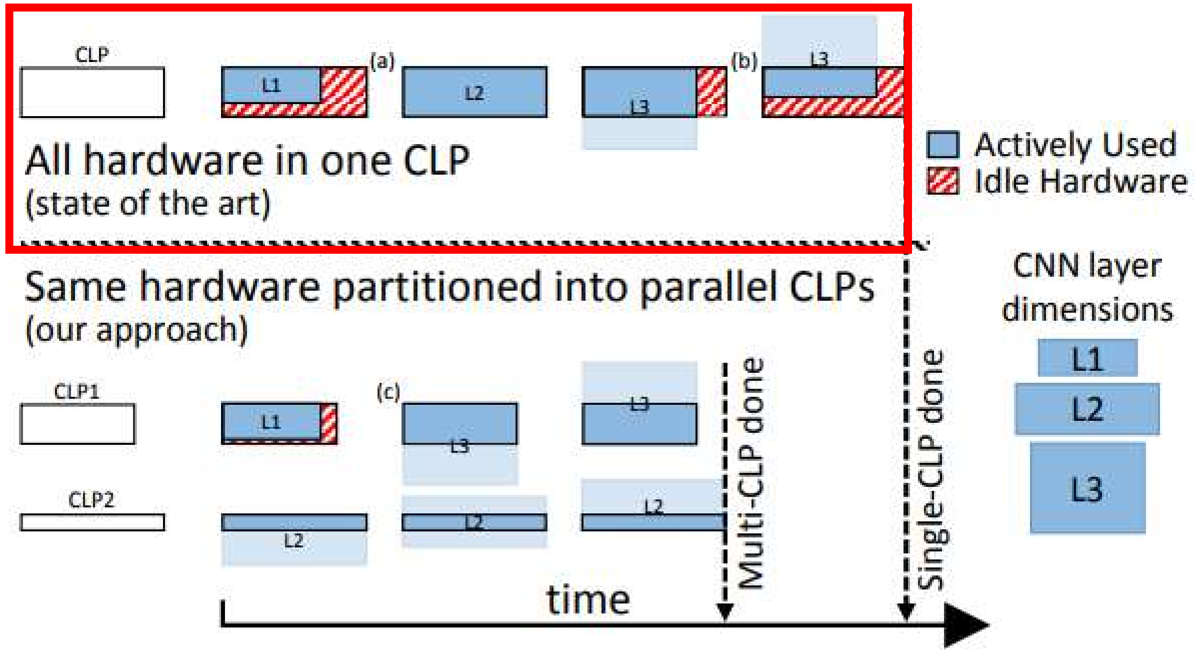


[2] C. Zhang, et al. Optimizing fpga-based accelerator design for deep convolutional neural networks. In ACM *FPGA* 2015.

Background

- Loop Optimization Technique [2]
  - ➢Tile-parallel architecture
  - ➢Single convolutional layer processor(CLP)
    - Resource underutilization problem

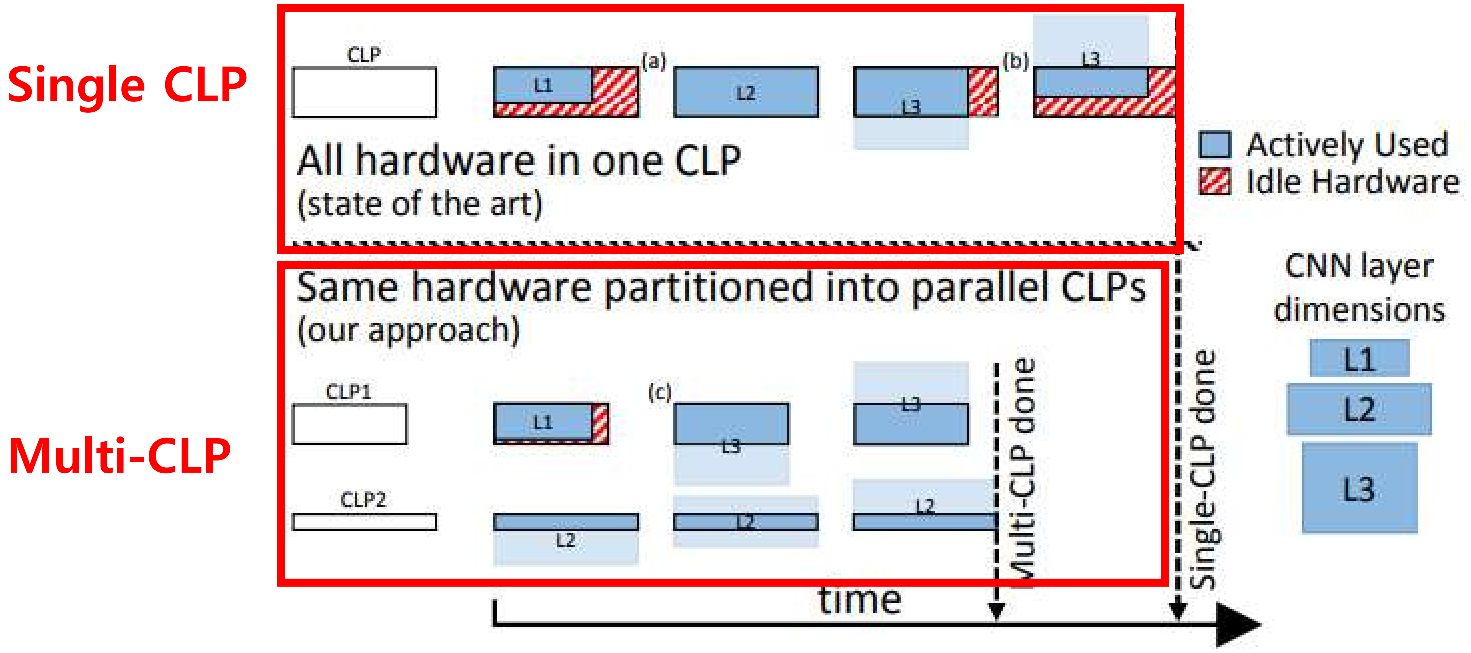
Single CLP



[2] C. Zhang, et al. Optimizing fpga-based accelerator design for deep convolutional neural networks. In ACM *FPGA* 2015.

Background

- Resource Partitioning [3]
  - ➢Tile-parallel architecture
  - ➢Multi-CLP method
    - Overcoming resource underutilization problem



[3] Y. Shen , et al. Maximizing CNN accelerator efficiency through resource partitioning. In ACM *ISCA* 2017.

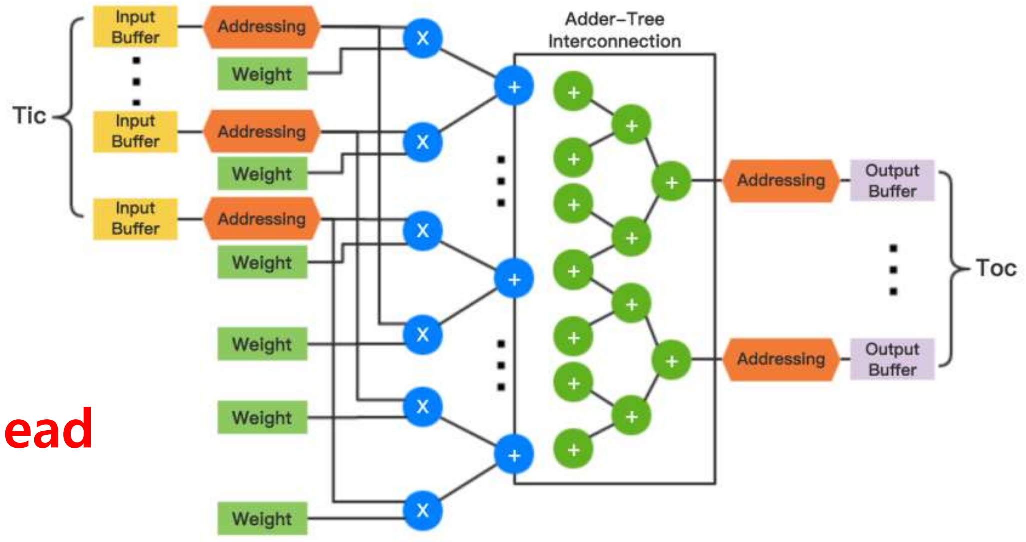
Background

- DCNN Accelerator [4]
  - Tile-based architecture
  - Reverse looping method
    - Overhead (extra operations, loop dimension)

Algorithm 2 Our FPGA Implementation of Deconvolution

```
procedure ReverseDeconvolution
    for $k_h = 0$ to $K - 1$ do
        for $k_w = 0$ to $K - 1$ do
            for $o'_h = 0$ to $\frac{T_{O_H}}{S} - 1$ do
                for $o'_w = 0$ to $\frac{T_{O_W}}{S} - 1$ do    ▷ loop $T_{O_W}$
                    for $o_c = 0$ to $T_{O_C} - 1$ do    ▷ loop $T_{O_C}$
                        for $i_c = 0$ to $T_{I_C} - 1$ do    ▷ loop $T_{I_C}$
                            COMPUTE($k_h, k_w, o'_h, o'_w, o_c, i_c$)
procedure Compute($k_h, k_w, o'_h, o'_w, o_c, i_c$)
    $f_h \leftarrow (S - ((P - k_h) \bmod S)) \bmod S$
    $f_w \leftarrow (S - ((P - k_w) \bmod S)) \bmod S$
    $o_h = o'_h \times S + P + f_h$
    $o_w = o'_w \times S + P + f_w$
    $i_h \leftarrow (o_h - k_h)/S$
    $i_w \leftarrow (o_w - k_w)/S$
    out[$o_c$][$o_h$][$o_w$] $\leftarrow$ in[$i_c$][$i_h$][$i_w$] $\times$ kernel[$o_c$][$i_c$][$k_h$][$k_w$]
```

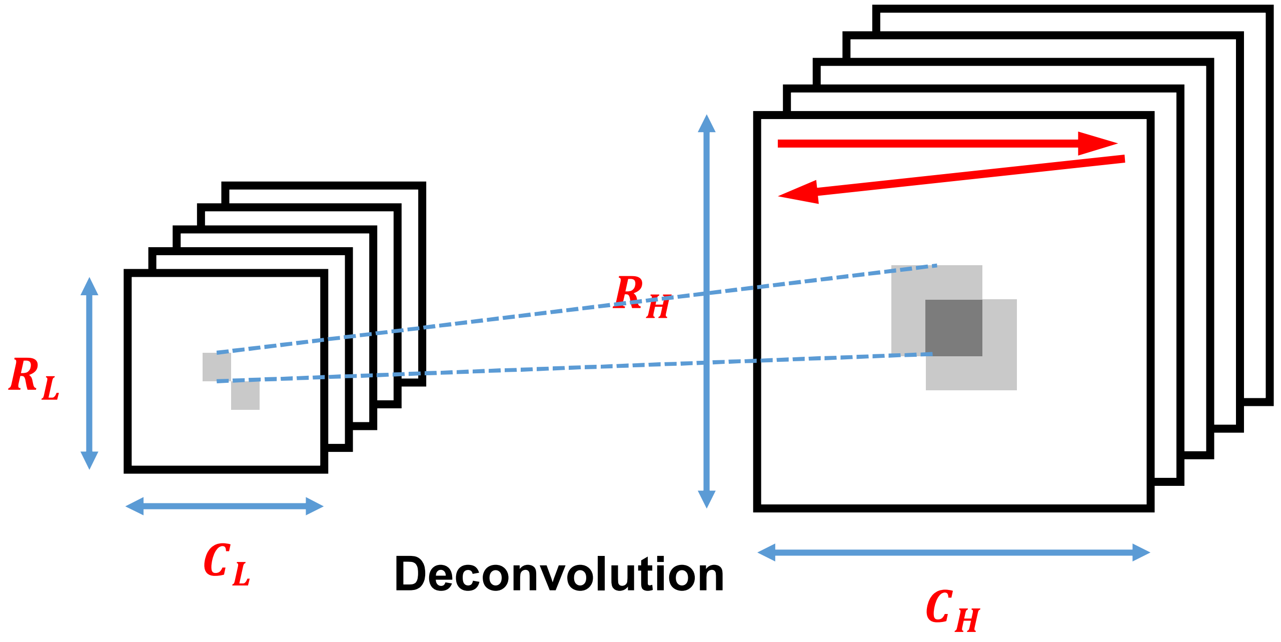
Overhead



[4] X. Zhang, et al. A design methodology for efficient implementation of deconvolutional neural networks on an FPGA. In arxiv: 1705.02583.

Background

- DCNN Accelerator [4]
  - ➢ Tile-based architecture
  - ➢ Reverse looping method
    - Overhead (extra operations, loop dimension)

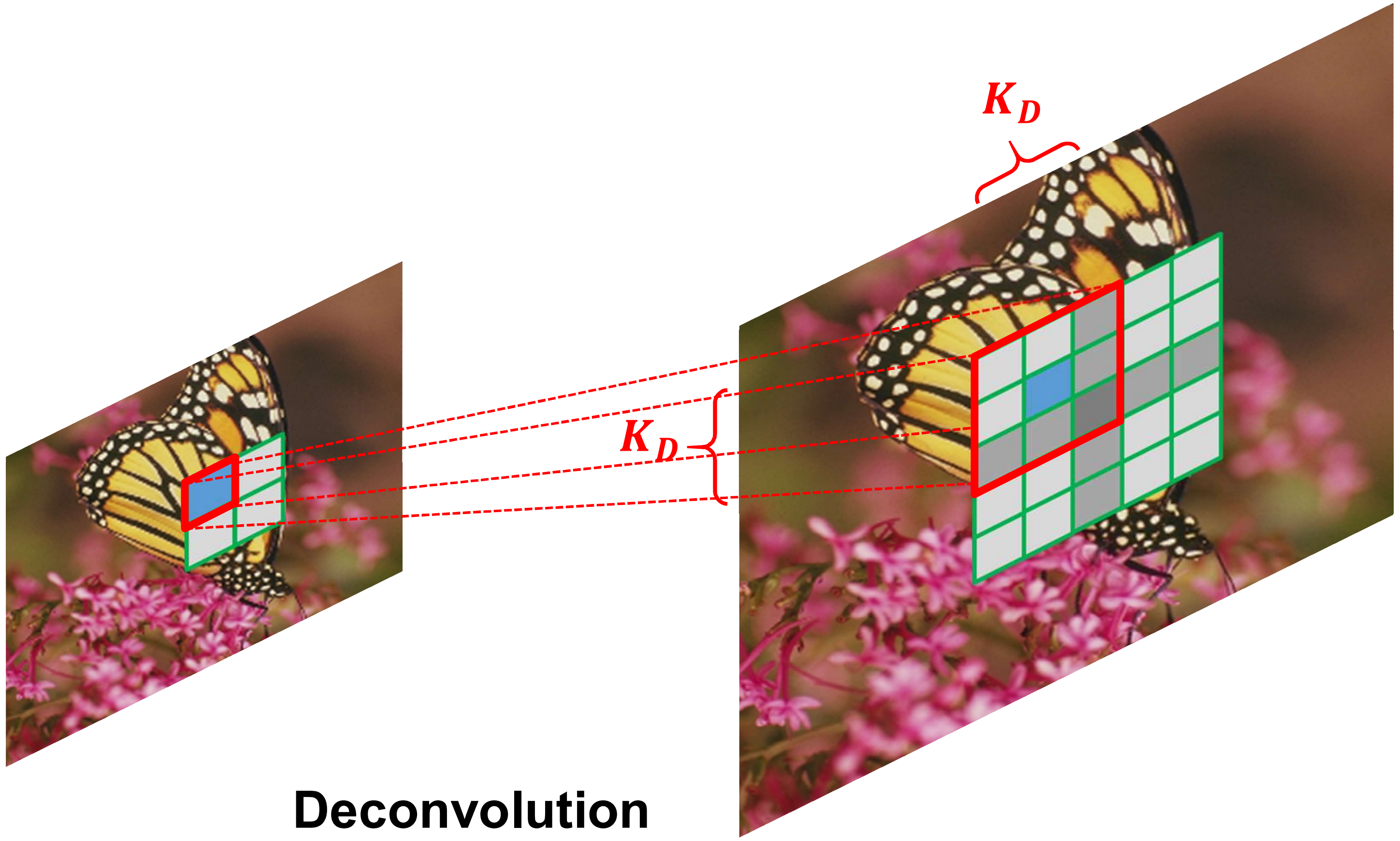


[4] X. Zhang, et al. A design methodology for efficient implementation of deconvolutional neural networks on an FPGA. In arxiv: 1705.02583.

Outline

Proposed Architecture – TDC method

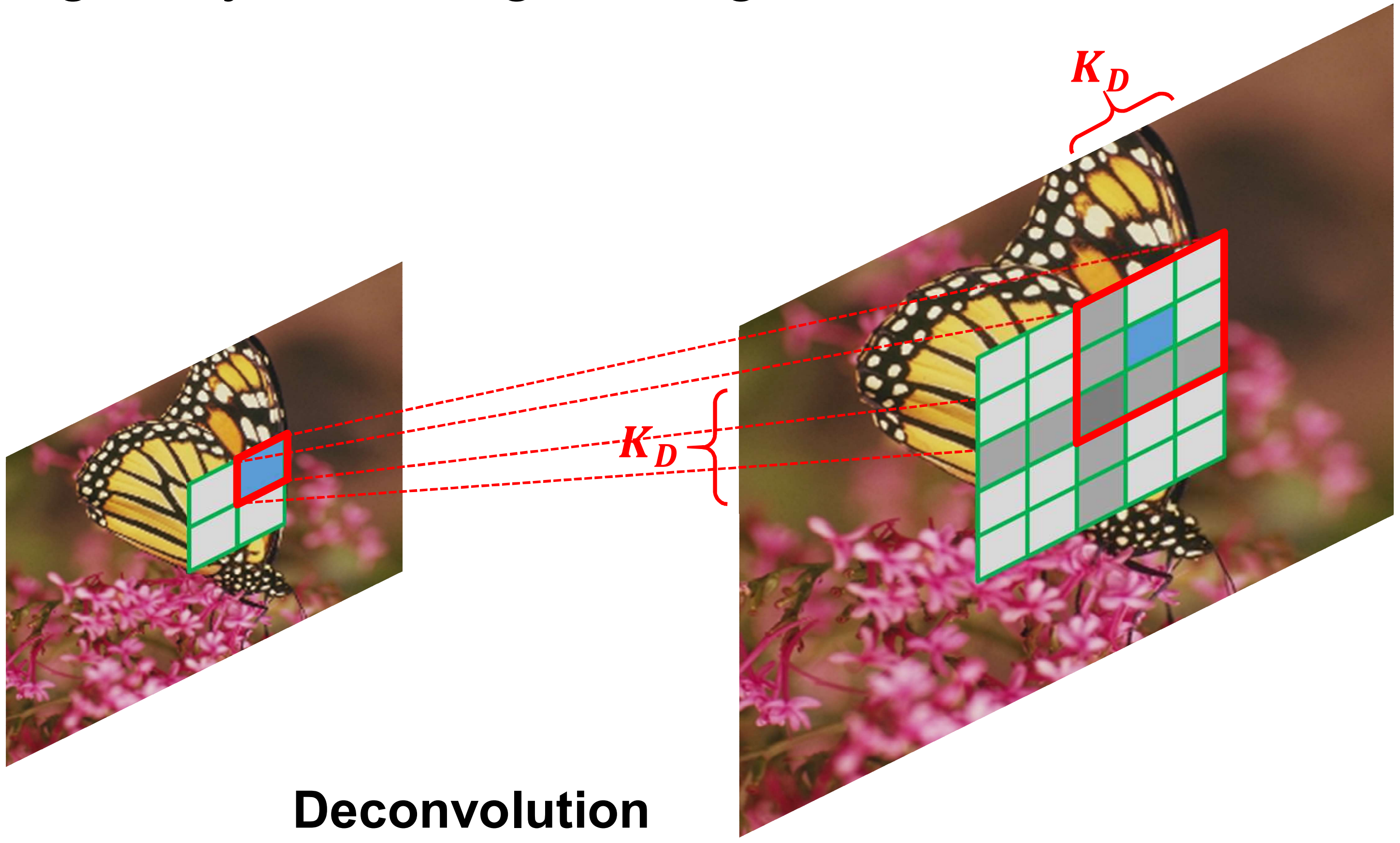
- Main idea
  - An output pixel is generated by overlapping regularly with neighboring blocks.



Deconvolution

Proposed Architecture – TDC method

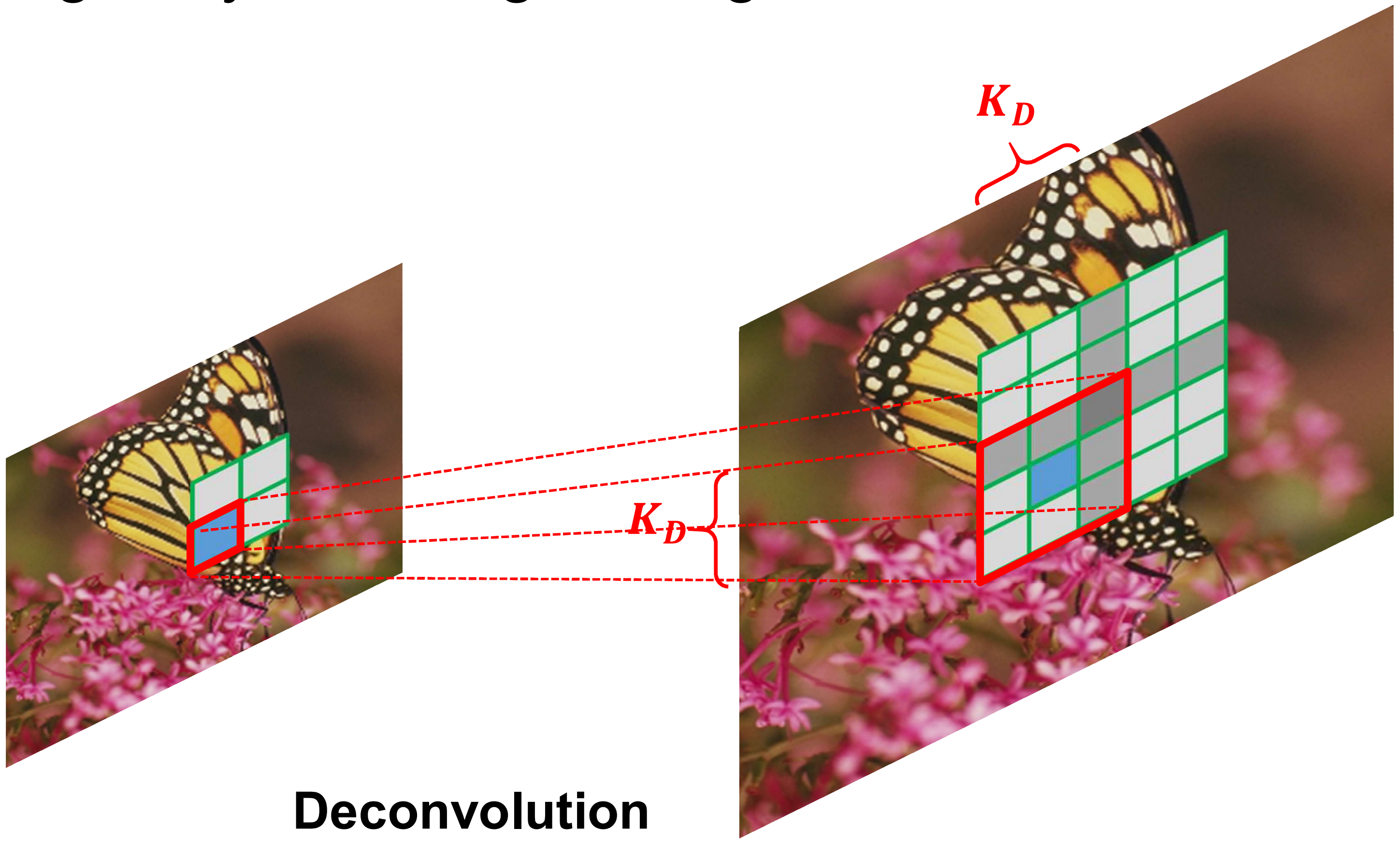
- Main idea
  - An output pixel is generated by overlapping regularly with neighboring blocks.



Deconvolution

Proposed Architecture – TDC method

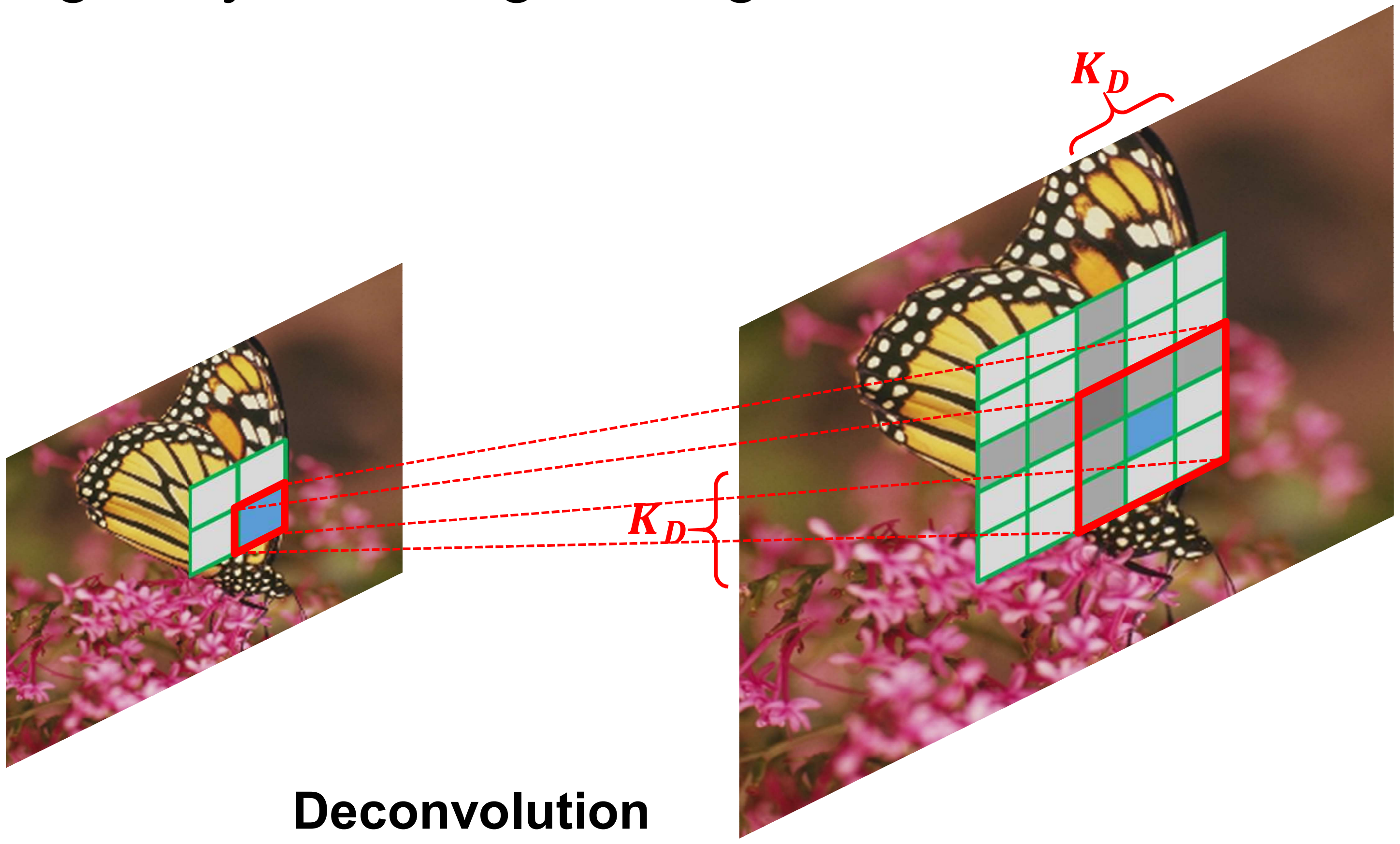
- Main idea
  - An output pixel is generated by overlapping regularly with neighboring blocks.



Deconvolution

Proposed Architecture – TDC method

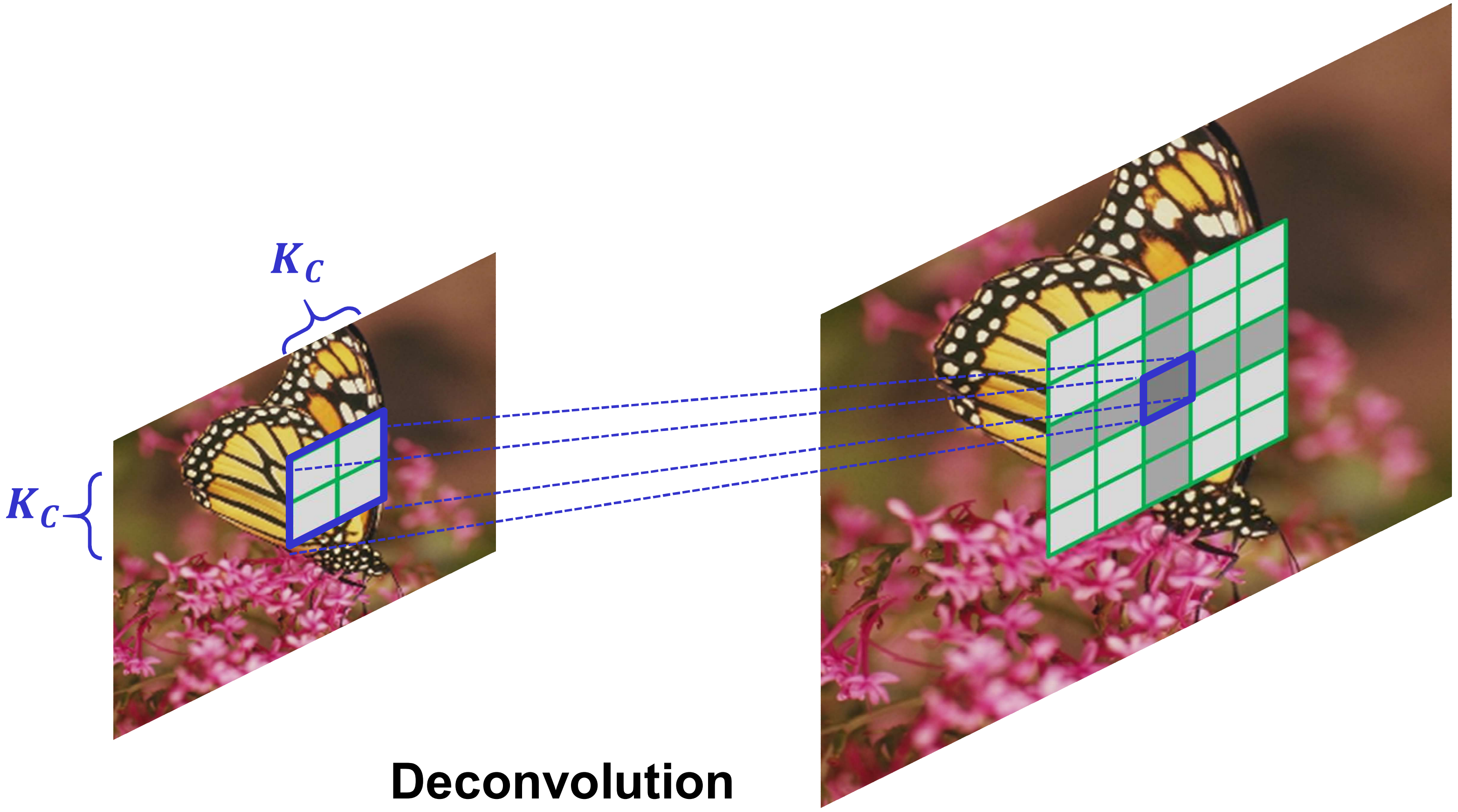
- Main idea
  - An output pixel is generated by overlapping regularly with neighboring blocks.



Deconvolution

Proposed Architecture – TDC method

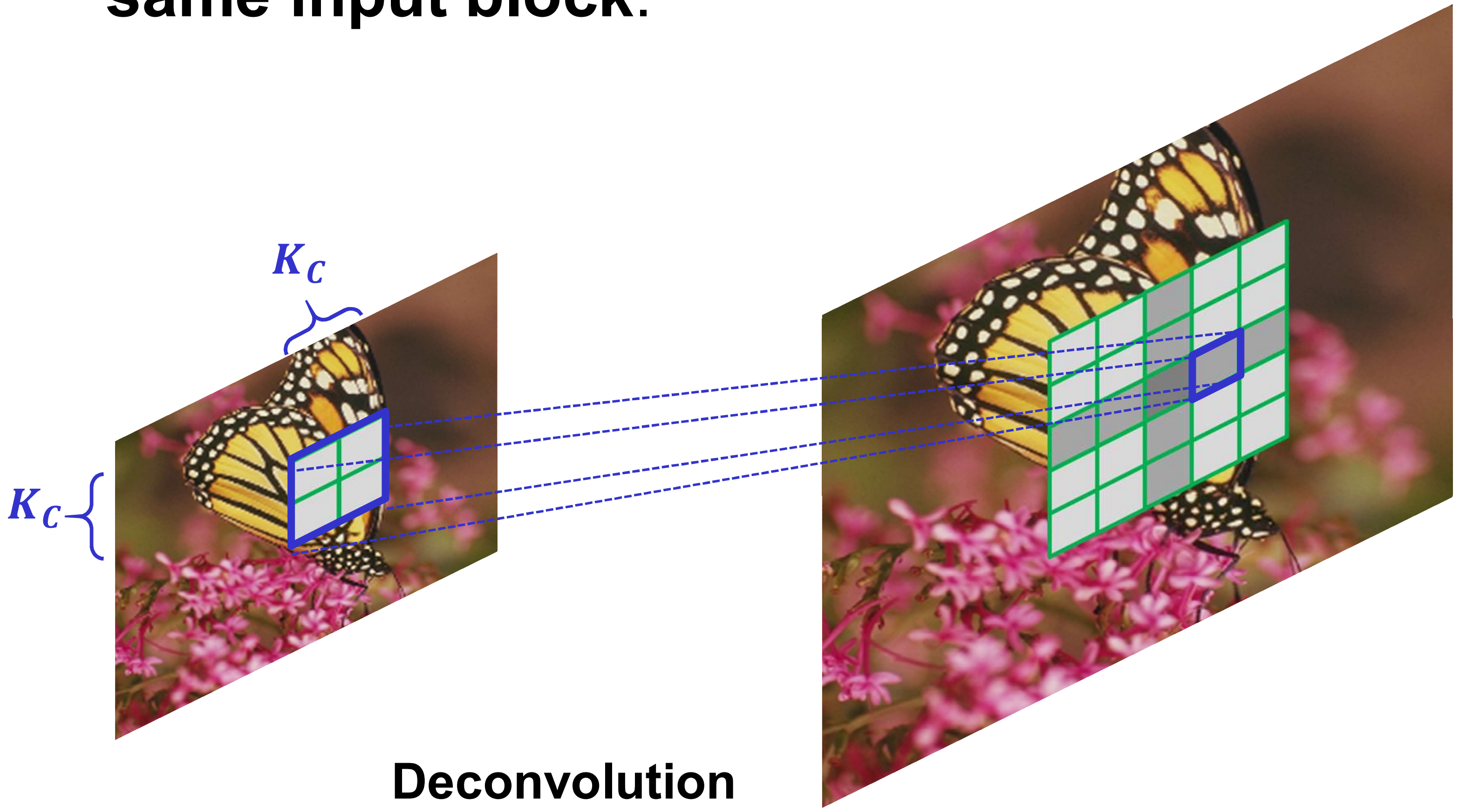
- Main idea
  - We can find the input block that determines the output pixels.



Deconvolution

Proposed Architecture – TDC method

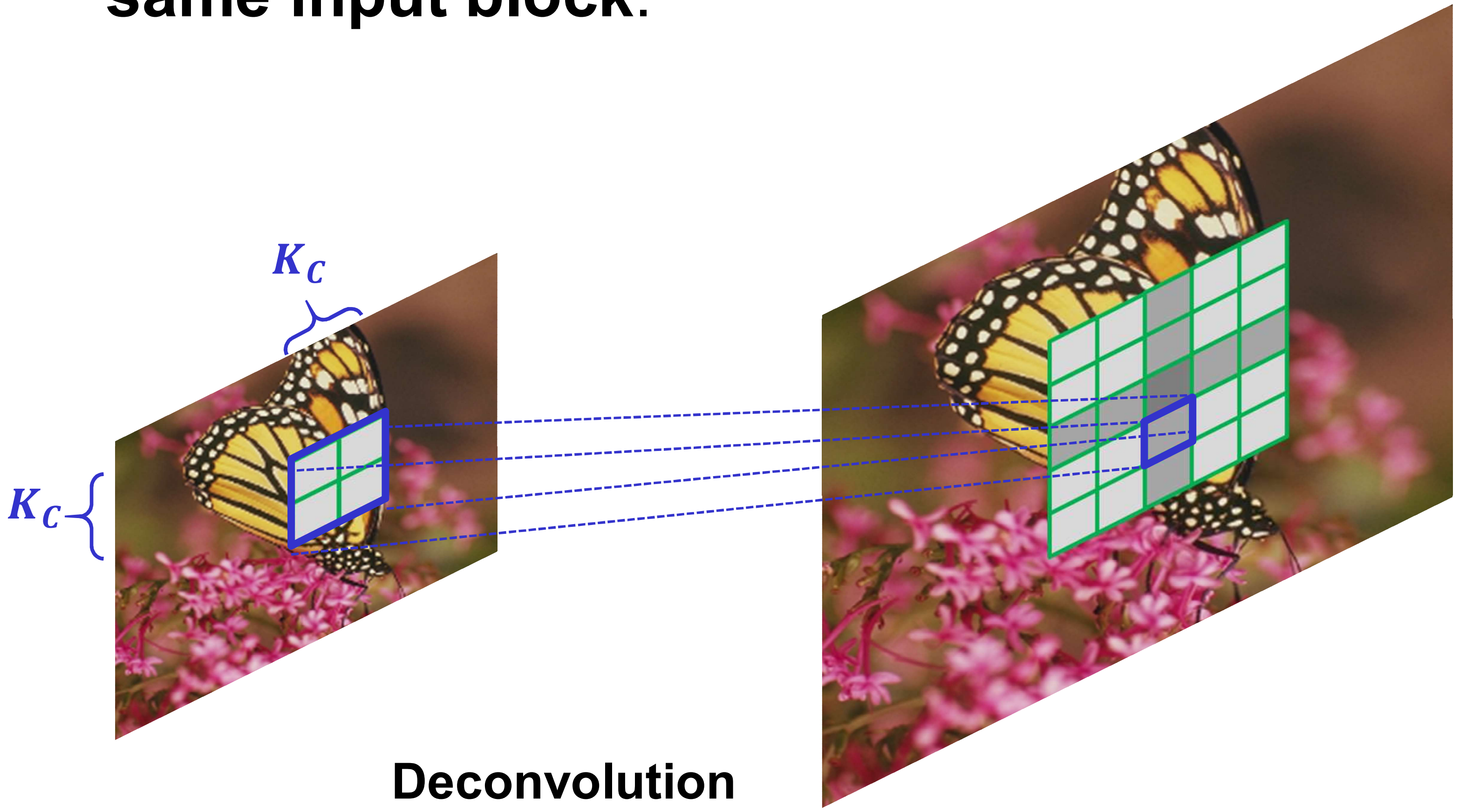
- Main idea
  - Especially, each input block moves by S_D pixels, so multiple output pixels can be created with **the same input block**.



Deconvolution

Proposed Architecture – TDC method

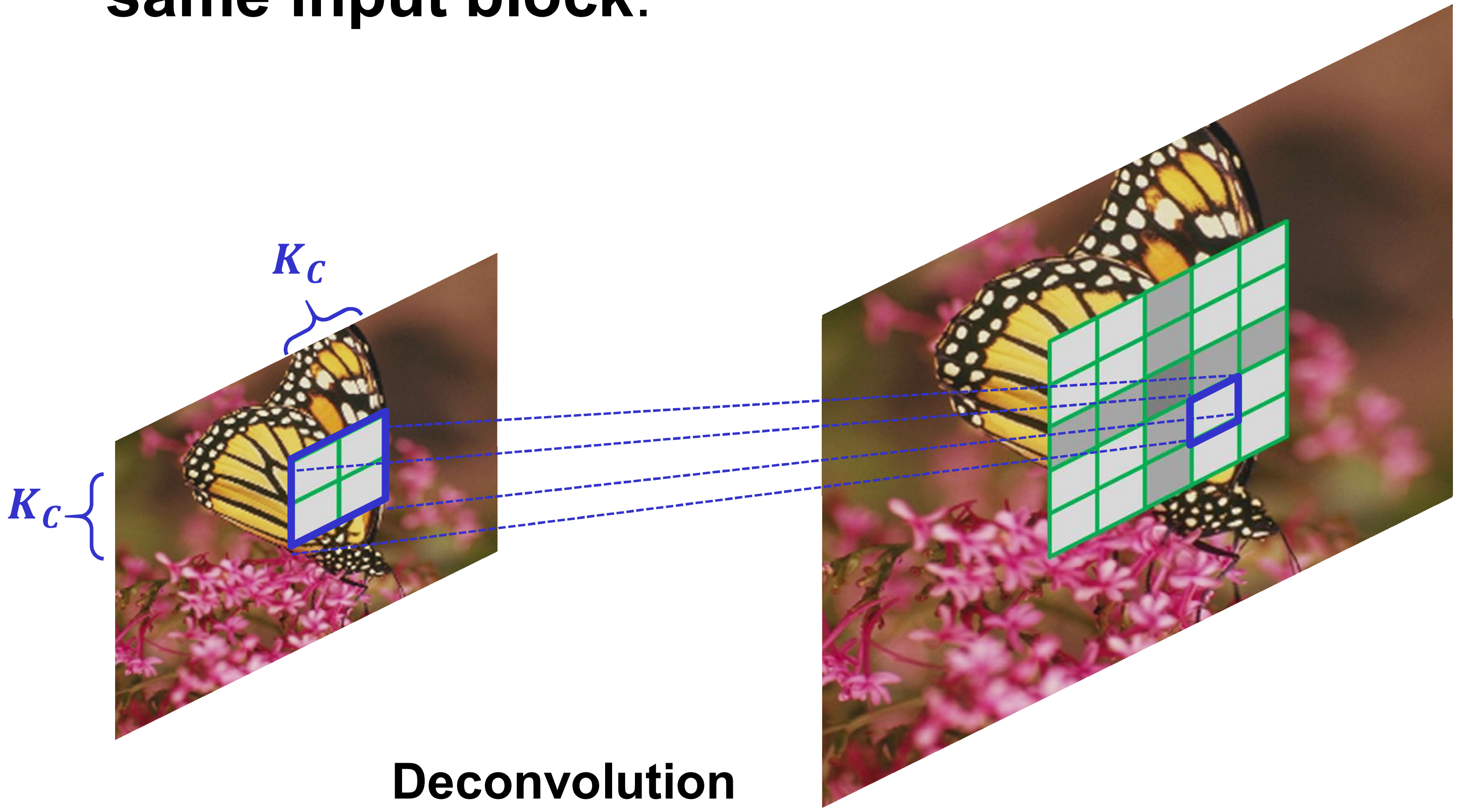
- Main idea
  - Especially, each input block moves by S_D pixels, so multiple output pixels can be created with **the same input block**.



Deconvolution

Proposed Architecture – TDC method

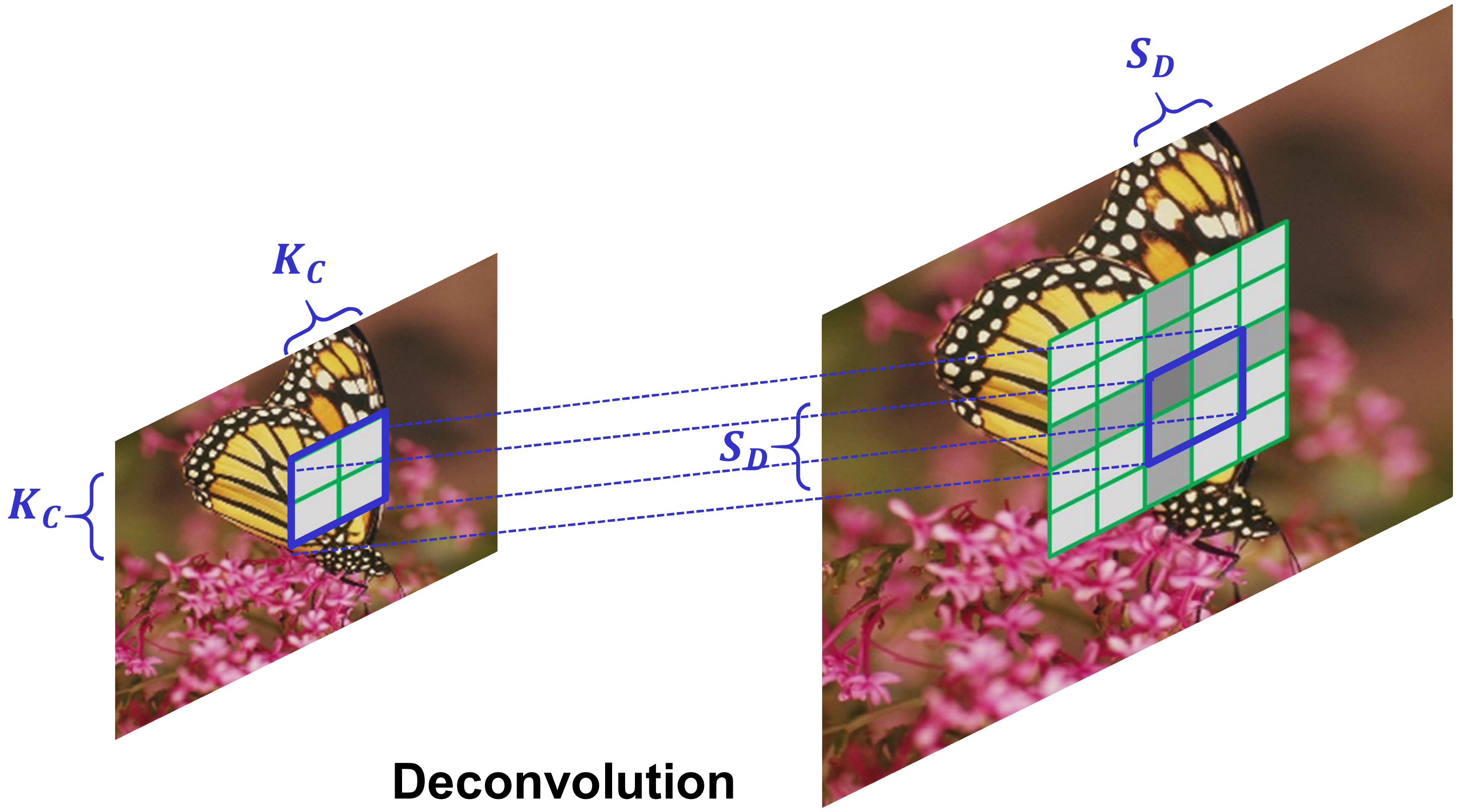
- Main idea
  - ➢Especially, each input block moves by S_D pixels, so multiple output pixels can be created with **the same input block**.



Deconvolution

Proposed Architecture – TDC method

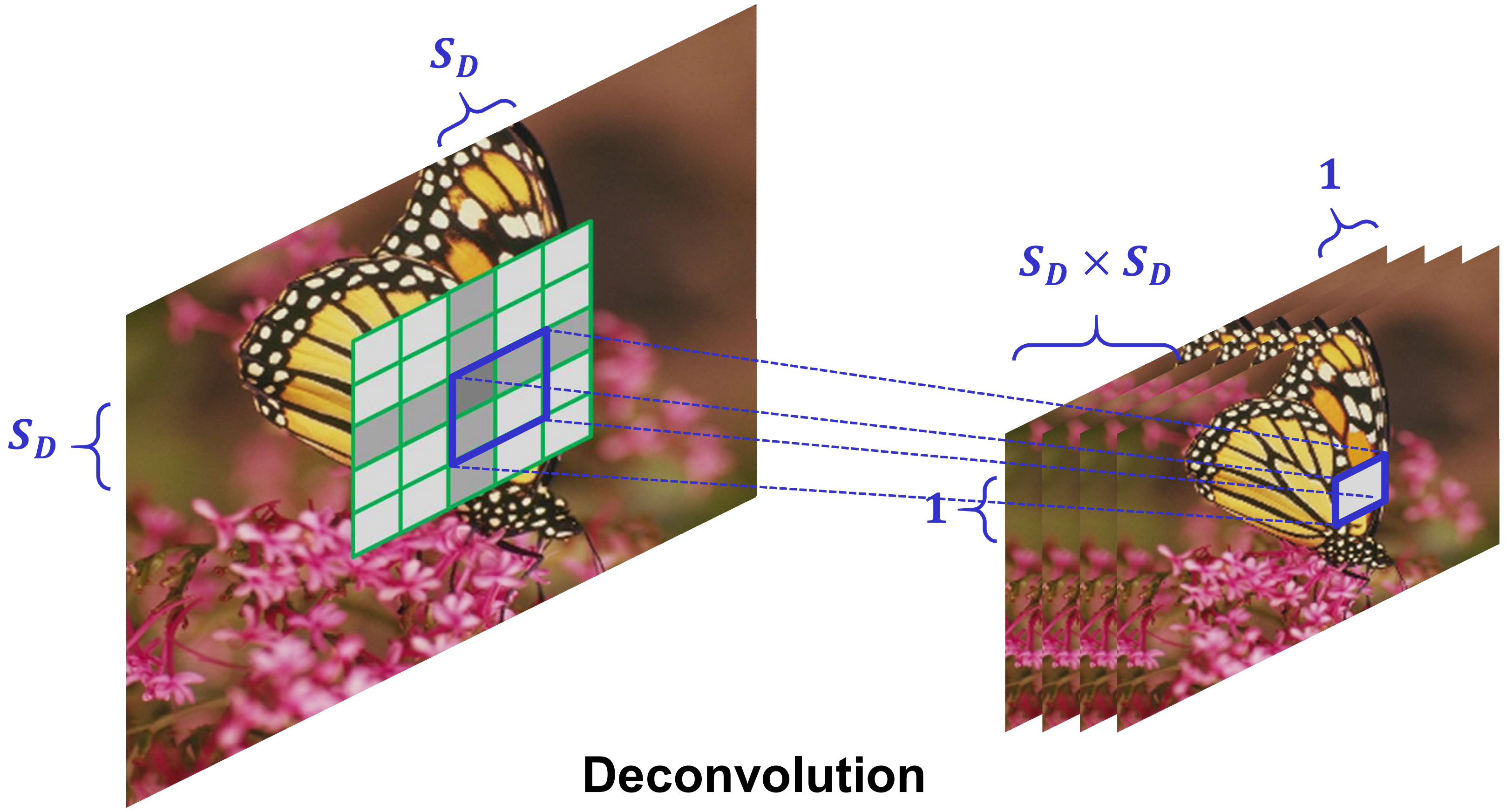
- Main idea
  - Thus, the $S_D \times S_D$ output block can be created with the same input block.



Deconvolution

Proposed Architecture – TDC method

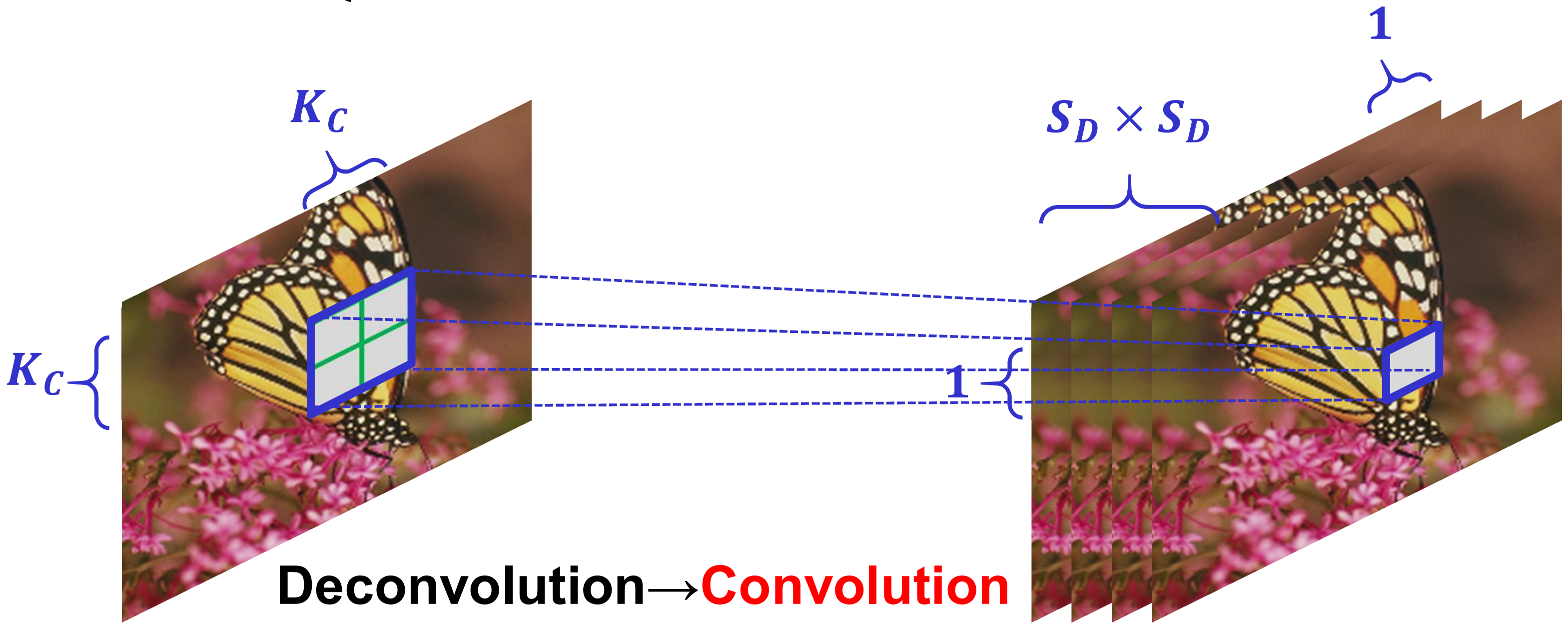
- Main idea
  - All pixels in the $S_D \times S_D$ output block can be created simultaneously.



Deconvolution

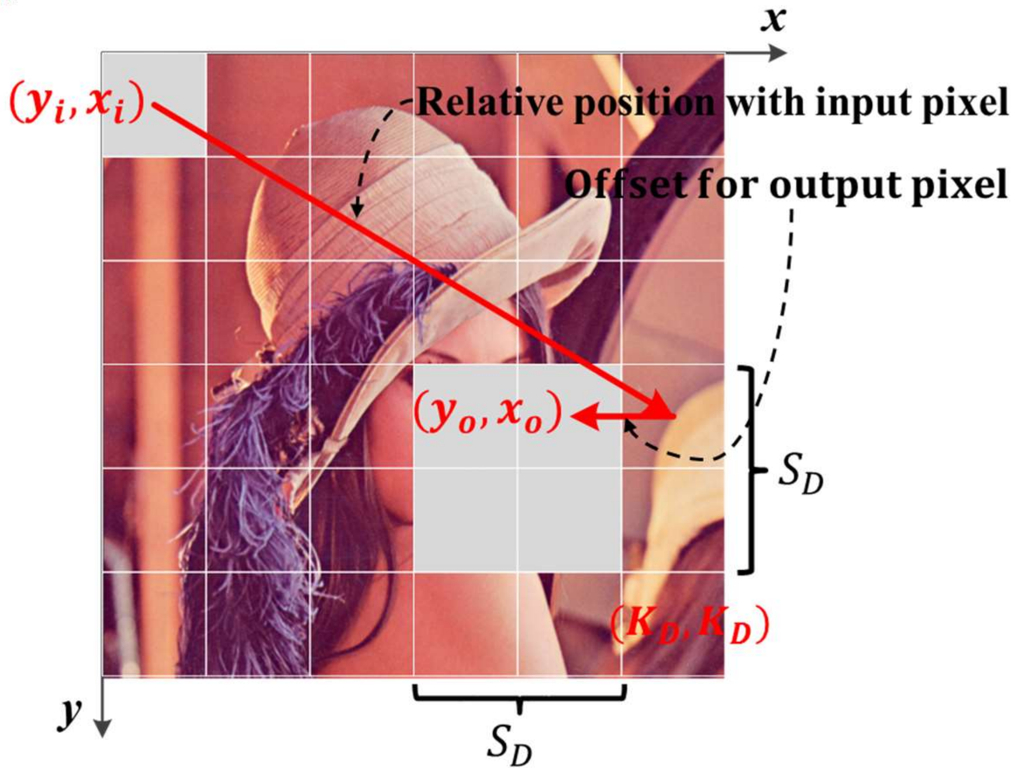
Proposed Architecture – TDC method

- Main idea
  - In conclusion, deconvolution is converted to convolution.
    - $K_C = \begin{cases} 2 \times \left\lfloor \left\lfloor \frac{K_D}{2} \right\rfloor \times \frac{1}{S_D} \right\rfloor + 1, \text{if } D < 0.5 \\ 2 \times \left\lfloor \left\lfloor \frac{K_D}{2} \right\rfloor \times \frac{1}{S_D} \right\rfloor, \quad \text{otherwise} \end{cases}$ where $D = \left\lfloor \frac{K_D}{2} \right\rfloor \times \frac{1}{S_D}$



Proposed Architecture – TDC method

- Main idea
  - Weights of newly created convolutional layer $\mathbf{W_C}$ can be mapped to the weights of the deconvolutional layer $\mathbf{W_D}$ using inverse coefficient mapping.



Proposed Architecture – TDC method

- Advantages

1. Kernel size is reduced from K_D to K_C.
   - ❖ Speed ↑, especially useful for kernel based CLP

2. All output pixels can be generated in parallel.
   - ❖ Speed ↑, DCNN→CNN

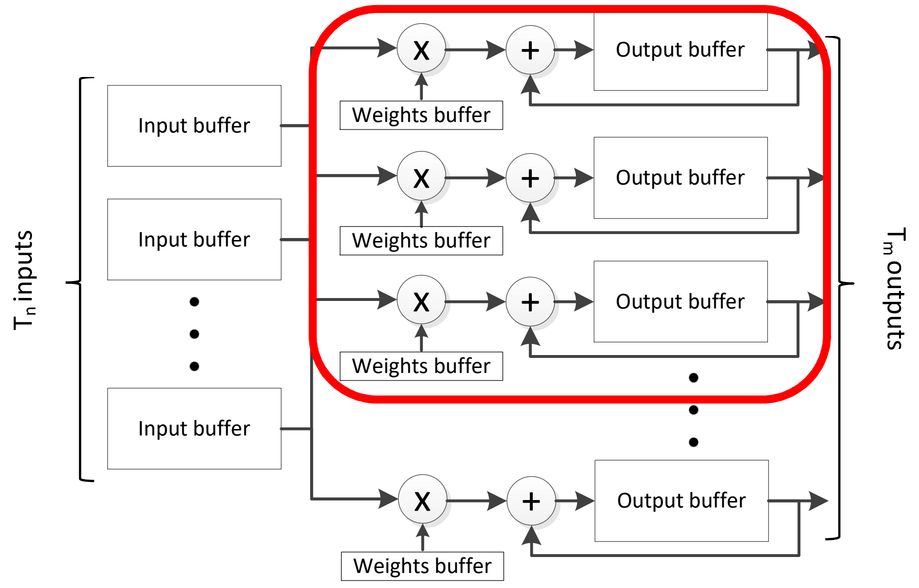
3. Compared with [4], TDC method does not need to calculate every loop iteration to obtain the position of the input pixels.
   - ❖ Speed ↑, overhead ↓

Proposed Architecture – TDC method

- Advantages

4. In addition, if there is a DSP underutilization problem in the deconvolutional layer, TDC method solves this problem.

Resource underutilization

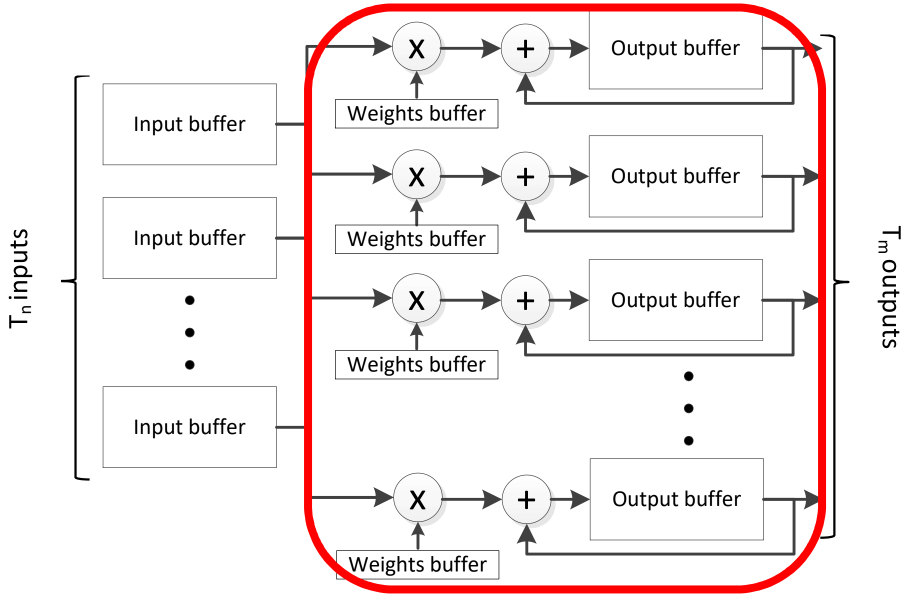


Proposed Architecture – TDC method

- Advantages

4. In addition, if there is a DSP under-utilization problem in the deconvolutional layer, TDC method solves this problem.

Activate disabled MAC !



Proposed Architecture – TDC method

- Evaluation of the Performance Compared with [4]

  - Number of execution cycles (Proposed) =

$$\left\lceil \frac{S_D^2 \times M}{T_m} \right\rceil \times \left\lceil \frac{N}{T_n} \right\rceil \times R_L \times C_L \times (K_C \times K_C + \frac{P}{T_r \times T_c})$$

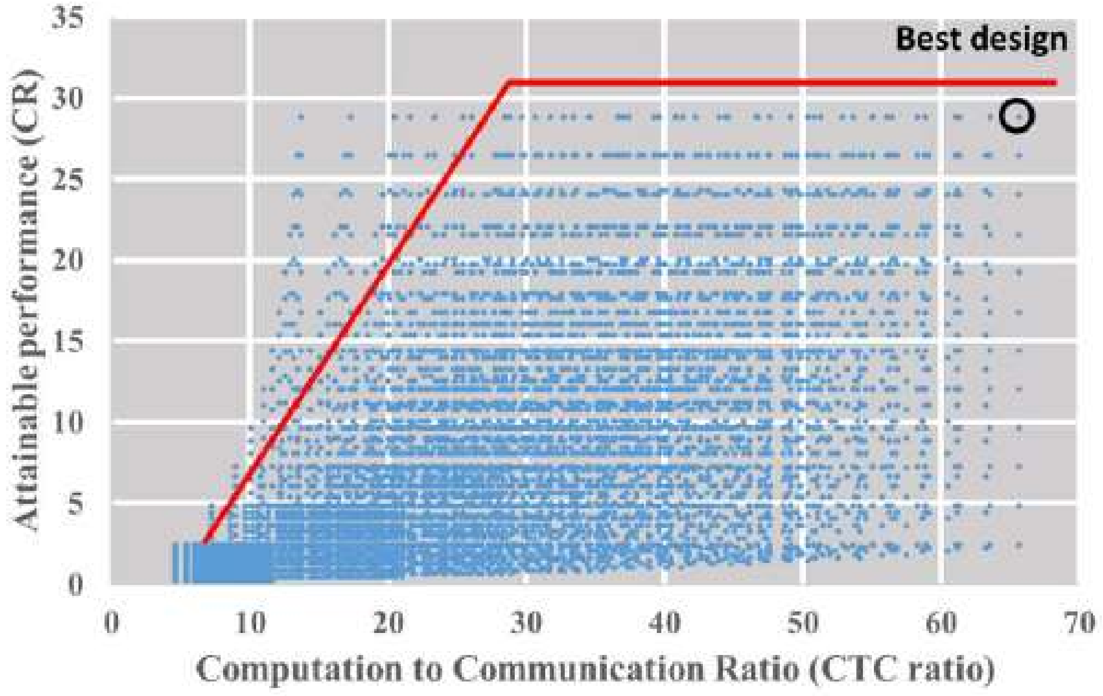
  - Number of execution cycles [4] =

$$\left\lceil \frac{M}{T_m} \right\rceil \times \left\lceil \frac{N}{T_n} \right\rceil \times (S_D \times R_L) \times (S_D \times C_L) \times (K_D \times K_D + \frac{P}{T_r \times T_c})$$

- Case 1. $M \le \frac{T_m}{S_D^2}$
  - Performance enhancement $= S_D^2 \times \frac{K_D^2}{K_C^2}$
- Case 2. $\frac{T_m}{S_D^2} < M < T_m$
  - Performance enhancement $= \frac{S_D^2}{\left\lceil \frac{S_D^2 \times M}{T_m} \right\rceil} \times \frac{K_D^2}{K_C^2}$
- Case 3. $M \ge T_m$
  - Performance enhancement $= \frac{K_D^2}{K_C^2}$

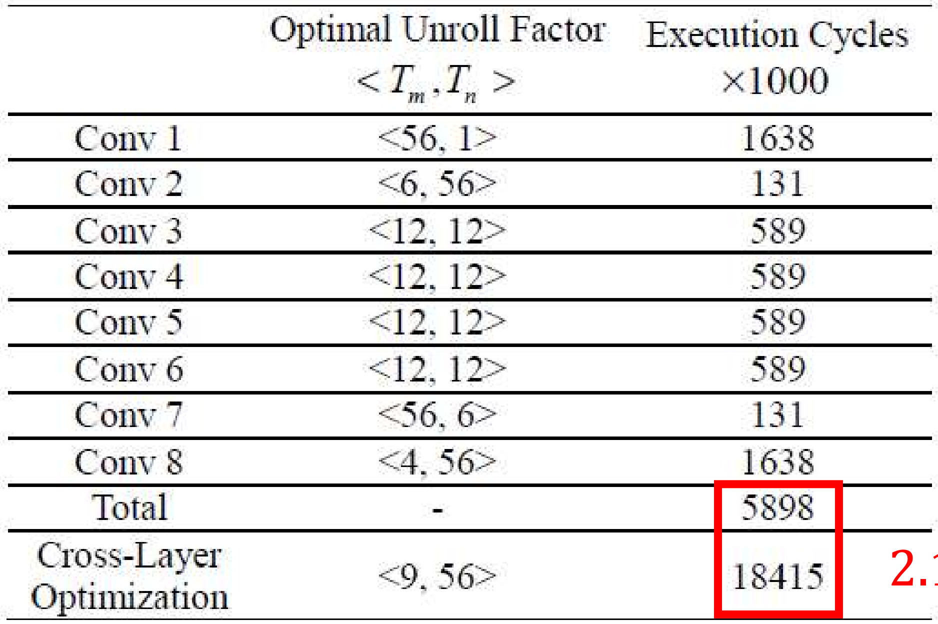
Proposed Architecture – Multi-CLP

- Hardware Implementation in FSRCNN
  - Roofline model
    - $CTC\ ratio = \frac{total\ number\ of\ operations}{total\ number\ of\ external\ memory\ access}$
    - $CR = \frac{total\ number\ of\ operations}{number\ of\ execution\ cycles}$



Proposed Architecture – Multi-CLP

- Hardware Implementation in FSRCNN
  - CNN with hourglass-type has a serious problem of **DSP underutilization** when using single CLP method.

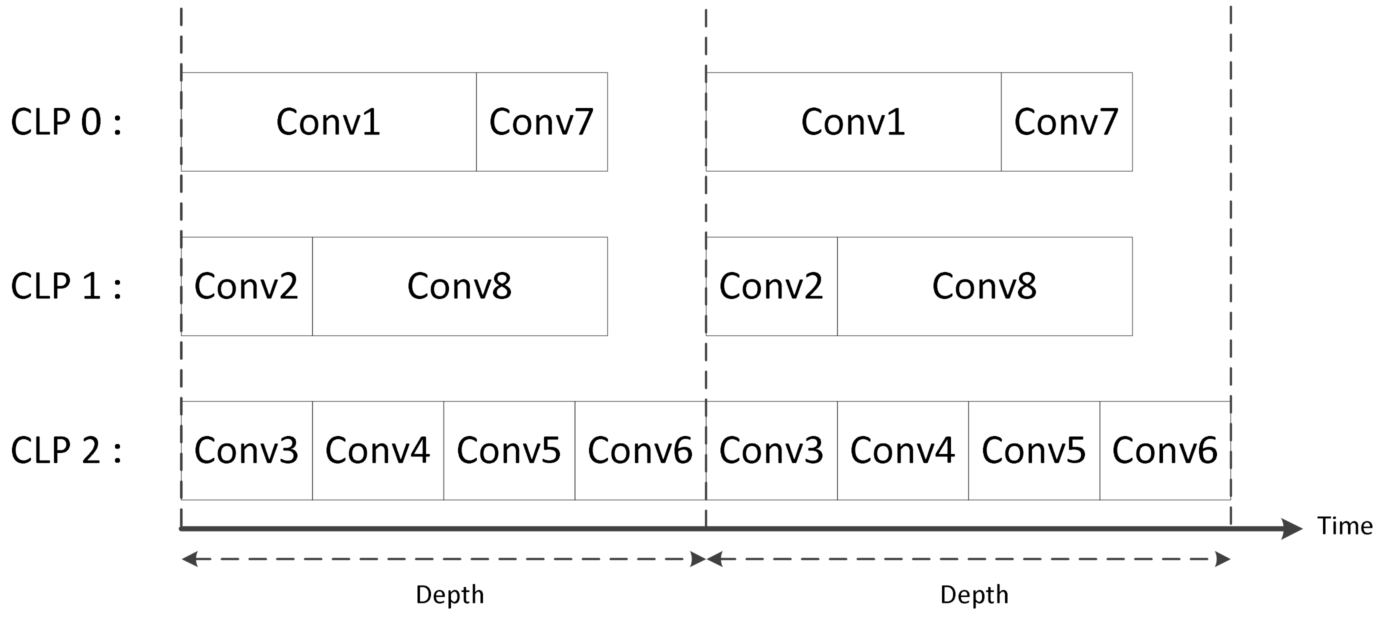


	Optimal Unroll Factor $< T_m, T_n >$	Execution Cycles $\times 1000$
Conv 1	<56, 1>	1638
Conv 2	<6, 56>	131
Conv 3	<12, 12>	589
Conv 4	<12, 12>	589
Conv 5	<12, 12>	589
Conv 6	<12, 12>	589
Conv 7	<56, 6>	131
Conv 8	<4, 56>	1638
Total	-	5898
Cross-Layer Optimization	<9, 56>	18415

2.12 × degradation

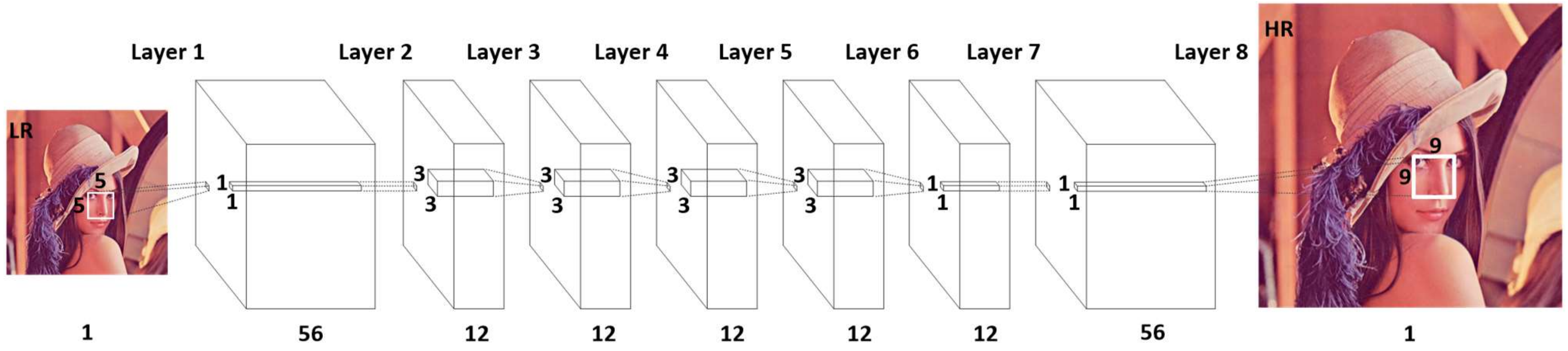
Proposed Architecture – Multi-CLP

- Hardware implementation in FSRCNN
  - We use the multi-CLP method.
  - We set T_n, T_m for the CLP by layer characteristics.



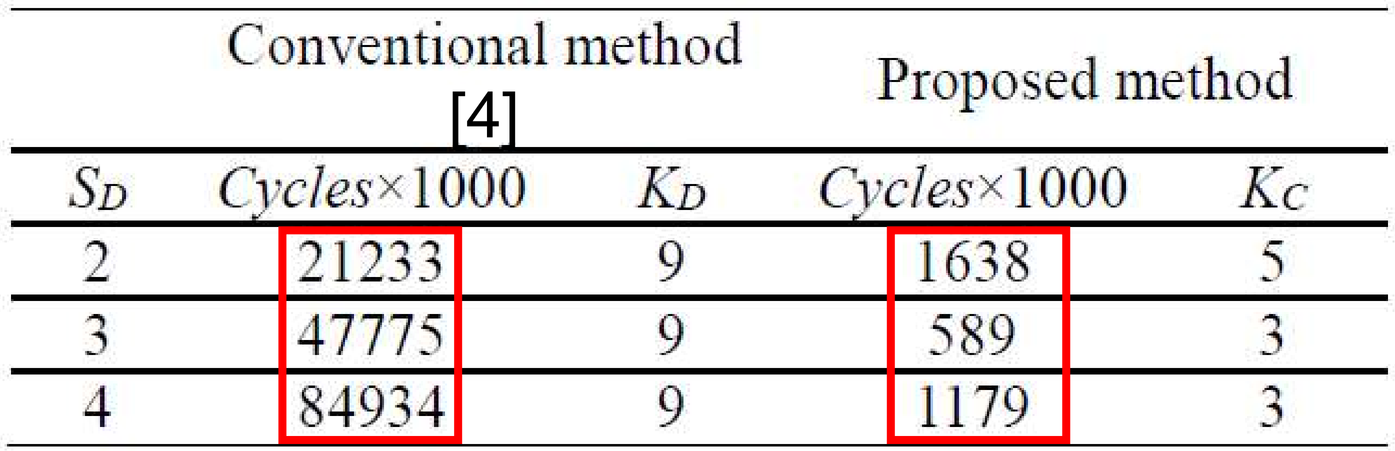
Experimental Environments

- Hardware Implementation Tool
  - High-Level Synthesis
  - Vivado HLS 2016. 4
  - Xilinx Virtex-7 485T FPGA
  - Single-precision floating point
- CNN model
  - FSRCNN
    - 7 convolution + 1 deconvolution



Experimental Results

- TDC Method
  - Loop tiling factor (T_n, T_m) was set to (56, 9).
  - Two reasons for the increase in the throughput
    - Kernel size
    - Resource underutilization problem is resolved



	Conventional method [4]		Proposed method	
S_D	$Cycles\times1000$	K_D	$Cycles\times1000$	K_C
2	21233	9	1638	5
3	47775	9	589	3
4	84934	9	1179	3

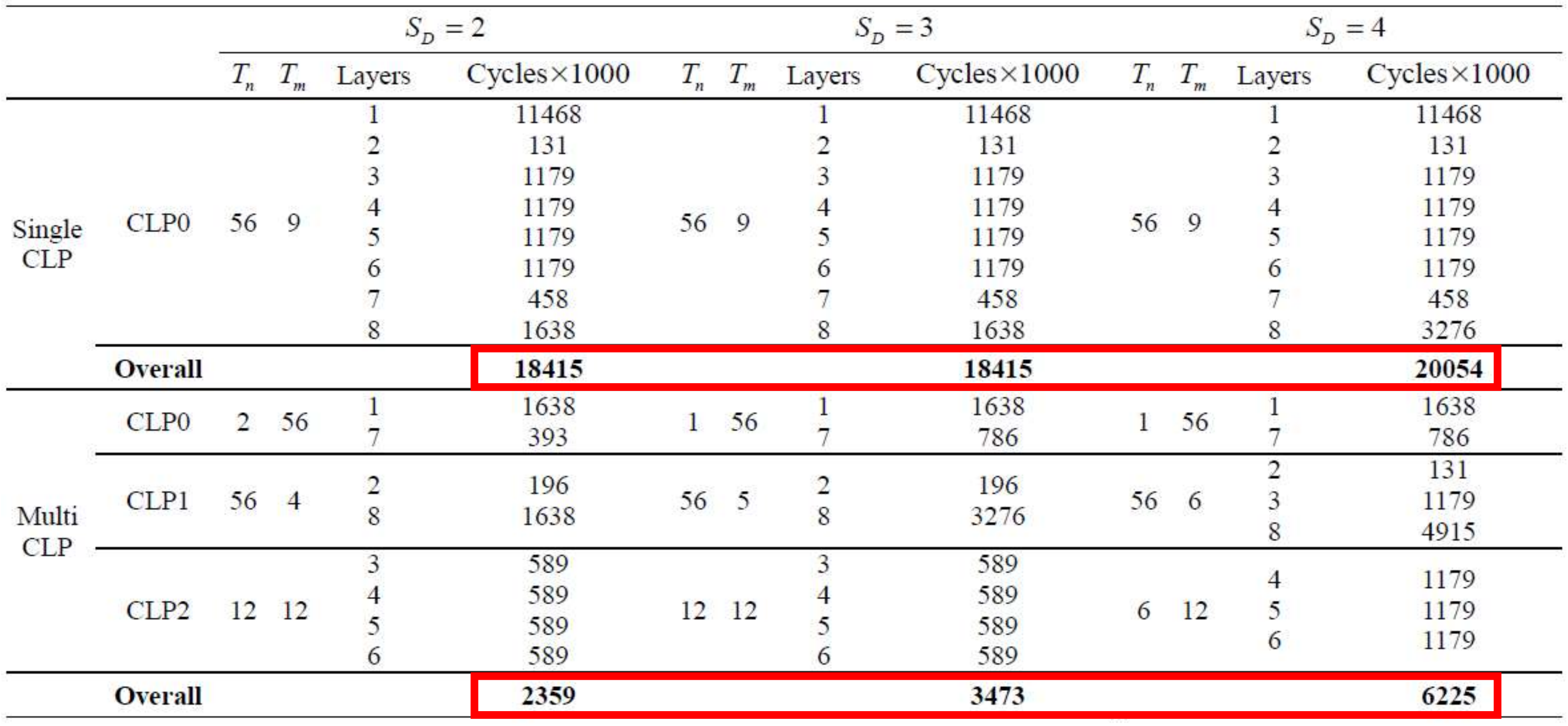
12.96 ×

81.11 ×

72.04 ×

Experimental Results

- Hardware Implementation in FSRCNN
  - ➤Single-CLP method vs Multi-CLP method

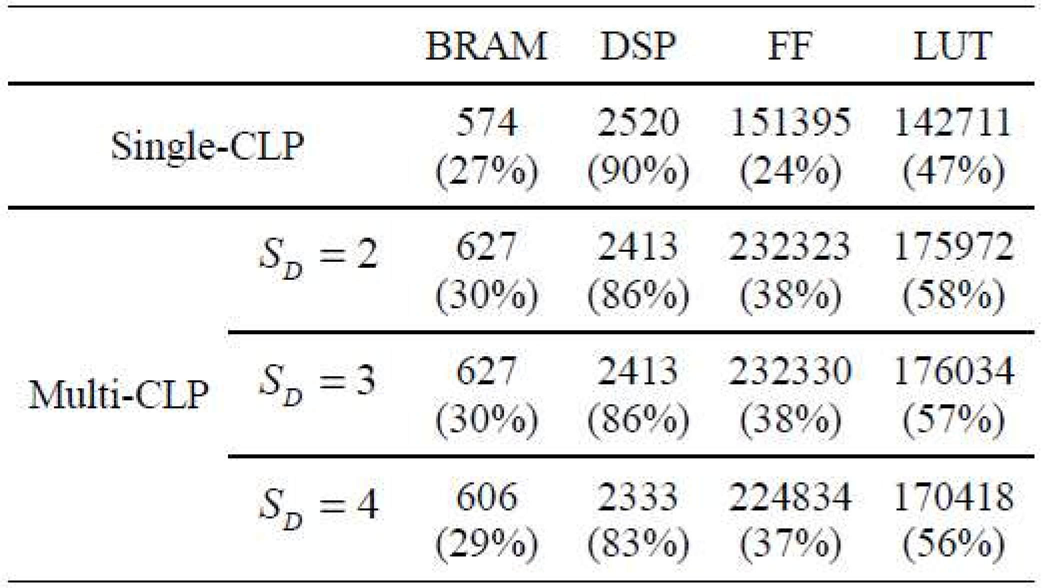


		$S_D = 2$				$S_D = 3$				$S_D = 4$			
		T_n	T_m	Layers	Cycles×1000	T_n	T_m	Layers	Cycles×1000	T_n	T_m	Layers	Cycles×1000
Single CLP	CLP0	56	9	1	11468	56	9	1	11468	56	9	1	11468
				2	131			2	131			2	131
				3	1179			3	1179			3	1179
				4	1179			4	1179			4	1179
				5	1179			5	1179			5	1179
				6	1179			6	1179			6	1179
				7	458			7	458			7	458
				8	1638			8	1638			8	3276
	Overall				**18415**				**18415**				**20054**
Multi CLP	CLP0	2	56	1	1638	1	56	1	1638	1	56	1	1638
				7	393			7	786			7	786
	CLP1	56	4	2	196	56	5	2	196	56	6	2	131
				8	1638			8	3276			3	1179
												8	4915
	CLP2	12	12	3	589	12	12	3	589	6	12	4	1179
				4	589			4	589			5	1179
				5	589			5	589			6	1179
				6	589			6	589				
	Overall				**2359**				**3473**				**6225**

7.8 × **5.32 ×** **3.22 ×**

Experimental Results

- Hardware Implementation in FSRCNN
  - Resource usage of each design
    - Single CLP method used more resources.
    - According to S_D, the tiling factors of multi-CLP does not change much, so there is little difference between resource usage.



		BRAM	DSP	FF	LUT
Single-CLP		574 (27%)	2520 (90%)	151395 (24%)	142711 (47%)
Multi-CLP	$S_D = 2$	627 (30%)	2413 (86%)	232323 (38%)	175972 (58%)
	$S_D = 3$	627 (30%)	2413 (86%)	232330 (38%)	176034 (57%)
	$S_D = 4$	606 (29%)	2333 (83%)	224834 (37%)	170418 (56%)

Conclusion

- Propose the TDC Method based DCNN Accelerator.
  - Reduce the size of the kernel from $K_D \times K_D$ to $K_C \times K_C$.
  - Improves throughput and could resolve the spatial problem.
  - Increase the parallelism of the output feature maps.
  - Outperform the state-of-the-art DCNN accelerator up to 81×.

- Propose the Efficient Architecture for Hourglass-type CNN.
  - Implement FSRCNN on Xilinx Virtex-7 FPGA.
  - Resolve resource underutilization problem.
  - Improve the CNN accelerator up to 7.8× using multi-CLP method when compared with single CLP method.