PointSplit: Towards On-device 3D Object Detection with Heterogeneous Low-power Accelerators

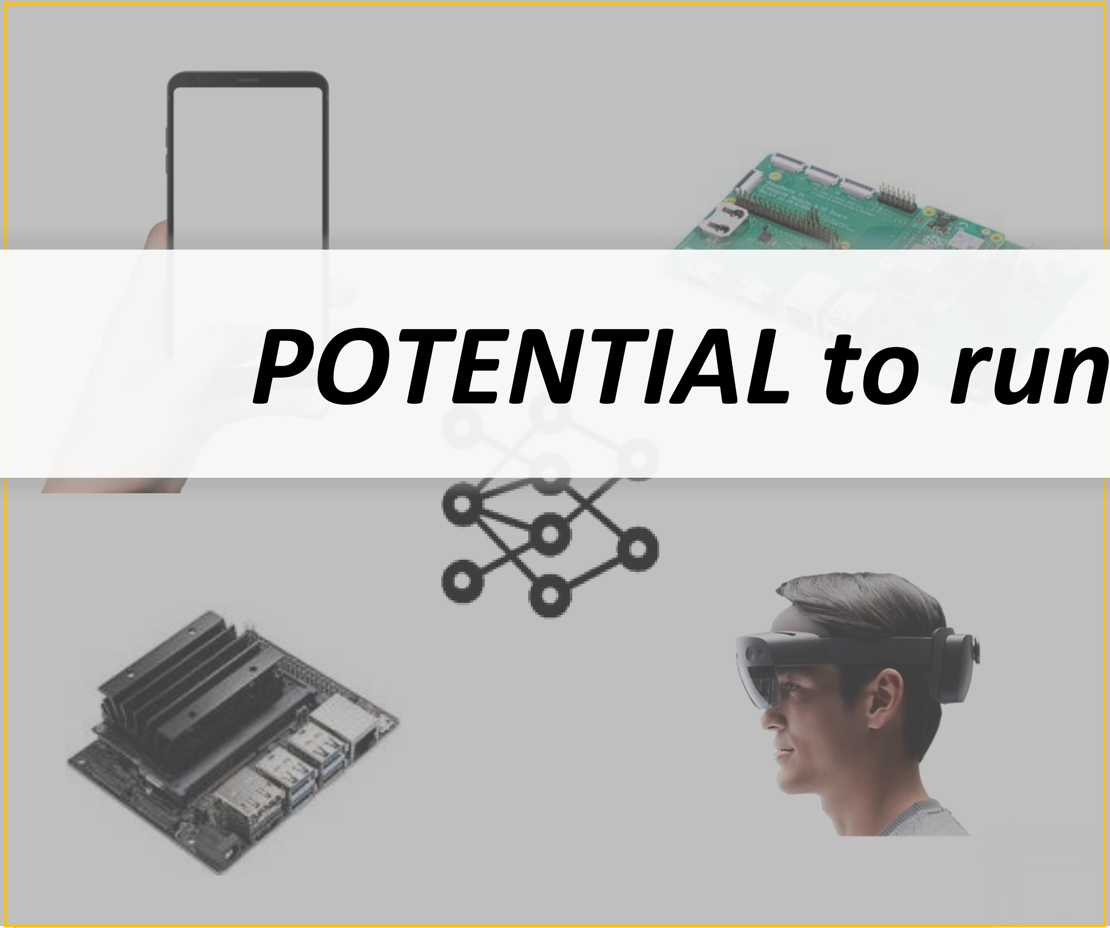
Presented by **Keondo Park**

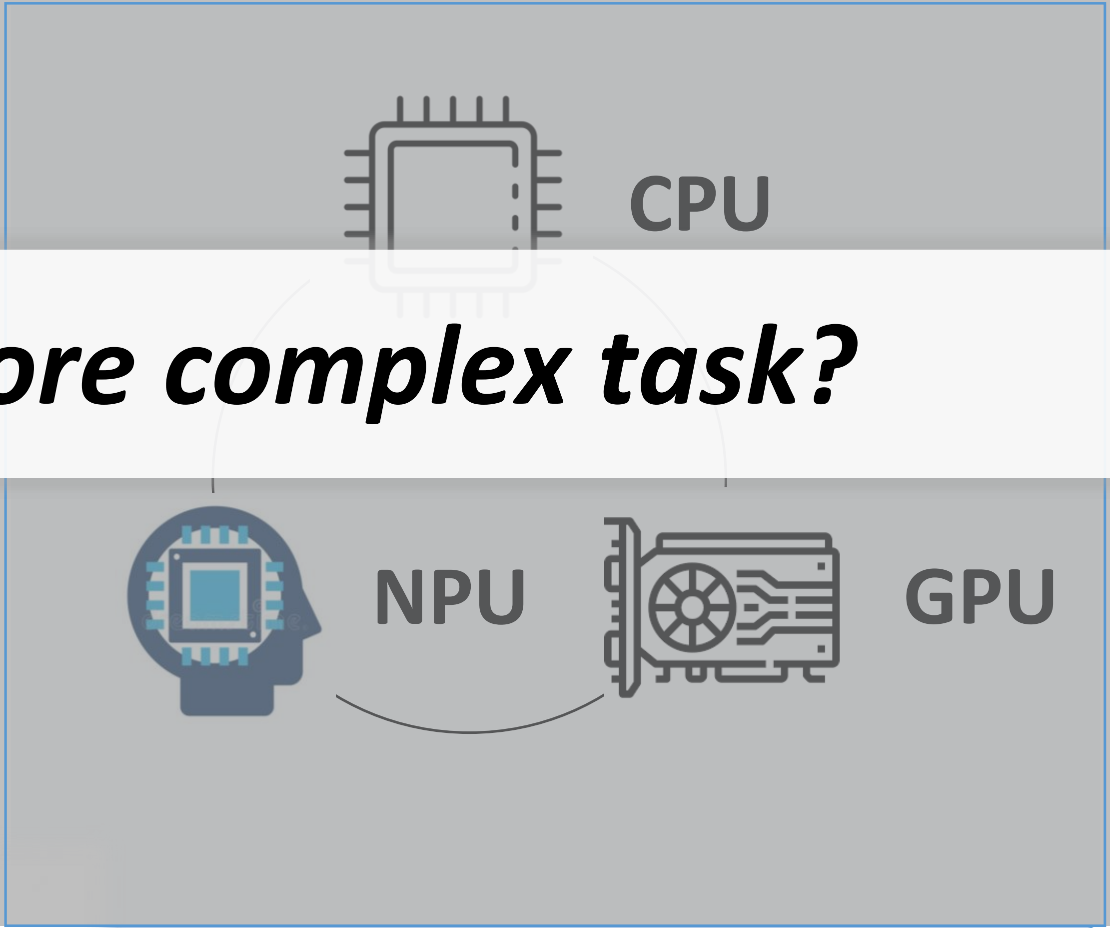
You Rim Choi, Inhoe Lee, Hyung-Sin Kim

May 10, 2023

Seoul National University
Graduate School of Data Science

POTENTIAL to run more complex task?

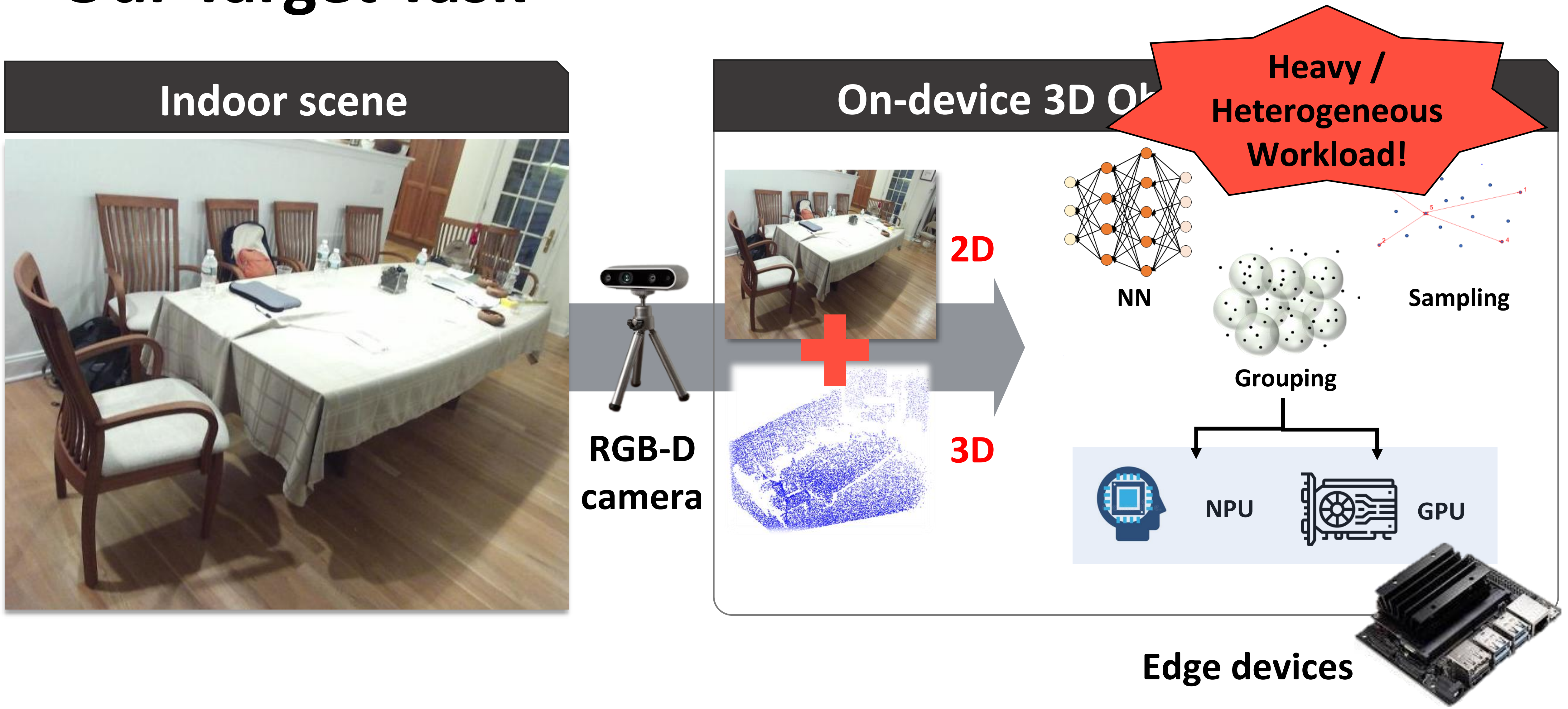




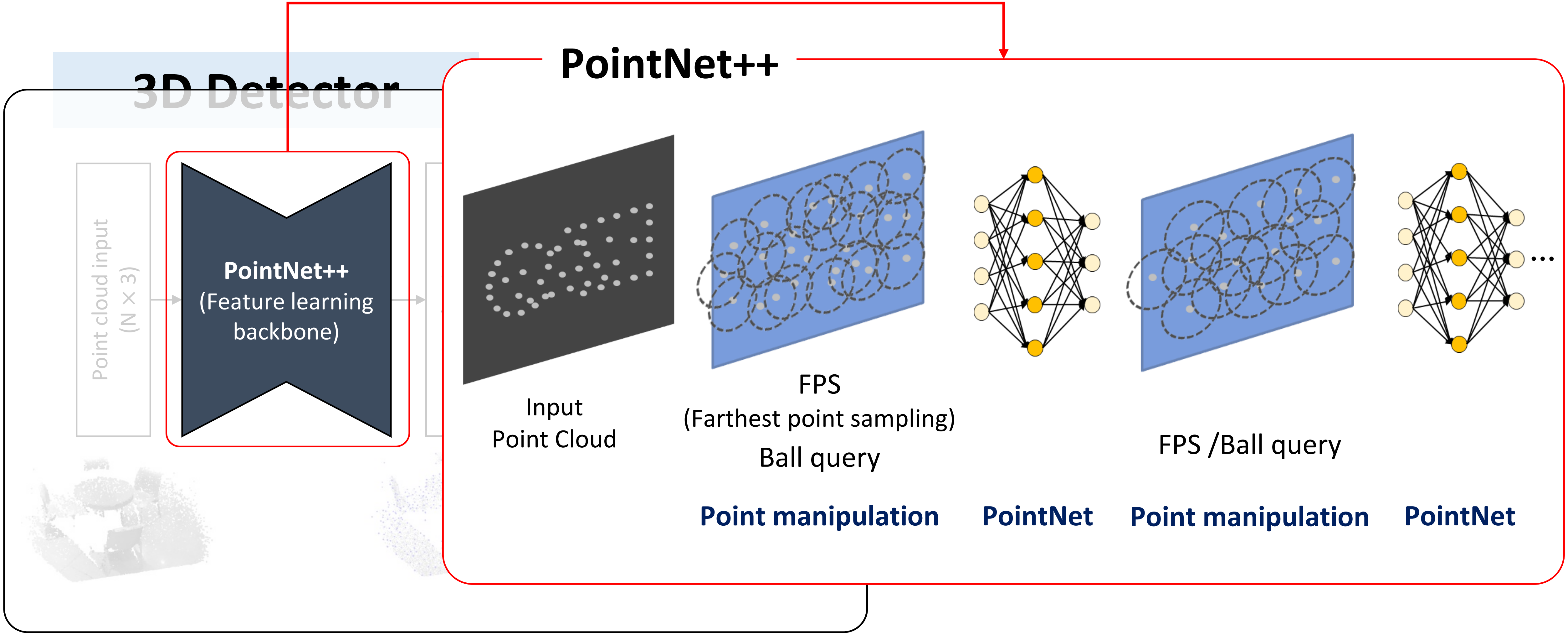
On-device machine learning

Mobile heterogeneous processors

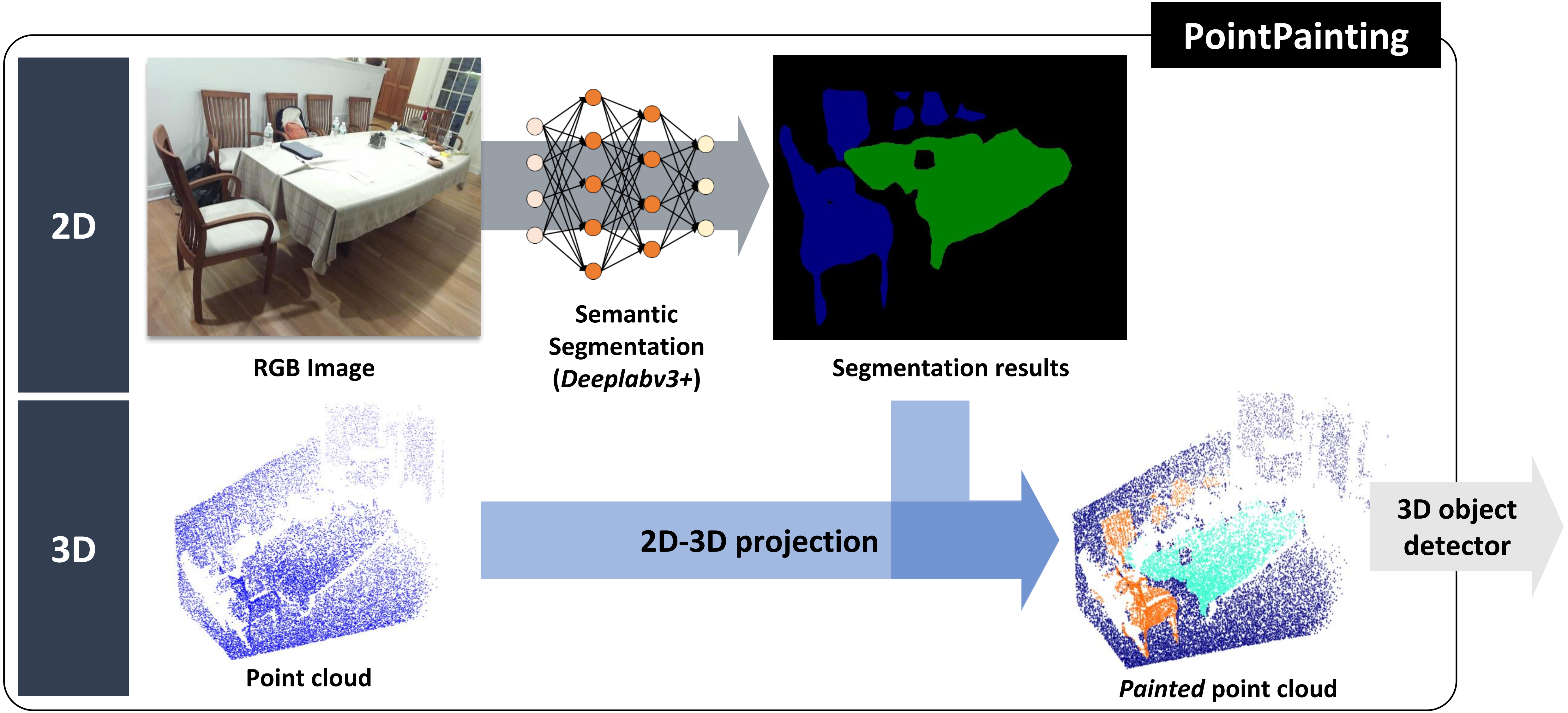
Our Target Task



Baseline: 3D Object Detector

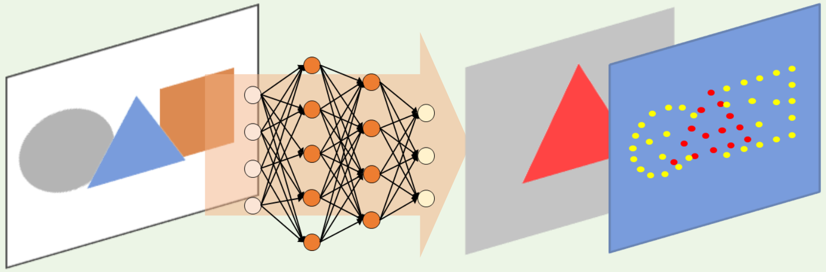


Baseline: 2D + 3D Fusion

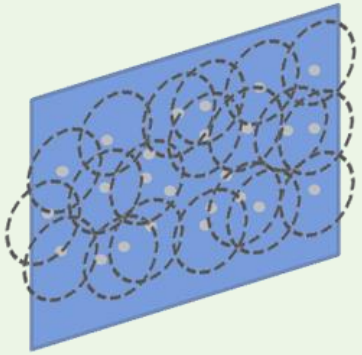


Which Job to Which Processor?

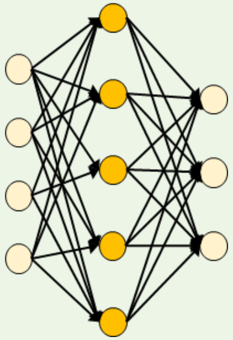
Jobs



PointPainting (NN)

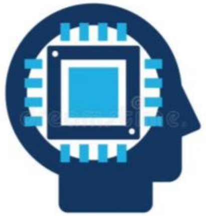


Point manipulation
(FPS, Ball query)

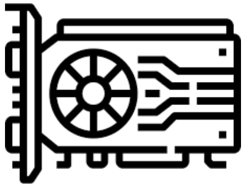


PointNet (NN)

Processor



NPU



GPU

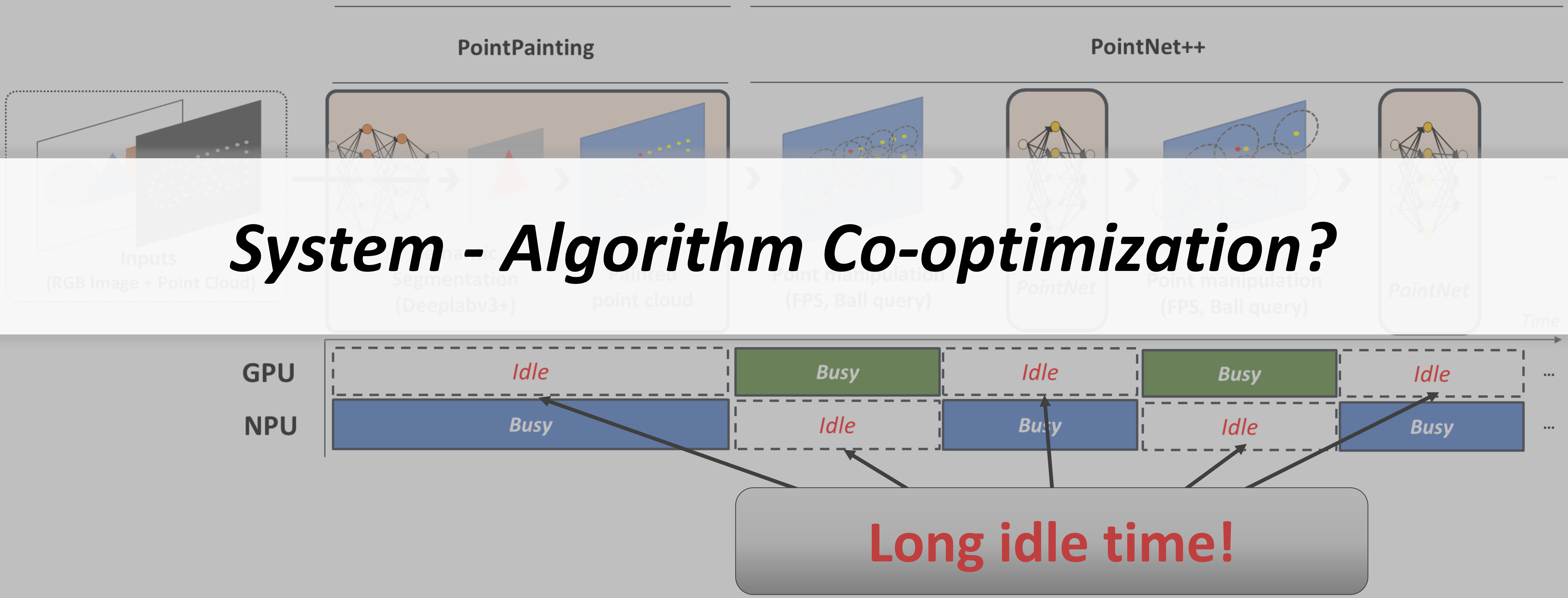
Features

NPU:
- Very fast NN inference
- Limited to NN operations

GPU:
- Slower NN inference
- Wide coverage of operations

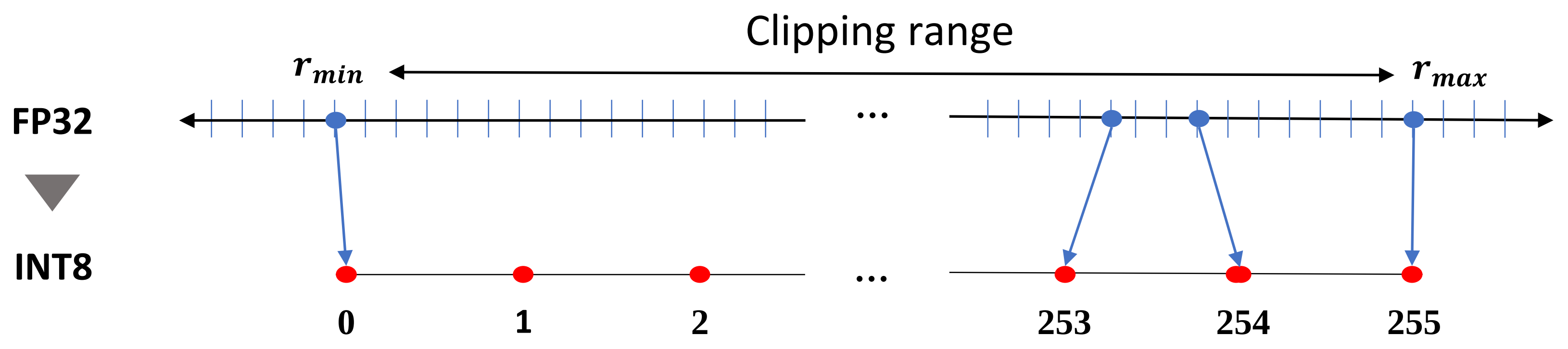
Challenge: Naïve Combination

- PointPainting and PoinetNet++ on GPU and NPU

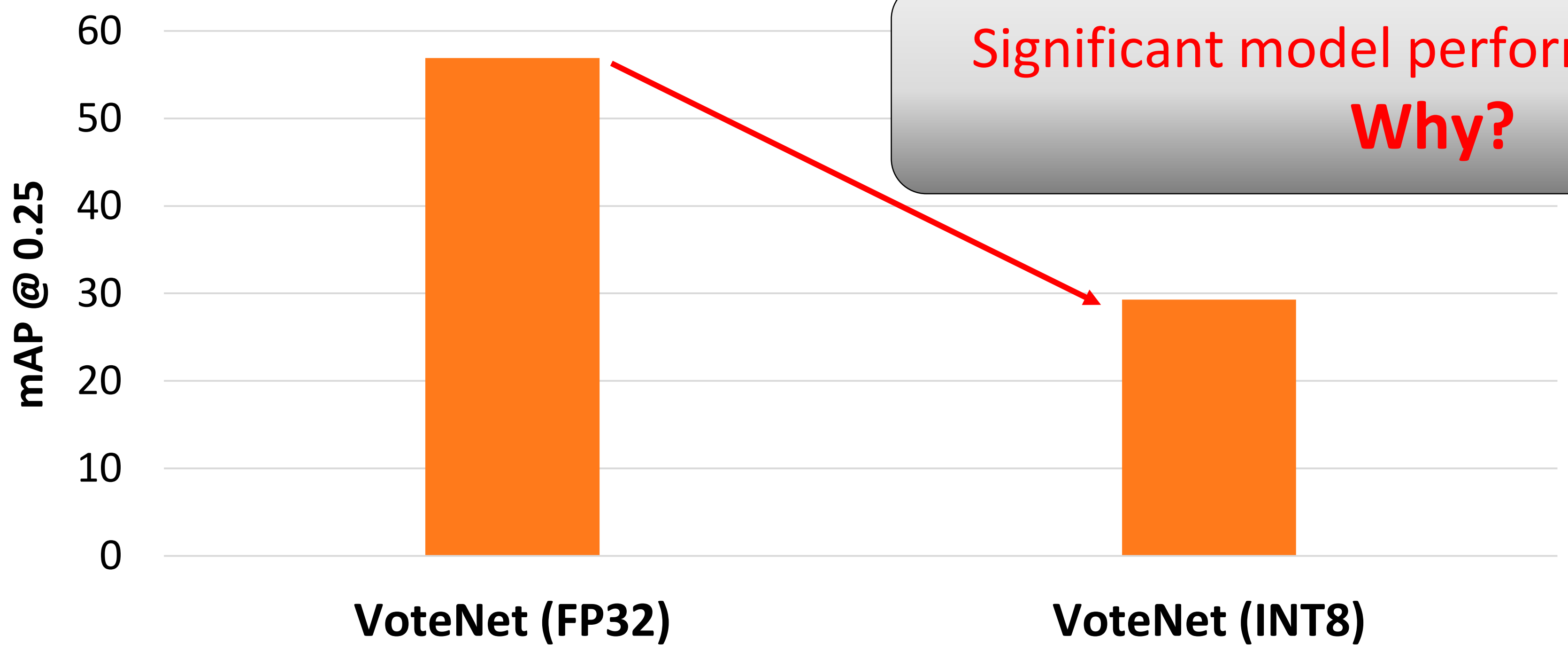


System - Algorithm Co-optimization?

Challenge: Quantization



- Memory ↓ Speed ↑
- Marginal accuracy drop
- Required for ASIC (*e.g.* EdgeTPU)



Significant model performance drop…
Why?

PointSplit

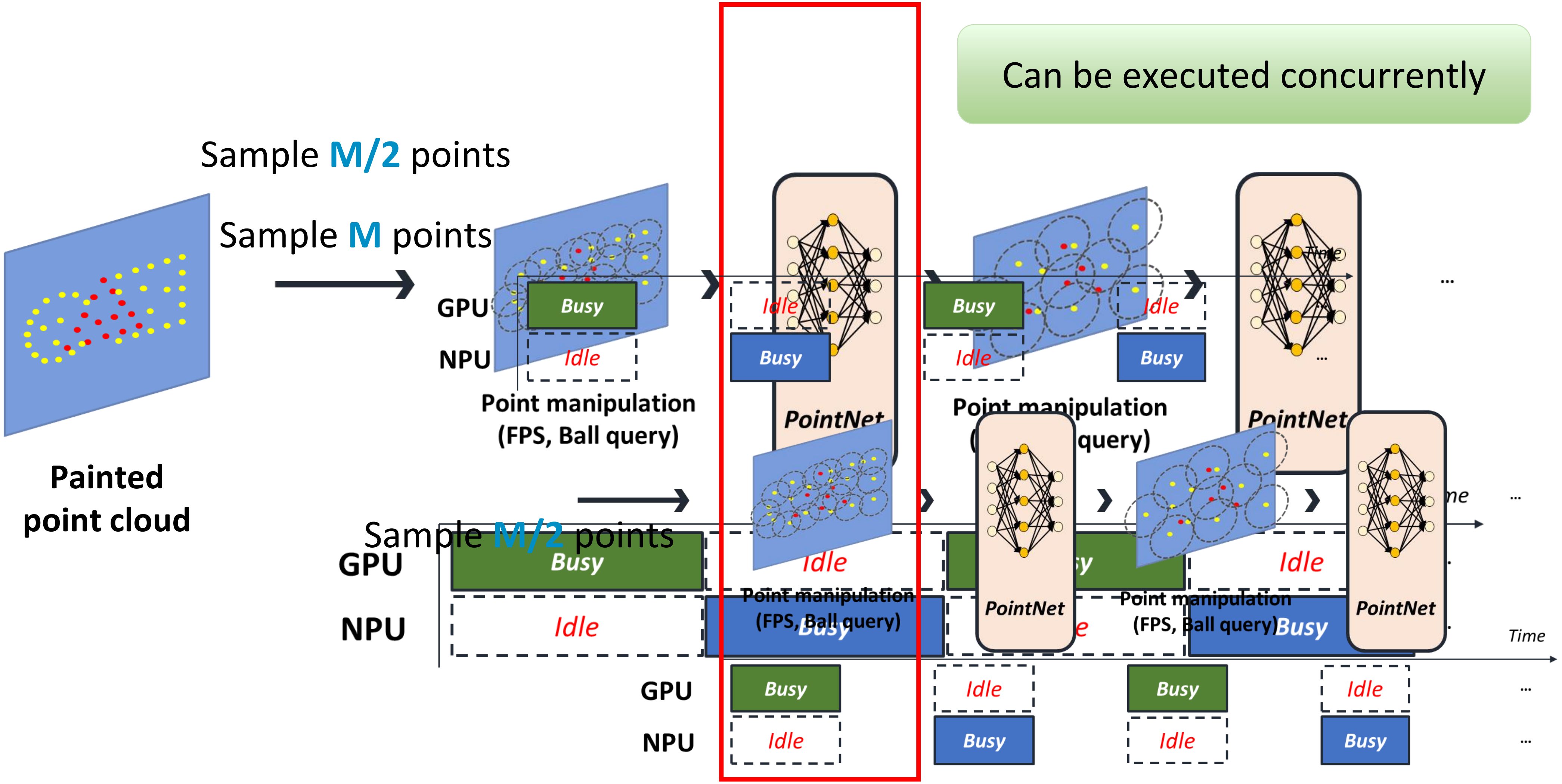
~~Low utilization of system under naïve combination~~

⬇

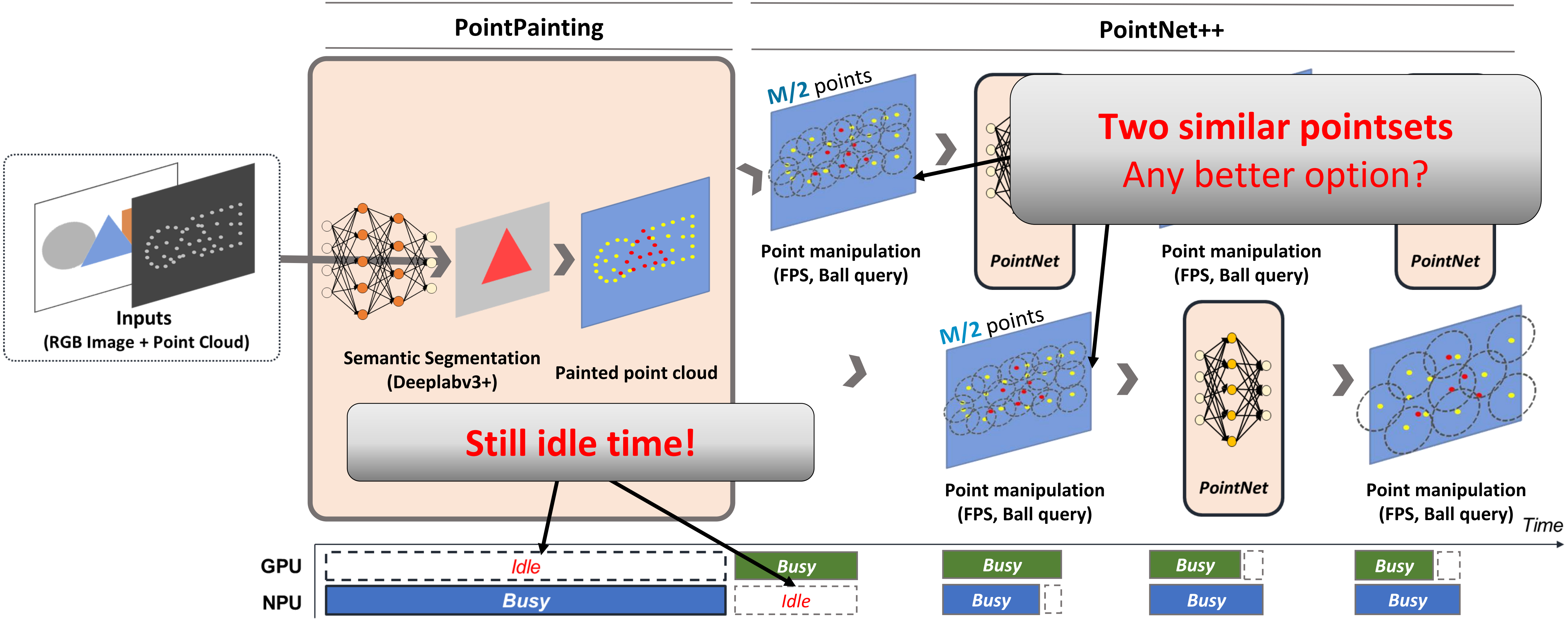
1. System - Algorithm joint optimization

- Biased farthest point sampling
- Parallelizable feature extractor

GPU/NPU Parallelization (Naïve Approach)

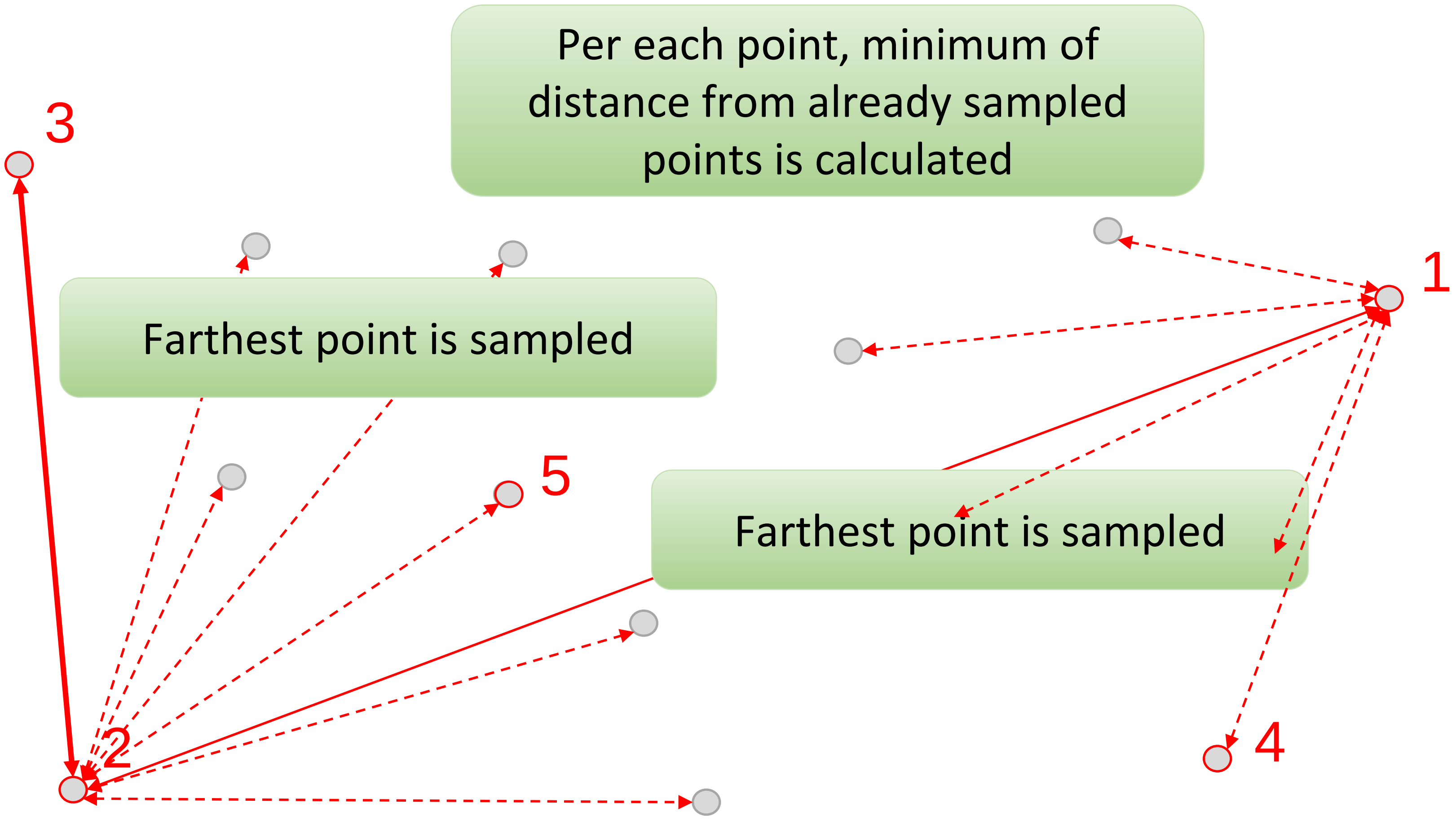


GPU/NPU Parallelization (Naïve Approach)



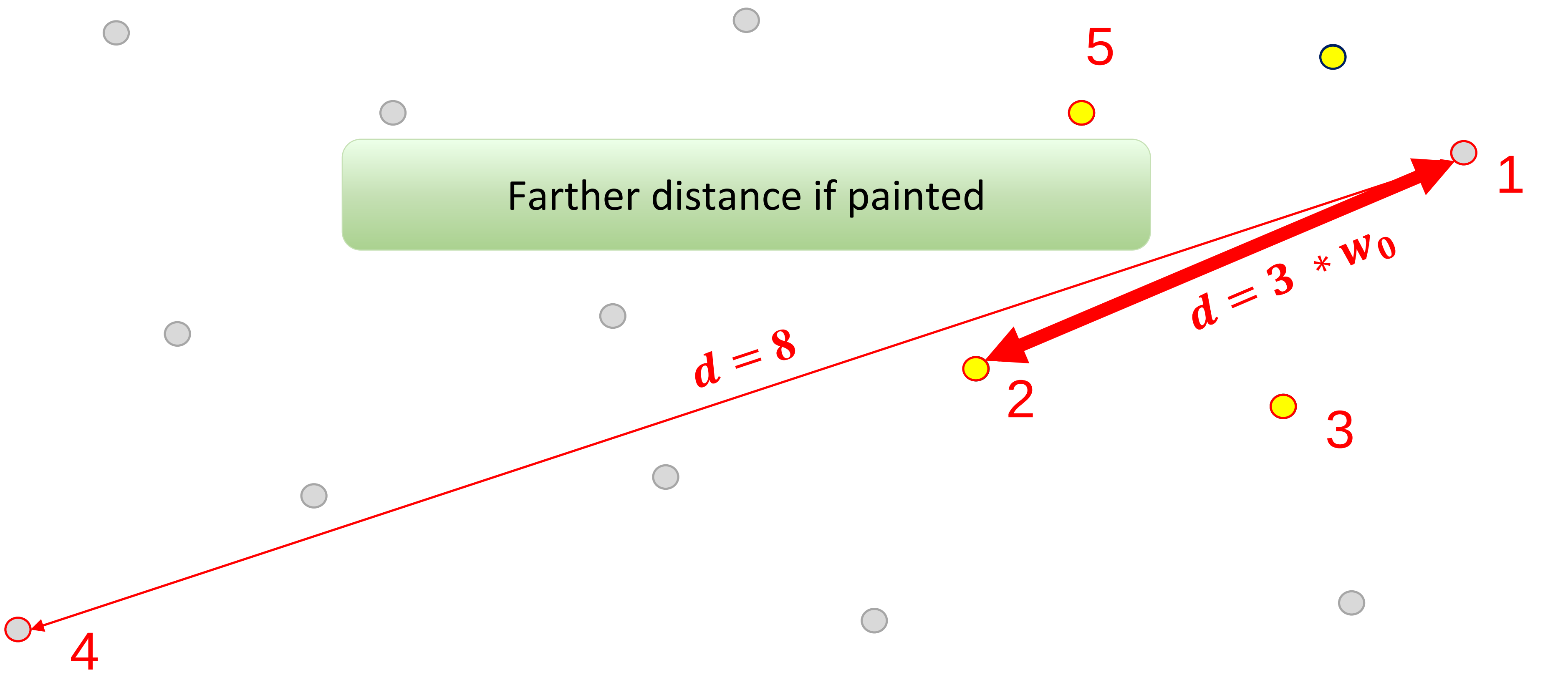
Farthest Point Sampling

- Base sampling technique in PointNet++.

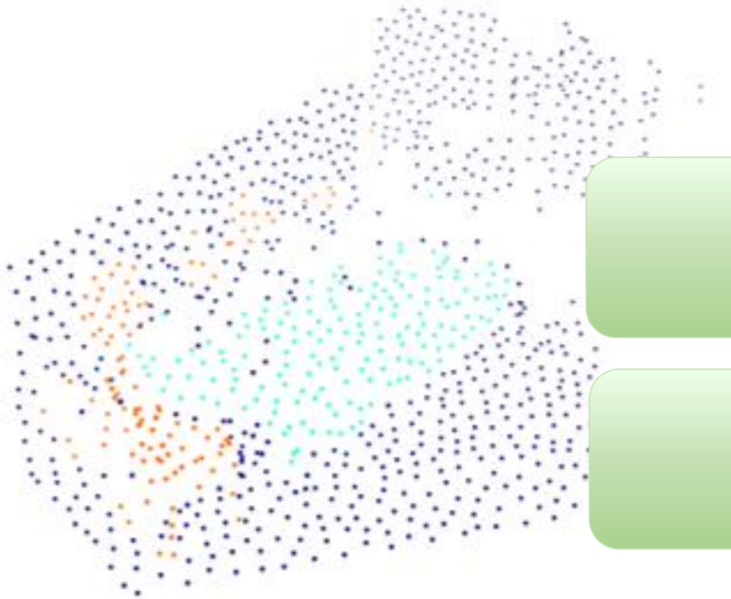


Biased Farthest Point Sampling

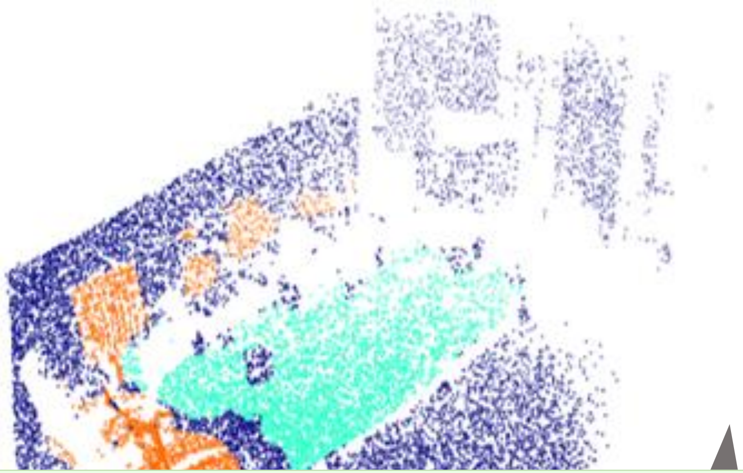
- *Biased* towards painted points using **foreground boundary**!



Comparison of Sampling Techniques



Normal FPS ($w_0 = 1$)



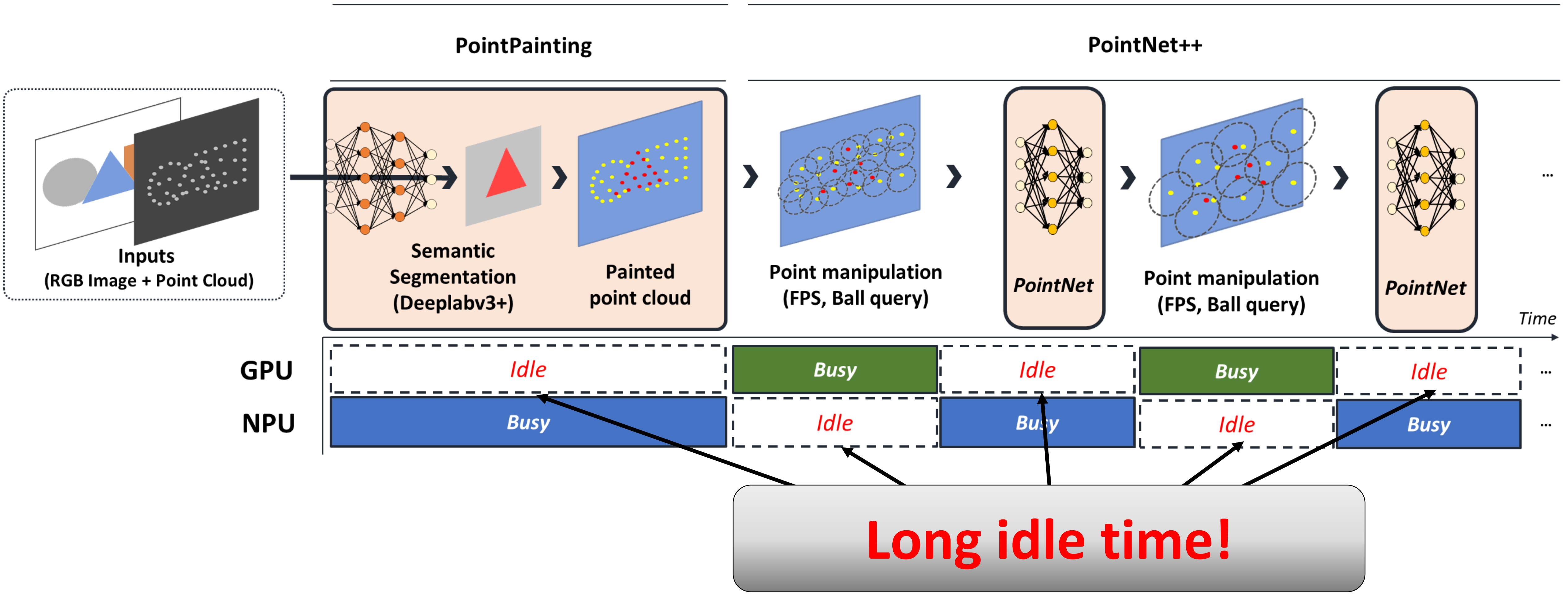
Complementary point sets

Augmentation during training

Biased FPS ($w_0 = 10$)

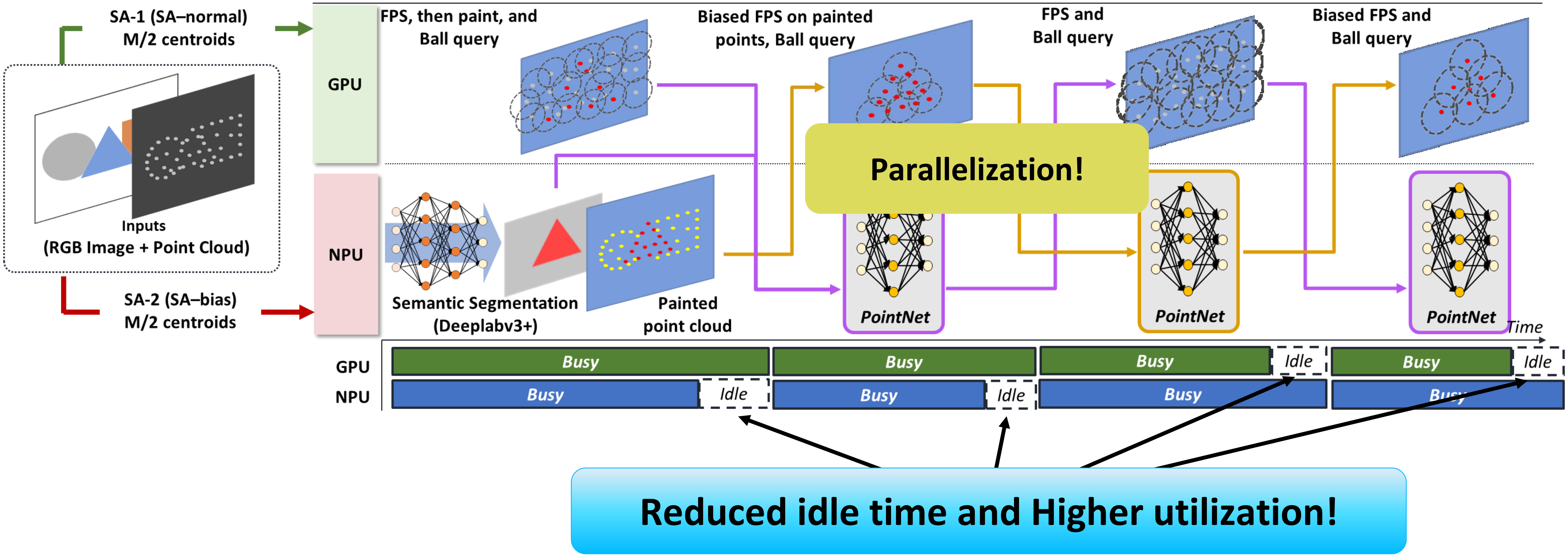
Reminder: Naïve Combination

- PointPainting and PoinetNet++ on GPU and NPU



Parallelizable Feature Extractor

- Runtime schedule of ***PointSplit*** on GPU and NPU



PointSplit

1. System - Algorithm joint optimization

- Biased farthest point sampling
- Parallelizable feature extractor

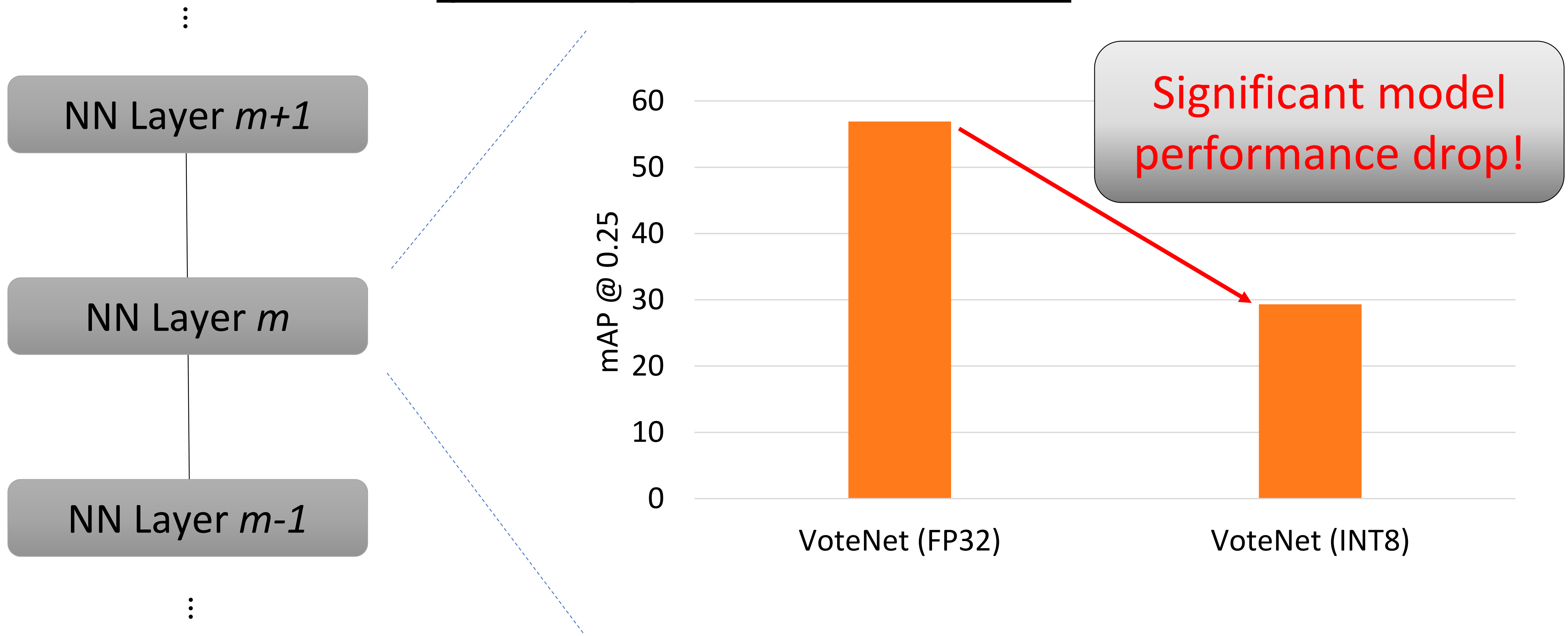
~~Large quant. errors without quant. granularity consideration~~

⬇

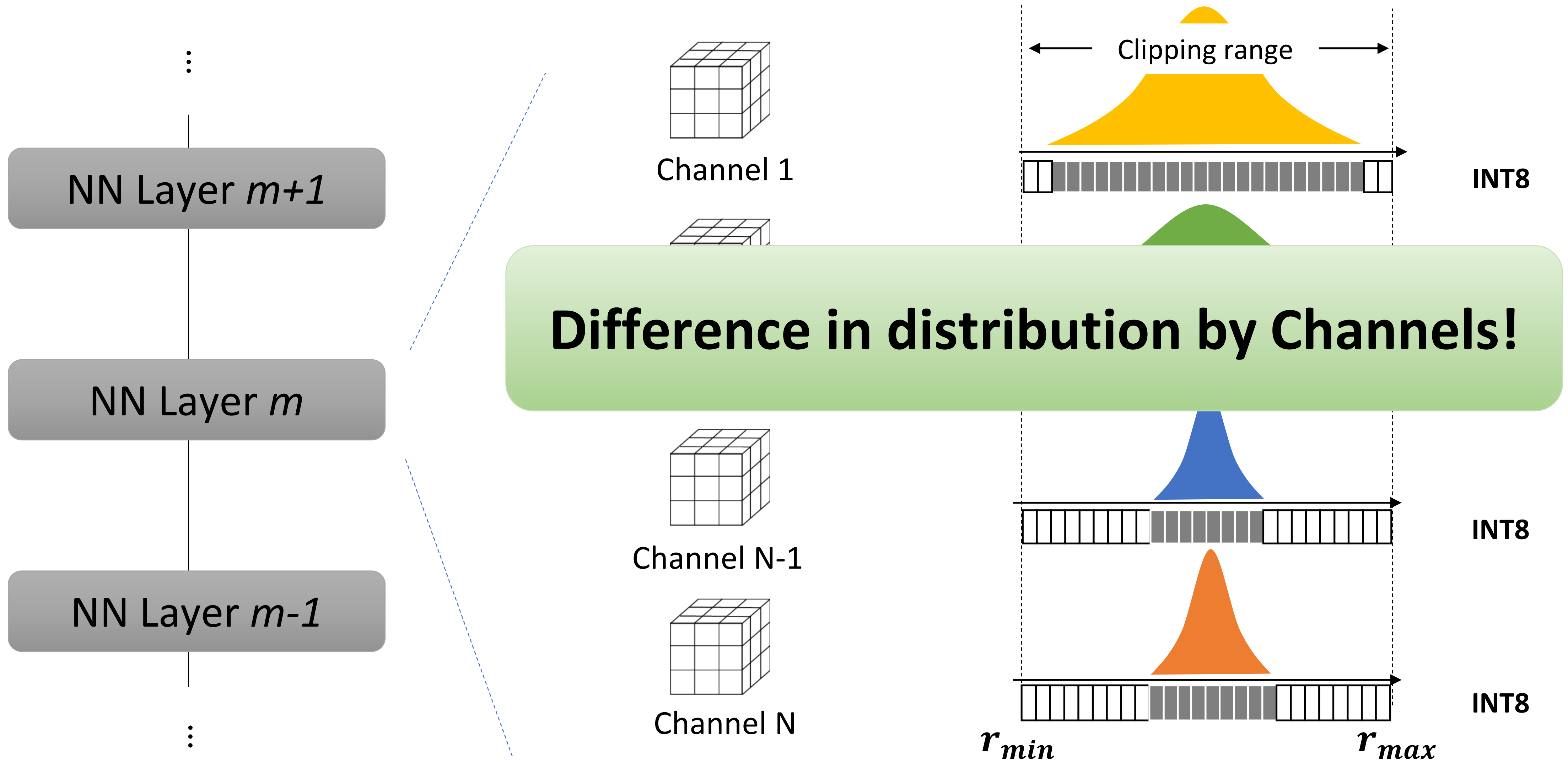
2. Role-based groupwise quantization

Large Quantization Errors in *tflite*

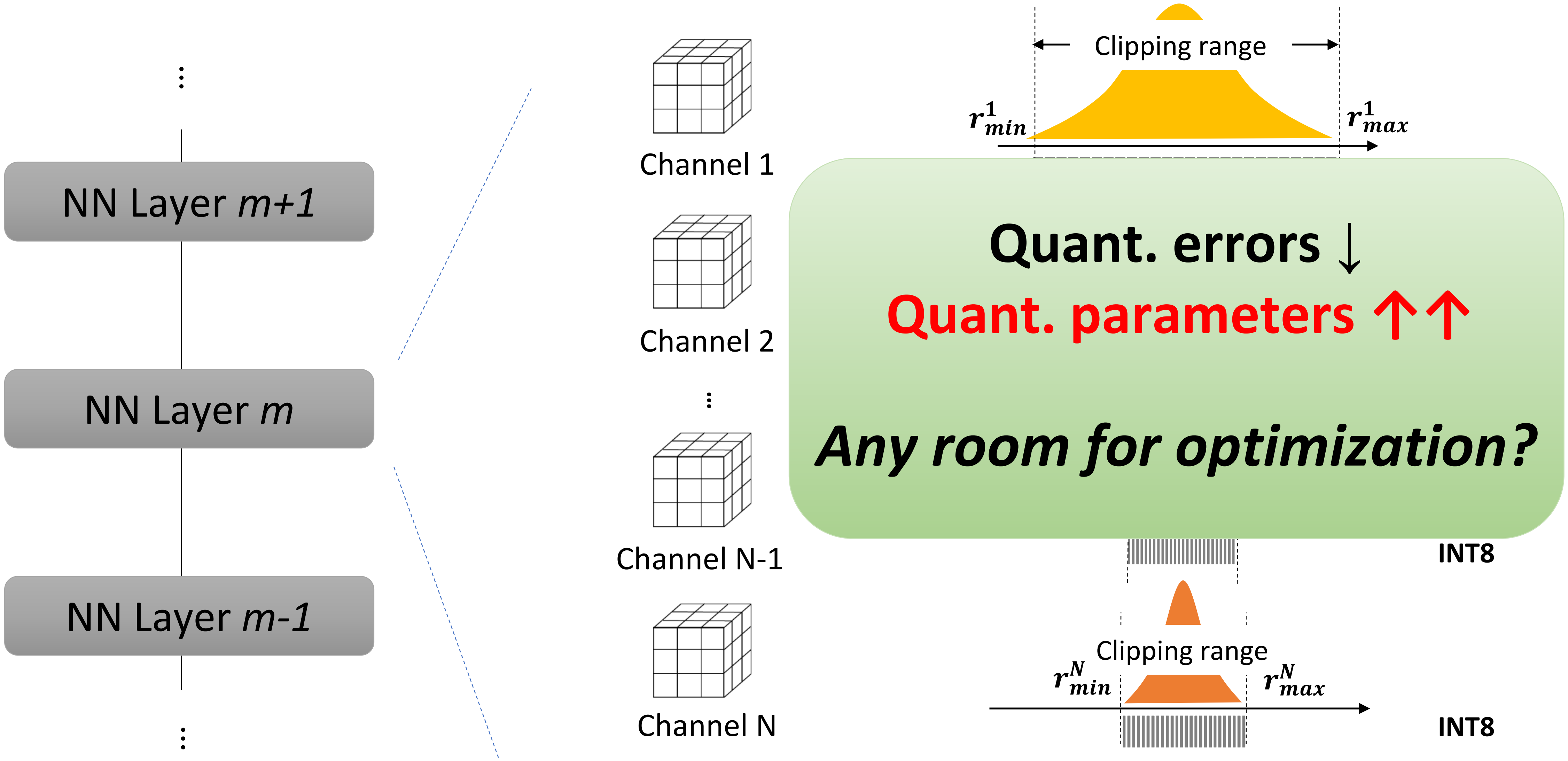
***tflite* - Layerwise Quantization**



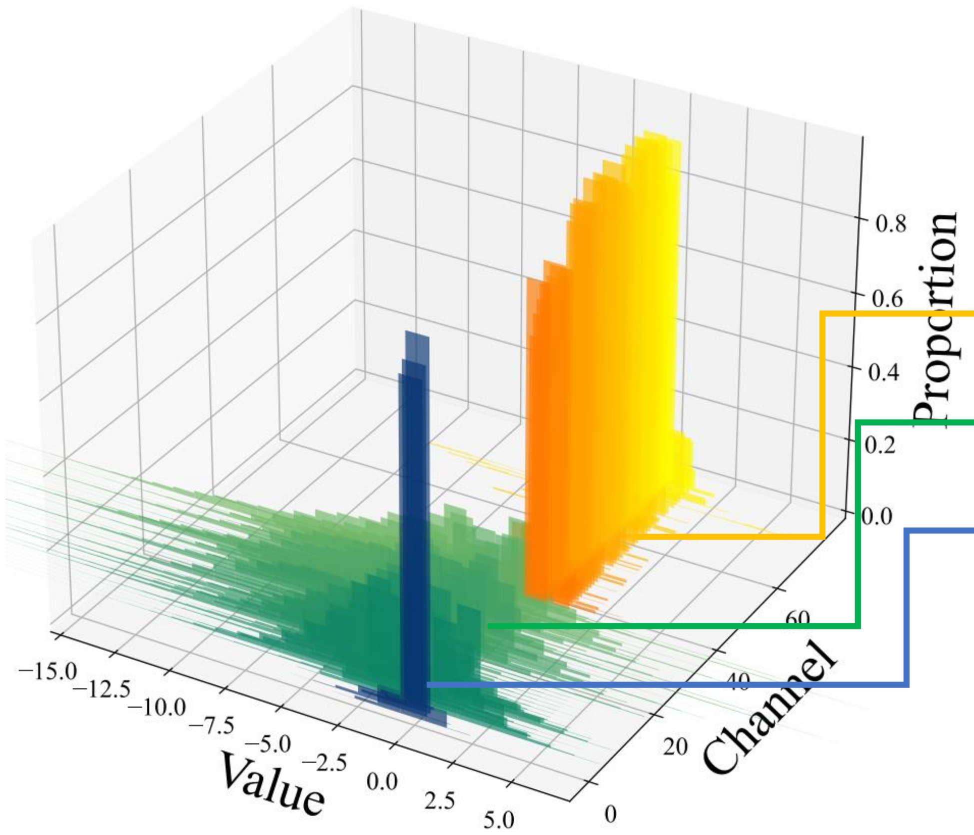
Large Quantization Errors in *tflite*



Channelwise Quantization?



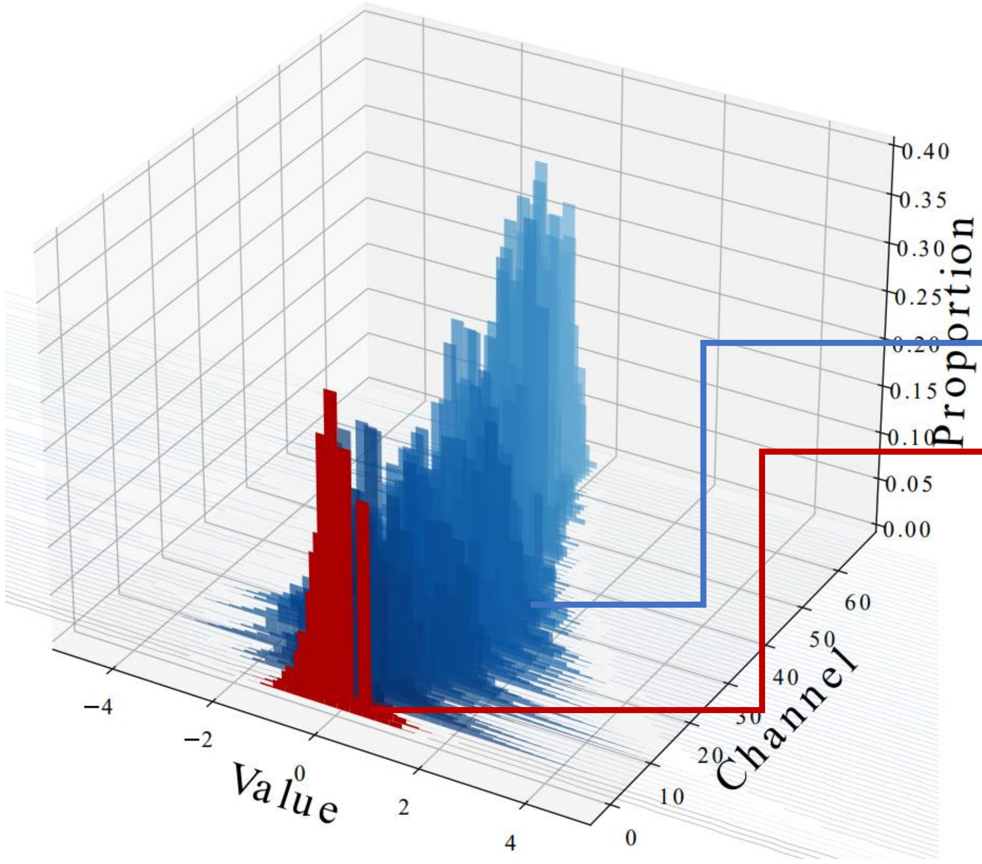
Role-based Groupwise Quantization



Distribution of activations by channel
in the last layer of *Proposal* Module

Role-Group	Roles
Group 1	Regression
Group 2	Classification
Group 3	Box center

Role-based Groupwise Quantization



Distribution of activations by channel
in the last layer of *Voting* Module

Role-Group	Roles
Group 1	Point Features
Group 2	Voting center coords.

**Quant. Granularity
based on Role-Group**

Implementation: Dataset

SUN RGB-D (Primary)

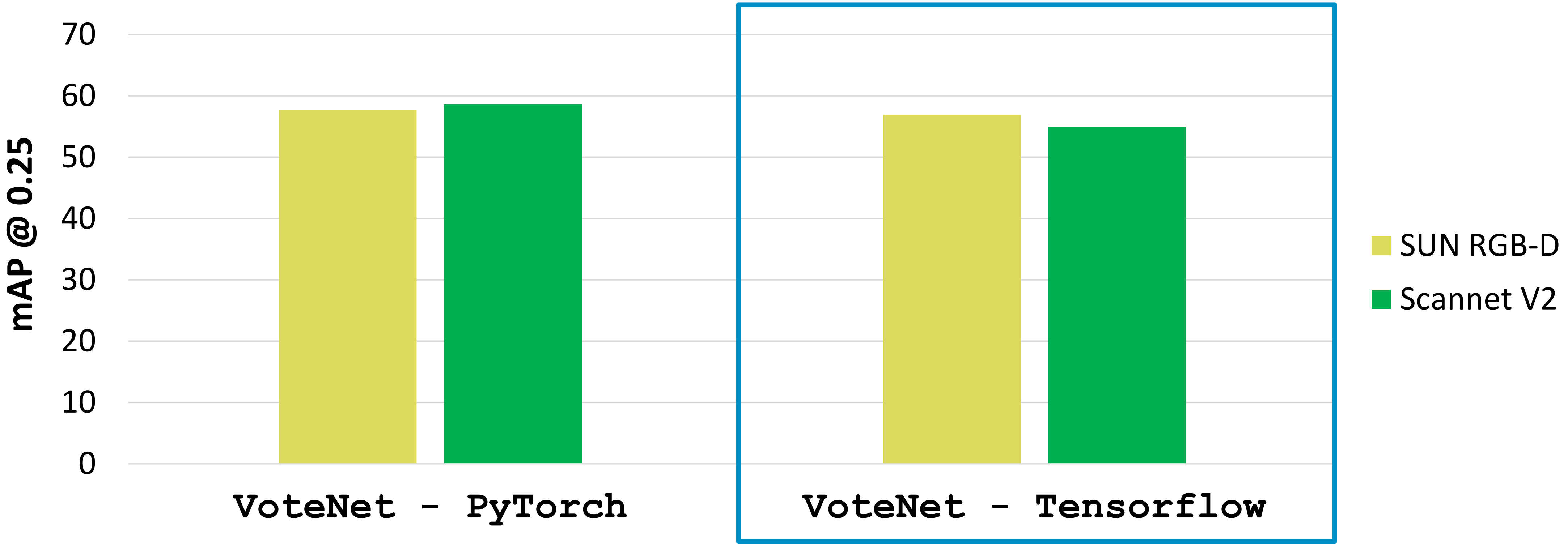
Snapshot

Scannet V2 (Secondary)

Reconstruction from >100 snapshots

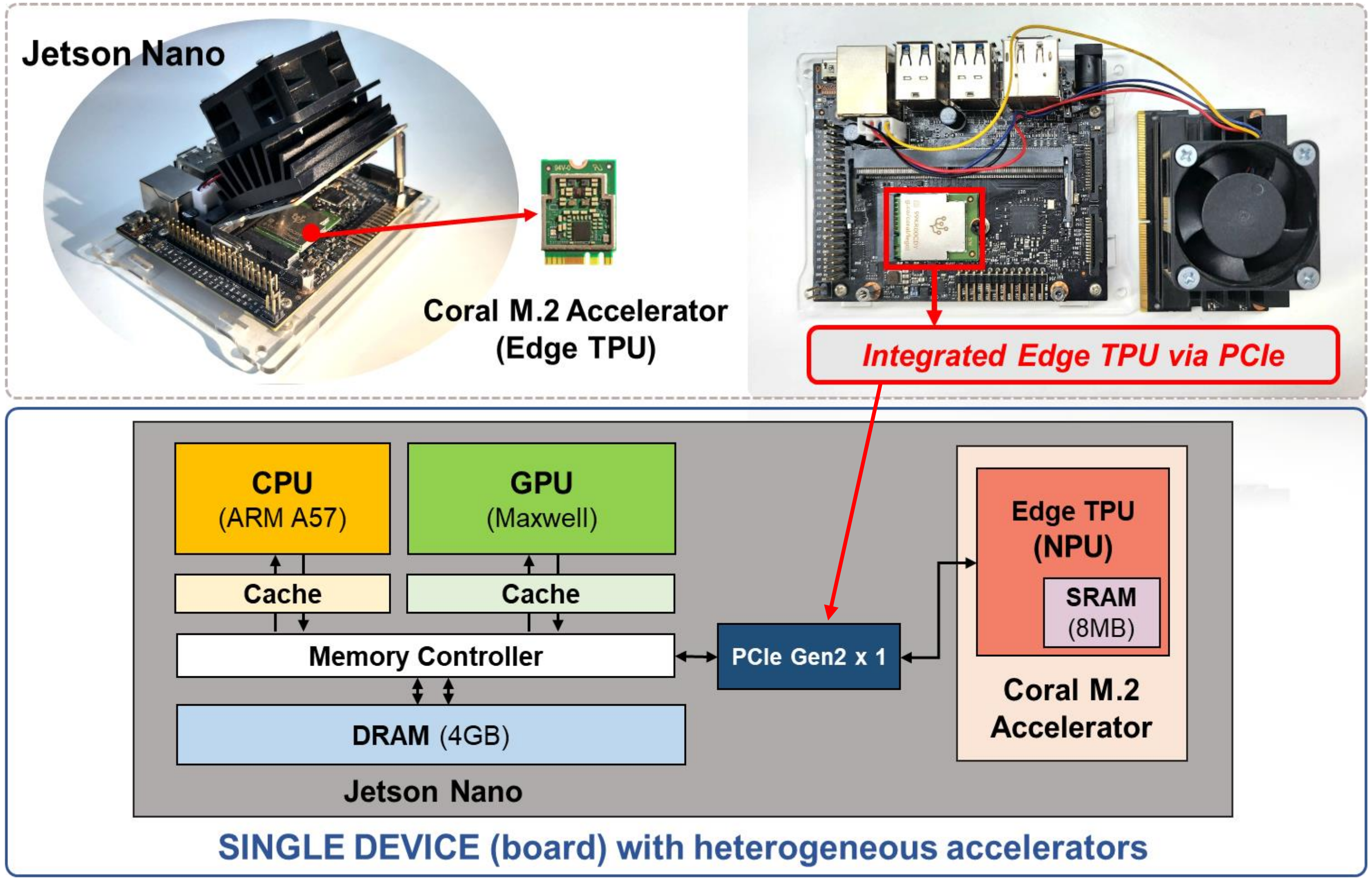
Implementation: Model

- *Deeplabv3+* finetuning for PointPainting.
- 3D object detector **`Tensorflow`** conversion from original **`PyTorch`** implementation.
  - For INT8 quantization and EdgeTPU-compiling supported by ***`tflite`***.

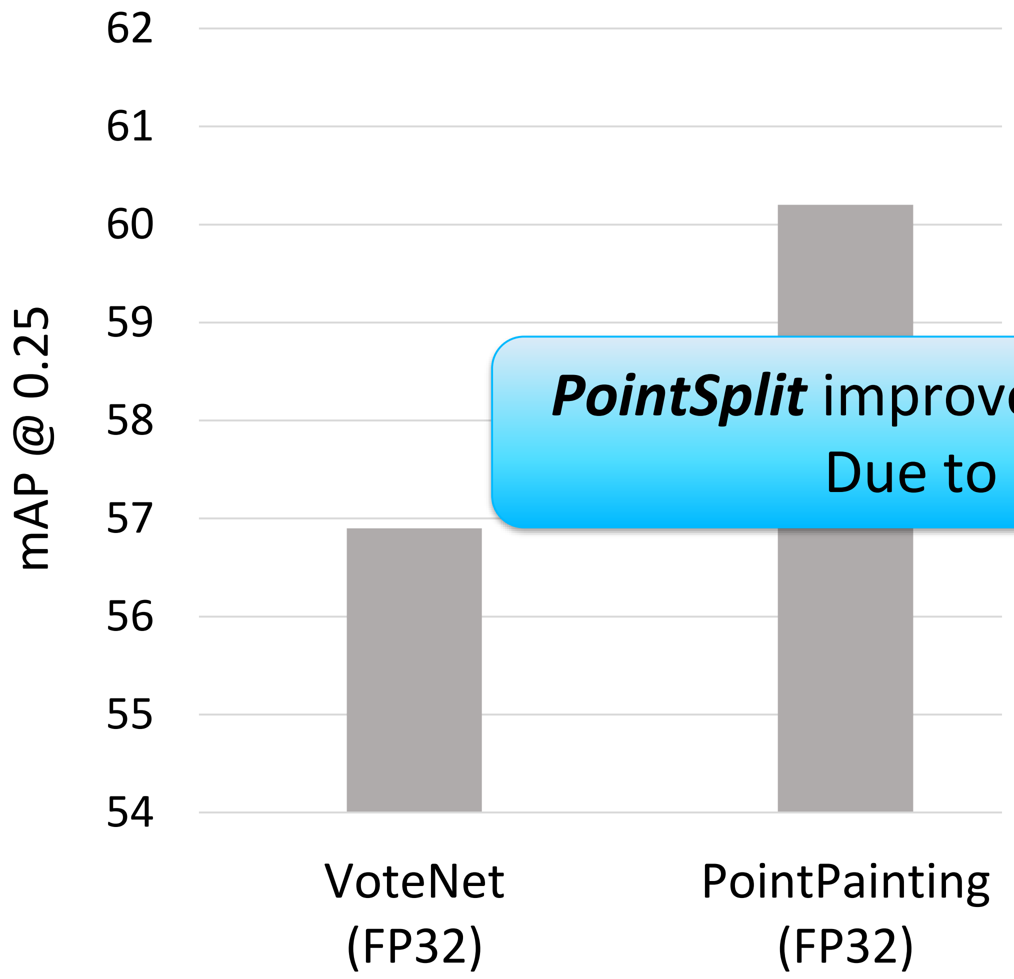


Implementation: Hardware platform

- Hardware platform with heterogeneous accelerators

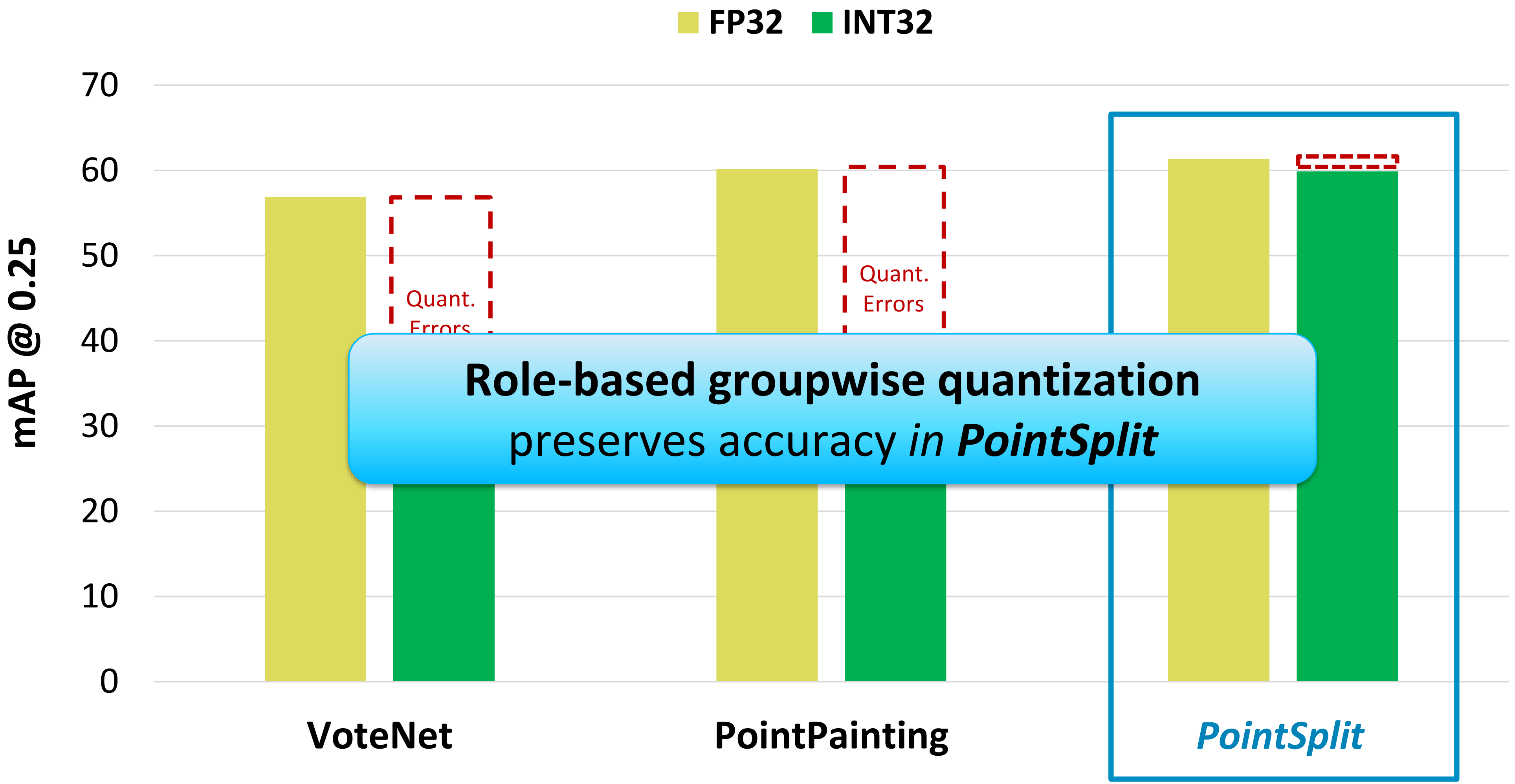


FP32 Detection Accuracy on SUN RGB-D



PointSplit improves mAP @ 0.25 by 1.2
Due to **Biased FPS**

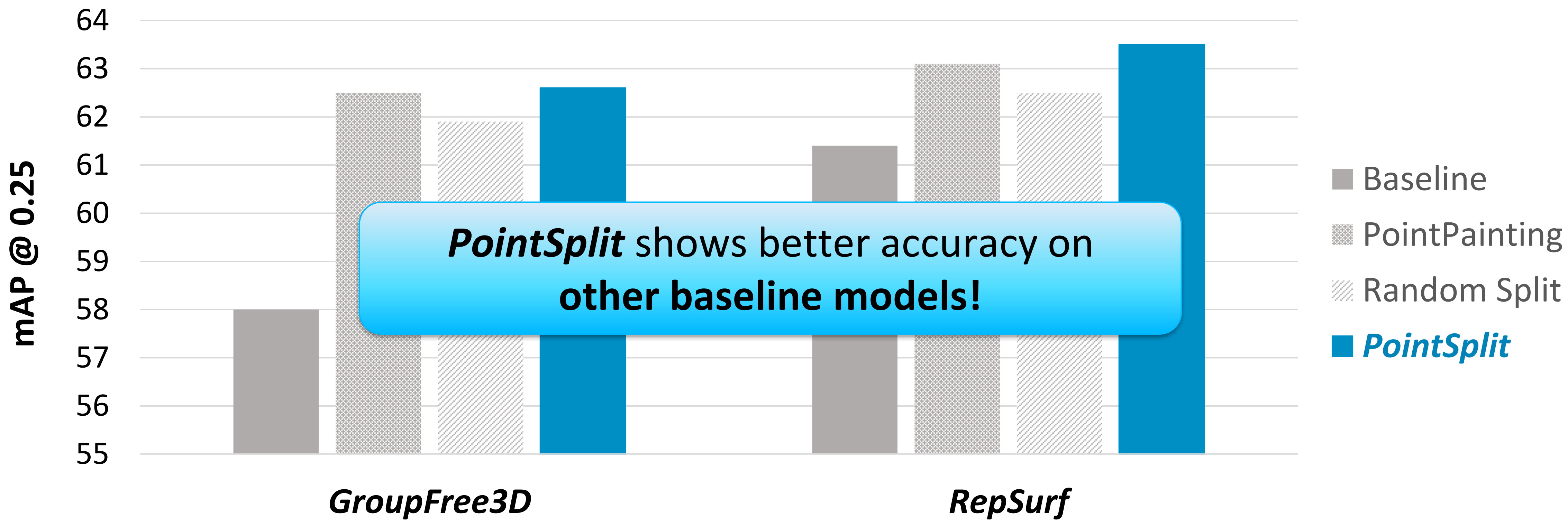
INT8 Detection Accuracy on SUN RGB-D



Role-based groupwise quantization
preserves accuracy *in* ***PointSplit***

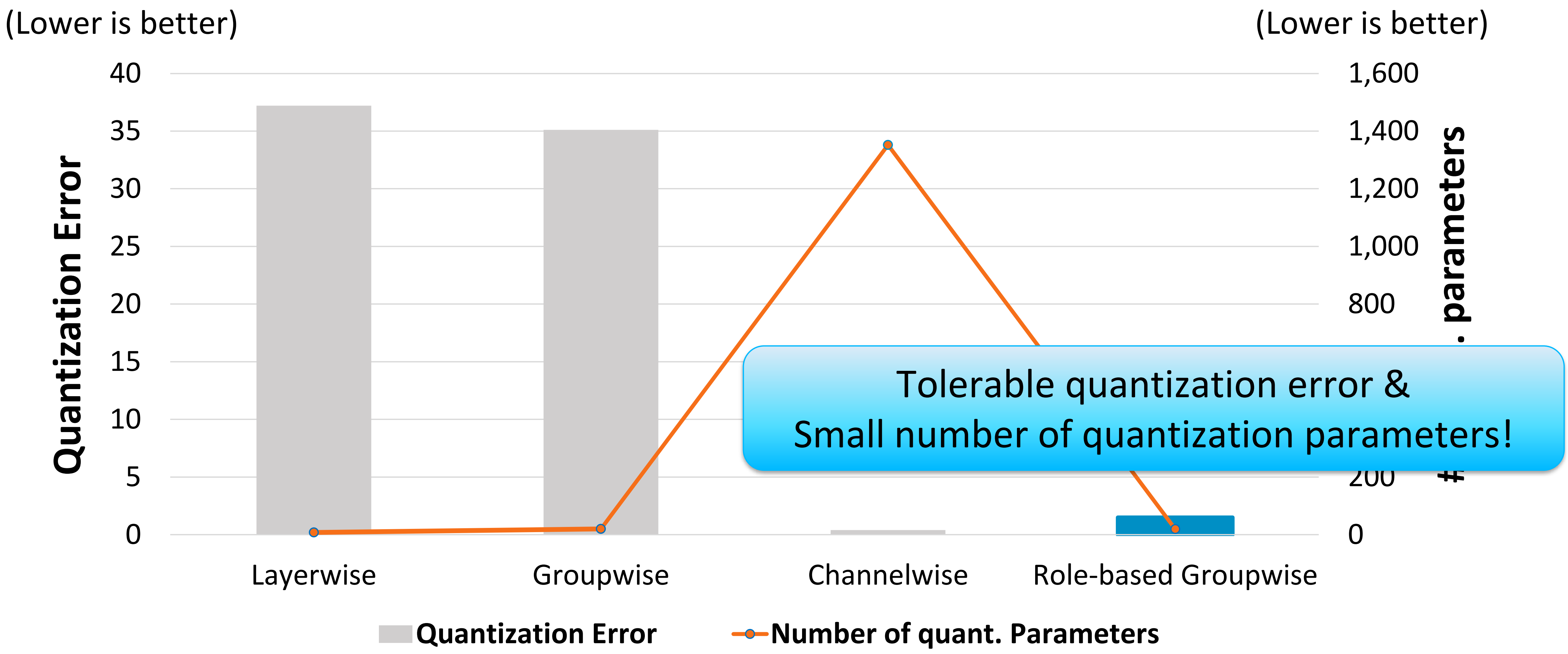
Detection Accuracy on Recent 3D Object Detectors

- ***GroupFree3D***: Uses Transformer modules.
- ***RepSurf***: Uses sophisticated 3D input representation.



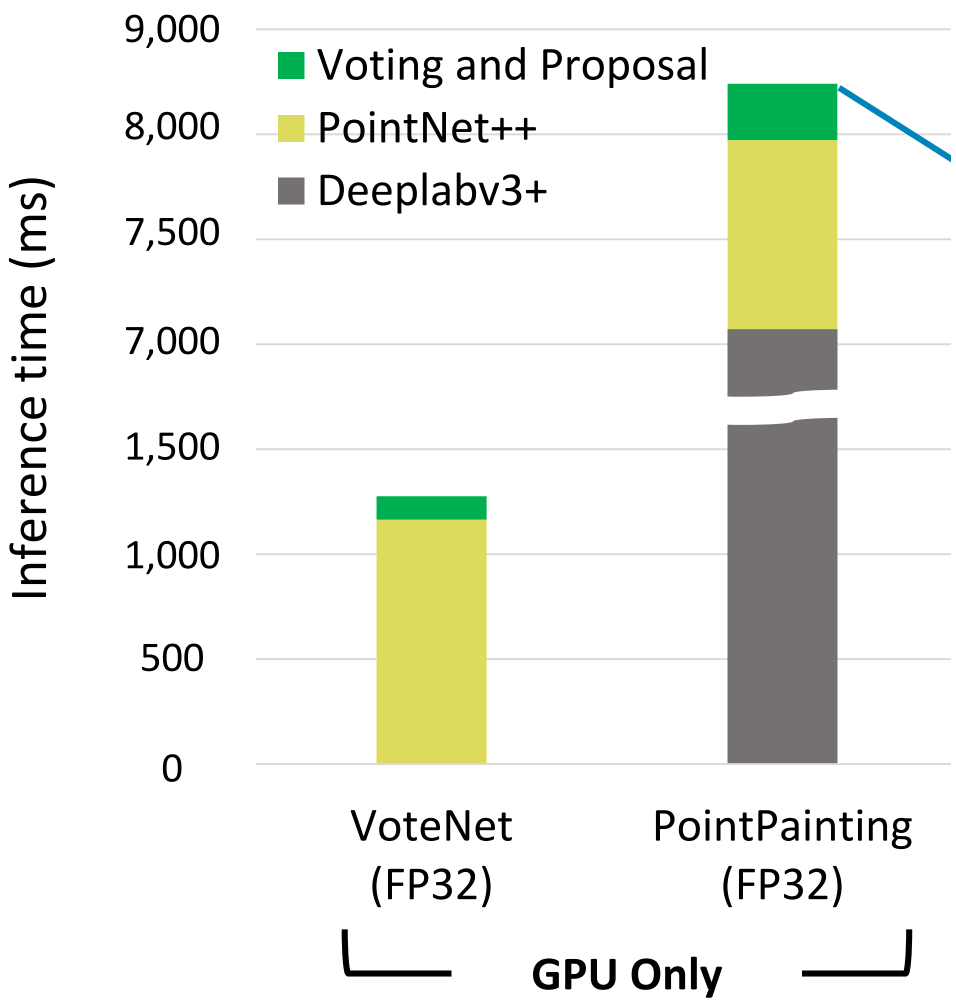
*Results on **FP32** (SUN RGB-D), implemented in Tensorflow*

Impact of Quantization Granularity

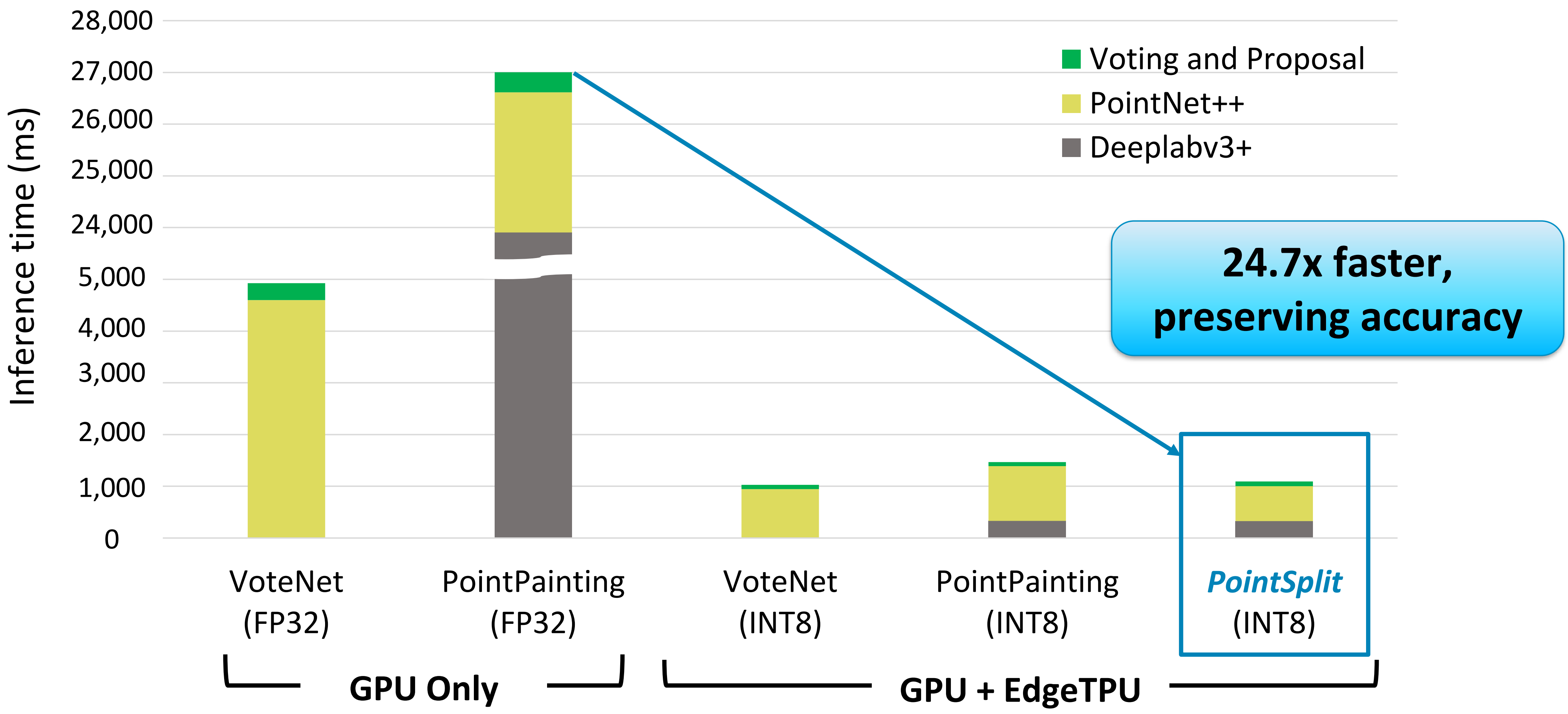


*Results on **PointSplit** (SUN RGB-D)*

Latency on SUN RGB-D



Latency on Scannet V2



Conclusion

- On-device 3D object detection with heterogeneous accelerators
- ***PointSplit***: system-algorithm joint optimization
  - Parallelizable feature extractor
  - Biased farthest point sampling
  - Role-based groupwise quantization
  - 11-25x latency reduction, preserving accuracy

Thank you very much!
감사합니다!

Biased Farthest Point Sampling: Algorithm

- Farthest point sampling(FPS): base sampling technique in VoteNet.
  - FPS twice? Two identical pointsets → Detection accuracy ▼
- Can we sample two complementary point sets?
  - Another important information from segmentation results: **Foreground boundary**!

Algorithm: Farthest Point Sampling

Initialization:

P: Input point cloud

$S = \{s_1\}$: Sampled point set. s_1 is randomly selected sample from P.

Distance from $p_k \in P$ to $S = \{s_1\}$:

$$d_S(p_k) = d(p_k, s_1) = \sqrt{(p_{k,x} - s_{1,x})^2 + (p_{k,y} - s_{1,y})^2 + (p_{k,z} - s_{1,z})^2}$$

For $i = 2, \dots, l$, repeat steps (a) – (c)

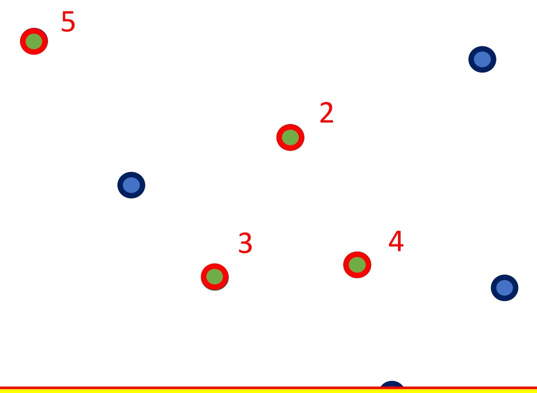
(a) Find the farthest sample away from S:

$$s_i = \ argmax\ d_S(p), \qquad p \in P$$

(b) Add s_i as a new seed into S.

(c) Update the distance from $p_k \in P$ to S:

$$d_S(p_k) \leftarrow \min\{d_S(p), d(p_k, s_i)\}$$



Biased FPS: More weights on favored (foreground) points (denoted as $\mathcal{A}$) in distance metric.

$$\boldsymbol{d(p_k, s_i) = w * \sqrt{(p_{k,x} - s_{i,x})^2 + (p_{k,y} - s_{i,y})^2 + (p_{k,z} - s_{i,z})^2}}$$

$$\boldsymbol{where\ w} = \begin{cases} w_0 & if\ p_{k,x} \in \mathcal{A}\ or\ s_{i,x} \in \mathcal{A} \\ 1 & otherwise \end{cases}$$

Supplementary

- Detection accuracy (SUN RGB-D, primary dataset)
  - PointPainting: sequential 2D/3D fusion improves mAP@0.25 by 3.3.
  - *PointSplit* achieves better mAP@0.25 than PointPainting or RandomSplit.
  - Even after quantization, *PointSplit* shows comparable mAP to PointPainting.

Item	Bathtub	Bed	Bookshelf	Chair	Desk	Dresser	Nightstand	Sofa	Table	Toilet	Overall
VoteNet (FP32)	**72.4**	84.0	25.3	74.1	24.2	30.0	61.4	61.6	49.7	86.8	56.9
PointPainting (FP32)	68.0	**86.5**	29.6	74.1	24.6	**39.9**	**61.8**	77.9	49.3	90.0	60.2

- Detection accuracy on multiple datasets, varying threshold or precision.
  - On Scannet, *PointSplit* also shows comparable performance.
  - At IoU thresholds of 0.5, *PointSplit* also shows comparable performance.
  - Regardless of precision, *PointSplit* shows good performance.

Precision	Method	Dataset SUN RGB-D @0.25 / @0.5	Dataset Scannet V2 @0.25 / @0.5
FP32	VoteNet	56.9 / 31.1	54.9 / 30.4
	PointPainting	60.2 / **32.8**	**56.4** / 31.7
	RandomSplit	60.4 / 32.0	55.2 / 31.2
	PointSplit	**61.4** / 32.7	56.1 / **32.4**
INT8	VoteNet	29.3 / 3.0	41.7 / 11.6
	PointPainting	32.3 / 3.2	48.8 / 18.2
	PointSplit	**59.9** / **32.5**	**55.7** / **30.3**

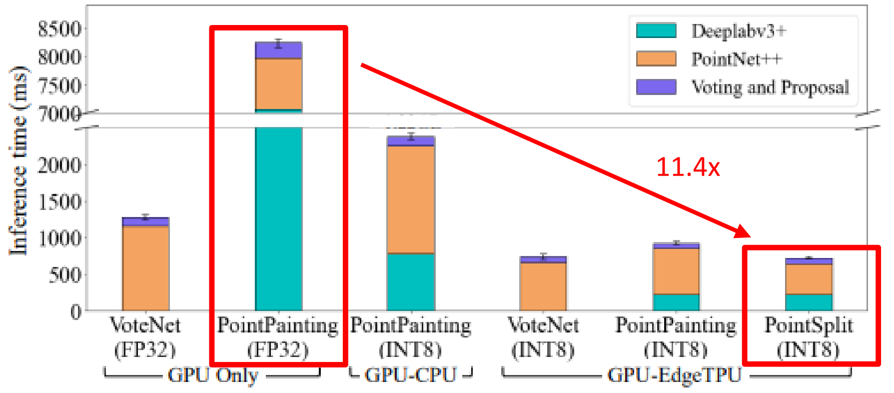
Supplementary

- Communication overhead
  - Alternating GPU / NPU may incur communication overhead. This is our limitation.
- How to measure comm. overhead?
  - GPU memory copy time could be measured with NVIDIA profiler.
  - Such a tool is not provided for EdgeTPU.
  - Our trick:
    - (1) Measure original tflite time: t_comm + t_comp
    - (2) Create another tflite with same input/output but twice computation: t_comm + t_comp*2
    - (2) – (1) = t_comp, then we can also calculate t_comm

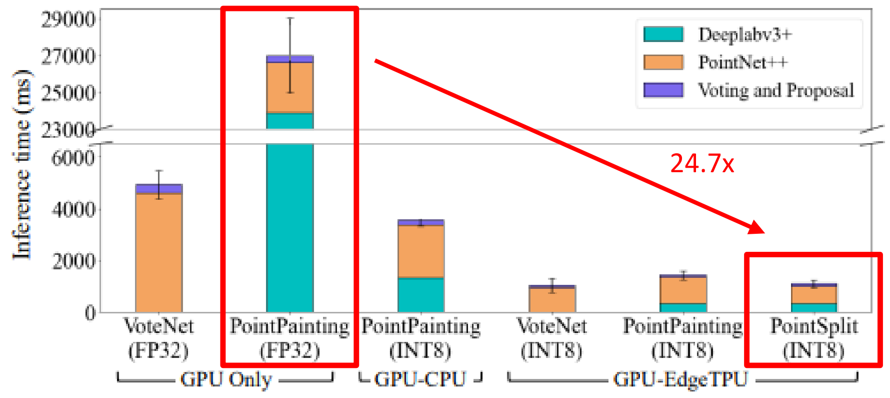
Processor	Latency (ms)		
	Communication	Computation	Total
GPU	80	248	328
EdgeTPU (estimates)	360	121	481

Supplementary

- Latency
  - To inference a single scene (2D + 3D) from SUN RGB-D with a GPU Only, it takes > 8,000ms.
  - *PointSplit* decreases the latency to 750ms (11.4x faster), while keeping comparable detection accuracy.
    - Use of EdgeTPU increases the inference speed by 8.9x, and pipelining increases the inference speed further by 1.3x.
  - On ScannetV2, the final latency decreases from 27,000ms to 1,400ms (24.7x).
- Peak memory
  - Peak memory consumption decreased from 2.25GB to 1.18GB, thanks to lightweight software platform(*tflite*) as well as quantization.
  - Parallelizing across heterogeneous processors does not sacrifice memory consumption.



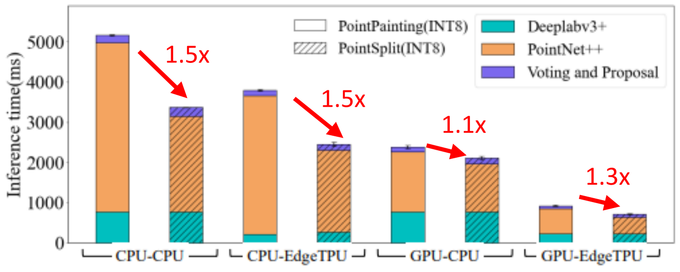
(a) Latency on SUN RGB-D



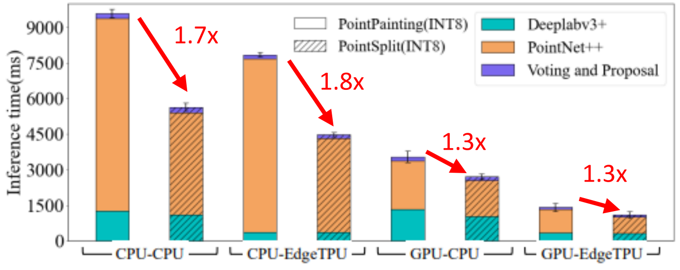
(b) Latency on Scannet V2

Supplementary

- Latency on more hardware configuration
  - The operations are assigned to different processor combinations (e.g. CPU – EdgeTPU: Point manipulation on CPU and Neural nets on EdgeTPU).
  - Across all combinations, *PointSplit* improved the inference time by up to 1.8x on both SUN RGB-D and Scannet V2.



(a) Latency on SUN RGB-D

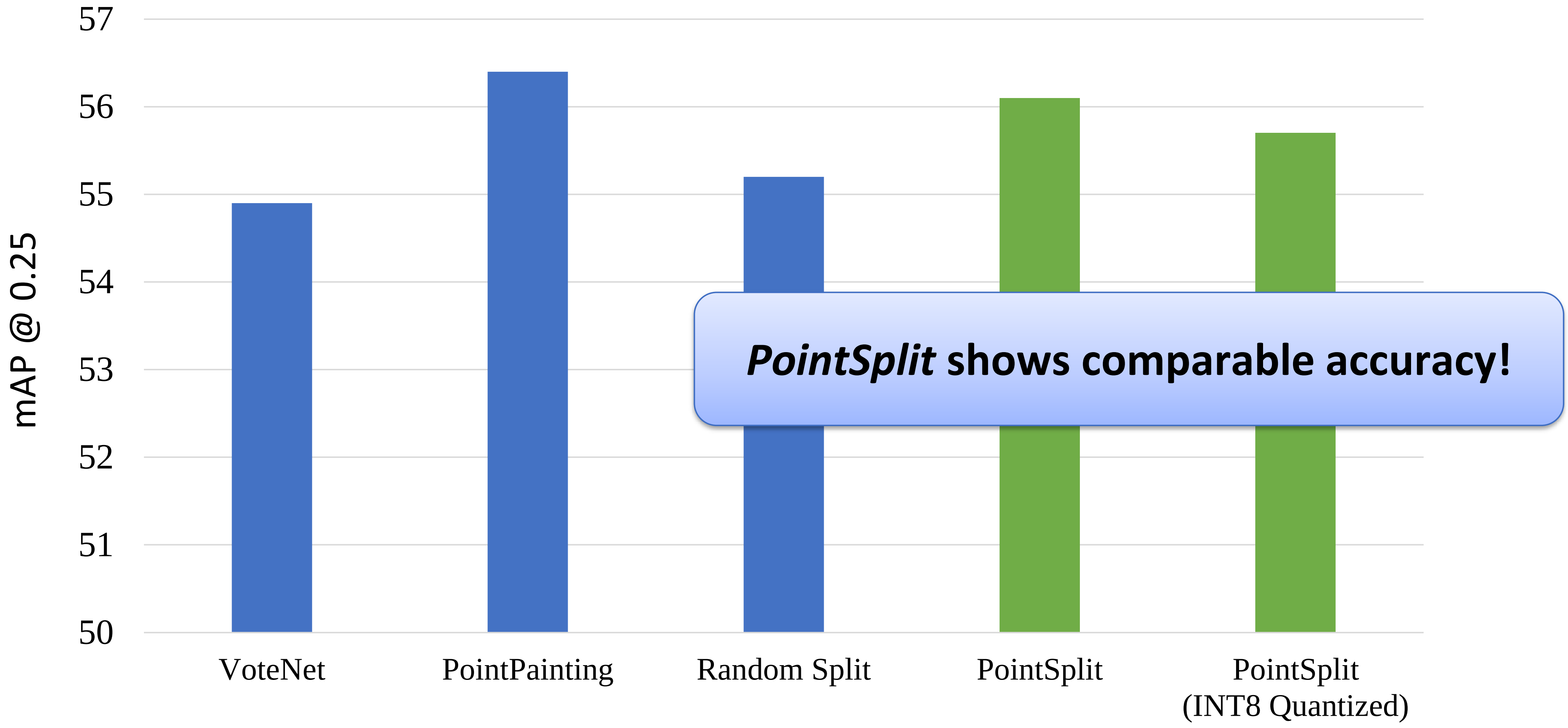


(b) Latency on Scannet V2

- Layerwise latency analysis
  - Latency on each processor per layer shows that the largest gain in inference time comes from parallelizing 2D-3D fusion (PointPainting) and SA1 point manipulation.
  - At later SA layers, GPU time decreases but EdgeTPU time increases then decrease. This indicates further optimization room for job allocation.

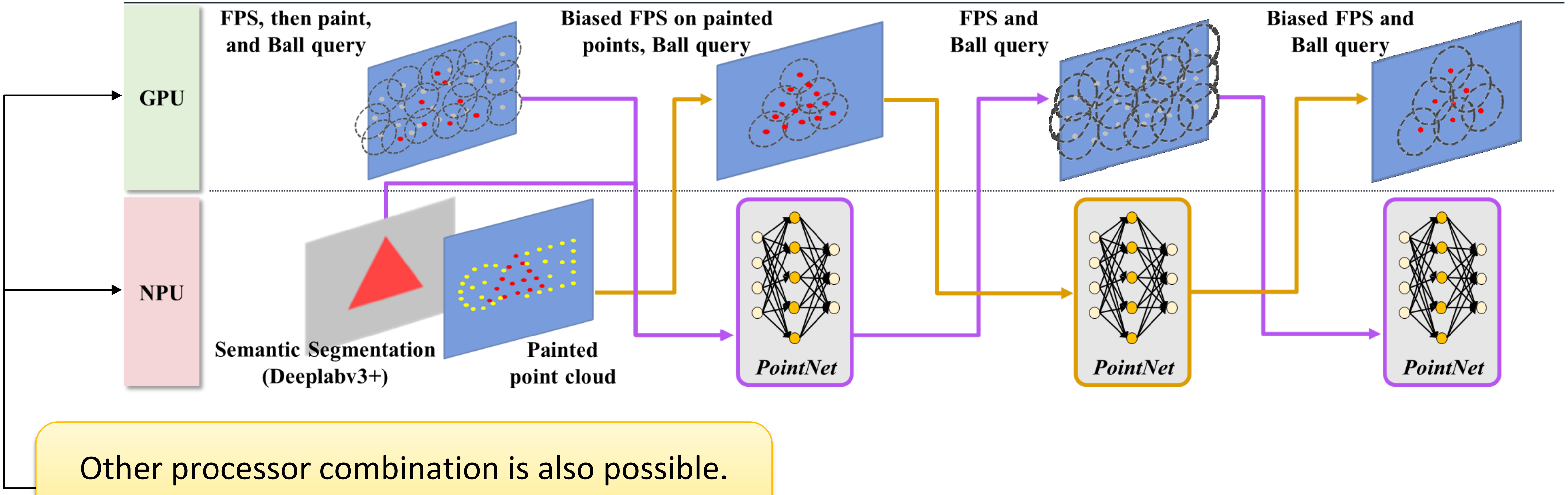
Layers	GPU	EdgeTPU
2D-3D fusion	-	222 ms
SA1	199 ms	47 ms
SA2	52 ms	71 ms
SA3	25 ms	84 ms
SA4	20 ms	21 ms

Detection Accuracy on Scannet V2



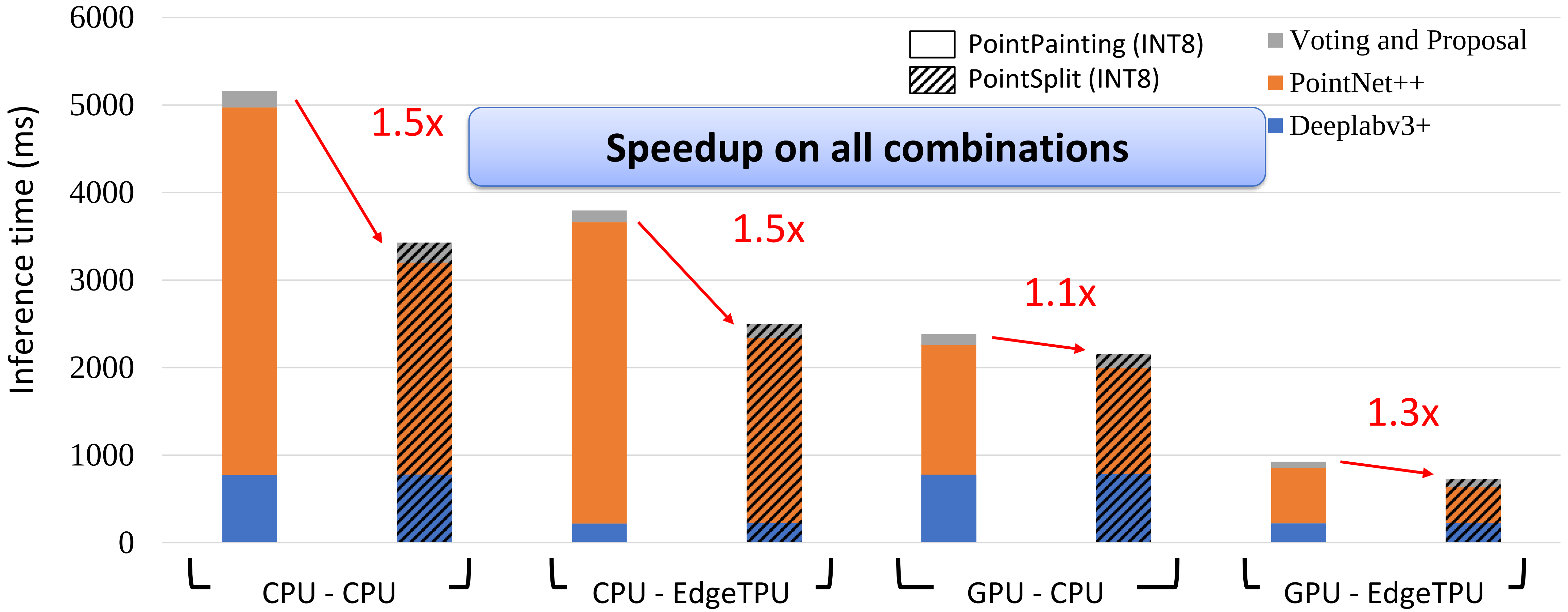
Latency on more hardware configuration

Recall PointSplit's pipelining scheme…



Other processor combination is also possible.
(CPU – CPU, CPU – NPU, …)

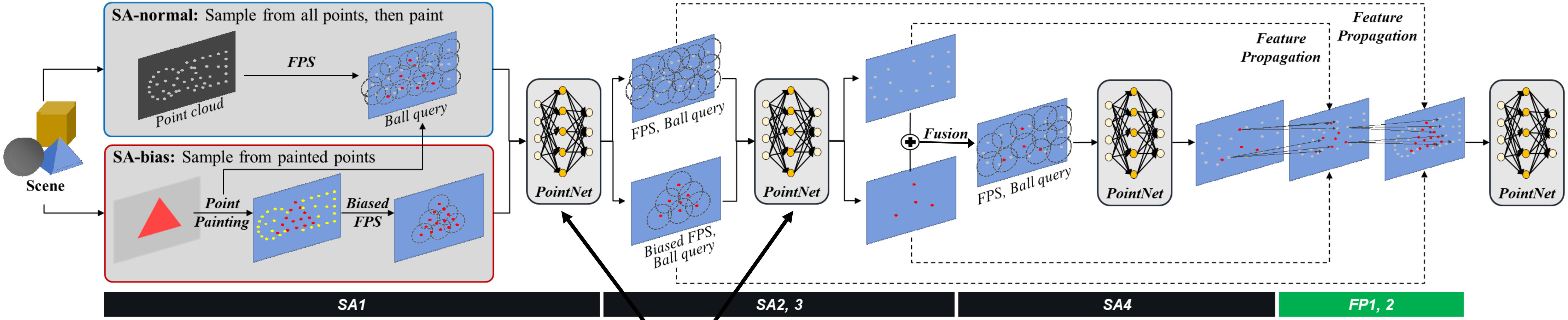
Latency on more hardware configuration



Results on SUN RGB-D

PointSplit

- Can we optimize the model structure to have higher utilization on GPU/NPU?
- Let's sample 2 point sets from the input point cloud, then process **independently**.
  - Sample M centroids → (Sample M/2 centroids) × 2.



Same *PointNet* for both pointsets:
- Prevent model size from increasing
- More robust network training