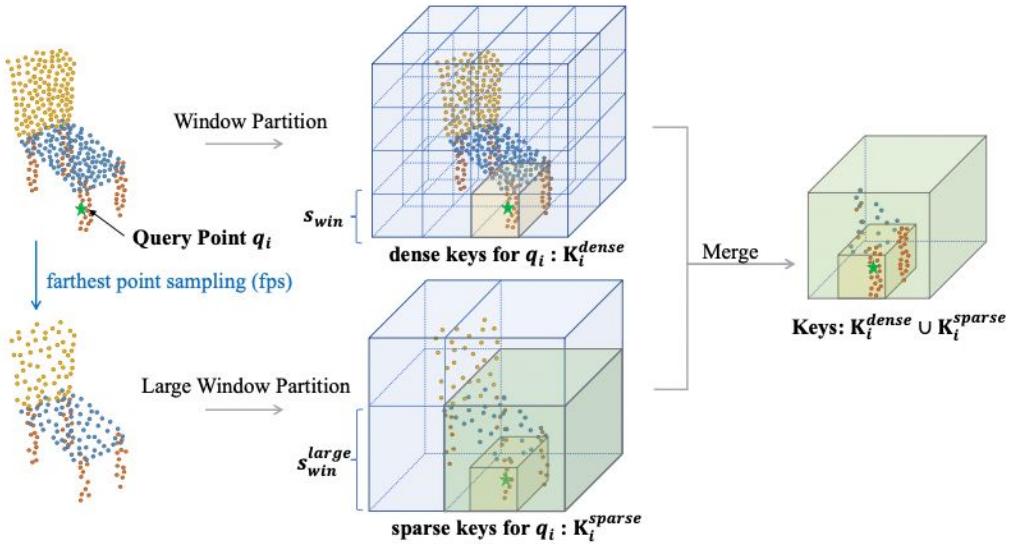
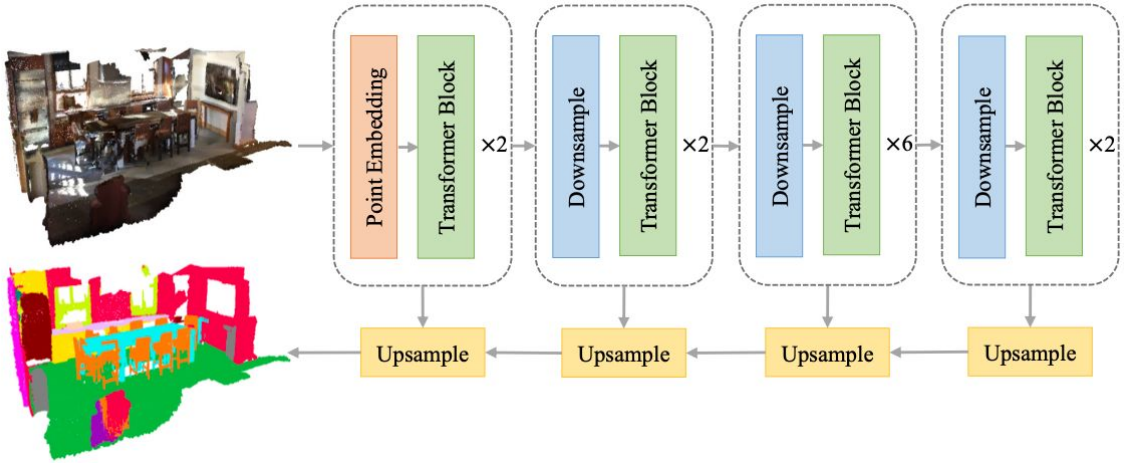
Paper Review

Stratified Transformer for 3d Point Cloud Segmentation

CVPR, 2022 Accept



Architecture

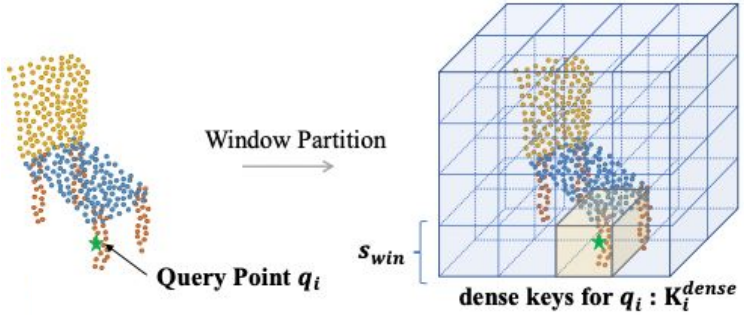


Capture long-range contexts

- Input : both xyz coordinates and rgb colors
- Encoder-Decoder structure
- **Point embedding** for each point to aggregate local information in the first layer
- **Relative position encoding** to capture richer position information.
  - Positional bias dynamically with **contexts** through the interaction with the semantic features.

Vanilla Version

- **Window-based Self-attention**



: Each query point only needs to consider neighbors **in the same window**

- Multi-head self-attention is performed in each window independently

$$\mathbf{q} = Linear_q(\mathbf{x}), \quad \mathbf{k} = Linear_k(\mathbf{x}), \quad \mathbf{v} = Linear_v(\mathbf{x}),$$

$$\mathbf{attn}_{i,j,h} = \mathbf{q}_{i,h} \cdot \mathbf{k}_{j,h},$$

$$\hat{\mathbf{attn}}_{i,.,h} = softmax(\mathbf{attn}_{i,.,h}),$$

$$\mathbf{y}_{i,h} = \sum_{j=1}^{k_t} \hat{\mathbf{attn}}_{i,j,h} \times \mathbf{v}_{j,h},$$

$$\hat{\mathbf{z}} = Linear(\mathbf{y}),$$

Input : $\mathbf{x} \in \mathbb{R}^{k_t \times (N_h \times N_d)}$

- N_h : #heads
- N_d : Dimension of each head
- k_t : #points within the t-th window

$\mathbf{q}, \mathbf{k}, \mathbf{v} \in \mathbb{R}^{k_t \times N_h \times N_d}$

$\mathbf{attn} \in \mathbb{R}^{k_t \times k_t \times N_h}$

$\mathbf{y} \in \mathbb{R}^{k_t \times N_h \times N_d}$

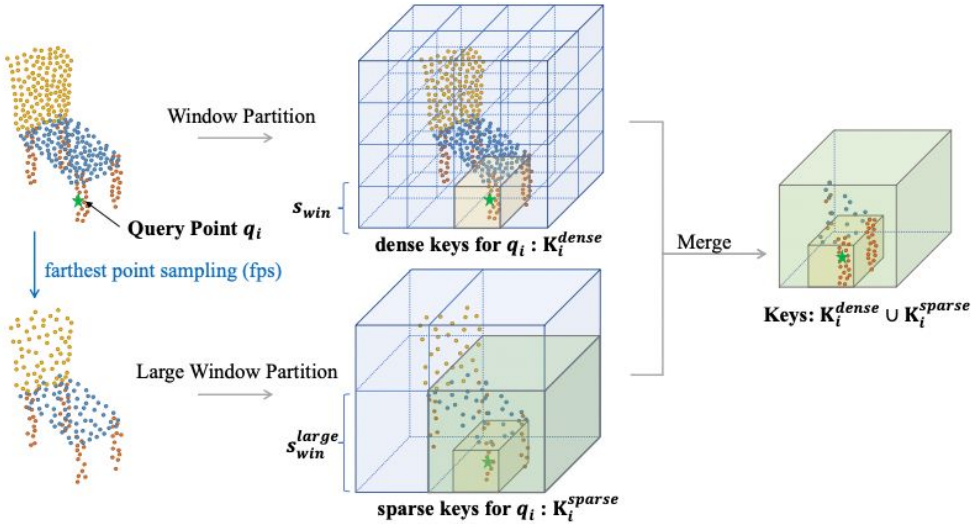
$\hat{\mathbf{z}} \in \mathbb{R}^{k_t \times (N_h \times N_d)}$

Stratified Key-sampling Strategy

- **Cons** of Vanilla version
  : Limited effective receptive field
  - Fails to capture long-range contextual dependencies over distant objects

- Stratified Strategy
  : For each query point, **both denser** nearly points($\mathbf{K}_i^{dense}$) and **sparser** distant points($\mathbf{K}_i^{sparse}$) are sampled to form the keys **all together**.
  - Enlarging the effective receptive field and building direct long-range dependency

- Shift
  : To further complement the information interaction across windows, the original window is **shifted** by $\frac{1}{2}s_{win}$ while the large window is shifted by $\frac{1}{2}s_{win}^{large}$



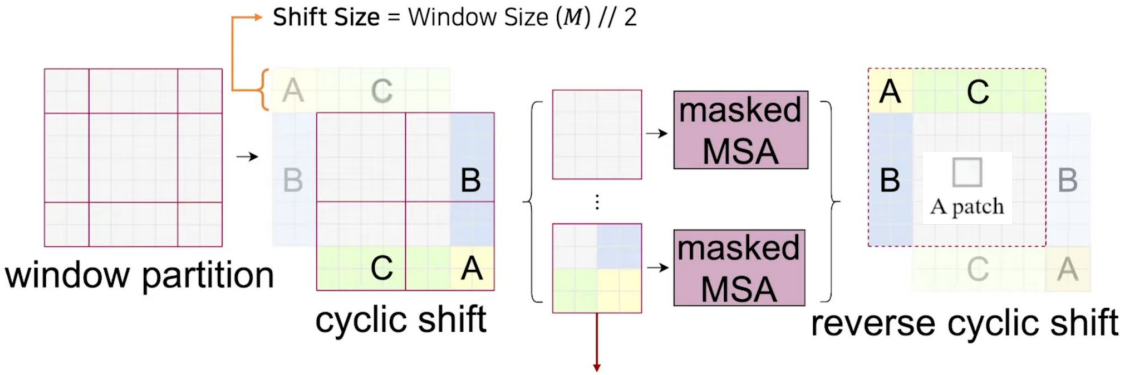
Swin Transformer

- **W-MSA :** Self attention within local window
  - Cons : Pixels near the boundary of each window do not perform the self-attention even though they are adjacent to each other.

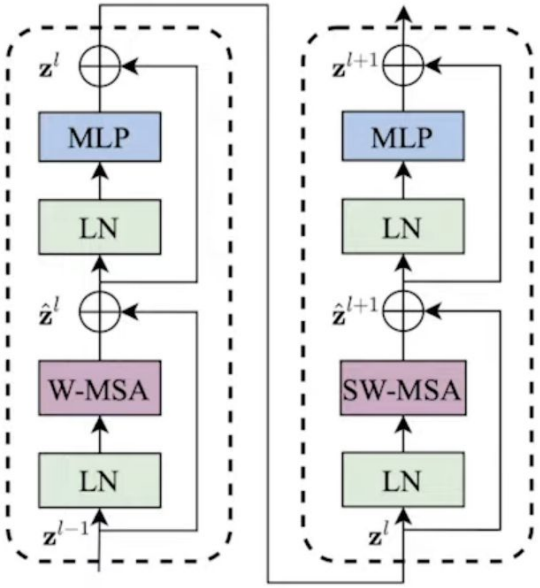
  ↓ **Shift** half the window size to perform a self-attention

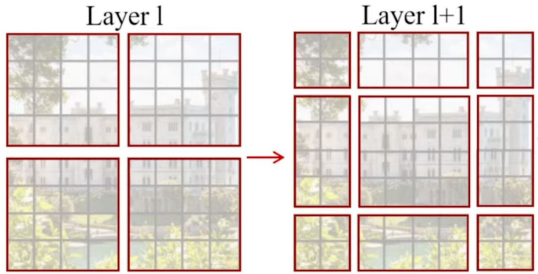
- **SW-MSA :** Self-attention between local window

- Use same the number of window through Cyclic Shift & Attention Mask

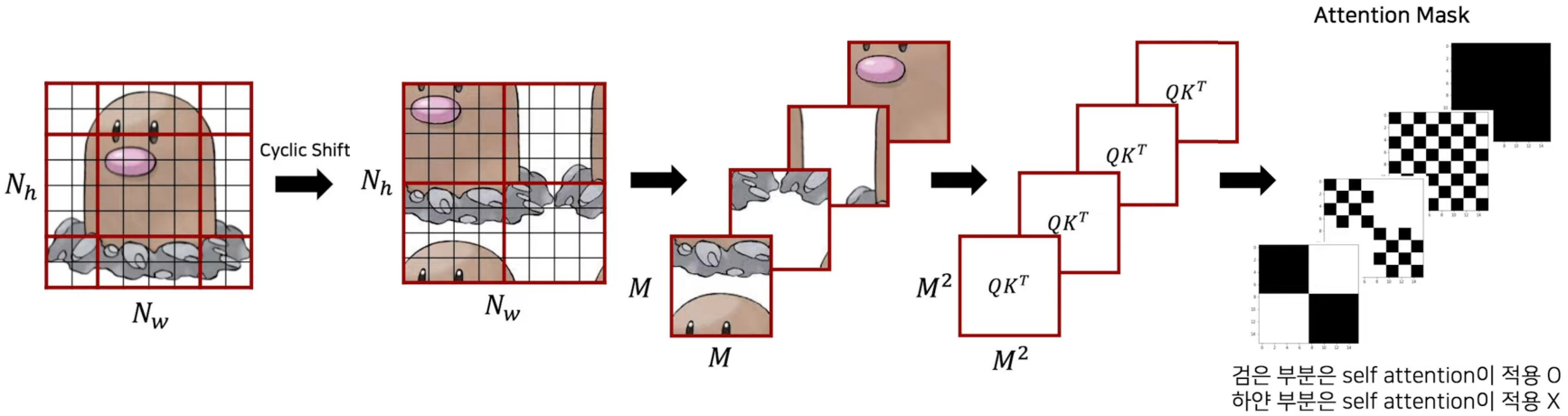


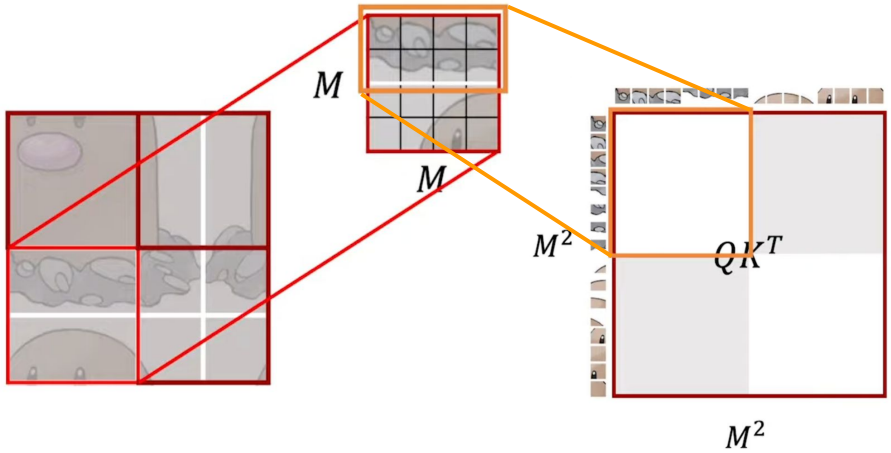
서로 인접한 위치가 아니었기 때문에 Mask를 적용하여 mask 부분에서만 self attention을 적용





Swin Transformer

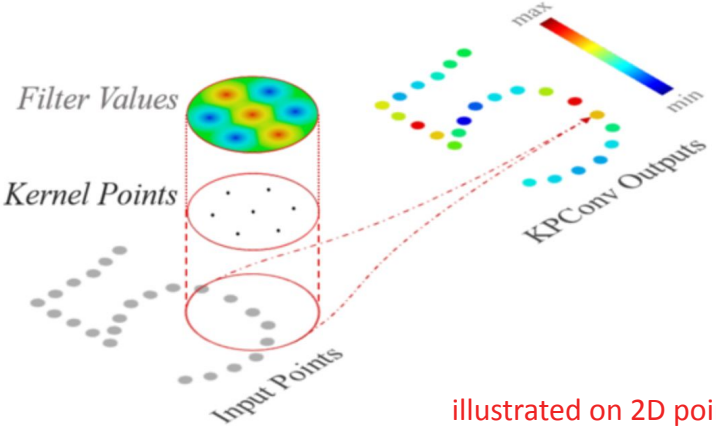




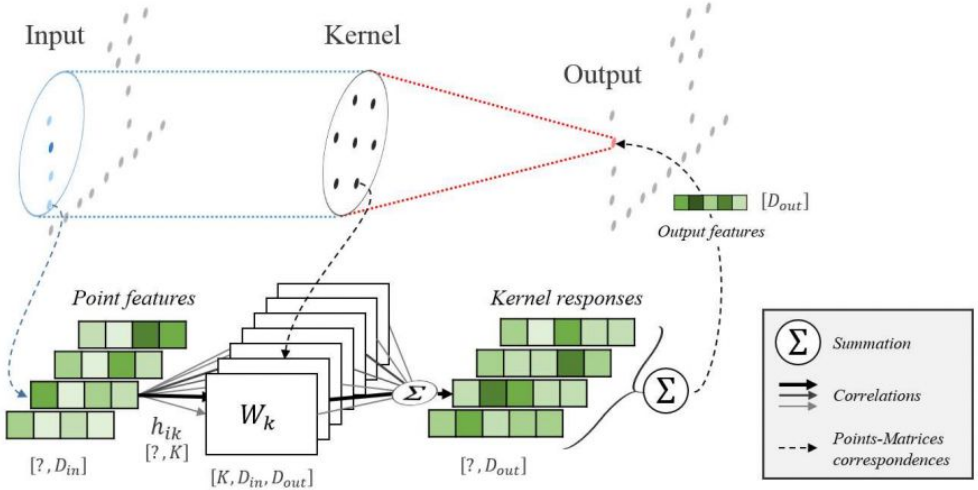
First-layer Point Embedding

- **Kernel Point Convolution** : Perform kernel point convolution on points within a radius r

- Place **kernel points** within the **kernel**
  - **Rigid** Kernel Point Convolution : Using a fixed kernel point location
  - **Deformable** Kernel Point Convolution : Kernel point position is learned and expresses the distribution of the surrounding point cloud
- Each kernel point have a **kernel weight**, which is correlation function(linear function that increases as the distance between each kernel point and the point)

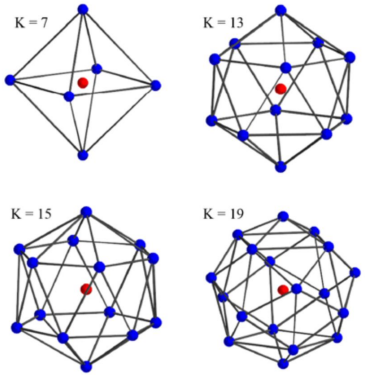


illustrated on 2D points

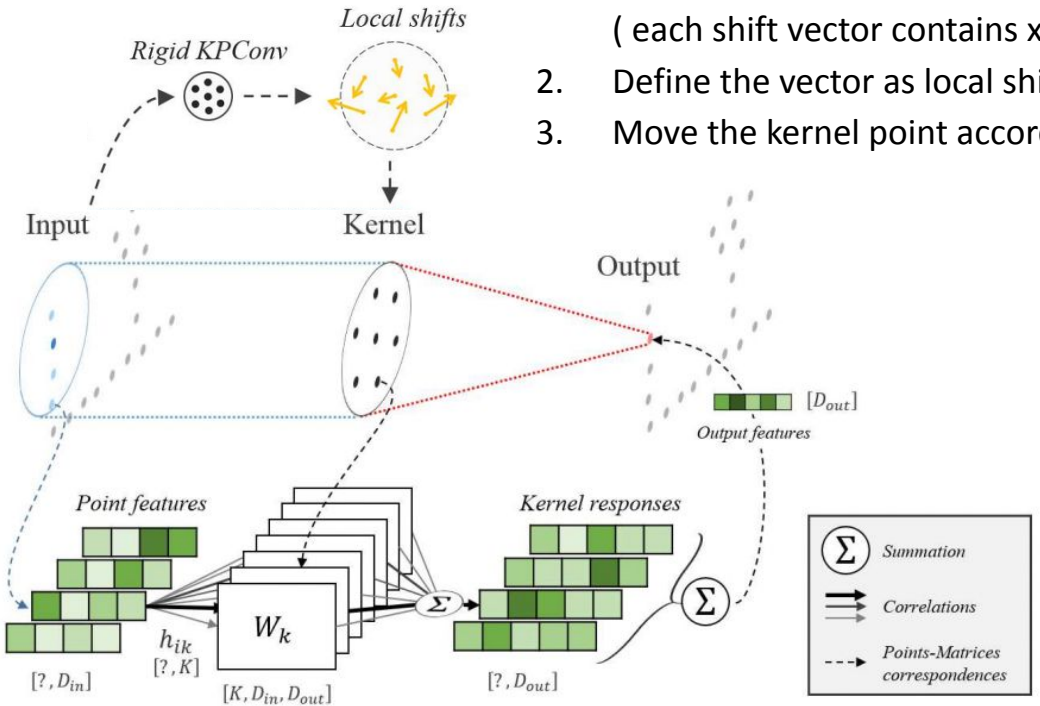


First-layer Point Embedding

- **Deformable Kernel Point Convolution**



1. Added a layer to extract shift vectors for K points using the rigid KPConv operation
   (each shift vector contains xyz coordinates)
2. Define the vector as local shift
3. Move the kernel point accordingly



illustrated on 2D points

Contextual Relative Position Encoding

Why use the Position Encoding?

Although the input of the Transformer block has already contained the *xyz* position,
fine-grained position information may be lost in high-level features when going deeper through the network.

- **Contextual**

: Considers the interaction with queries, keys or values.

- Change mode changes the encoding with the input feature

- **Relative Position Encoding**

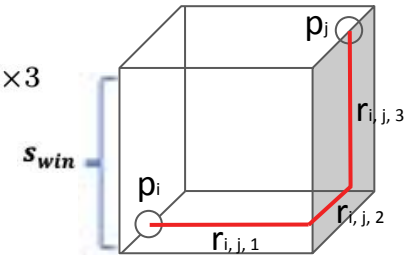
: Calculated via a look-up table with learnable parameters interacting with queries and keys in self-attention modules. Such scheme allows the modules to capture very long dependencies between tokens

Contextual Relative Position Encoding

Relative coordinates : continuous floating-point numbers

$$\mathbf{r}_{i,j,m} = \mathbf{p}_{i,m} - \mathbf{p}_{j,m}, \quad 1 \le i,j \le k_t, m \in \{1,2,3\}$$

$$\mathbf{r} \in \mathbb{R}^{k_t \times k_t \times 3}$$



Uniformly quantization

$(-s_{win}, s_{win})$ into L discrete parts

$$s_{quant} = \frac{2 \cdot s_{win}}{L}$$

$$\mathbf{idx}_{i,j,m} = \lfloor \frac{\mathbf{r}_{i,j,m} + s_{win}}{s_{quant}} \rfloor$$

it makes index to start at 0

relative coordinates **map** the learnable look-up tables **t**

$$\mathbf{t}[idx] \in \mathbb{R}^{N_h \times N_d}$$

$$\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z \in \mathbb{R}^{L \times (N_h \times N_d)}$$

$$\mathbf{e}_{i,j} = \mathbf{t}_x[\mathbf{idx}_{i,j,1}] + \mathbf{t}_y[\mathbf{idx}_{i,j,2}] + \mathbf{t}_z[\mathbf{idx}_{i,j,3}]$$

$$\mathbf{e} \in \mathbb{R}^{k_t \times k_t \times N_h \times N_d}$$

Contextual Relative Position Encoding

$$\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z \in \mathbb{R}^{L \times (N_h \times N_d)}$$

$$\mathbf{e}_{i,j} = \mathbf{t}_x[\mathbf{idx}_{i,j,1}] + \mathbf{t}_y[\mathbf{idx}_{i,j,2}] + \mathbf{t}_z[\mathbf{idx}_{i,j,3}]$$

$$\mathbf{pos\_bias}^{cRPE}_{i,j,h} = \mathbf{q}_{i,h} \cdot \mathbf{e}^{q}_{i,j,h} + \mathbf{k}_{j,h} \cdot \mathbf{e}^{k}_{i,j,h},$$

$$\mathbf{attn}^{cRPE}_{i,j,h} = \mathbf{q}_{i,h} \cdot \mathbf{k}_{j,h} + \mathbf{pos\_bias}^{cRPE}_{i,j,h},$$

$$\hat{\mathbf{attn}}^{cRPE}_{i,.,h} = softmax(\mathbf{attn}^{cRPE}_{i,.,h}),$$

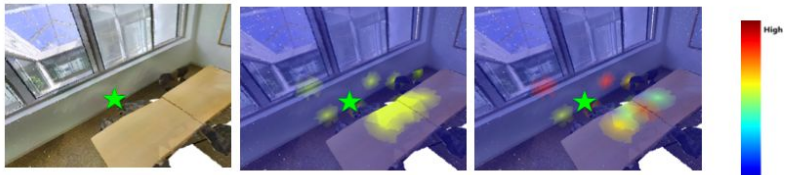
$$\mathbf{y}^{cRPE}_{i,h} = \sum_{j=1}^{k_t} \hat{\mathbf{attn}}^{cRPE}_{i,j,h} \times (\mathbf{v}_{j,h} + \mathbf{e}^{v}_{i,j,h}).$$

ID	PointEmb	Aug	cRPE	Stratified	S3DIS	ScanNet
I					56.8	56.8
II	✓				61.3	69.6
III	✓	✓			67.2	70.6
IV	✓	✓	✓		70.1	72.5
V	✓	✓	✓	✓	**72.0**	**73.7**
VI		✓	✓	✓	70.0	69.7
VII	✓		✓	✓	66.1	72.3
VIII	✓	✓		✓	68.0	71.4

$$\mathbf{t}[idx] \in \mathbb{R}^{N_h \times N_d}$$

$$\mathbf{e} \in \mathbb{R}^{k_t \times k_t \times N_h \times N_d}$$

$$\mathbf{pos\_bias} \in \mathbb{R}^{k_t \times k_t \times N_h}$$

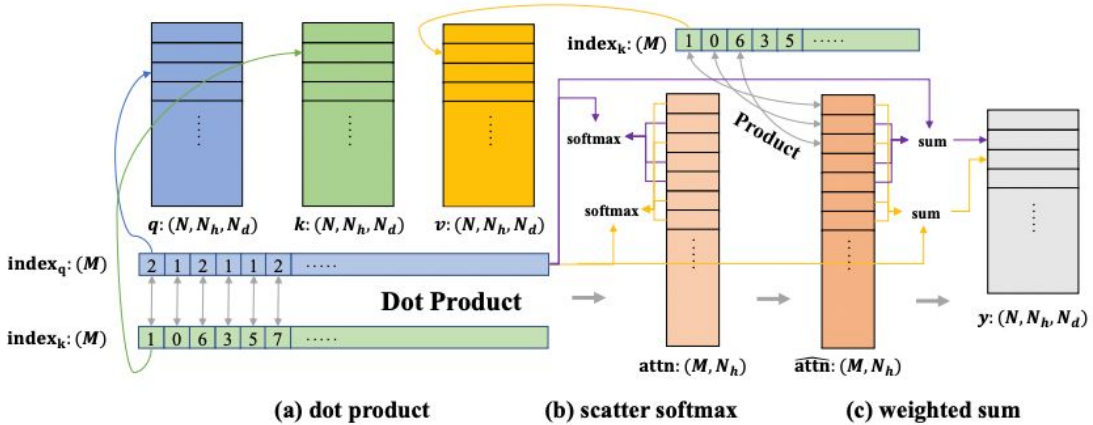


Input MLP-based cRPE

- MLP-based model
  : Positional bias is similar among the keys. So it makes little difference to the attention weights.
- cRPE
  : the positional bias varies a lot for different keys.

Memory-efficient Implementation ?

Each of the steps is implemented by a single **CUDA kernel**



So the intermediate variables inside each step hardly occupy memory

- **Cons** : Due to the irregular point arrangements in 3D, the number of the tokens in each window varies a lot
  - It cause unnecessary memory occupation for windows with a small number of points

Pre-compute all pairs of query and key that need to perform dot product

1. Perform dop product between the entries indexed by index$_q$ and index$_k$
2. Softmax function is applied on the entries in attn with the same index in indexq
3. Use index$_k$ to index the values v and multiply them with the attention map attn
4. Sum up the entries with the same index in indexq