

# **A 54.7 fps 3D Point Cloud Semantic Segmentation Processor with Sparse Grouping based Dilated Graph Convolutional Network for Mobile Devices**



**Sangjin Kim**, Sangyeob Kim, Juhyoung Lee,  
and Hoi-Jun Yoo

**Semiconductor System Lab.  
School of Electrical Engineering, KAIST**

**2020 IEEE International Symposium on Circuits and Systems  
Virtual, October 10-21, 2020**

# Outline

---

## **I. Introduction**

## **II. Overall Architecture**

## **III. Key Features**

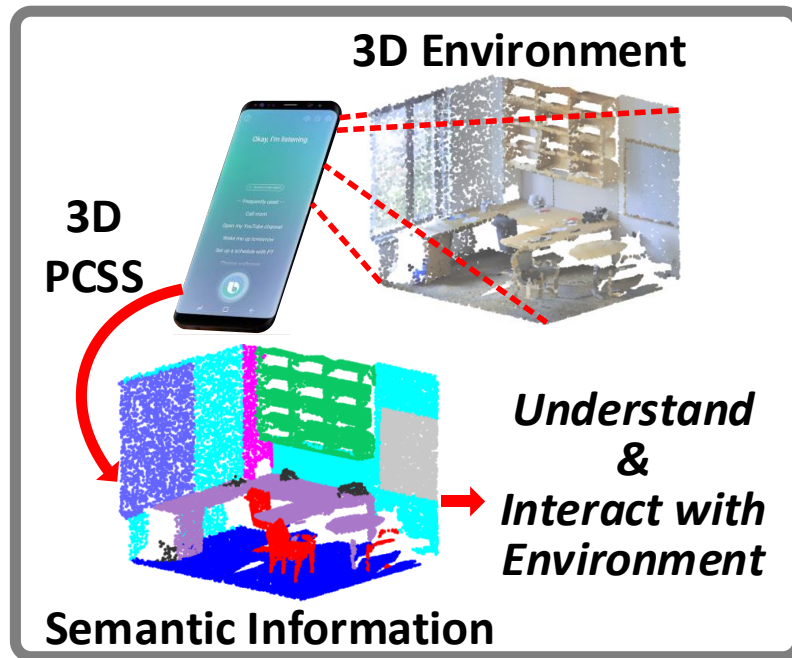
- I. Sparse Grouping based Dilated Graph Convolution
- II. Group-level Pipelining

## **IV. Implementation Results**

## **V. Conclusion**

# 3D Point Cloud Semantic Segmentation

- ❑ **Extension of semantic segmentation to 3D data**
  - Essential for applications that interact with the 3D environment



Ex)



Augmented Reality



Autonomous Drone

## ❑ Key Requirements

- Real-time Operation ( $> 30\text{fps}$ ), Low-power consumption ( $< 100\text{mW}^*$ )

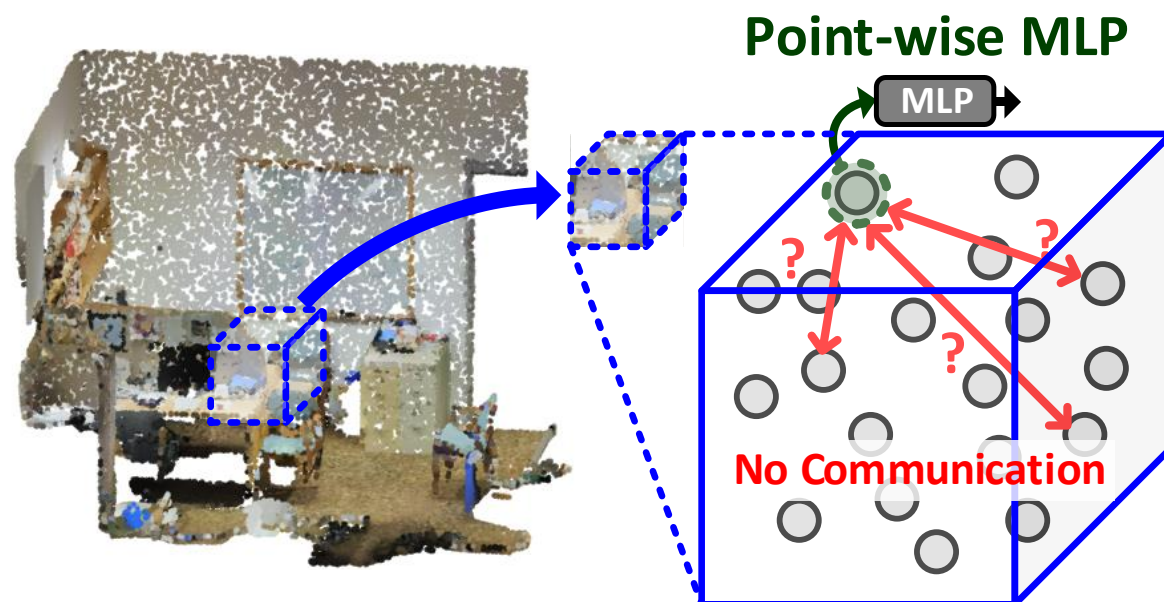
\* 5% power of mobile AP for AR device

# 3D PCSS Algorithm

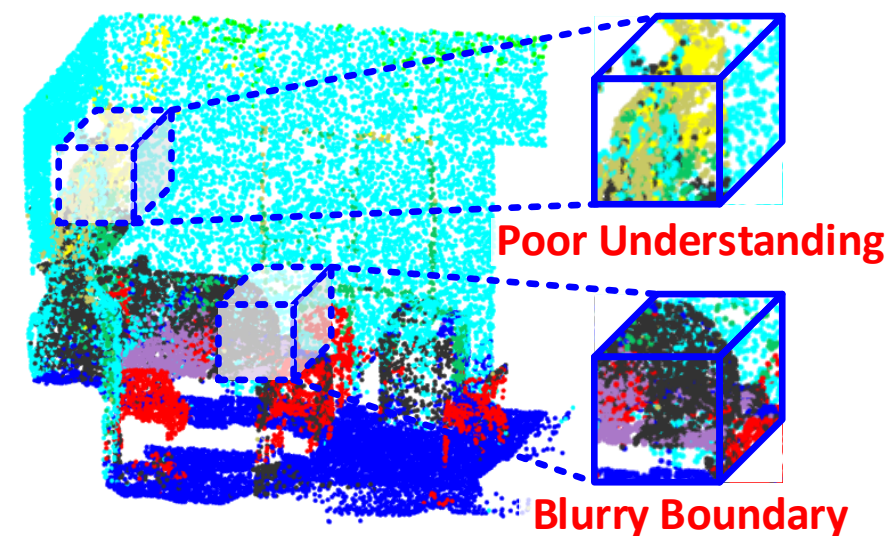
## ❑ Traditional Method - Point-wise Feature based 3D PCSS

– Limited communication between points

➔ Hard to understand contextual feature 😞, Unclear boundary 😞



Point-wise Feature Extraction



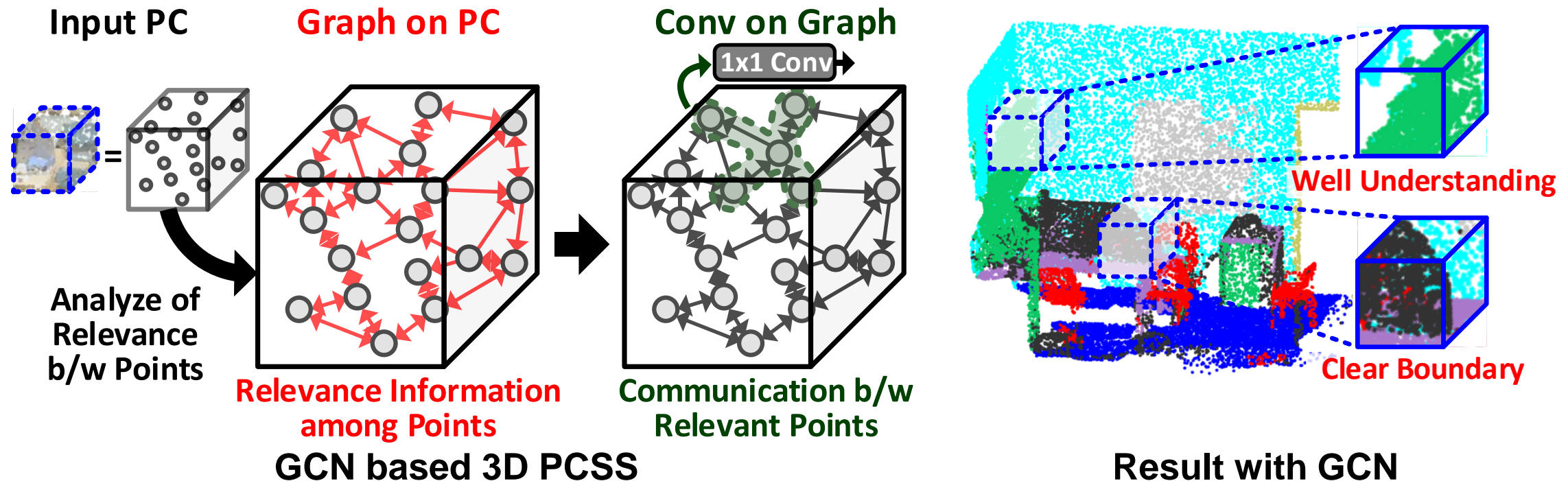
Result with Trad. Method

# 3D PCSS Algorithm

## ❑ Trend: Graph Convolutional Network based 3D PCSS

- Create a graph by connecting relevant points
- Understanding of contextual features through communication on graph

➔ Understanding contextual feature 😊, Clear boundary 😊



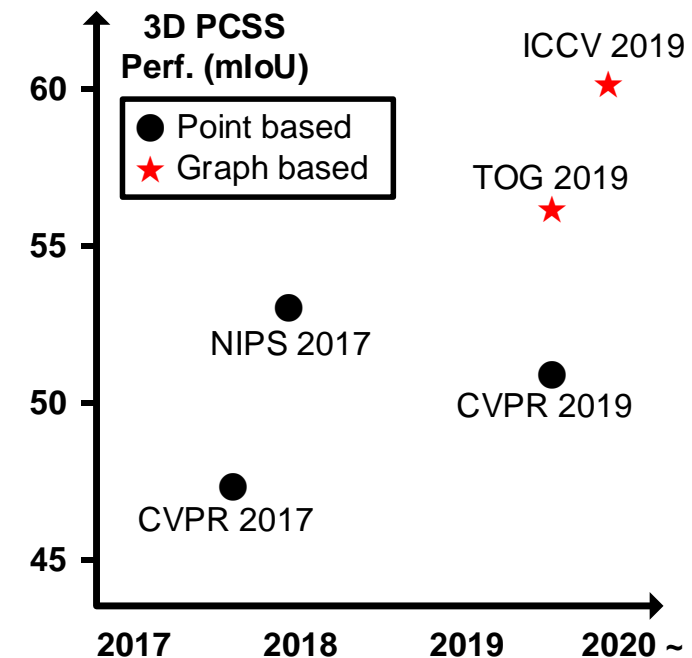
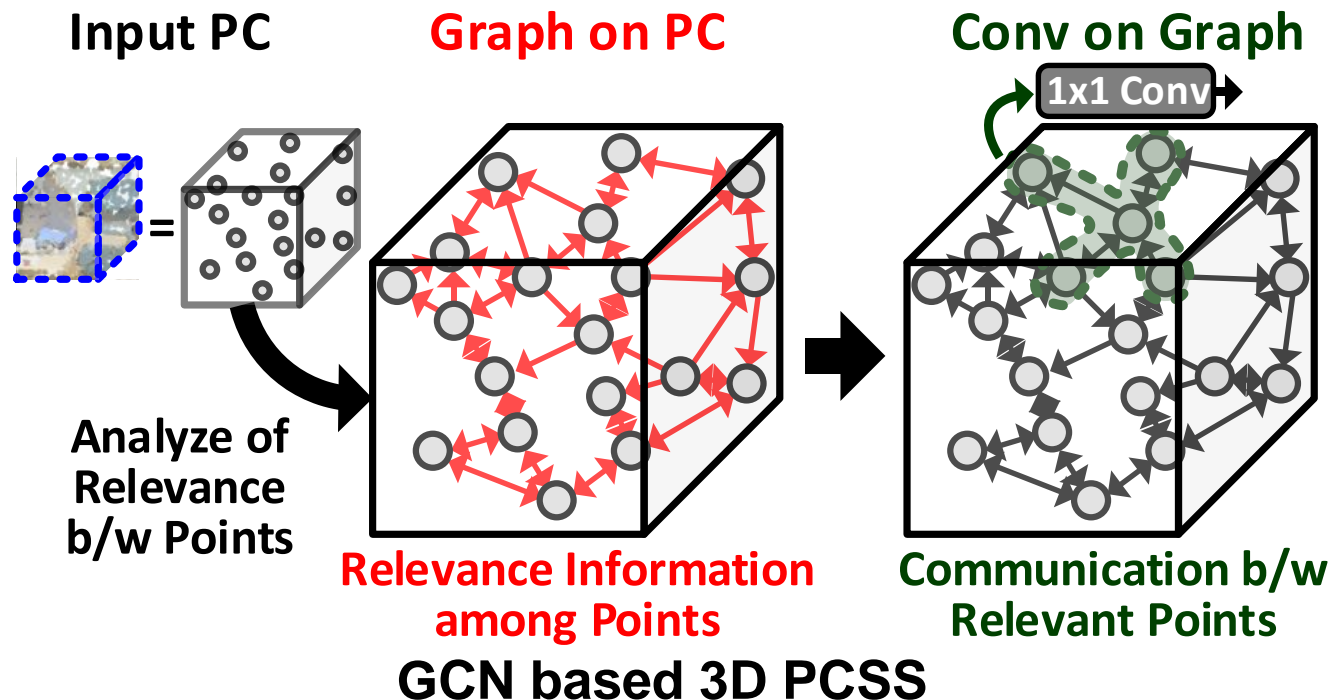


# 3D PCSS Algorithm

## □ Trend: Graph Convolutional Network based 3D PCSS

- Create a graph by connecting relevant points
- Understanding of contextual features through communication on graph

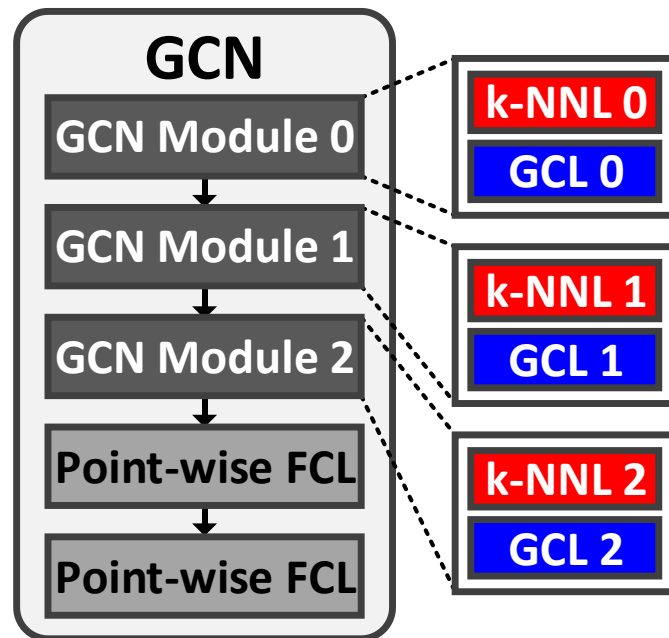
➔ Understanding contextual feature 😊, Clear boundary 😊



# Graph Convolutional Network (GCN)

## □ Architecture of GCN for 3D PCSS

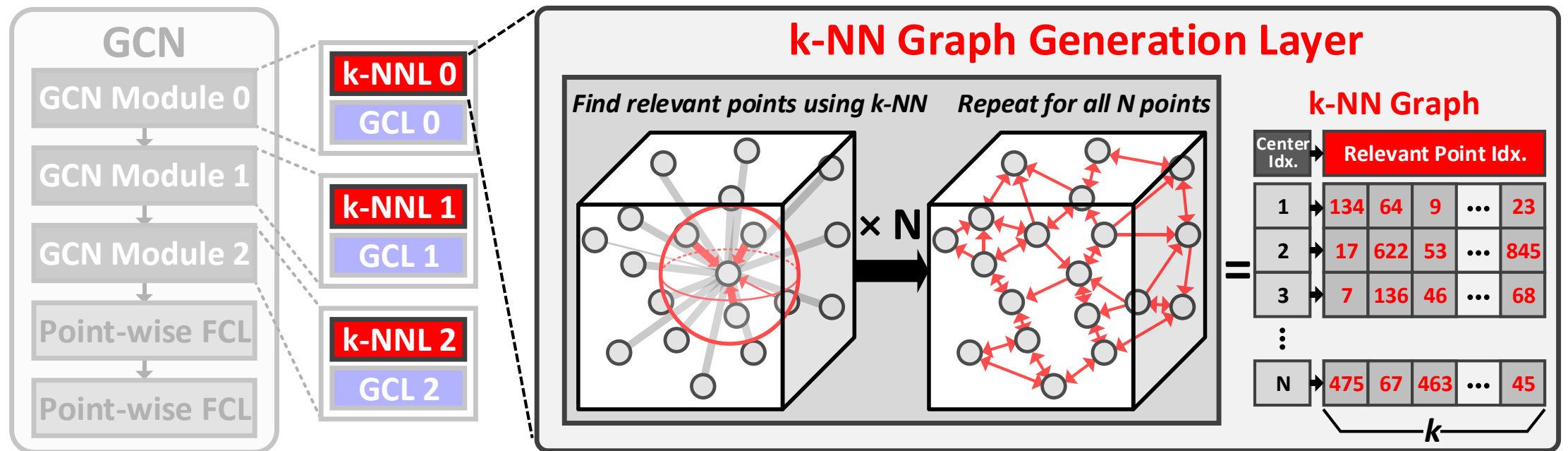
- ***k*-nearest neighbor graph generation layer (*k*-NNL)** for graph generation
- **Graph convolution layer (GCL)** for feature extraction



# Graph Convolutional Network (GCN)

## □ ***k*-Nearest Neighbor Graph Generation layer (*k*-NNL)**

- Find a neighbor by comparing the dist. on feature dimension with other points
- Generate *k*-NN graph by repeating for all *N* points of the point cloud.

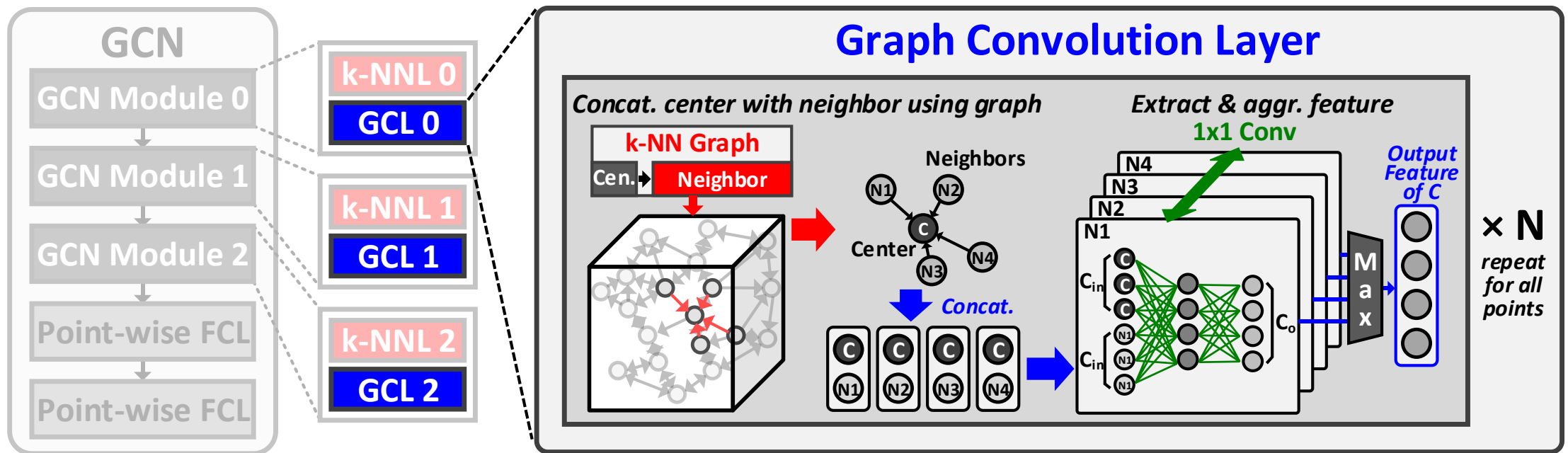




# Graph Convolutional Network (GCN)

## □ Graph Convolution Layer (GCL)

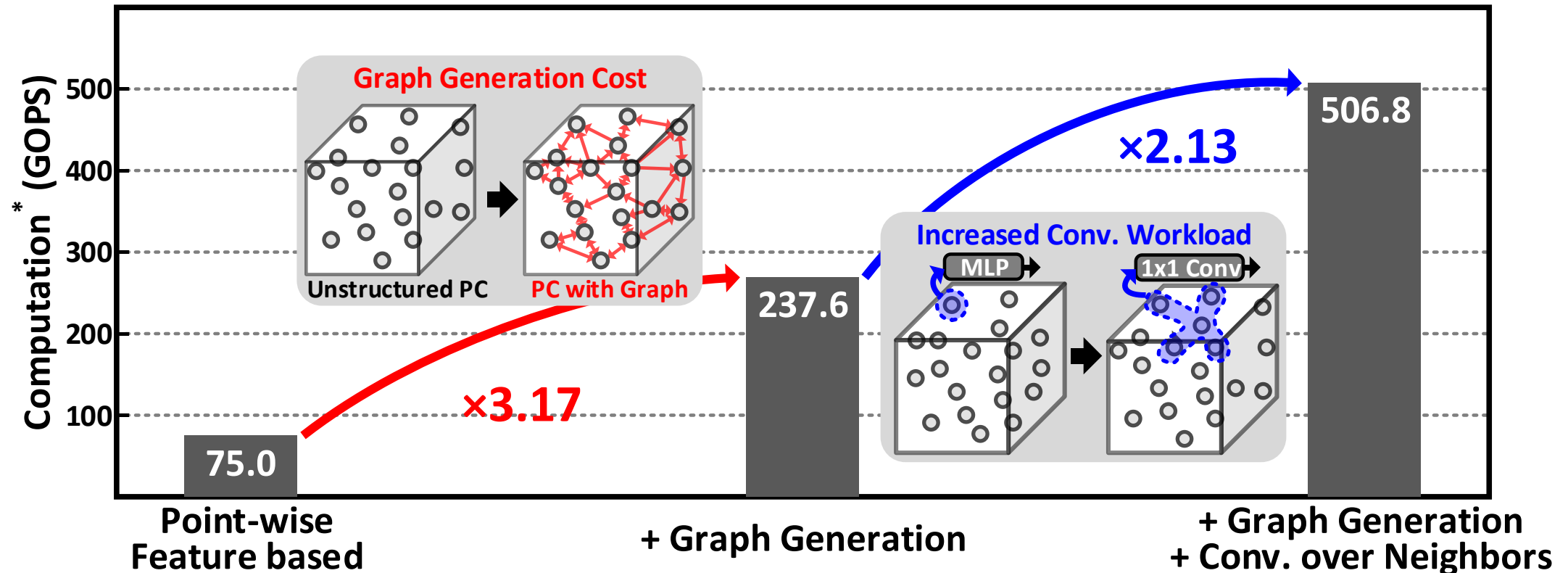
- Concatenate feature vector of center and neighbors
- Extract context feature using MLP & aggregate features using Max-pooling



# Design Challenge of GCN

## ❑ Heavy Computation Amount (~506.8 GOPS)

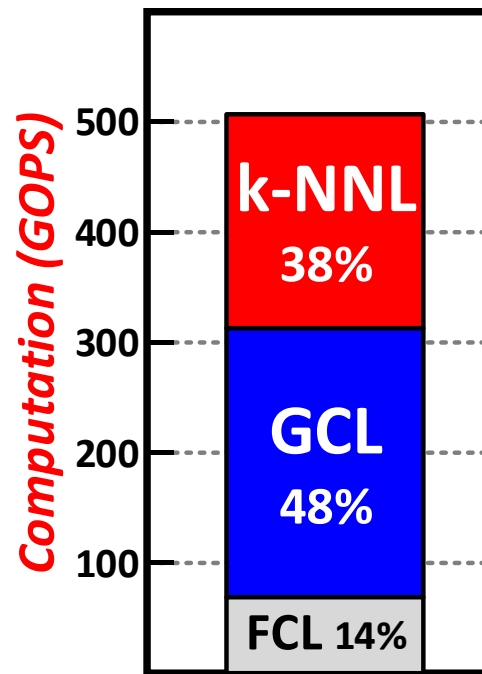
- **Graph generation:** Additional cost for graph generation ( **$k$ -NNL**)
- **Increased conv. workload:** Multiple point used for feature extraction (**GCL**)



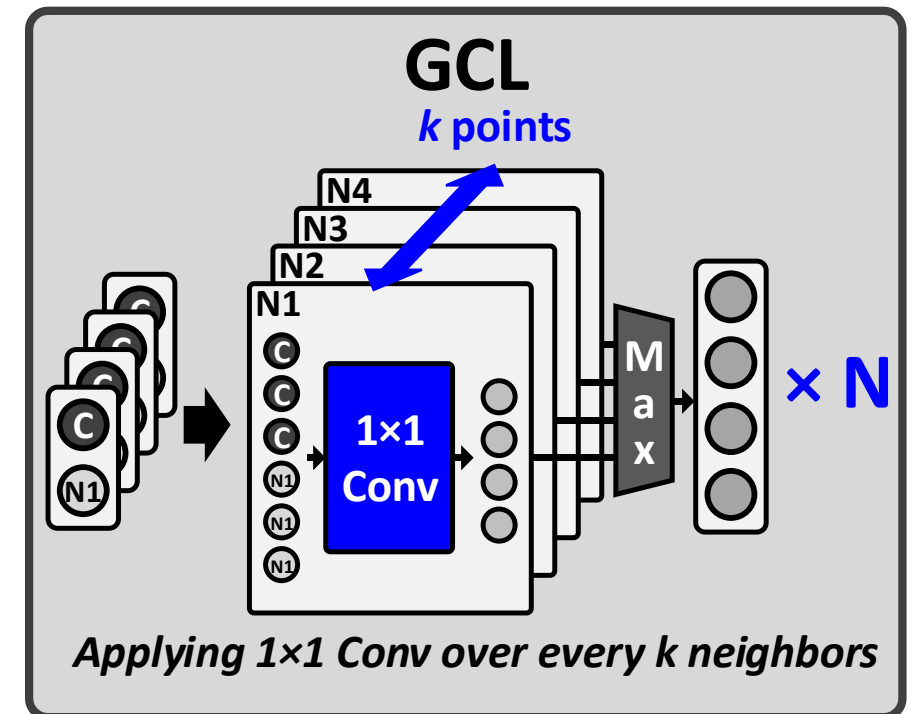
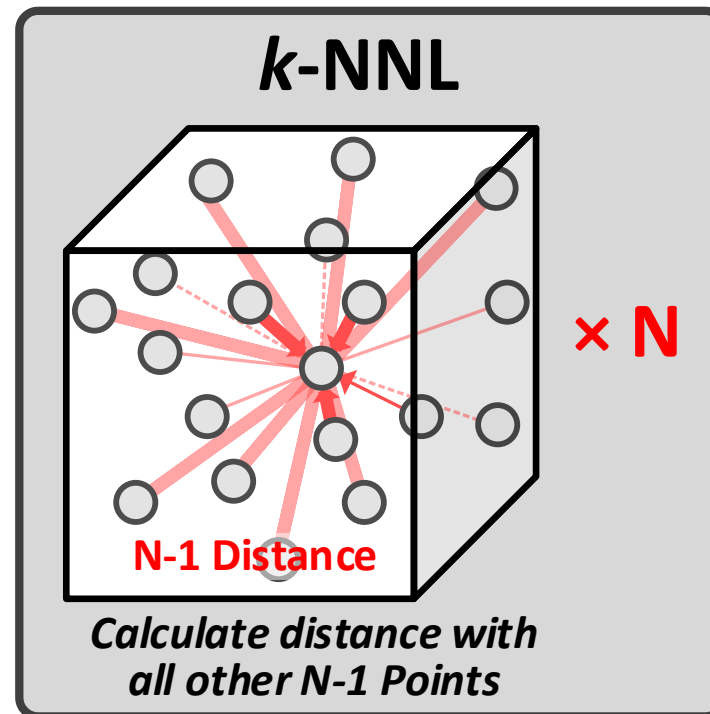
# Design Challenge of GCN

## ❑ Heavy Computation Amount (~506.8 GOPS)

- **k-NNL**: Calculate the distance from  $N$  points to all other  $N-1$  points ( $\propto N^2$ )
- **GCL**:  $1 \times 1$  Conv. over every  $k$  neighbors for a single center point ( $\propto k$ )



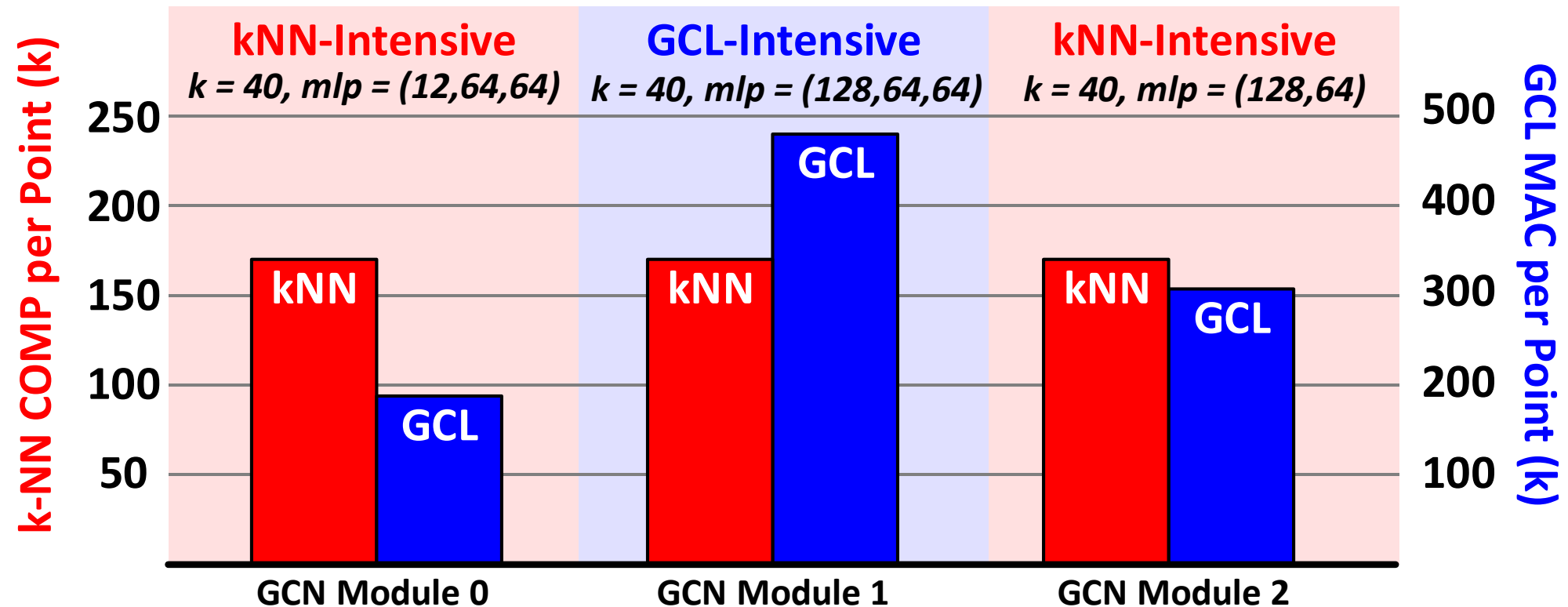
Computational Breakdown\*



# Design Challenge of GCN

## ❑ Heterogeneous Characteristics of the GCN Module

- Depending on the layer configuration, module can be either **kNN-intensive** & **GCL-intensive**



# Contribution of This Work

---

- ❑ **Sparse Grouping based Dilated Graph Convolution**
  - *For Computation Reduction*
  - Dividing input point cloud into multiple sparse point cloud
- ❑ **Group-level Pipelining**
  - *For high utilization*
  - Balancing workload of heterogeneous core

**A 98.9mW GCN based 3D PCSS Processor  
with 30 fps Real-time Operation**

# Overall Architecture

## ❑ $k$ -NN Core ( $k$ -NNC)

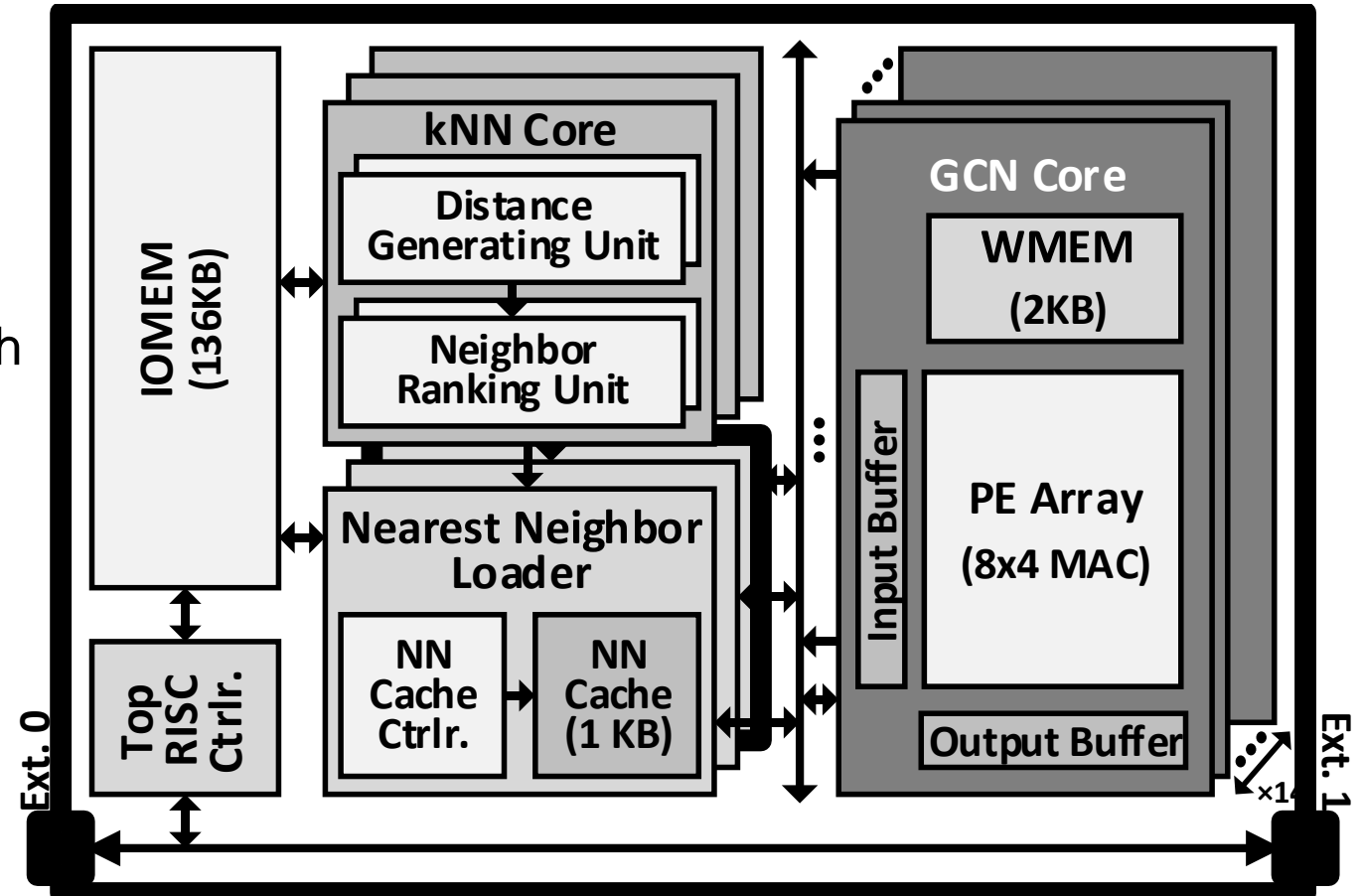
- For  $k$ -NN graph generation layer

## ❑ Nearest Neighbor Loader

- Load neighbor to cache using graph
- Hold neighbor to be used by GCC

## ❑ Graph Convolution Core (GCC)

- For graph convolution layer

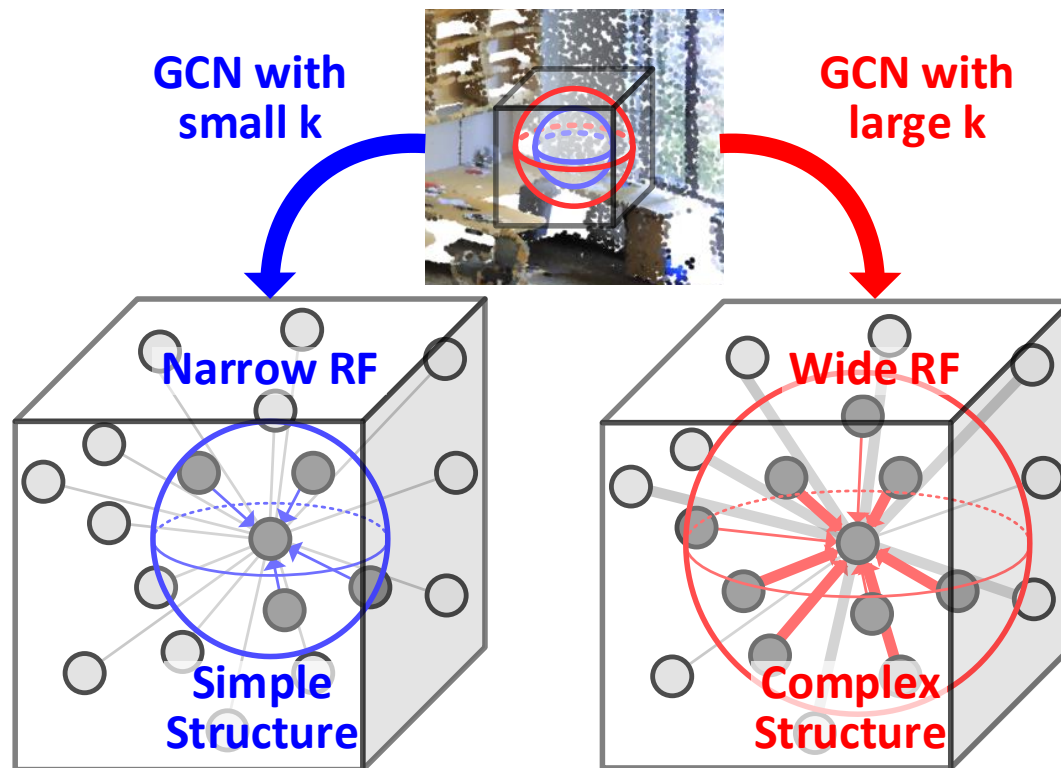




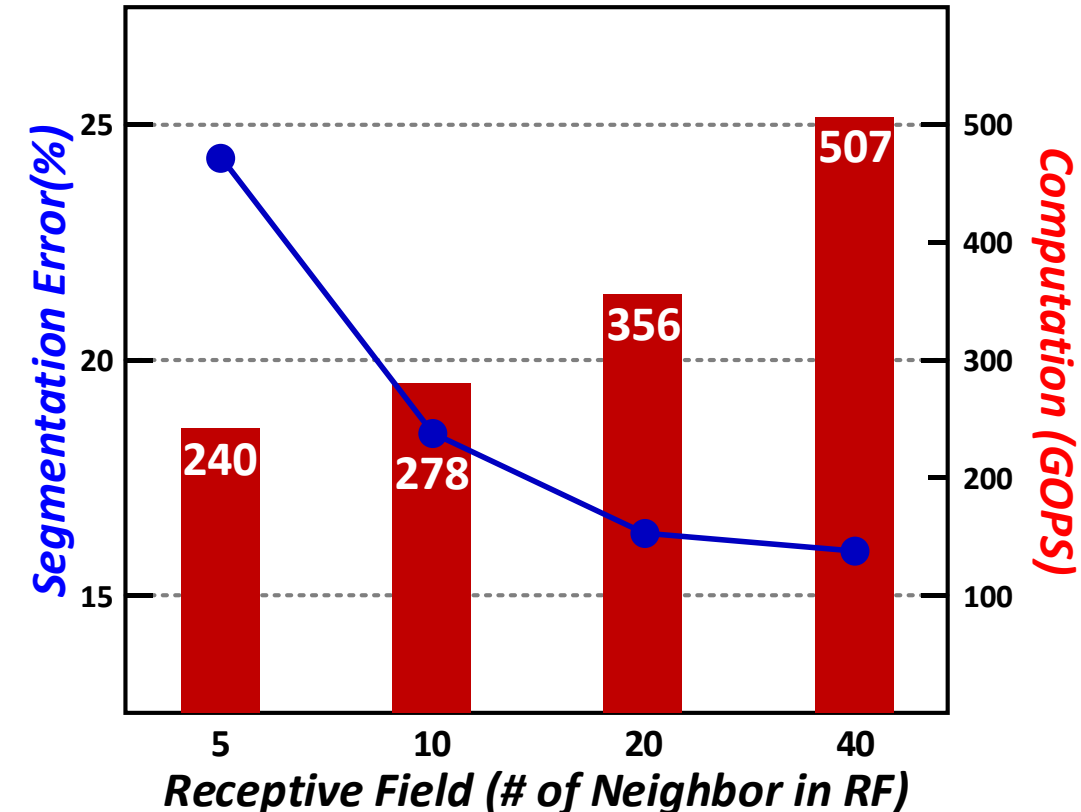
## Effect of $k$ on 3D PCSS

### □ Effect of $k$ on 3D PCSS

- Large  $k \rightarrow$  Inducing wide RF\*  $\rightarrow$  Understanding of complex structures 😊
- Large  $k \rightarrow$  # of points to be calculated $\uparrow \rightarrow$  High computational cost 😞



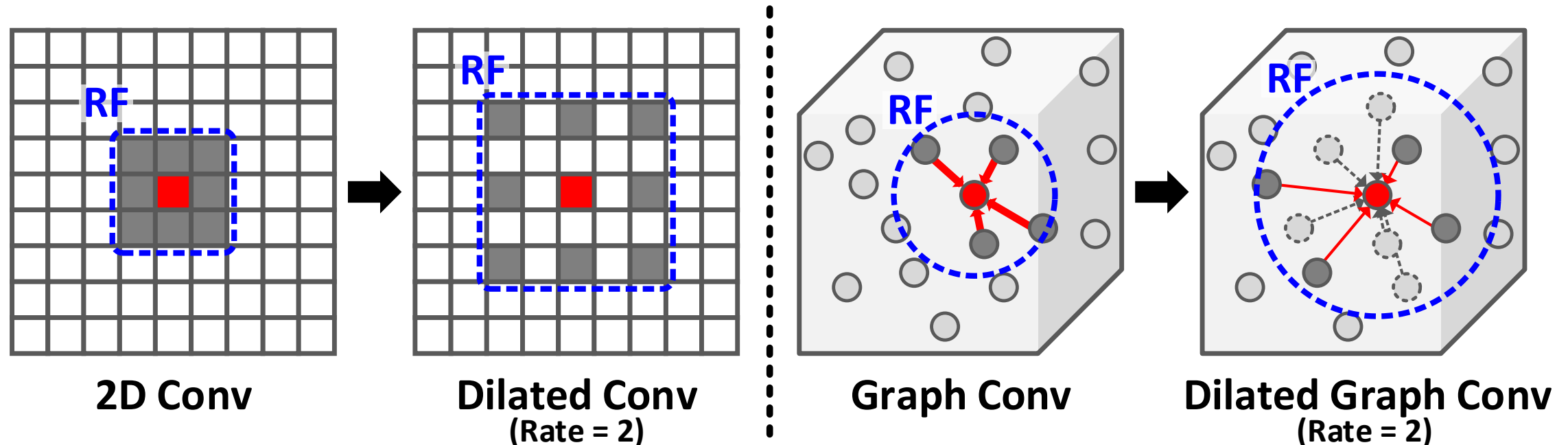
\* RF: Receptive field



# Dilated Graph Convolution

## ❑ Duplicate Information between Neighbor Points

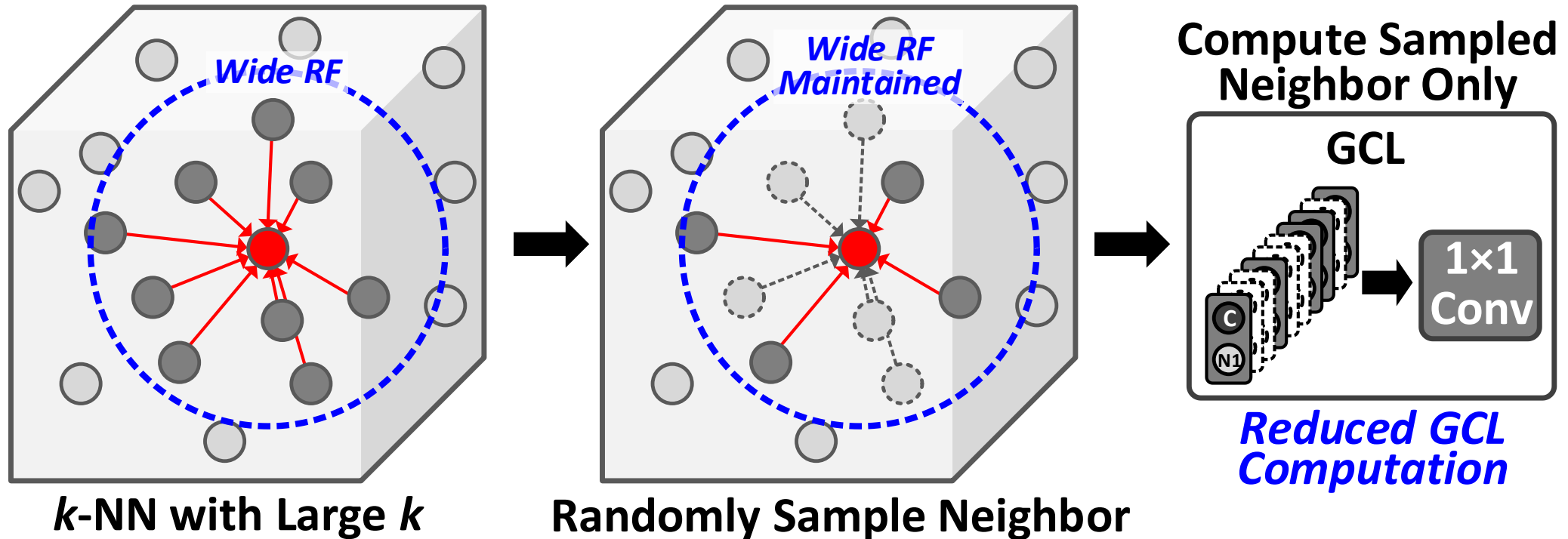
- Not all  $k$  points contain unique information
- Like dilated-conv in 2D CNN, receive information from sampled pixels



# Dilated Graph Convolution

## □ Dilated Graph Convolution (DGC) Operation

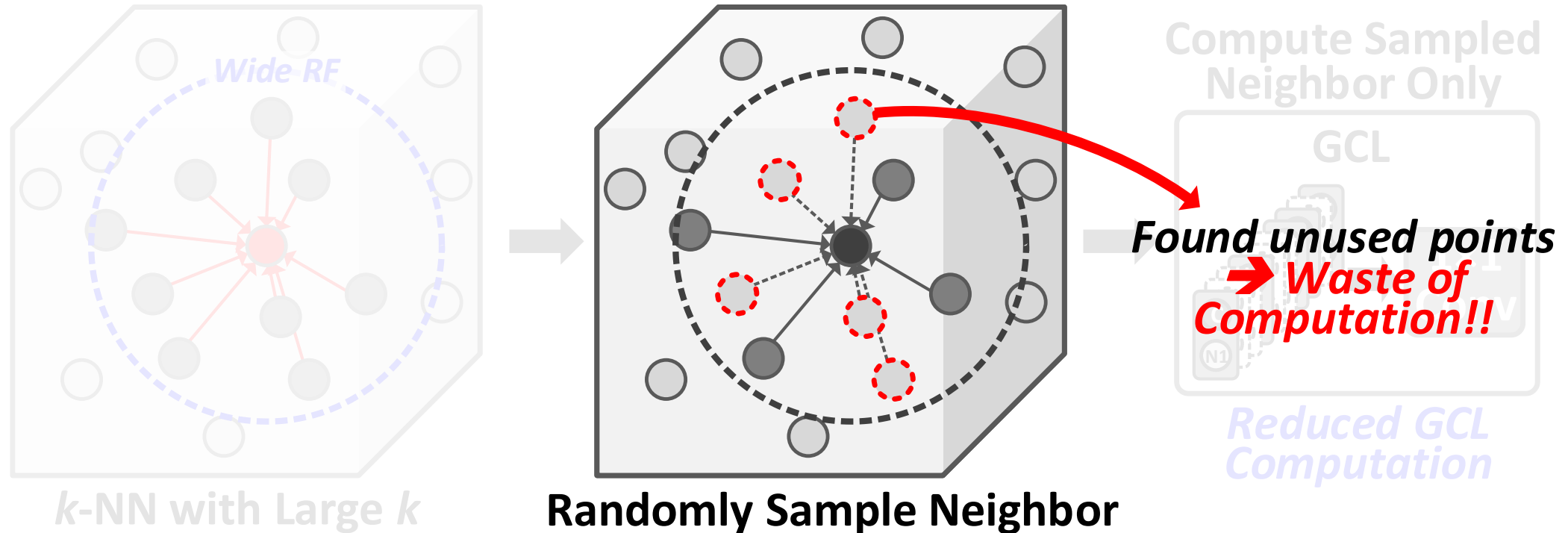
- Generate  $k$ -NN graph with large  $k$  for wide RF
- Randomly select neighbor among selected  $k$  points → **Reduced GCL Cost**



# Dilated Graph Convolution

## □ Dilated Graph Convolution (DGC) Operation

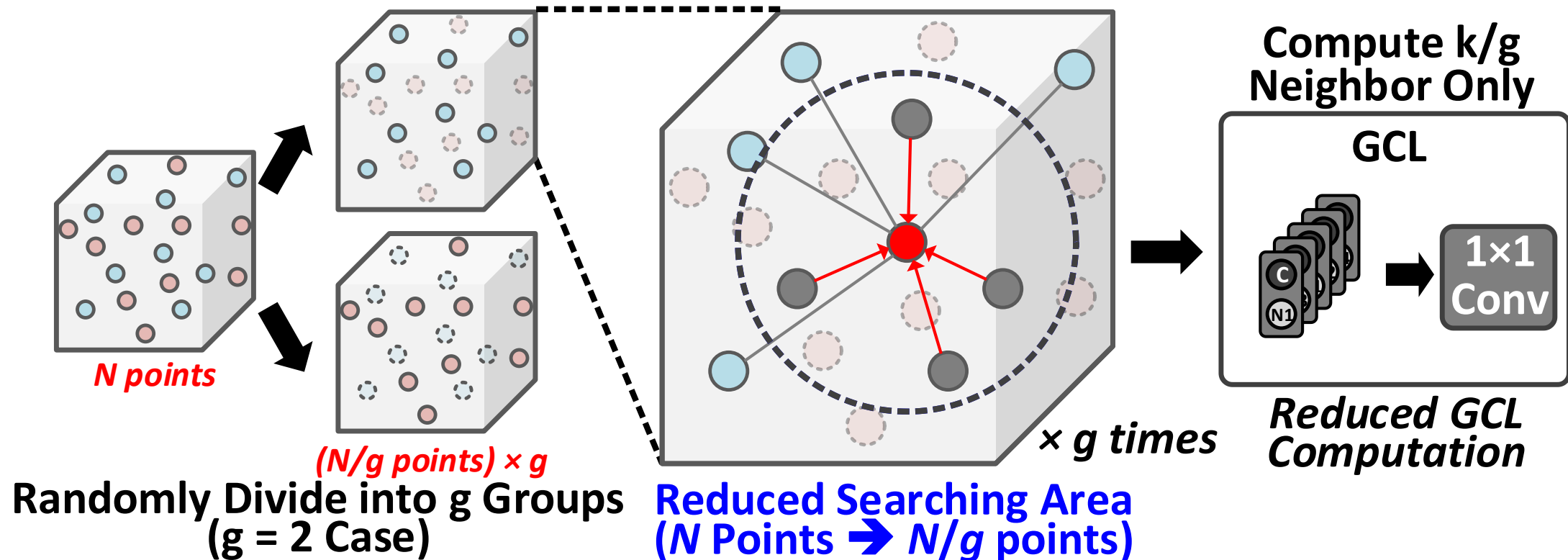
- Generate  $k$ -NN graph with large  $k$  for wide RF → **Same k>NNL Cost**
- Randomly select neighbor among selected  $k$  points → **Reduced GCL Cost**



# Sparse-Grouping based DGC

## □ Sparse Grouping based DGC (SG-DGC) Operation

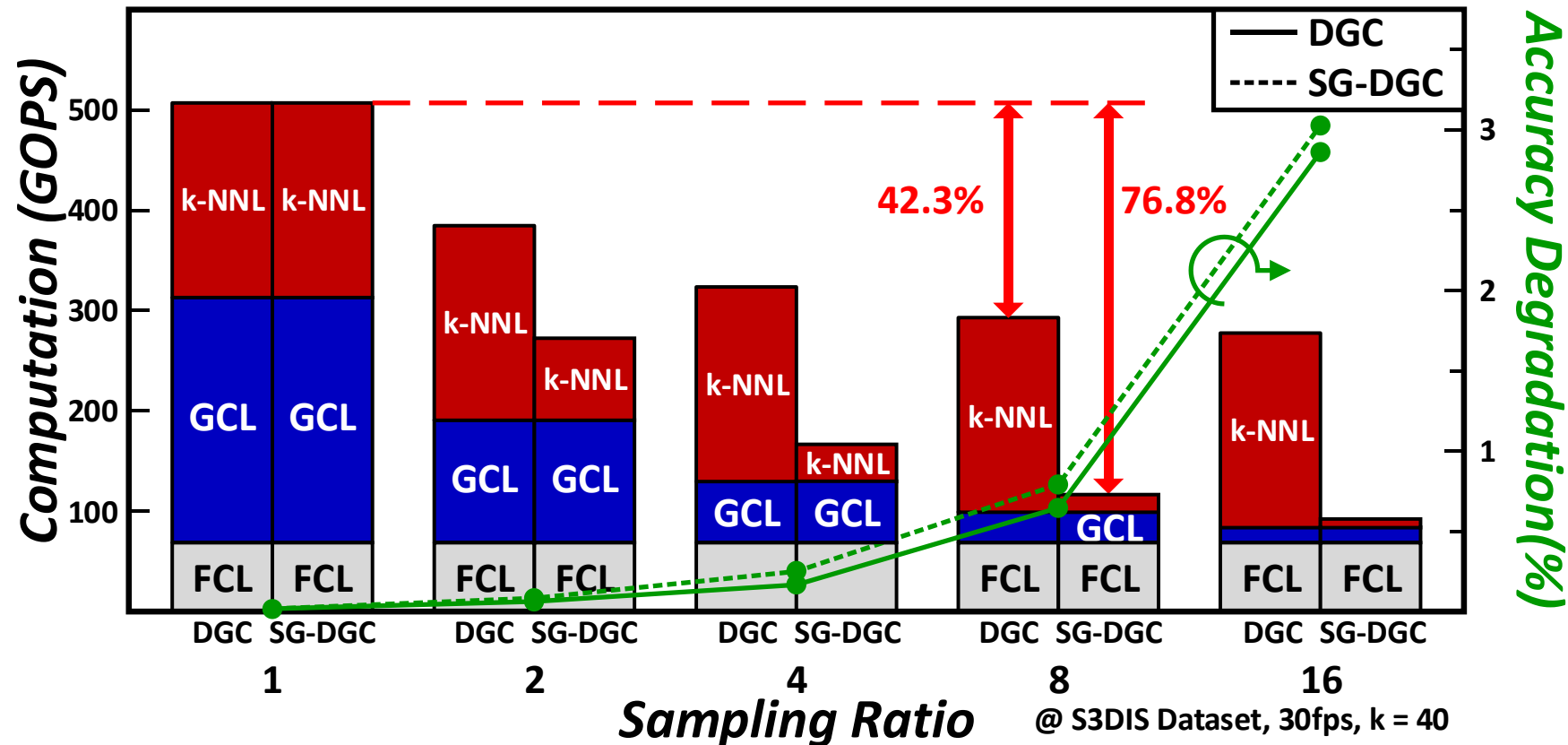
- Divide points into  $g$  groups, and find only  $(k/g)$  neighbor in a group ( $N/g$ )
- k-NN cost reduced from  $N^2$  to  $(N/g)^2 \times g \rightarrow$  **Reduced k-NN Cost**



## Sparse-Grouping based DGC

### □ Computation Reduction with SG-DGC

- Reduced both GCL and  $k$ -NN computation
- **Total 76.8% of computation reduced with  $\leq 1\%$  accuracy drop @  $SR^* = 8$**



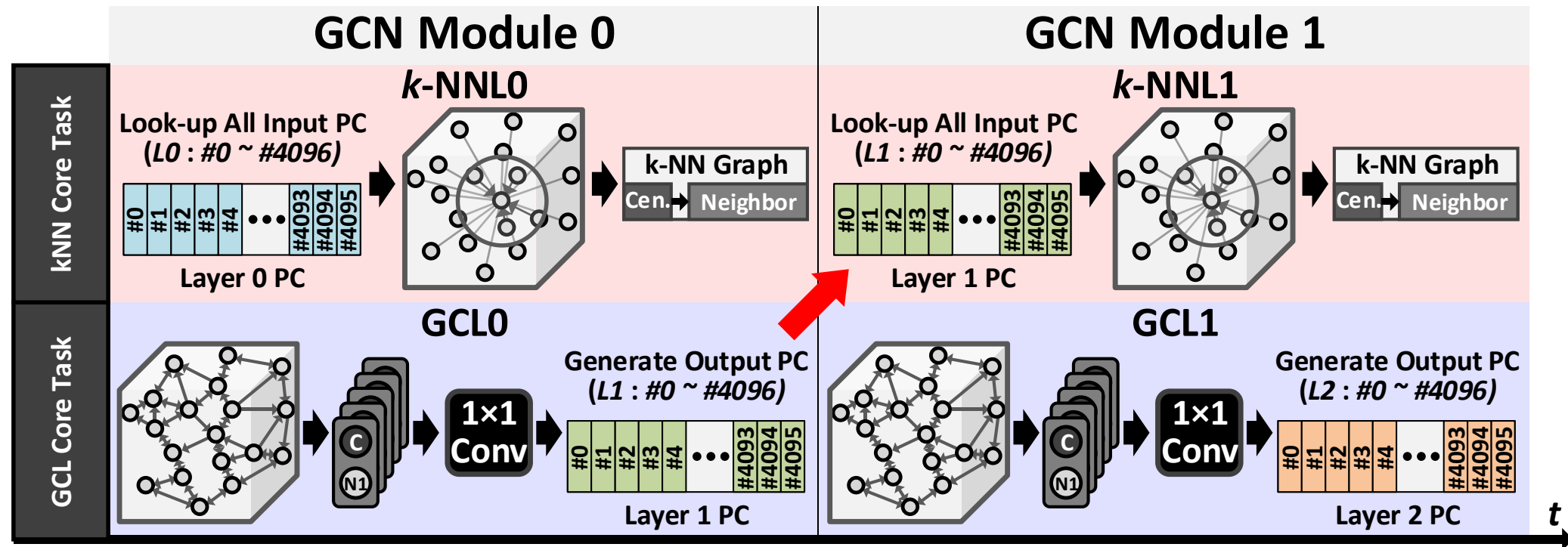
\* SR: Sampling Ratio



# Data Dependency of GCN Modules

## Serial Execution of GCN Modules by Data Dependency

- $k$ -NNL look-up all other points for finding neighbor
- For performing  $k$ -NN, previous GCL operation must be completed

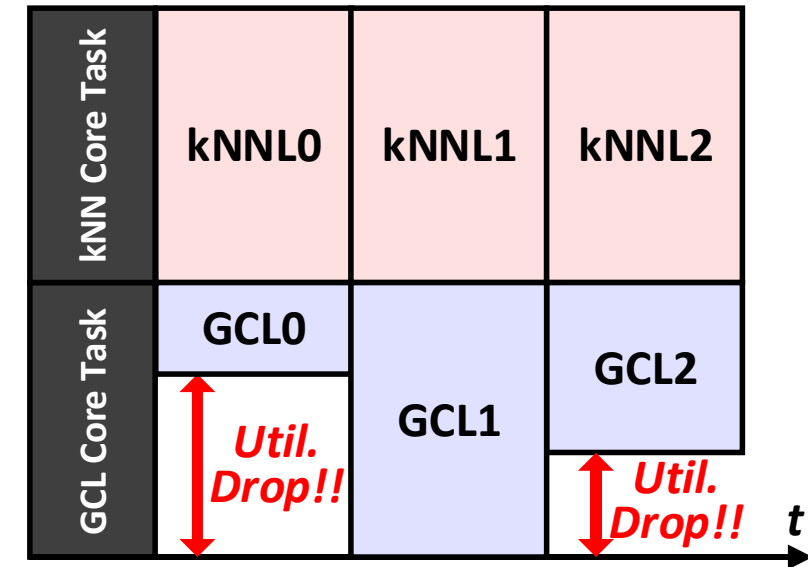
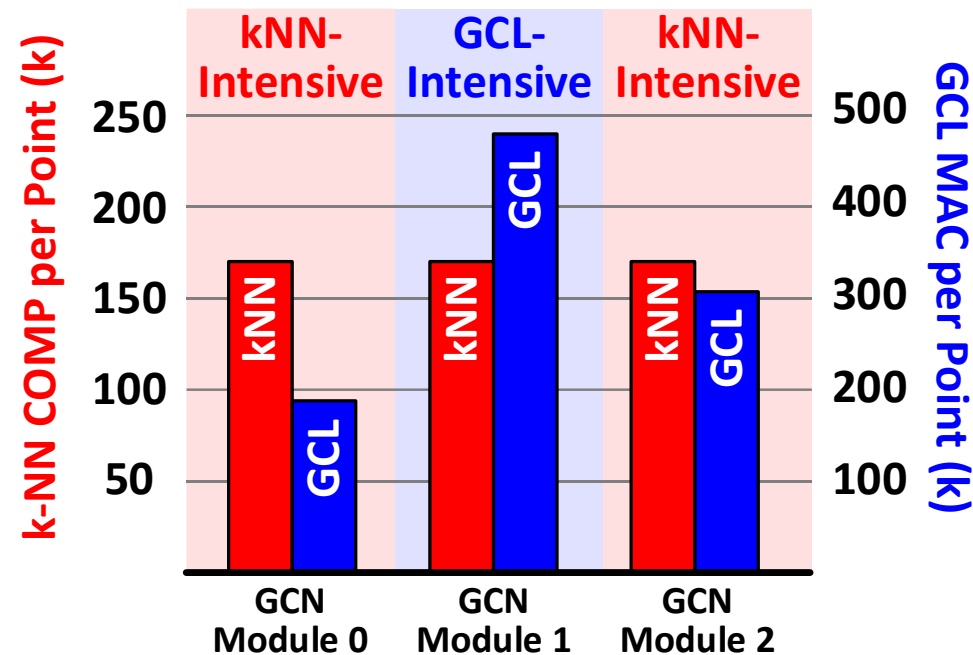


**Data Dependency!! → Sequential Execution**

# Heterogeneous Characteristic

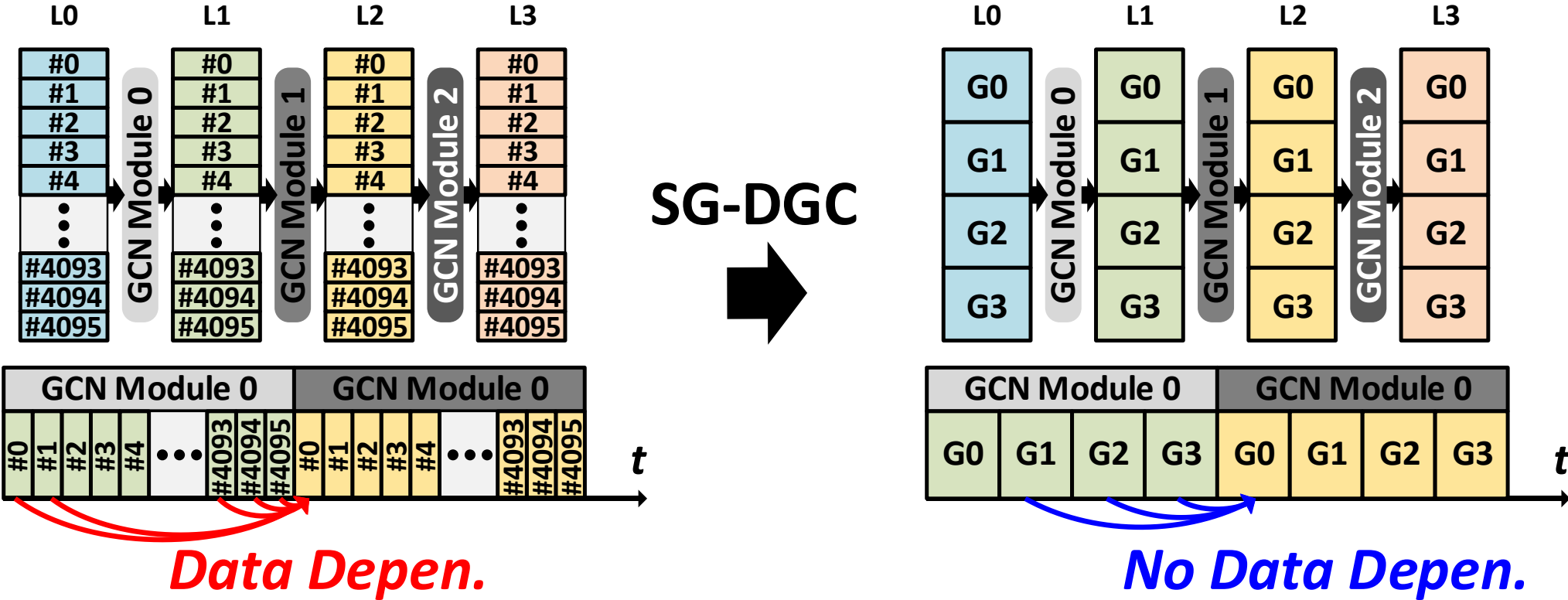
## ❑ Utilization Drop by Unbalanced Workload

- $k$ -NNL workload depends on only # of point in point cloud &  $k$
- GCL workload depends on layer config. (# layer,  $C_{in}$ ,  $C_{out}$ )
- ➔ Only workload of GCC varying and **GCC utilization drop**



# Data Dependency with SG-DGC

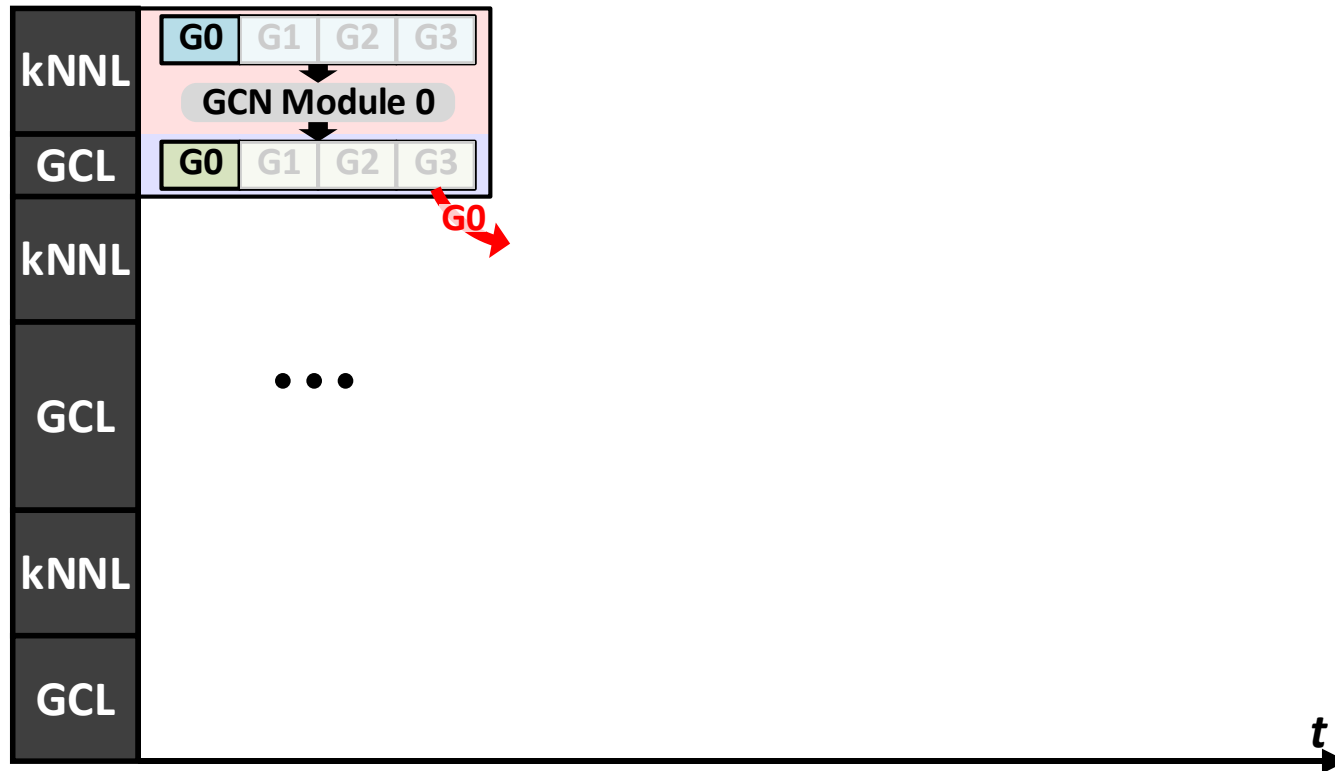
- ❑ Utilize Characteristic of SG-DGC
  - If GCL group  $g_i$  is completed, next  $k$ -NNL with  $g_i$  can be performed
  - No need to operate sequentially between different groups



## Group-level Pipelining

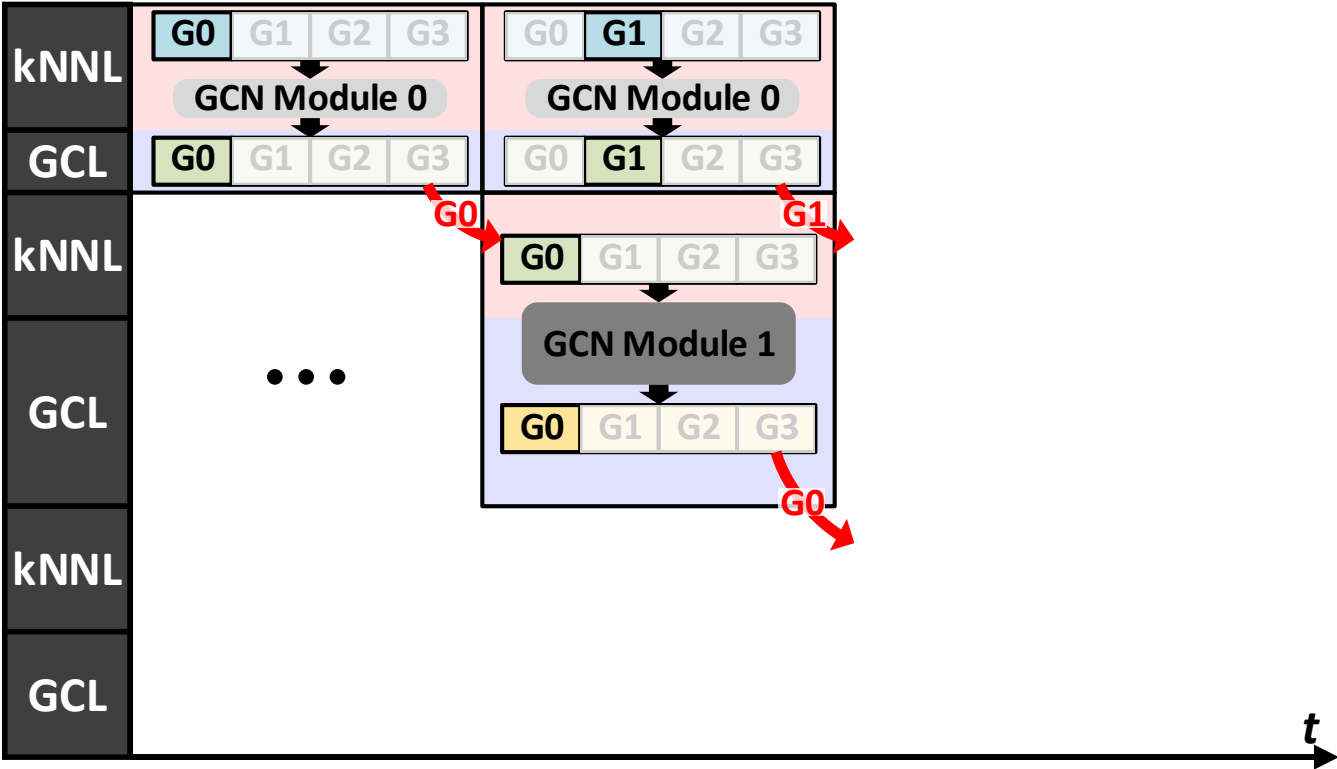
### □ Group-level Pipelining: Utilize Characteristic of SG-DGC

- Ensure all layers are always running with different group
- Utilization increased from 68.5% to 89.6%



# Group-level Pipelining

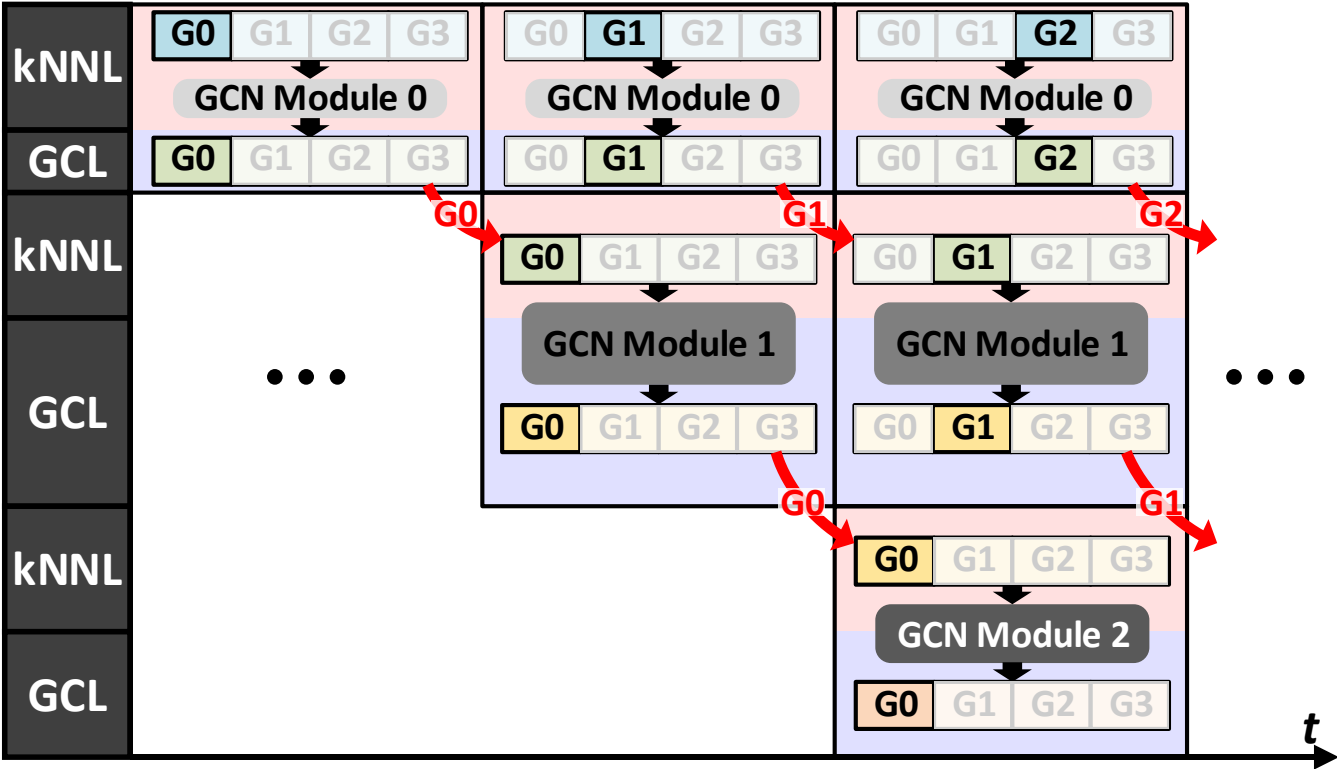
- ❑ **Group-level Pipelining: Utilize Characteristic of SG-DGC**
  - Ensure all layers are always running with different group
  - Utilization increased from 68.5% to 89.6%



# Group-level Pipelining

❑ **Group-level Pipelining: Utilize Characteristic of SG-DGC**

- Ensure all layers are always running with different group
- Utilization increased from 68.5% to 89.6%

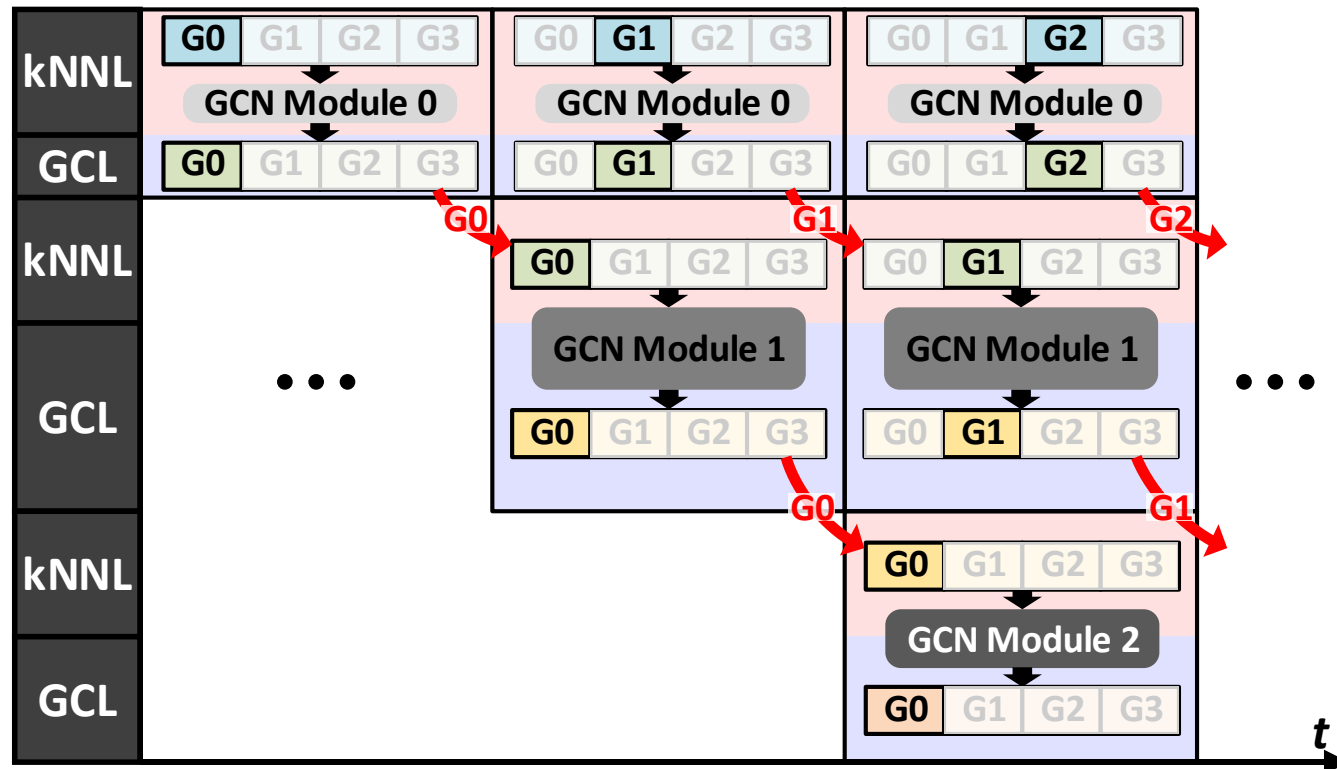




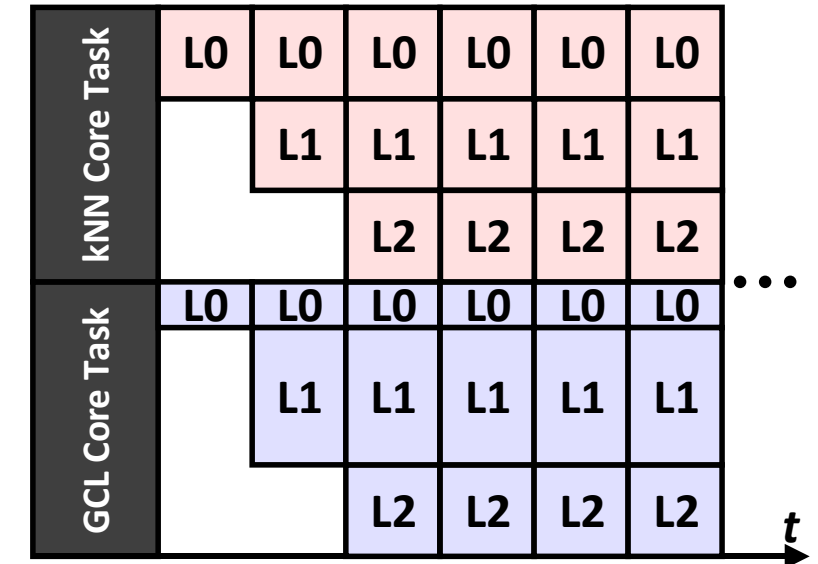
# Group-level Pipelining

- ❑ **Group-level Pipelining: Utilize Characteristic of SG-DGC**

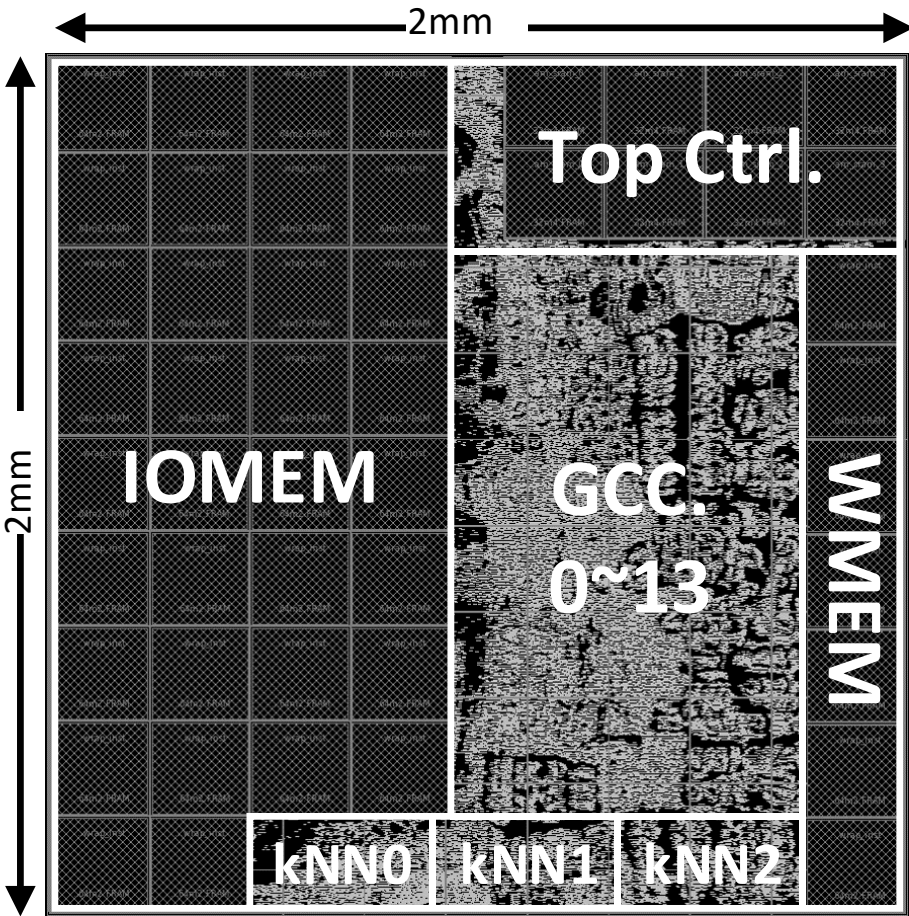
- Ensure all layers are always running with different group
- Utilization increased from 68.5% to 89.6%



**High Utilization with  
Group-level Pipelining!!**



# Chip Layout and Summary



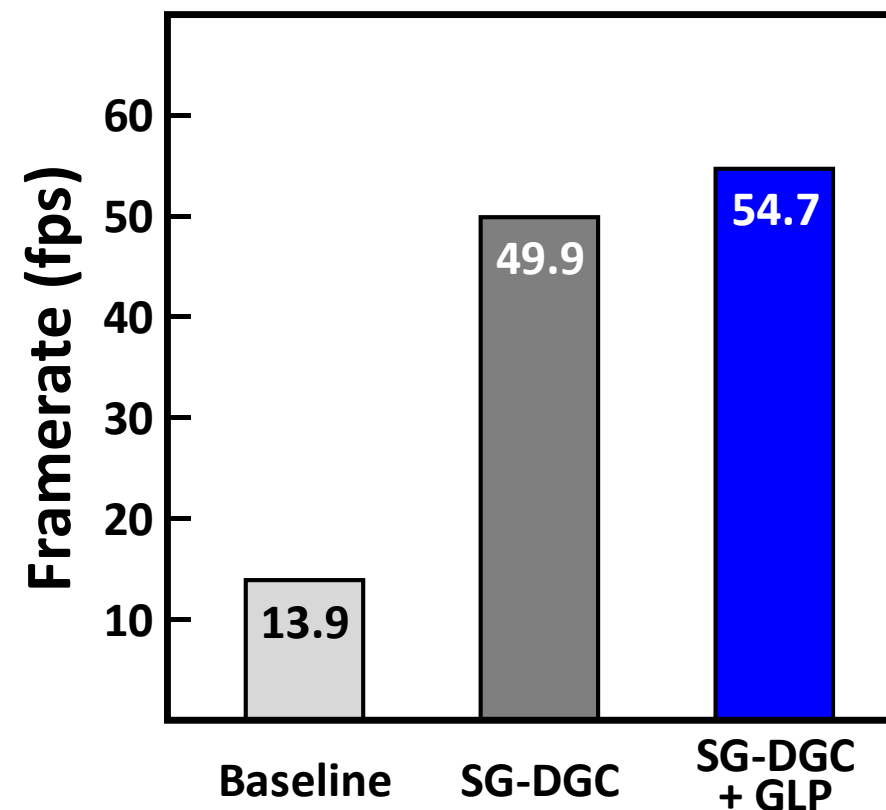
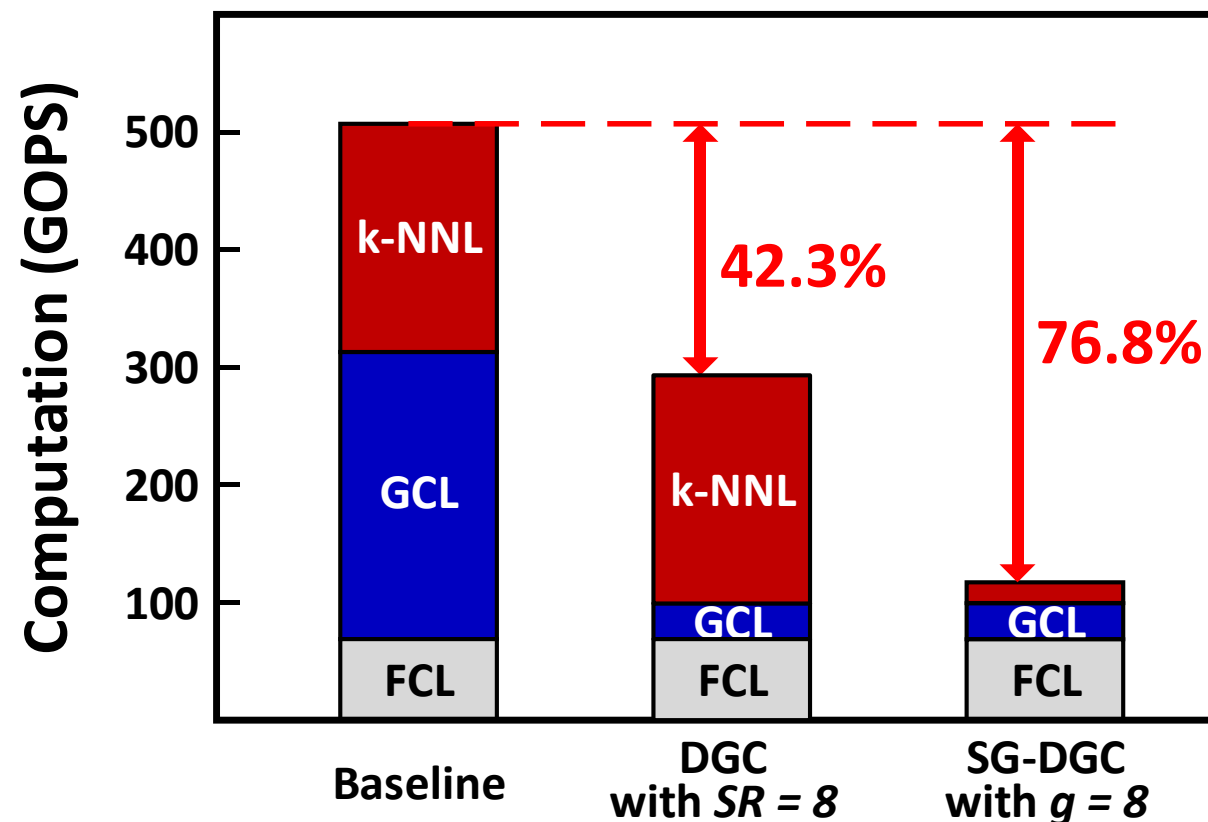
Technology	65nm CMOS
Operating Condtion	1.1V, 111-200MHz
Area	4mm <sup>2</sup>
Power	98.9mW @ 30fps 176mW @ 54.7fps
Framerate	30fps @ 111MHz 54.7 fps @ 200MHz
Power Efficiency	0.97TOPS/W <sup>*</sup> 2.76TOPS/W <sup>**</sup>
On-Chip SRAM Size	164KB

\* without dilation  
\*\* @ dilation size = 8

# Performance Summary


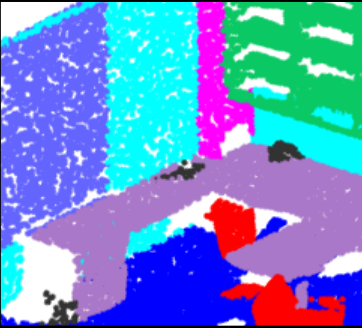
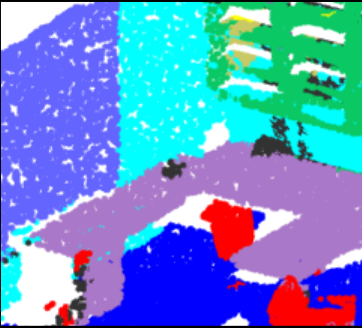
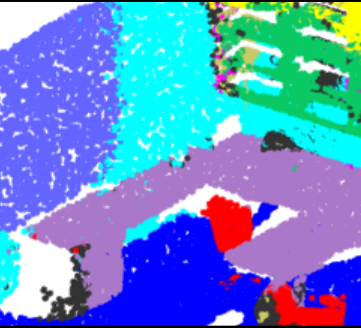

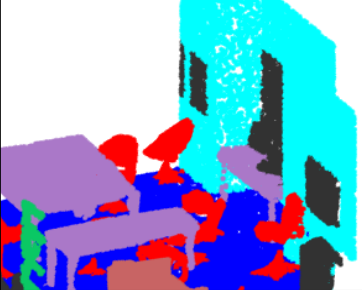
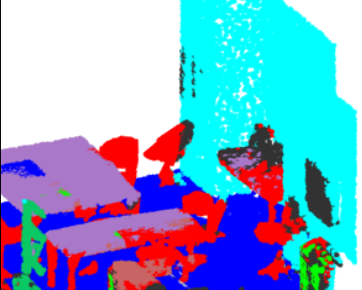
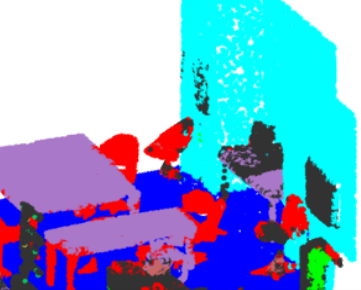
## ❑ Computation Reduction & Framerate Enhancement

- Reduce 76.8% of computation
- Satisfies 54.7fps real-time operation



# Point Cloud Segmentation Result

## ❑ Test Result on Indoor Semantic Segmentation

Input	Point Cloud	Ground Truth	Baseline	Ours
Scene 1				
Scene 2				
Acc.*	-	-	84.1%	83.2%

\* S3DIS Dataset, 4k point input

# Performance Comparison

	TCAS-I D. Im et al	TCAS-I Lyu et al	TCAS-II G. Park et al	This work
Processor/ Process	65nm CMOS Technology	Xilinx UltraScale XCKU115	65nm CMOS Technology	65nm CMOS Technology
Operating Condition	1.2V, 200MHz	350MHz	1.2V, 200MHz	1.1V, 200MHz
Target Application	2D Segmentation	3D Semantic Segmentation (Single Class)	3D Part Segmentation	3D Semantic Segmentation
Algorithm	CNN	3D CNN	Capsule Net	GCN
Semantic Segmentation Accuracy @ 4k S3DIS	-	-	-	83.2
Part Segmentation Accuracy @ 2k Shapenet	-	-	82.44	85.2
Resolution	128×192 Image	180×64×14 3D Image	2K Point Cloud	4K Point Cloud
Framerate (fps)	211	79.4	32.5	30~54.7
Power(mW)	196	12595	94.3 @ 32.5fps	176 @ 54.7 fps 98.9 @ 30 fps
Logical Throughput (GOPS)	639.7 (8b)	685.5 (18b)	108.8 (4b/8b)	486.5 (8b)
Energy Efficiency (TOPS/W)	3.26	0.054	1.15	2.76

# Conclusion

---

❑ A Real-time Point Cloud Semantic Segmentation Processor

❑ Key Features:

**I. Sparse Grouping based Dilated Graph Convolution**

- Reduce computation by 76.8% (@4K input points)

**II. Group-level Pipelining**

- Increase utilization by 21.1%

**A 98.9mW GCN based 3D PCSS Processor  
with 30 fps Real-time Operation**