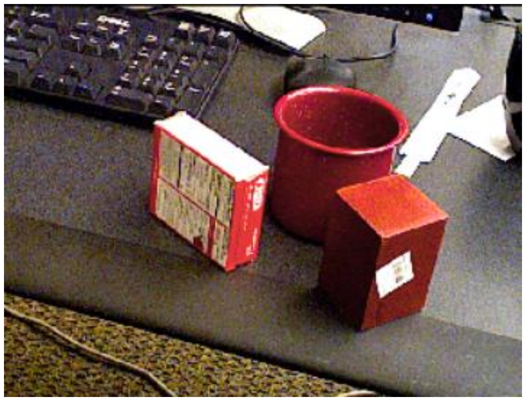
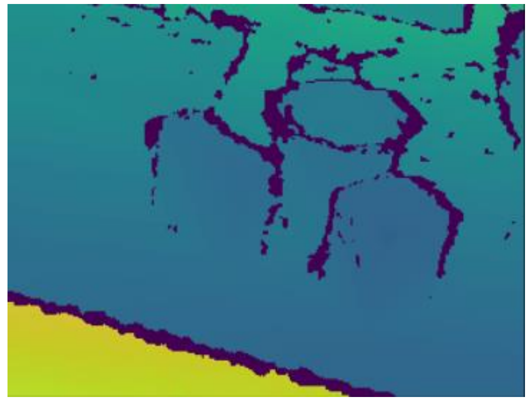
DenseFusion

KIST
송명하

Dataset

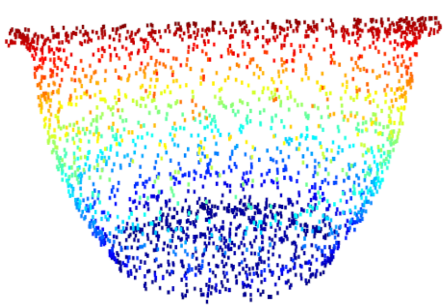


RGB



Depth

Label(seg)



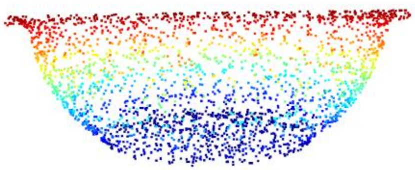
Point cloud

```
array([[[ 0.03187663,  0.49093292, -0.85895714],
        [-0.69099828, -0.86569373,  0.12561264],
        [ 0.7221546 ,  0.09777778, -0.49640202],
        [-0.02953567,  0.05711933,  0.08187473]],

       [[ 0.92482315, -0.43908223,  0.18477846],
        [-0.25362624, -0.34280246, -0.82808361],
        [-0.28350625, -0.83047825, -0.52927736],
        [-0.0077438 , -0.02501409,  0.0351884 ]],

       [[ 0.37905814,  0.75245806, -0.47754636],
        [ 0.67690162,  0.36477607, -0.54635045],
        [ 0.63096654, -0.54840286,  0.68807699],
        [ 0.58834638,  0.61850619,  0.53858151]]])
```

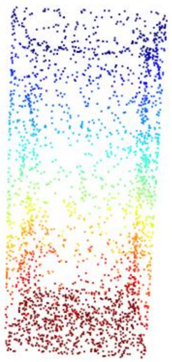
Pose(Matrix)

Bounding Box

Dataset (Symmetric Object)



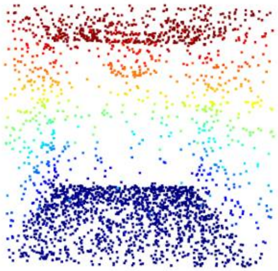
Bowl



Wood_block

Large_clamp

Extra_large_clamp

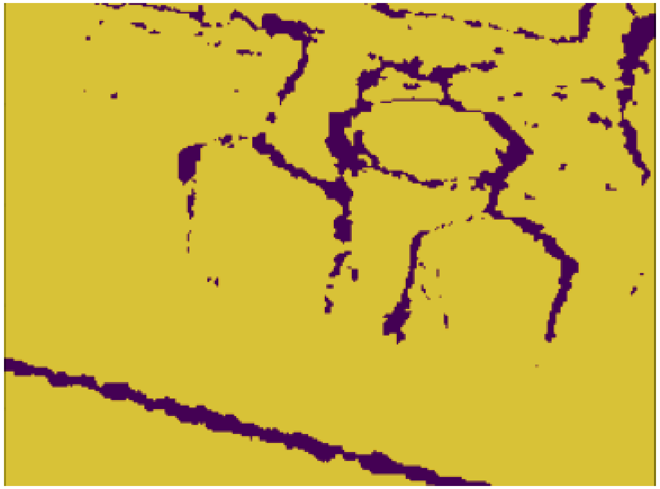


Foam_brick

Use the Segmentation label(PoseCNN result) to remove the background

-> Data augmentation(adding noise)

DenseFusion



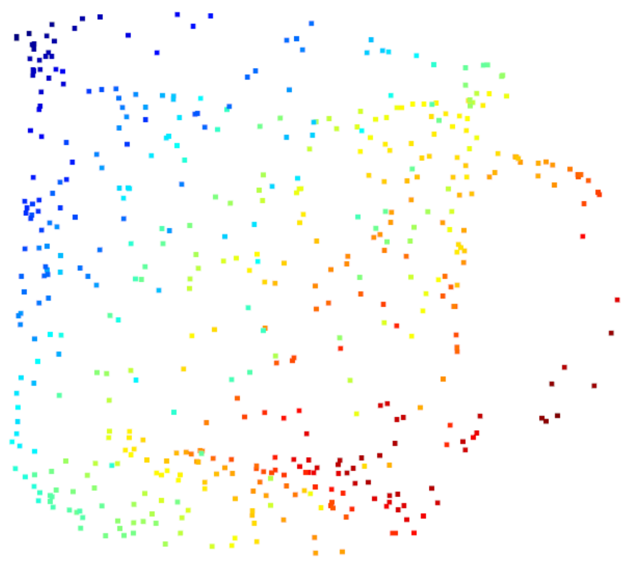
In the Depth image, only nonzero parts are made True and then multiplied by the semantic label of the object. The result is shown in the image below

DenseFusion

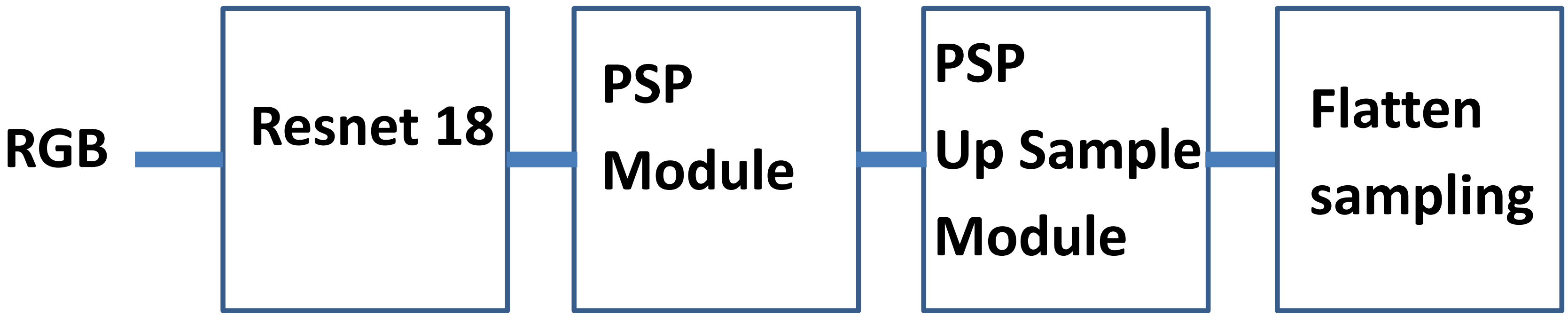
1. Use the bound box's Corrdinates(PoseCNN result) to cut out the image and the depth mask
2. Only a nonzero value is selected after flattening the depth mask value
3. Randomly sampling 1000 out of the values created in step2, The location of the selected value is stored in a variable named choose
4. In the depth masks, indexing is done using the choose variable
5. Create a variable named xmap and ymap indexing it in the same way, and create a pointcloud.

The number of points in the object – 500 randomly sampled. Remove from model point and save 500 points as model point.

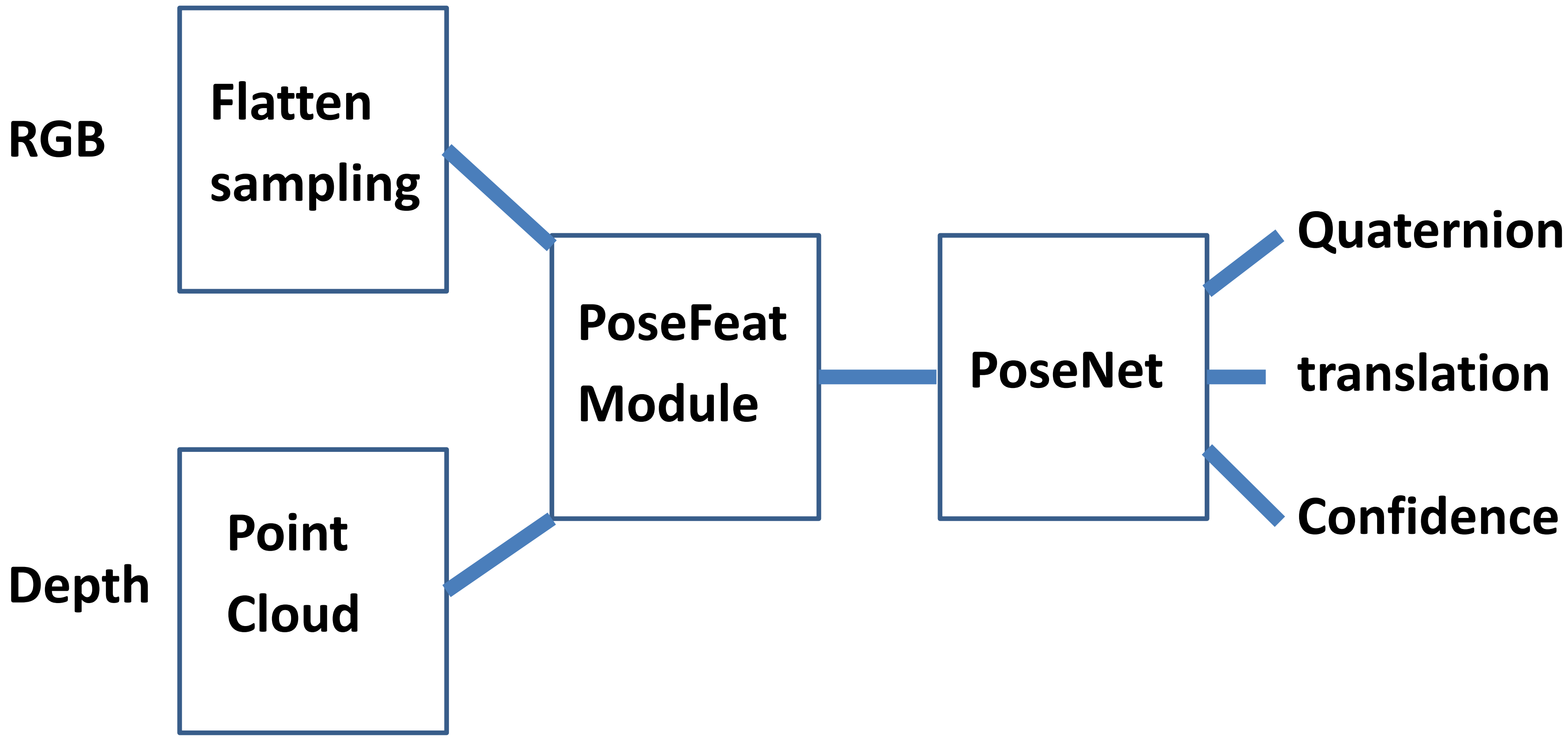
Multiply the rotation matrix at the created model point and add translation.



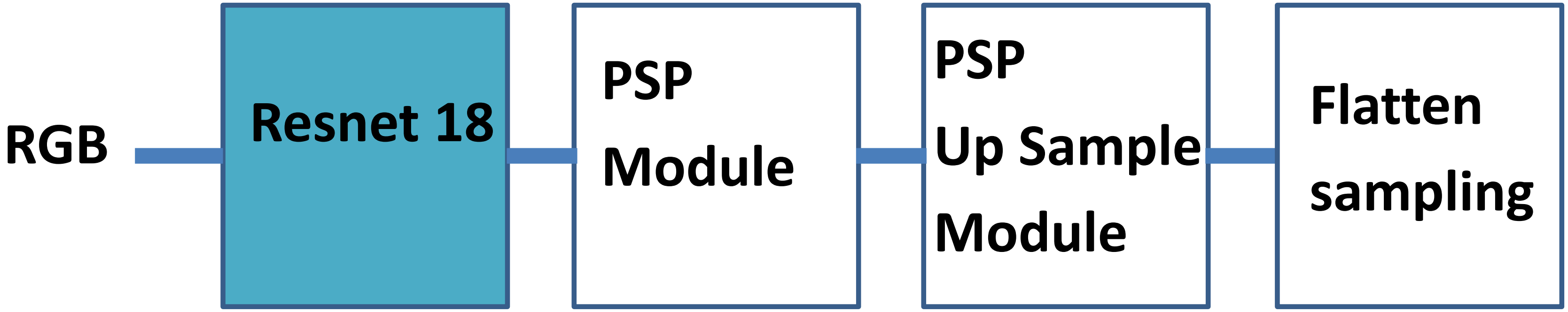
Network Architecture



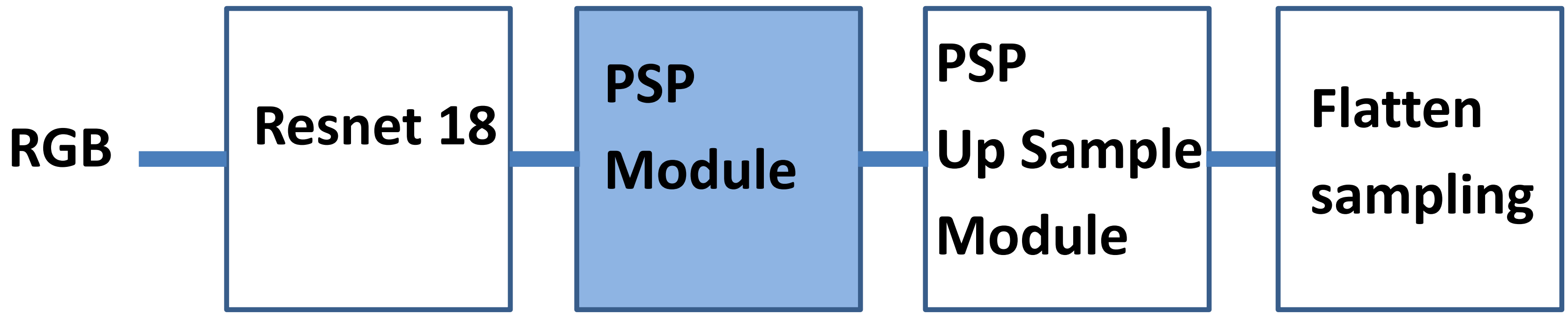
Network Architecture



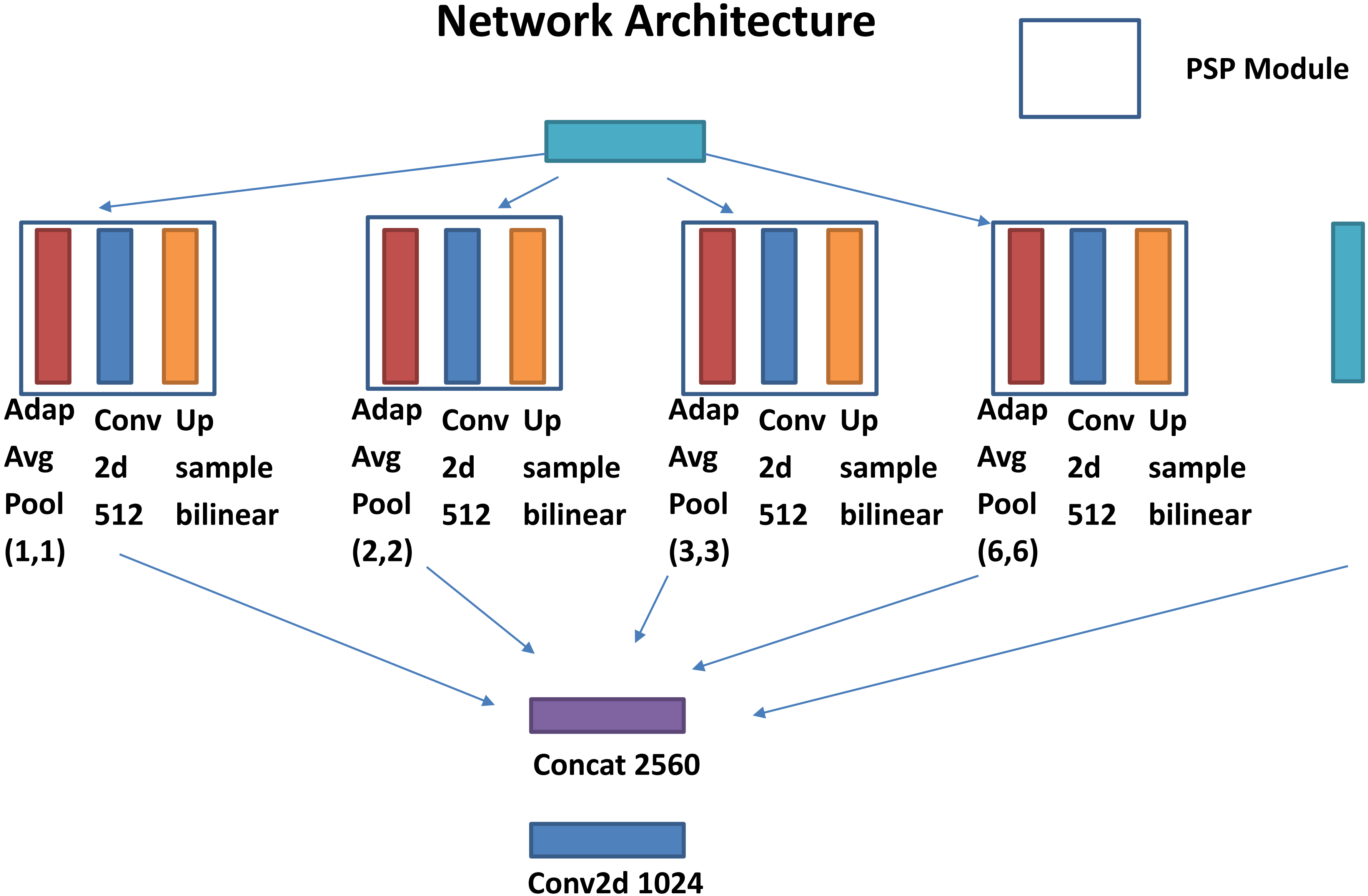
Network Architecture



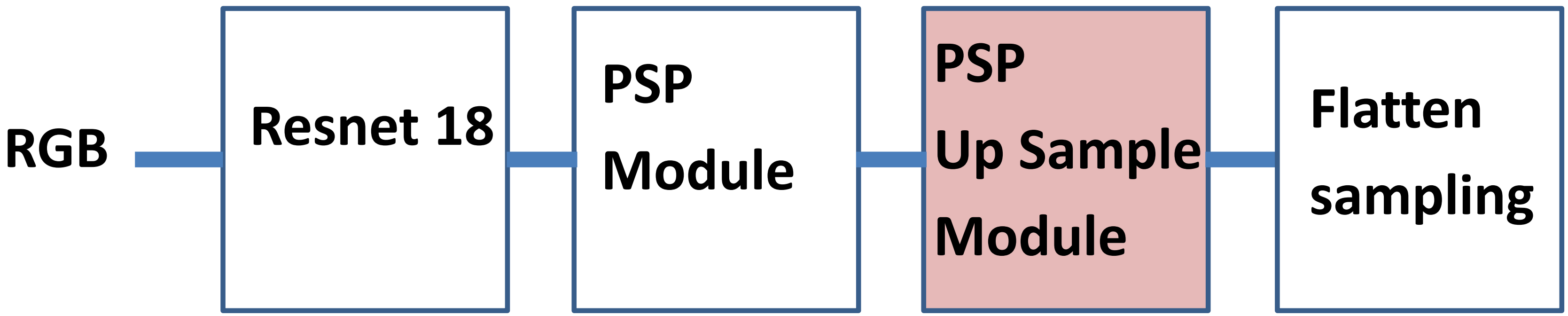
Network Architecture



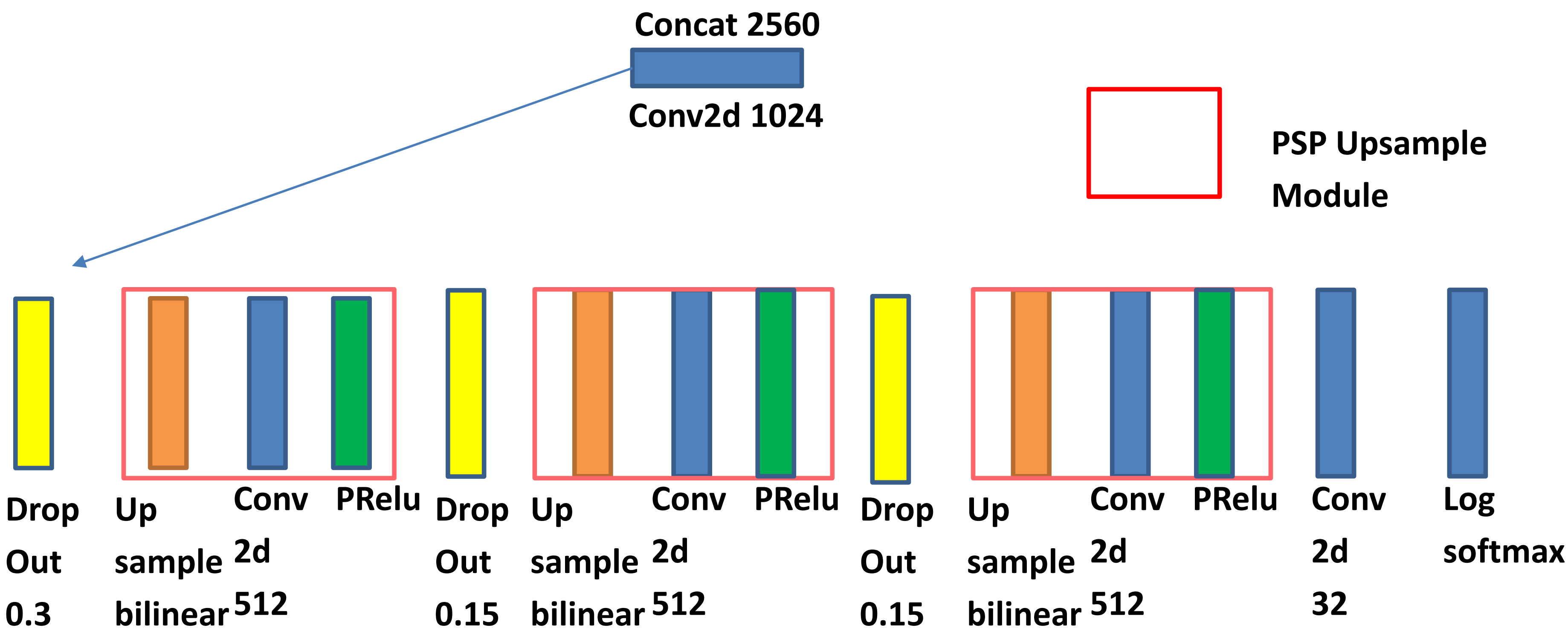
Network Architecture



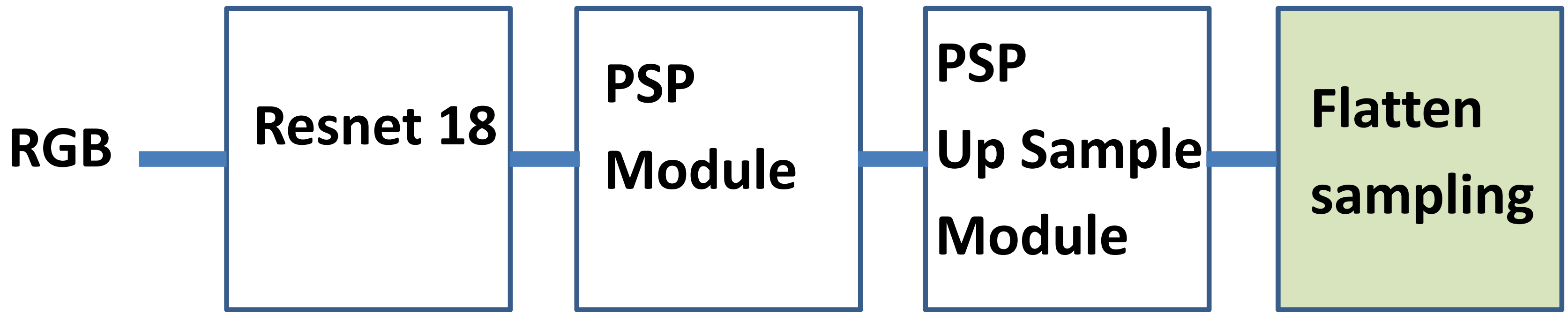
Network Architecture



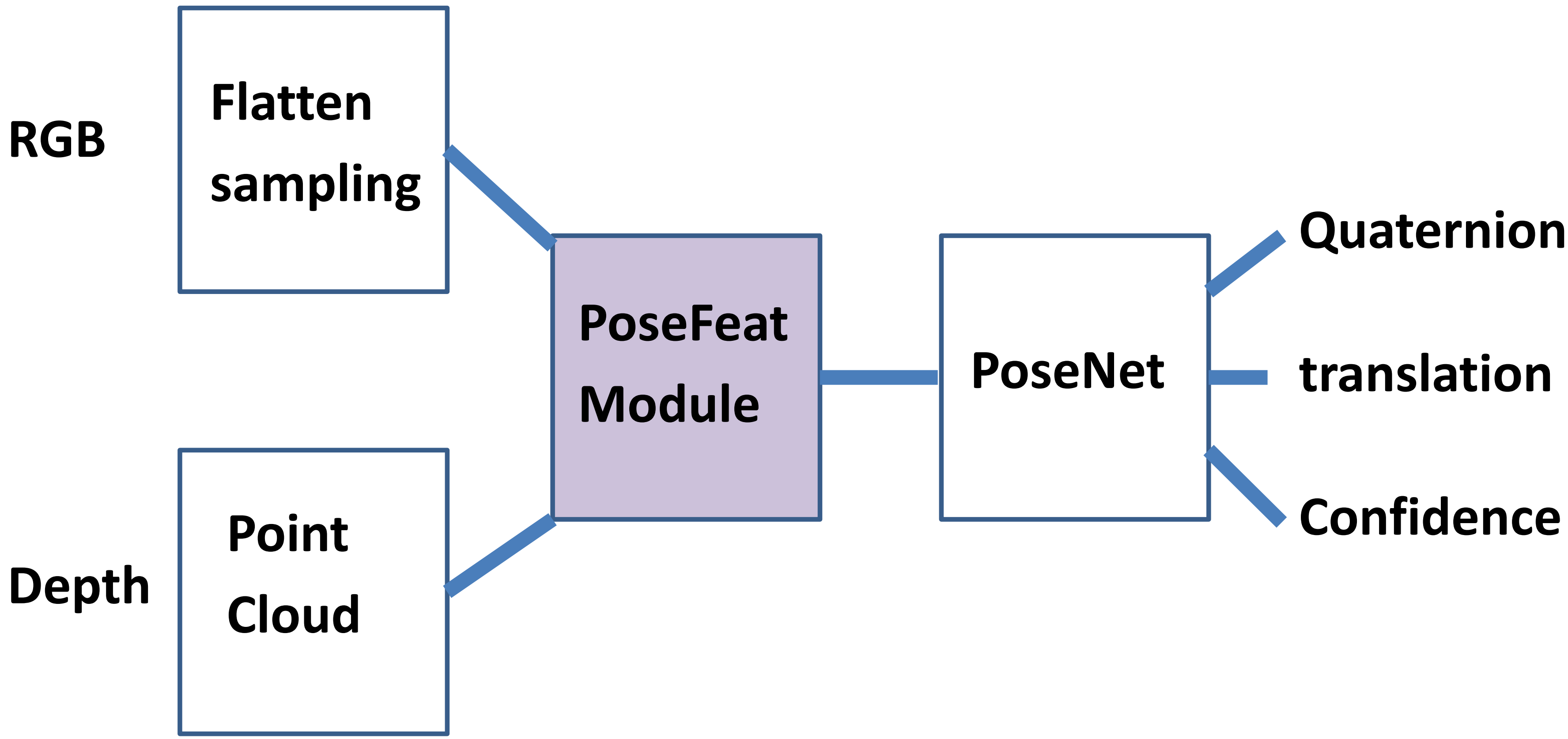
Network Architecture



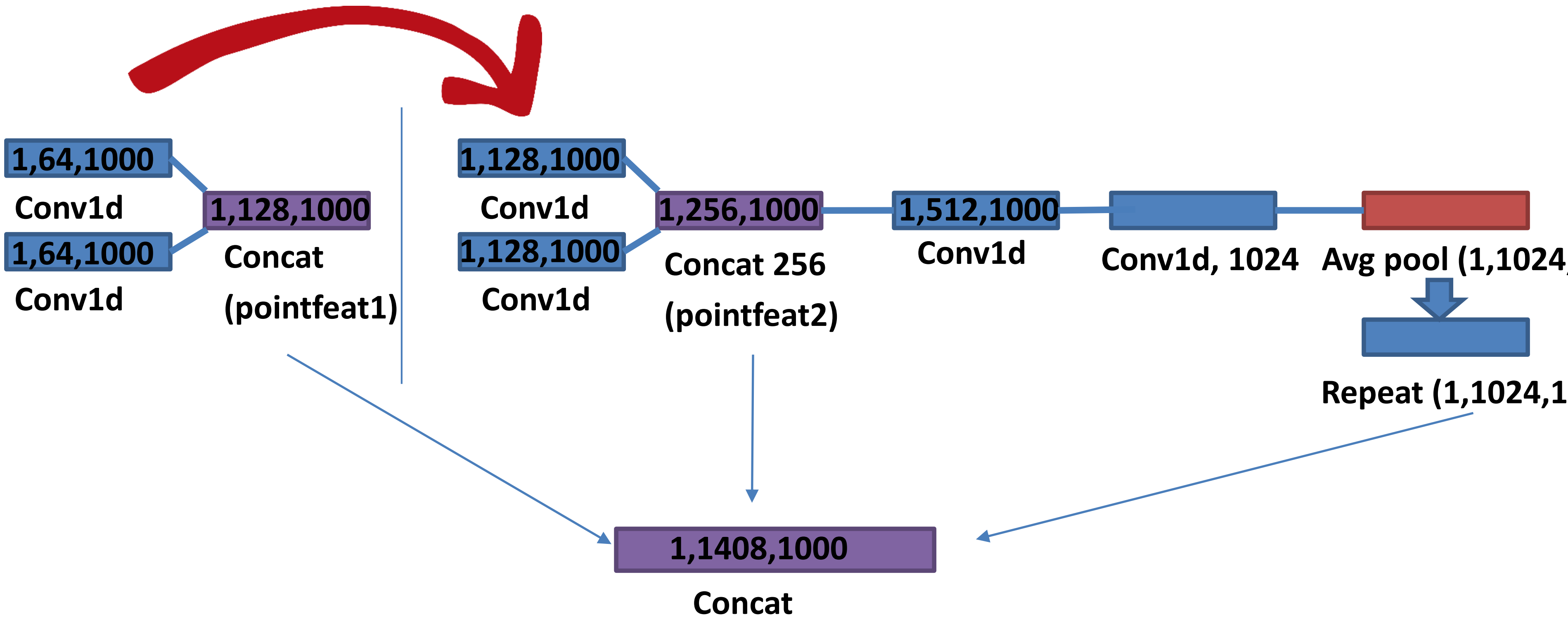
Network Architecture



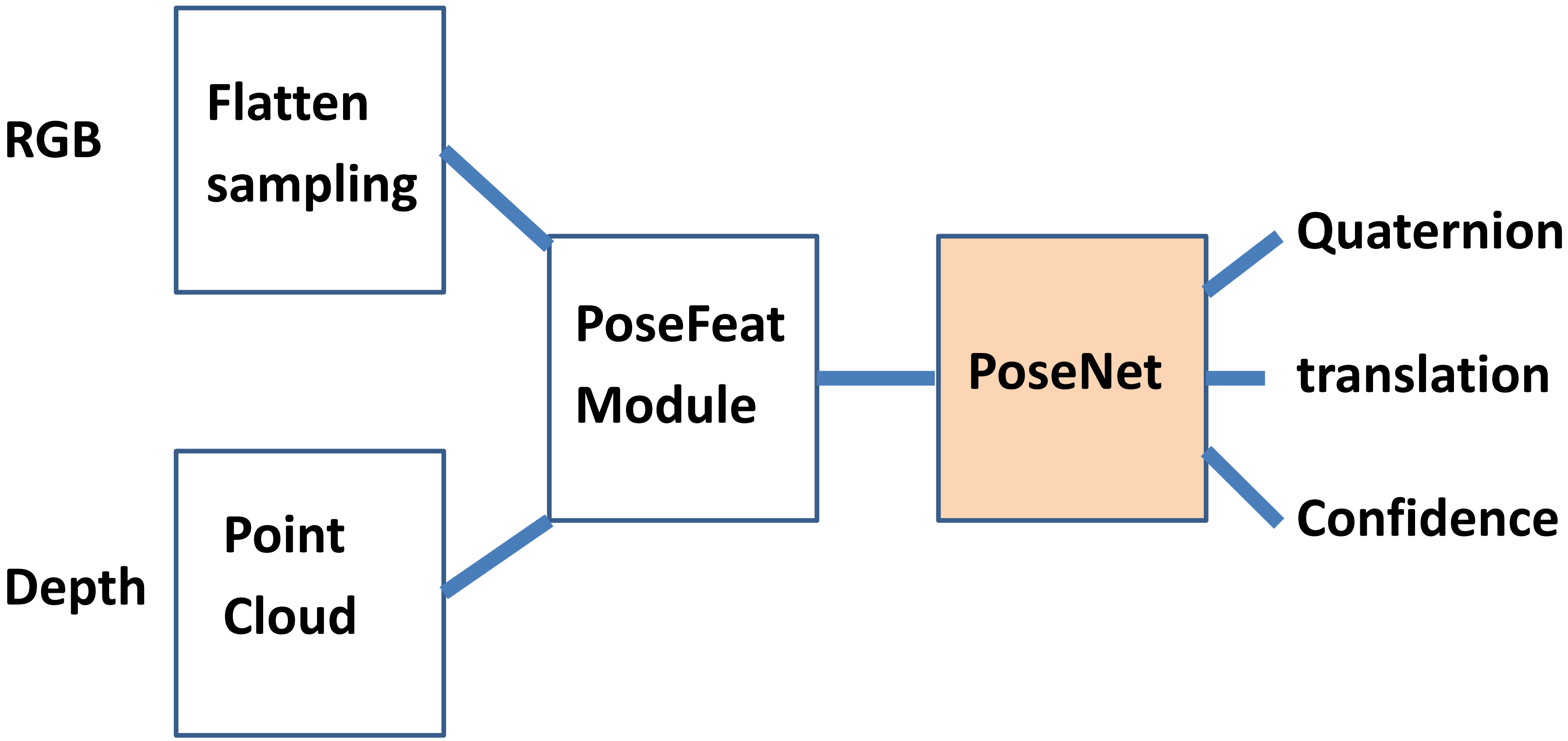
Network Architecture



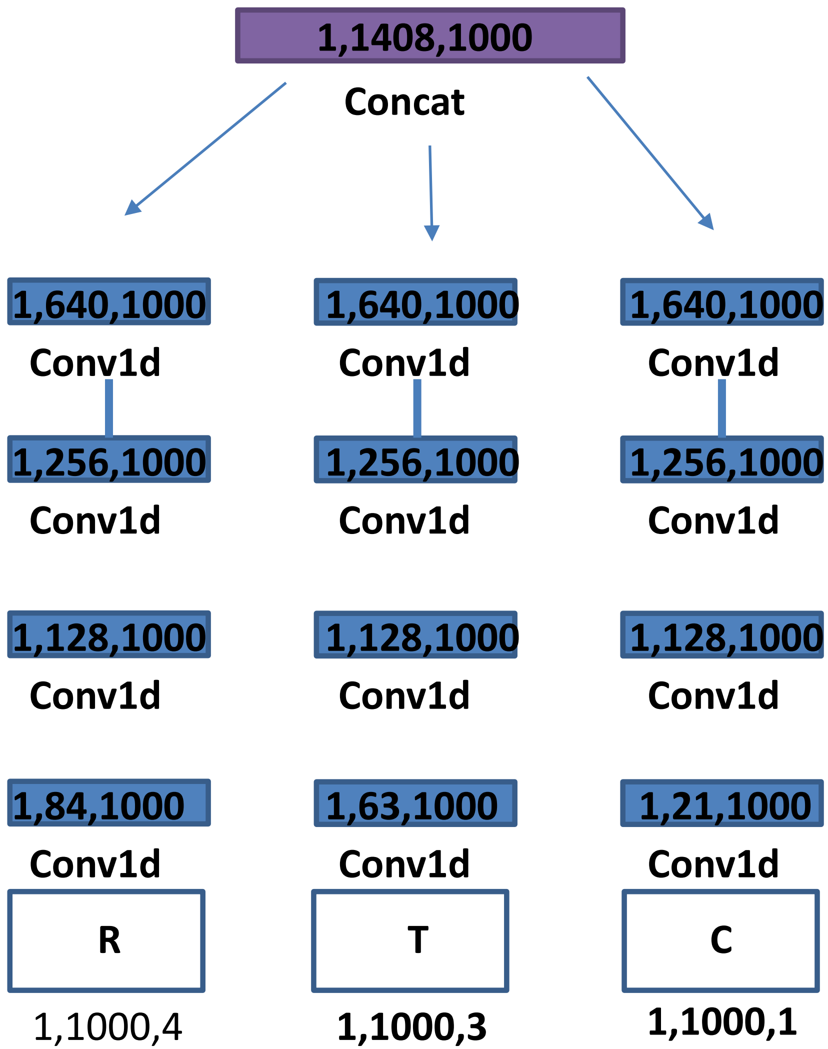
Network Architecture



Network Architecture



Network Architecture

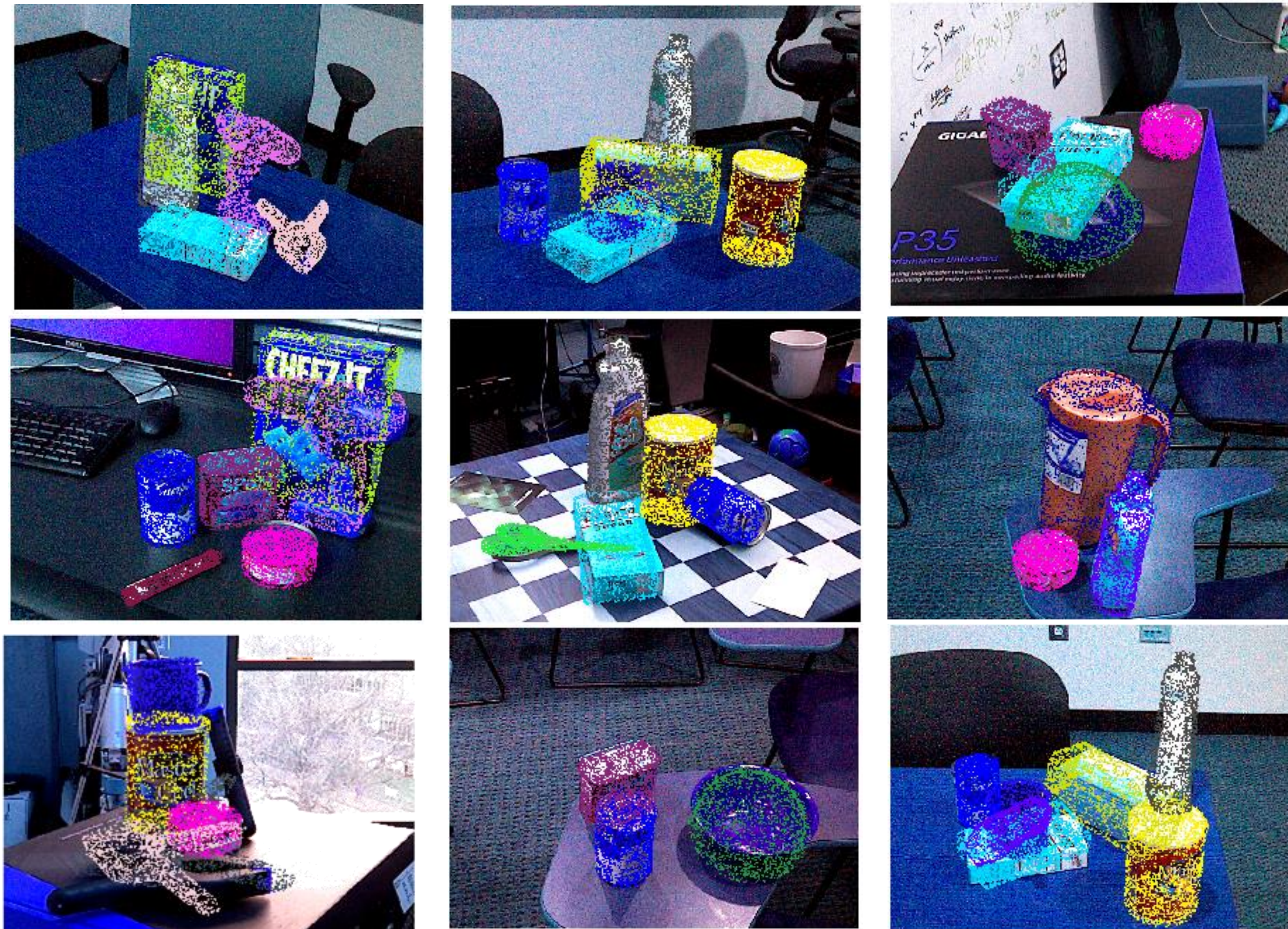


Loss

$$L_i^p = \frac{1}{M} \sum_j ||(Rx_j + t) - (\hat{R}_i x_j + \hat{t}_i)|| \qquad (1)$$

$$L = \frac{1}{N} \sum_i (L_i^p c_i - w \log(c_i)),$$

Result



속도

Table 3. Runtime breakdown (second per frame on YCB-Video Dataset). Our method is approximately 200x faster than PoseCNN+ICP. Seg means Segmentation, and PE means Pose Estimation.

PoseCNN+ICP [40]				Ours			
Seg	PE	ICP	ALL	Seg	PE	Refine	ALL
0.03	0.17	10.4	10.6	0.03	0.02	0.01	0.06

16FPS, about 5 objects in each frame (16.6fps)

End