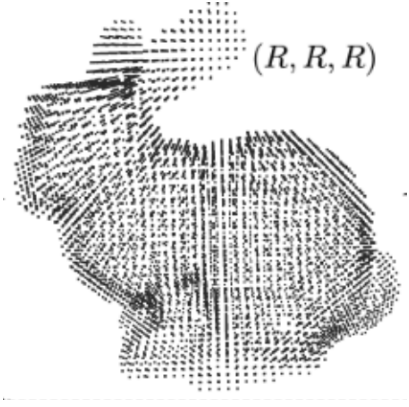
CENG5030 Lab 03: Sparse Conv

Bei Yu
Department of Computer Science & Engineering
Chinese University of Hong Kong
byu@cse.cuhk.edu.hk

November 22, 2023

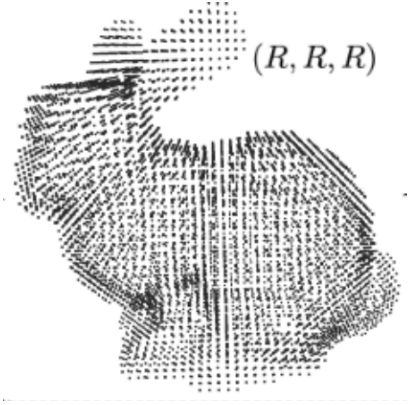
Voxel Data

In real world, we have to handle voxel data sometimes. For example, in point cloud analysis, 3D voxel data is widely used. A simple example is shown here and it can be viewed as $V \in (1, R, R, R)$.



Voxel Data

Here is a rabbit with shape $V \in (1, 64, 64, 64)$. If using traditional convolution to extract its feature, the GPU will out of memory very soon because the input $V \in (1, 64, 64, 64)$ can be viewed as an image $I \in (1, 2048, 2048)$.

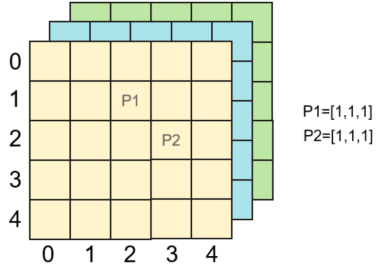


Submanifold Sparse Convolution

To overcome this issue, we use 3D sparse convolution for voxel data analysis.
Sparse convolution only calculate the data points where voxel data exists.

Submanifold Sparse Convolution

In this Lab, we are going to build a sparse convolution from scratch. Here we use the example input:



where P1 and P2 has pixel value of 1 in 3 channels.

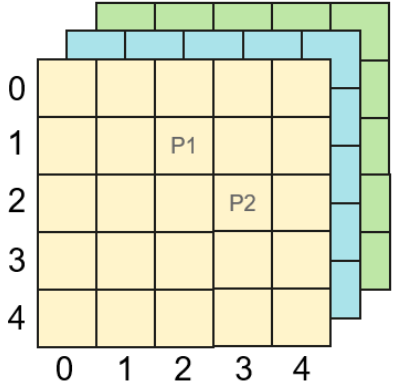
Submanifold Sparse Convolution

Firstly, we build a hash table to store the input data. Considering the following case:

conv2D(kernel_size=3, out_channels=2, stride=1, padding=0)

Submanifold Sparse Convolution

We can build an input table H_{in} like this:



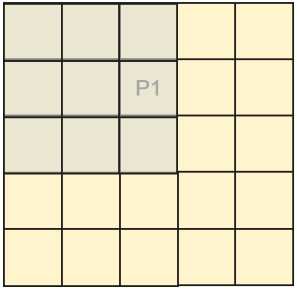
P1=[1,1,1]
P2=[1,1,1]

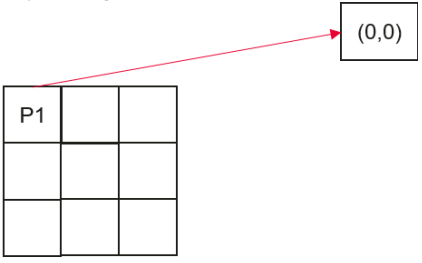
H_in

0	(2,1)
1	(3,2)

Submanifold Sparse Convolution

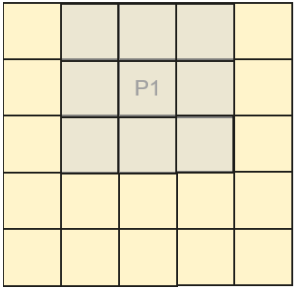
Then we build an output hash table. Firstly, we generate a P_{out} table as follow:

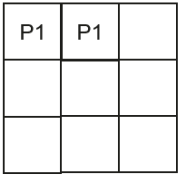




Submanifold Sparse Convolution

Then we build an output hash table. Firstly, we generate a P_{out} table as follow:

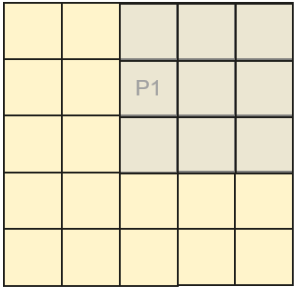




(0,0)
(1,0)

Submanifold Sparse Convolution

Then we build an output hash table. Firstly, we generate a P_{out} table as follow:

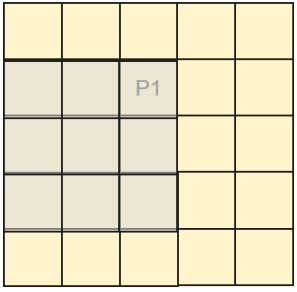


P1	P1	P1

(0,0)
(1,0)
(2,0)

Submanifold Sparse Convolution

Then we build an output hash table. Firstly, we generate a P_{out} table as follow:

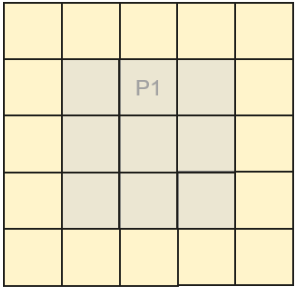


P1	P1	P1
P1		

(0,0)
(1,0)
(2,0)
(0,1)

Submanifold Sparse Convolution

Then we build an output hash table. Firstly, we generate a P_{out} table as follow:

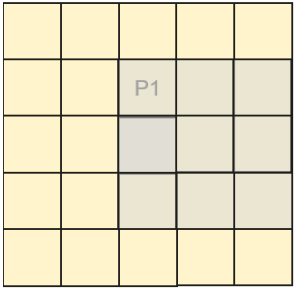


P1	P1	P1
P1	P1	

(0,0)
(1,0)
(2,0)
(0,1)
(1,1)

Submanifold Sparse Convolution

Then we build an output hash table. Firstly, we generate a P_{out} table as follow:

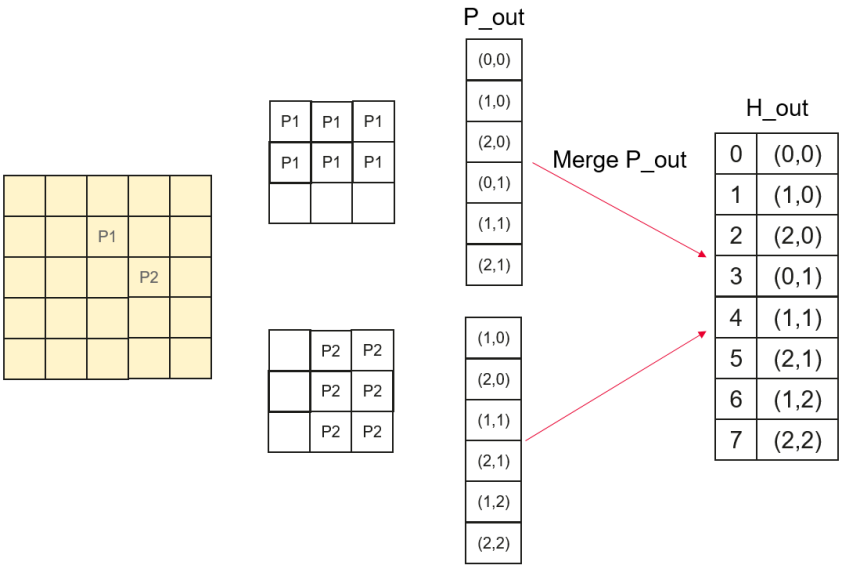


P1	P1	P1
P1	P1	P1

(0,0)
(1,0)
(2,0)
(0,1)
(1,1)
(2,1)

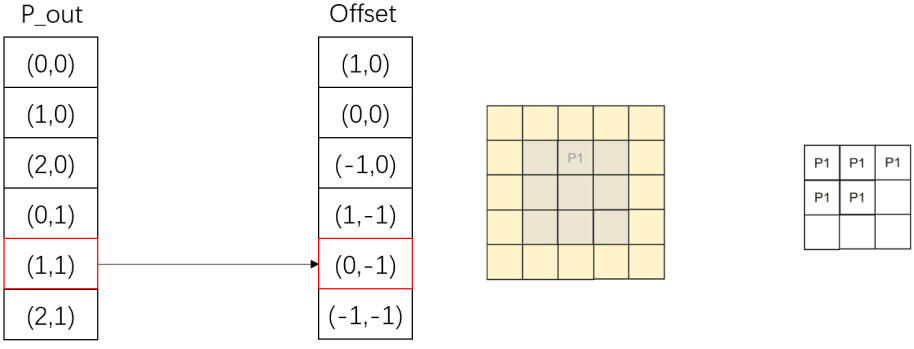
Submanifold Sparse Convolution

After applying the same process to P_2, we get an output hash table H_{out} via P_{out} merging:



Submanifold Sparse Convolution

- Next we build up a Rulebook to realize H_{in} to H_{out}.
- To build the rule book, we have to build an offset map like this:



Submanifold Sparse Convolution

Quick question, please write the offset map of $P2$ by yourself.

Submanifold Sparse Convolution

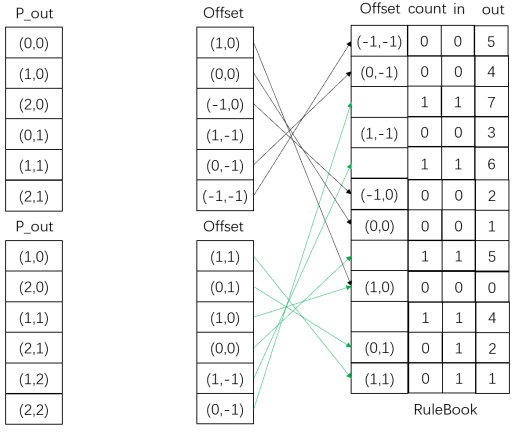
The offset map of $P2$ is:

P_out
(1,0)
(2,0)
(1,1)
(2,1)
(1,2)
(2,2)

Offset
(1,1)
(0,1)
(1,0)
(0,0)
(1,-1)
(0,-1)

Submanifold Sparse Convolution

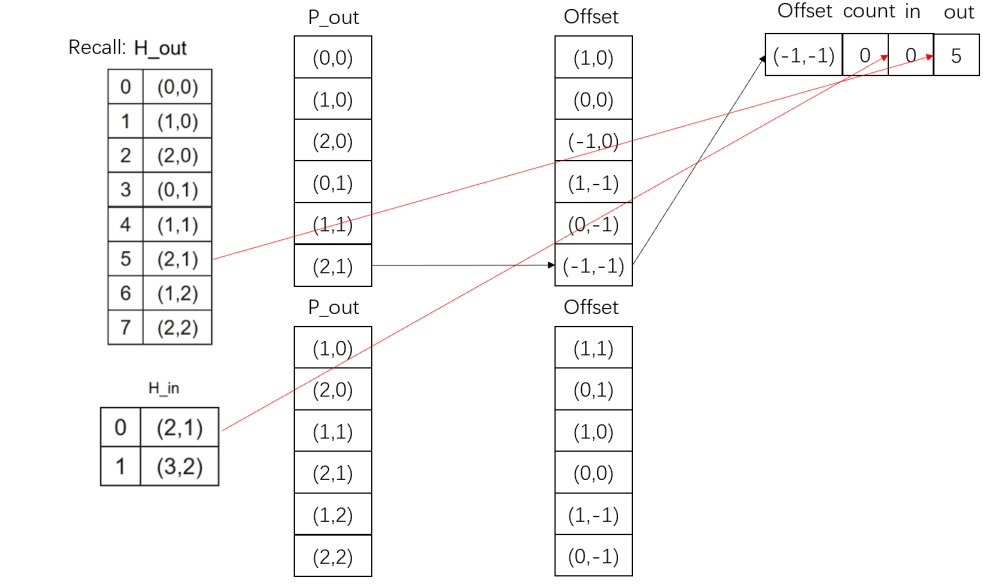
After obtaining the offset map, we can finally build up the rule book as follow:



The rulebook has 4 cloumn. The first column is offset and the second is the count. The third and fourth columns are the index of H_{in} and H_{out}.

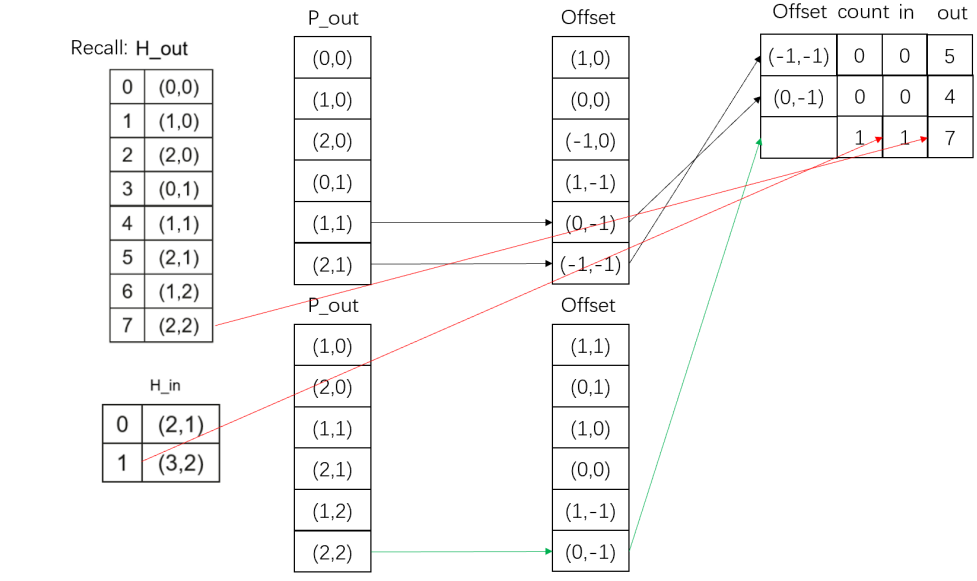
Submanifold Sparse Convolution

Recalling the H_{in} and H_{out}, the rulebook is generated as follow:



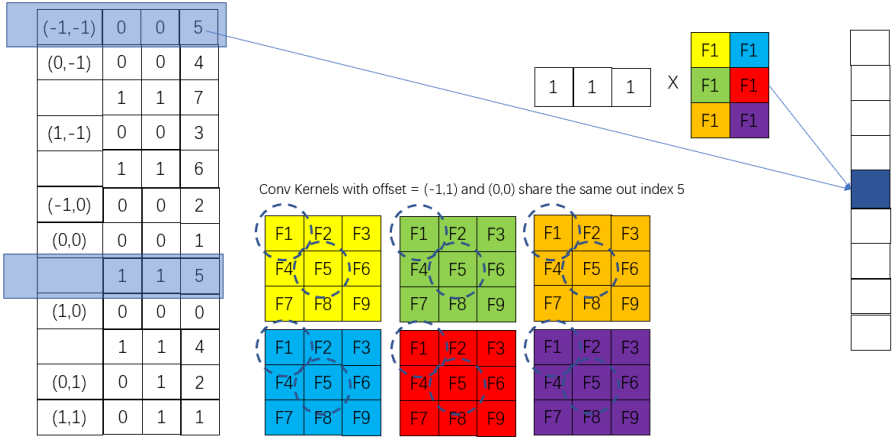
Submanifold Sparse Convolution

If the offset already exists, we simply add 1 in count:



Submanifold Sparse Convolution

After getting rulebook, we can apply sparse convolution:



For P_1, the reults is shown above, which is the blue points in 5-th row. Please practice P_2 by yourself

THANK YOU!