Korea Advanced Institute of Science and Technology

PSMent

Pyramid stereo matching network, 2018 CVPR, Spatial pyramid pooling module implementation

한국과학기술원

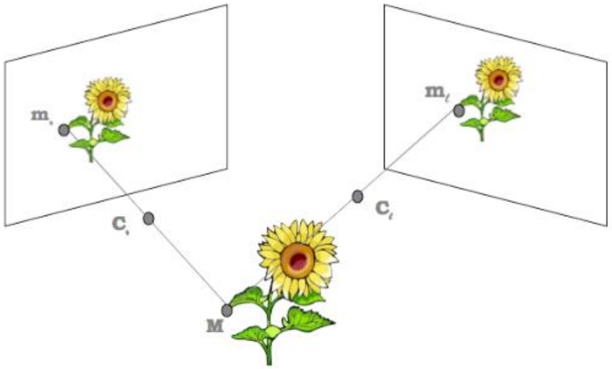
시각지능연구실

조훈희

Introduction – Stereo matching

What is stereo matching?

- Stereo matching is process of finding correspondence from one image (left) with the other image (right)
- We can call the difference in x axis as "disparity"
- By computing all this "disparity" for all pixels, we can get disparity image
- Using this disparity image along with the baseline information and focal length, we can acquire the depth



Pyramid stereo Matching Network.

The major problem with current CNN-based stereo matching method

- It is still hard to find accurate correspondences in ill-posed regions.
(occlusion areas, repeated patterns, texture-less regions, and reflective surfaces, etc.)

- Solely applying the intensity-consistency constraint b/w different viewpoints is insufficient for accurate correspondence estimation in such ill-posed regions.

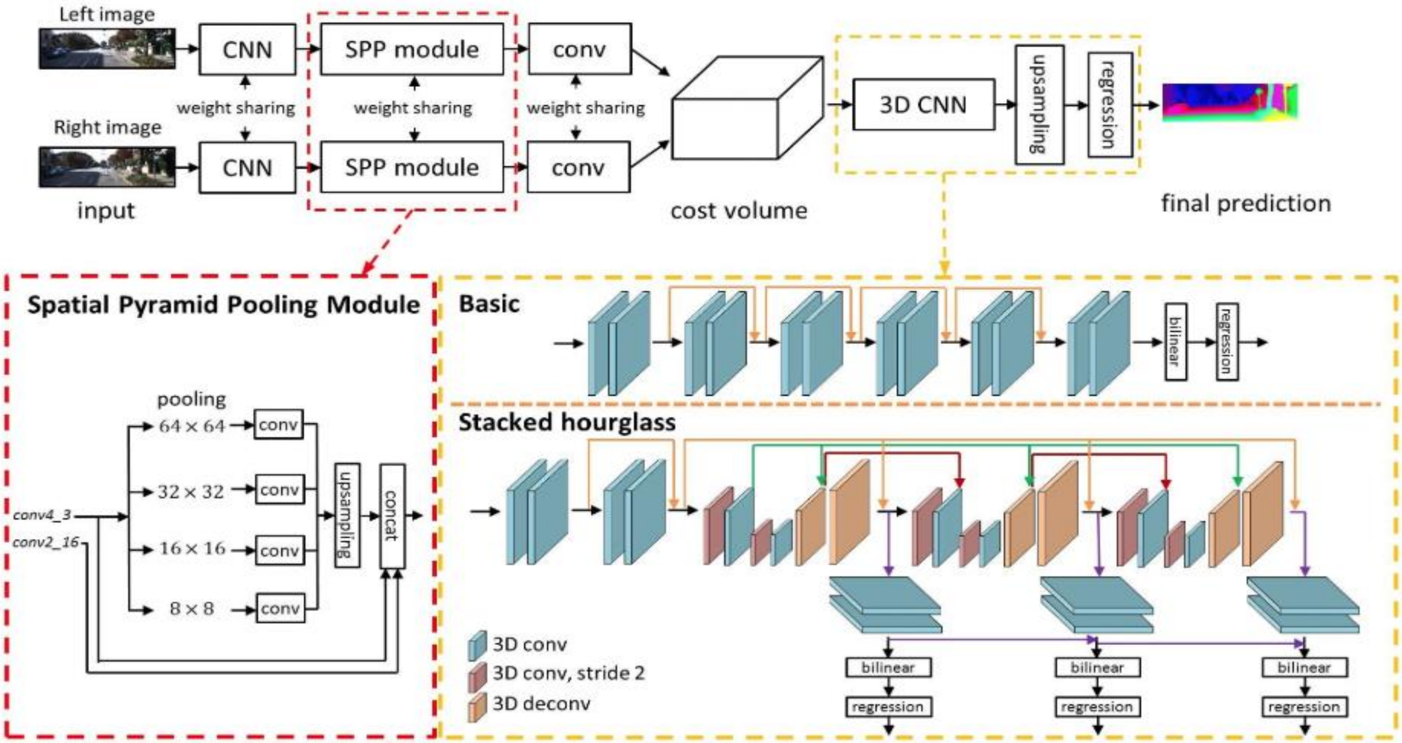
- Global context information must be incorporated into stereo matching(ex, DispNet, CRL, GC-Net)

Then, how to effectively exploit context information?

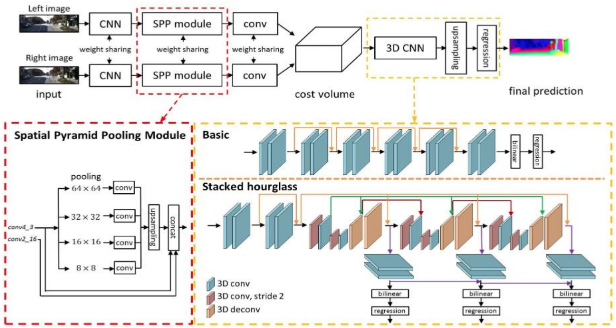
Pyramid stereo Matching Network.

Pyramid stereo matching network

- Spatial Pyramid Pooling module
- 3D CNN(stacked hourglass)



Pyramid stereo Matching Network.



Network layers

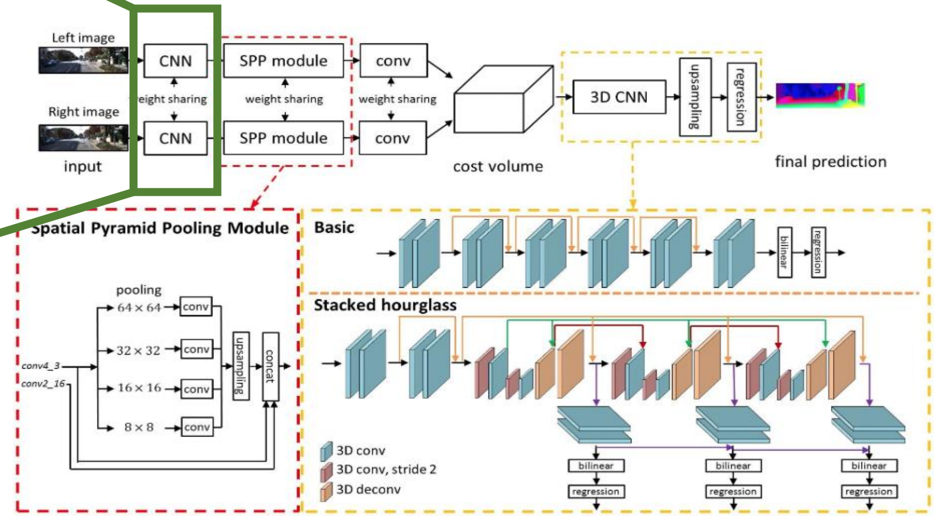
Name	Layer setting	Output dimension
input		$H \times W \times 3$
CNN		
conv0_1	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_2	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_3	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv1_x	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 16$	$\frac{1}{4}H \times \frac{1}{4}W \times 64$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila = 2	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
conv4_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila= 4	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
SPP module		
branch_1	64×64 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_2	32×32 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_3	16×16 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_4	8×8 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
concat[conv2_16, conv4_3, branch_1, branch_2, branch_3, branch_4]		$\frac{1}{4}H \times \frac{1}{4}W \times 320$
fusion	$3 \times 3, 128$ $1 \times 1, 32$	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
Cost volume		
Concat left and shifted right		$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 64$

3D CNN (stacked hourglass)		
3Dconv0	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 32$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dconv1	$\begin{bmatrix} 3 \times 3 \times 3, 32 \\ 3 \times 3 \times 3, 32 \end{bmatrix}$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dstack1_1	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack1_2	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 64$
3Dstack1_3	deconv $3 \times 3 \times 3, 64$ add **3Dstack1_1**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack1_4	deconv $3 \times 3 \times 3, 32$ add **3Dconv1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dstack2_1	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$ add **3Dstack1_3**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack2_2	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 64$
3Dstack2_3	deconv $3 \times 3 \times 3, 64$ add **3Dstack1_1**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack2_4	deconv $3 \times 3 \times 3, 32$ add **3Dconv1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dstack3_1	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$ add **3Dstack2_3**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack3_2	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 64$
3Dstack3_3	deconv $3 \times 3 \times 3, 64$ add **3Dstack1_1**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack3_4	deconv $3 \times 3 \times 3, 32$ add **3Dconv1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
output_1	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 1$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 1$
output_2	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 1$ add **output_1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 1$
output_3	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 1$ add **output_2**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 1$

Pyramid stereo Matching Network.

Network layers – feature extraction

Name	Layer setting	Output dimension
input		$H \times W \times 3$
CNN		
conv0_1	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_2	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_3	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv1_x	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 16$	$\frac{1}{4}H \times \frac{1}{4}W \times 64$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila = 2	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
conv4_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila= 4	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
SPP module		
branch_1	64×64 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_2	32×32 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_3	16×16 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_4	8×8 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
concat[conv2_16, conv4_3, branch_1, branch_2, branch_3, branch_4]		$\frac{1}{4}H \times \frac{1}{4}W \times 320$
fusion	$3 \times 3, 128$ $1 \times 1, 32$	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
Cost volume		
Concat left and shifted right		$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 64$



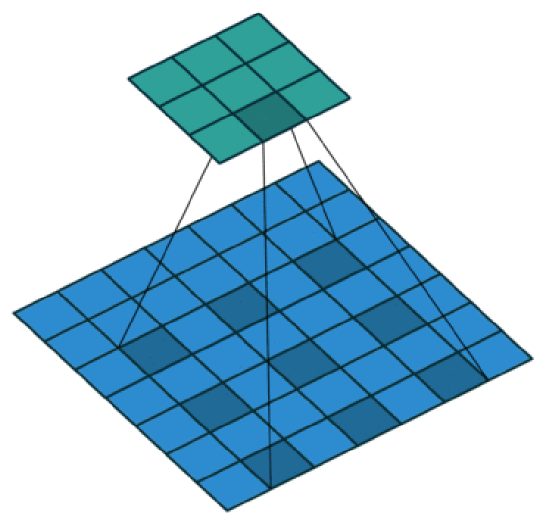
Network layers – Dilated convolution

- In dilated convolution, we define dilation as the spacing between kernel.

- As 3x3 kernel with a dilation rate of 2 uses 9 parameters and has the same receptive field as a 5x5 kernel.

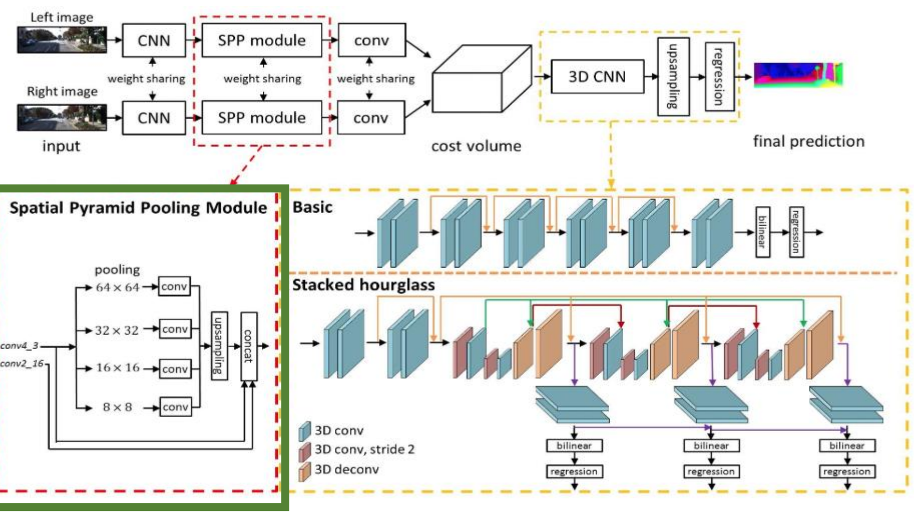
Dilated Convolutions (확장된 Convolution)

(a.k.a. atrous convolutions)



Network layers – SPP module

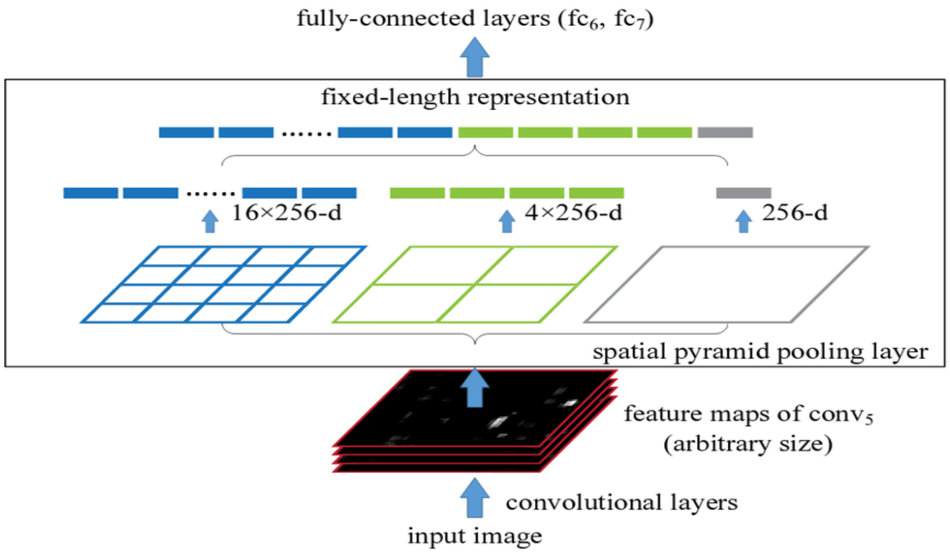
Name	Layer setting	Output dimension
input		$H \times W \times 3$
CNN		
conv0_1	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_2	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_3	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv1_x	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 16$	$\frac{1}{4}H \times \frac{1}{4}W \times 64$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila = 2	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
conv4_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila= 4	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
SPP module		
branch_1	64×64 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_2	32×32 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_3	16×16 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_4	8×8 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
concat[conv2_16, conv4_3, branch_1, branch_2, branch_3, branch_4]		$\frac{1}{4}H \times \frac{1}{4}W \times 320$
fusion	$3 \times 3, 128$ $1 \times 1, 32$	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
Cost volume		
Concat left and shifted right		$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 64$



Pyramid stereo Matching Network.

Spatial pyramid pooling (SPP)

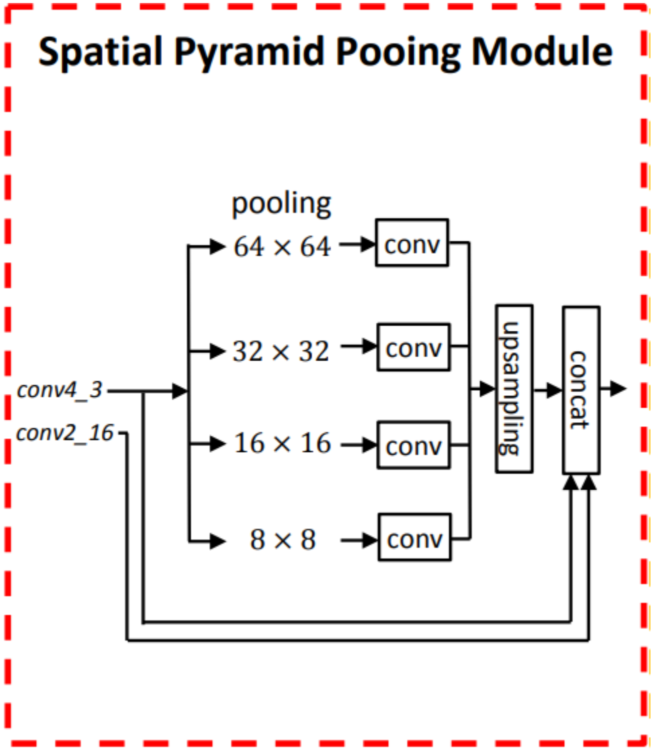
- To enlarge the receptive field.
- Enables PSMNet to **extend pixel-level features to region-level features** with different scales of receptive fields
- **Global and local feature clues are used** to form the cost volume for reliable disparity estimation.



Pyramid stereo Matching Network.

Network layers - Spatial pyramid pooling (SPP)

: <u>The relationship between an object and its sub-regions</u> is learned by the SPP module to incorporate hierarchical context information.

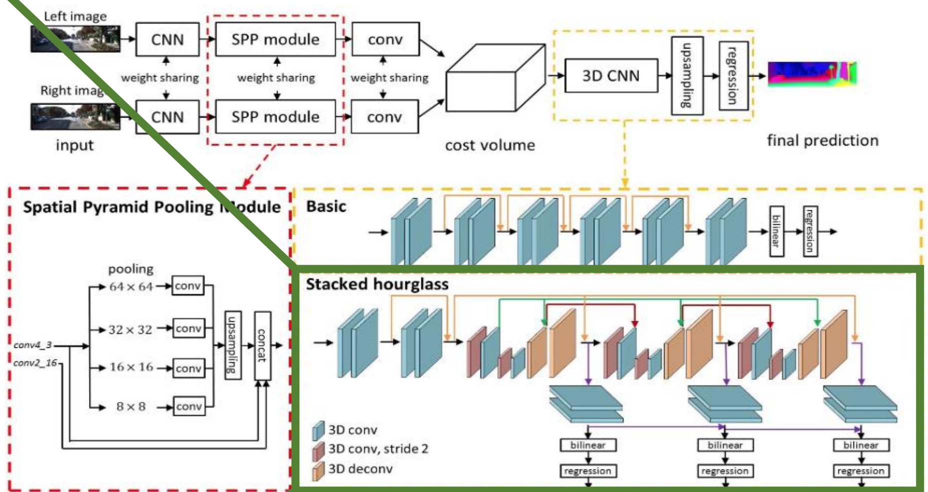


SPP module		
branch_1	64 × 64 avg. pool 3 × 3, 32 bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_2	32 × 32 avg. pool 3 × 3, 32 bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_3	16 × 16 avg. pool 3 × 3, 32 bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_4	8 × 8 avg. pool 3 × 3, 32 bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
concat[conv2_16, conv4_3, branch_1, branch_2, branch_3, branch_4]		$\frac{1}{4}H \times \frac{1}{4}W \times 320$
fusion	3 × 3, 128 1 × 1, 32	$\frac{1}{4}H \times \frac{1}{4}W \times 32$

Pyramid stereo Matching Network.

Network layers – 3D CNN

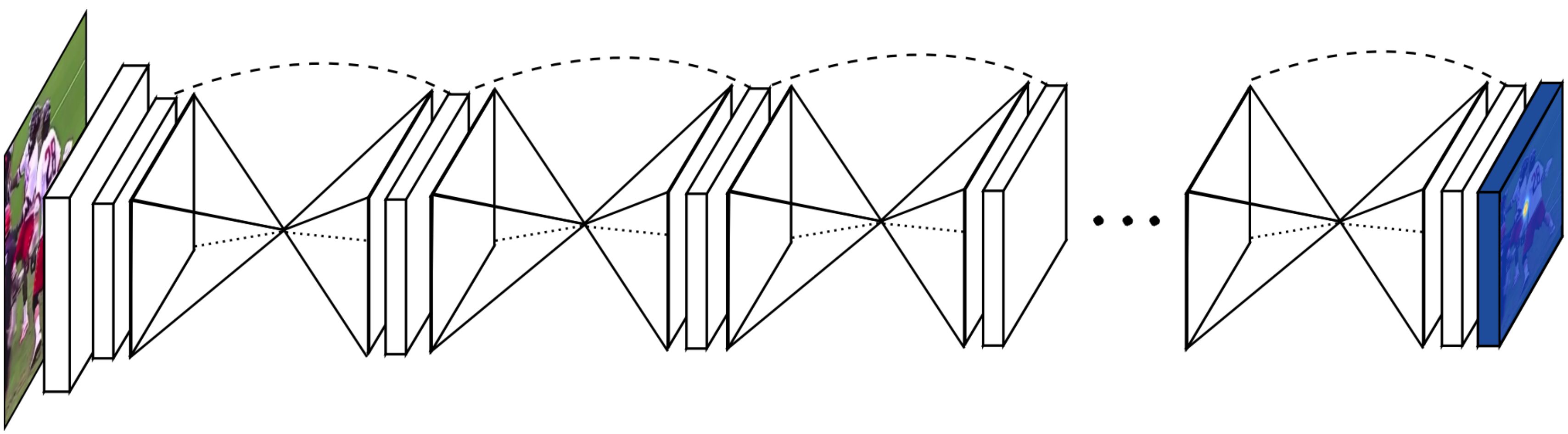
3D CNN (stacked hourglass)		
3Dconv0	$3\times3\times3, 32$ $3\times3\times3, 32$	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times32$
3Dconv1	$\begin{bmatrix}3\times3\times3, 32\\3\times3\times3, 32\end{bmatrix}$	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times32$
3Dstack1_1	$3\times3\times3, 64$ $3\times3\times3, 64$	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack1_2	$3\times3\times3, 64$ $3\times3\times3, 64$	$\frac{1}{16}D\times\frac{1}{16}H\times\frac{1}{16}W\times64$
3Dstack1_3	deconv $3\times3\times3, 64$ add **3Dstack1_1**	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack1_4	deconv $3\times3\times3, 32$ add **3Dconv1**	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times32$
3Dstack2_1	$3\times3\times3, 64$ $3\times3\times3, 64$ add **3Dstack1_3**	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack2_2	$3\times3\times3, 64$ $3\times3\times3, 64$	$\frac{1}{16}D\times\frac{1}{16}H\times\frac{1}{16}W\times64$
3Dstack2_3	deconv $3\times3\times3, 64$ add **3Dstack1_1**	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack2_4	deconv $3\times3\times3, 32$ add **3Dconv1**	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times32$
3Dstack3_1	$3\times3\times3, 64$ $3\times3\times3, 64$ add **3Dstack2_3**	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack3_2	$3\times3\times3, 64$ $3\times3\times3, 64$	$\frac{1}{16}D\times\frac{1}{16}H\times\frac{1}{16}W\times64$
3Dstack3_3	deconv $3\times3\times3, 64$ add **3Dstack1_1**	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack3_4	deconv $3\times3\times3, 32$ add **3Dconv1**	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times32$
output_1	$3\times3\times3, 32$ $3\times3\times3, 1$	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times1$
output_2	$3\times3\times3, 32$ $3\times3\times3, 1$ add **output_1**	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times1$
output_3	$3\times3\times3, 32$ $3\times3\times3, 1$ add **output_2**	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times1$



Pyramid stereo Matching Network.

(Stacked hourglass) 3D CNN

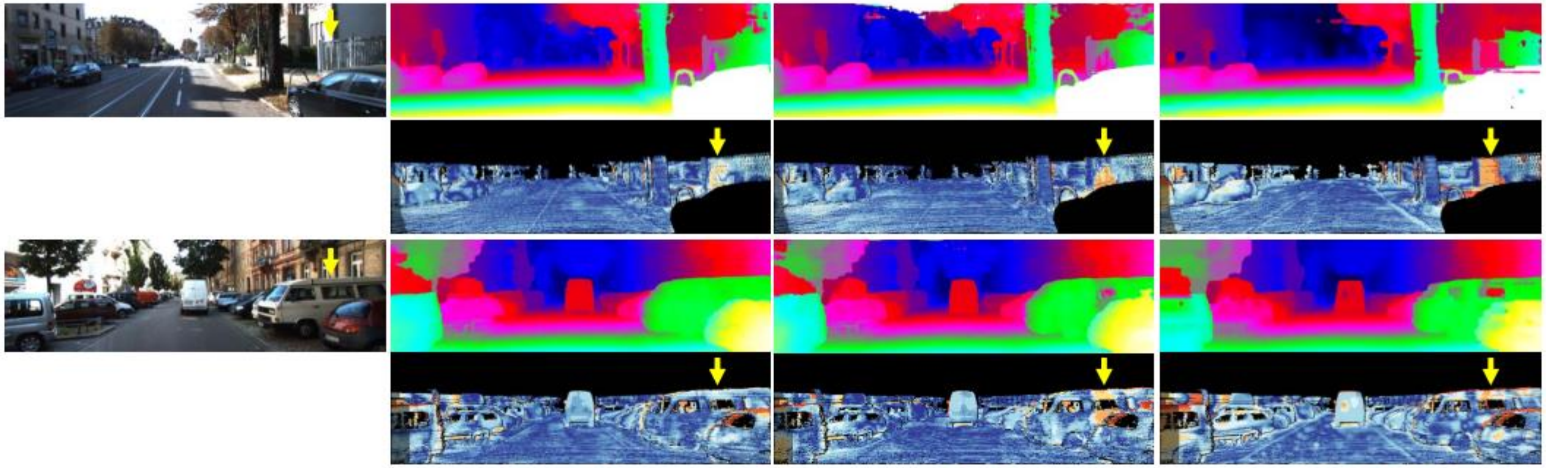
- **To regularize cost volume.**
- Repeatedly processes the cost volume in a top-down/bottom-up manner **<u>to further improve the utilization of global context information</u>**
- Extends the regional support of context information in the cost volume.



Results of disparity estimation for KITTI 2015

Table 4. The KITTI 2015 leaderboard presented on March 18, 2018. The results show the percentage of pixels with errors of more than three pixels or 5% of disparity error from all test images. Only published methods are listed for comparison.

Rank	Method	All (%)			Noc (%)			Runtime (s)
		D1-bg	D1-fg	D1-all	D1-bg	D1-fg	D1-all	
1	PSMNet (ours)	**1.86**	4.62	**2.32**	**1.71**	4.31	**2.14**	0.41
3	iResNet-i2e2 [14]	2.14	3.45	2.36	1.94	3.20	2.15	0.22
6	iResNet [14]	2.35	**3.23**	2.50	2.15	**2.55**	2.22	**0.12**
8	CRL [21]	2.48	3.59	2.67	2.32	3.12	2.45	0.47
11	GC-Net [13]	2.21	6.16	2.87	2.02	5.58	2.61	0.90



(a) PSMNet (b) GC-Net (c) MC-CNN

Figure 2. Results of disparity estimation for KITTI 2015 test images. The left panel shows the left input image of stereo image pair. For each input image, the disparity maps obtained by (a) PSMNet, (b) GC-Net [13], and (c) MC-CNN [30] are illustrated together above their error maps.

prerequisite

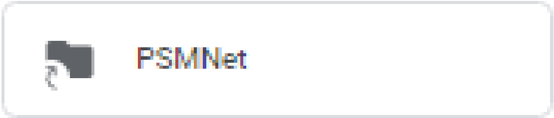
Linking with google drive for colab

1. Google 계정을 로그인 하세요

2. 다음 링크 클릭
https://drive.google.com/drive/folders/1UgthQcbGnMq0A_wguC9px5qrmCbs07Wc?usp=sharing

3. 좌측 상단의 PSMNet 우클릭-> 드라이브에 바로가기 추가-> 내 드라이브 -> 바로가기 추가

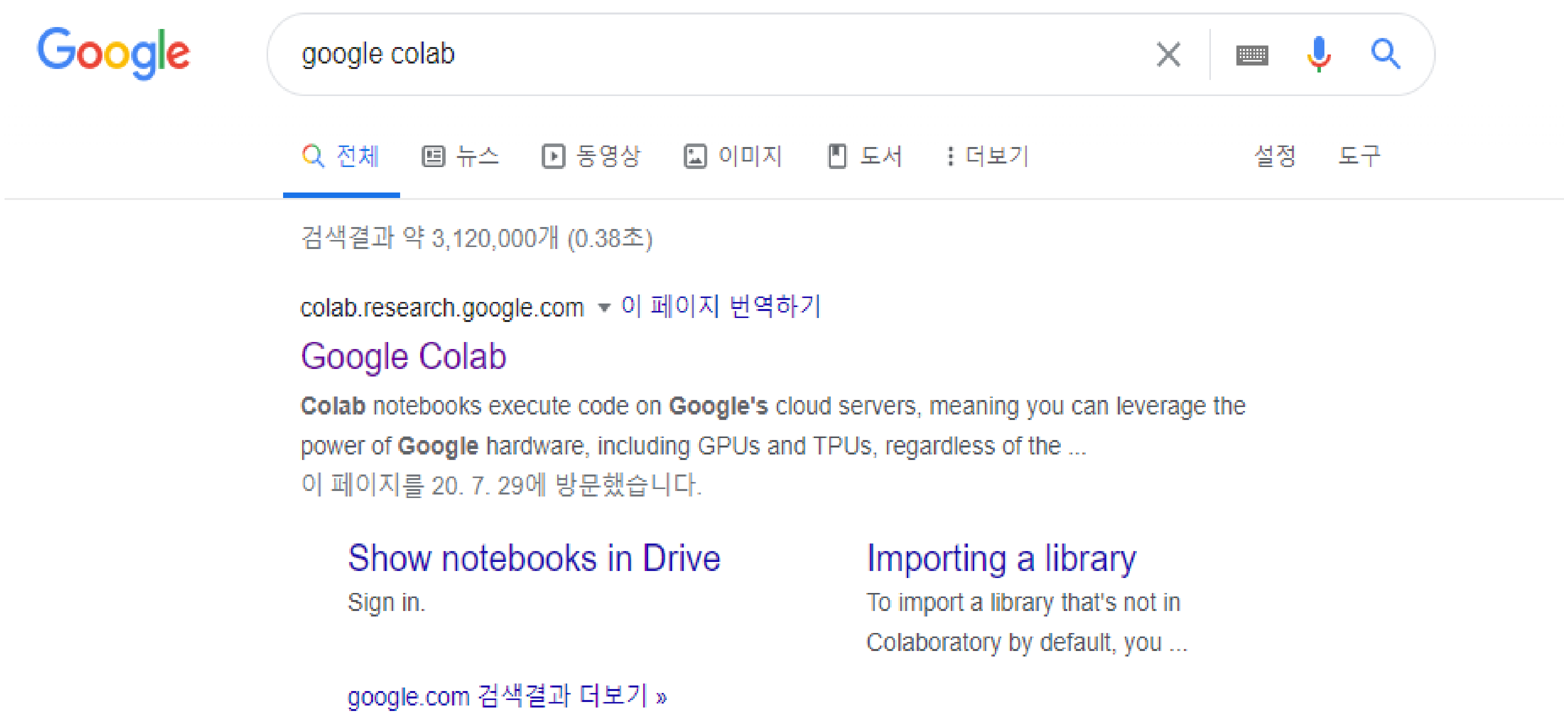
4. 내 드라이브에서 공유된 폴더 확인



Prerequisite - Colab

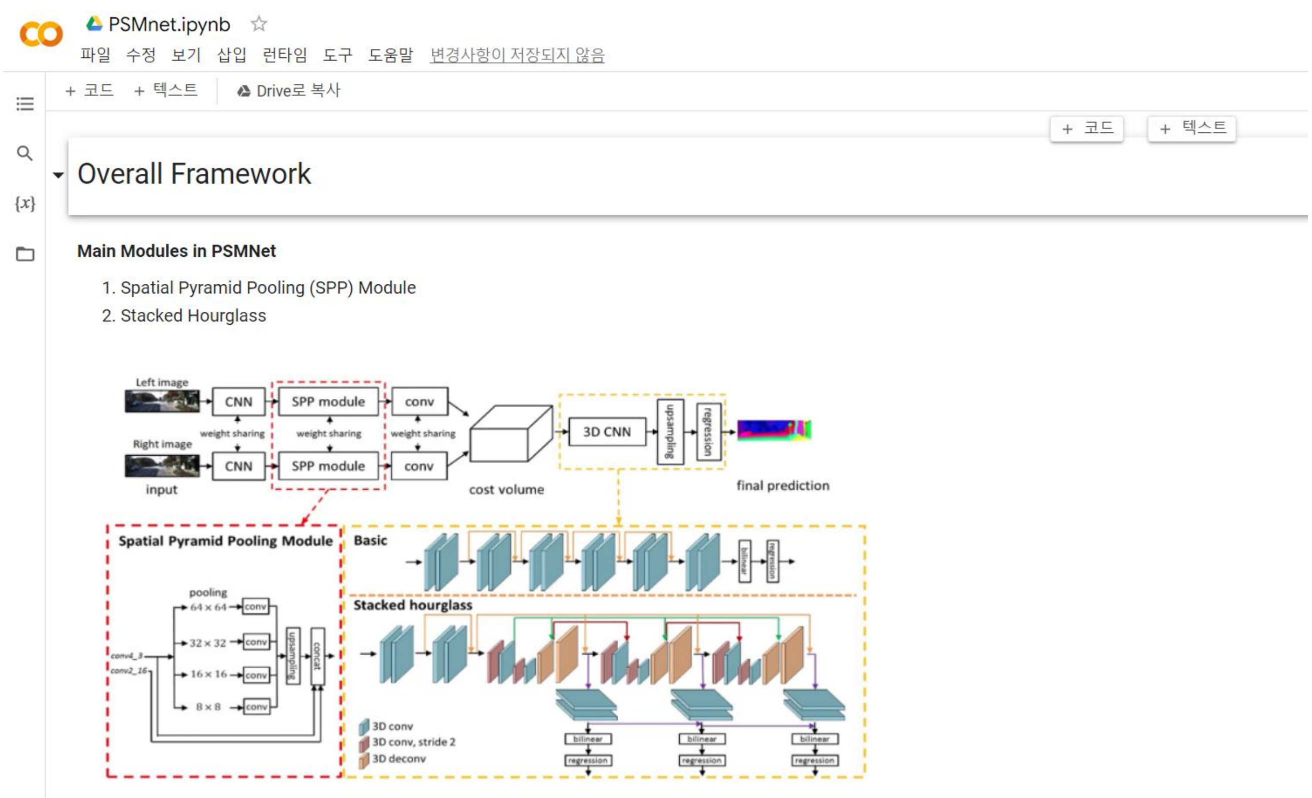
1. Colab으로 들어가기

https://colab.research.google.com/notebooks/intro.ipynb
또는 구글 코랩 검색해서 사이트 들어가기



Prerequisite - Colab

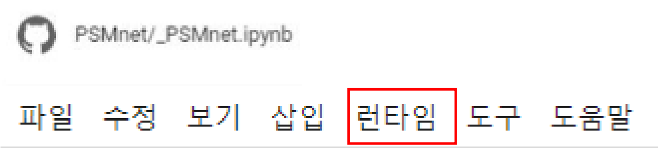
2. 아래 링크를 통해 Colab 접속 및 또는 ipynb 파일 다운로드
https://colab.research.google.com/drive/1y-f9_ee5APy5rPKhwZorQq8Q_bothx24?usp=drive_link

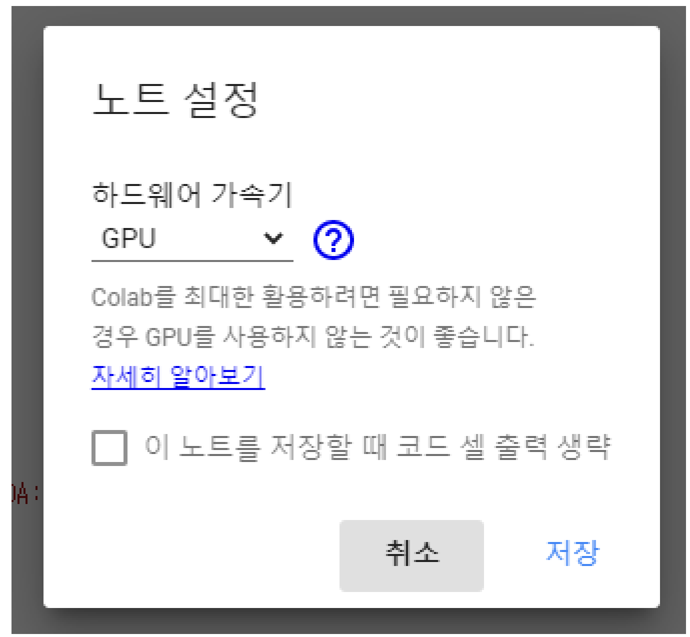


Prerequisite - Colab

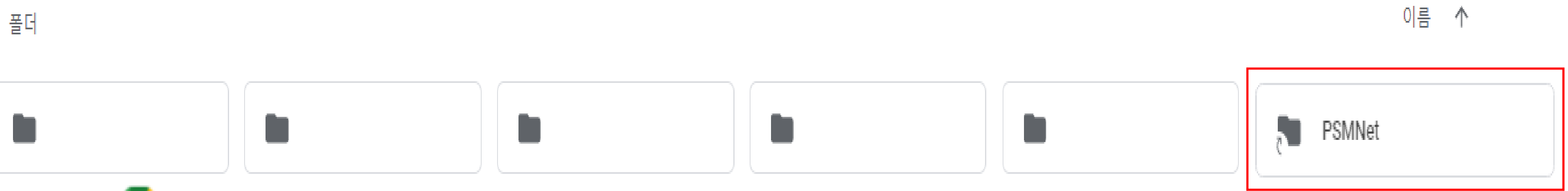
런타임 -> 런타임 유형 변경 -> 하드웨어 가속기(GPU)

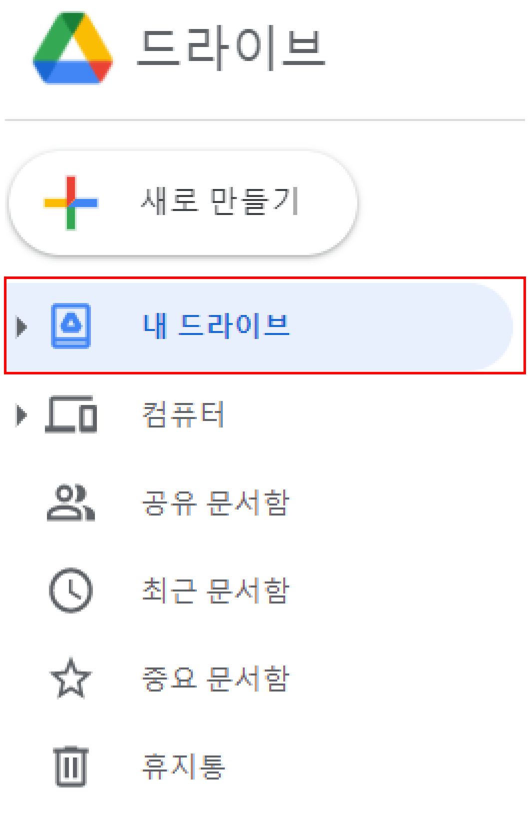
(eng) runtime -> change runtime type -> GPU





Prerequisite - Colab



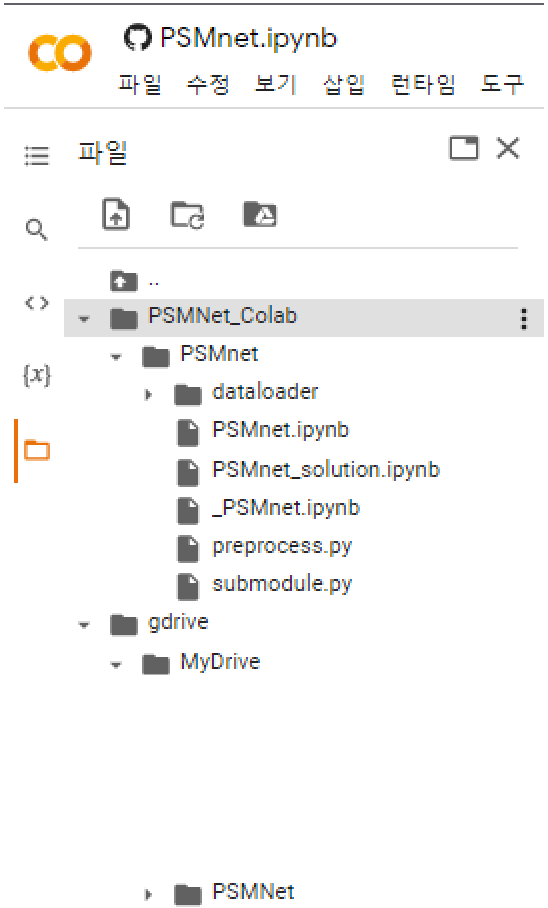


- 1) I will provide dataset, checkpoint download link and download the file.
- 2) Put the file in your google drive (at **Mydrive**, root directory)

Prerequisite - Colab

```
!git clone https://github.com/Chohoonhee/PSMNet_Colab
%cd /content/PSMNet_Colab/PSMnet/


from google.colab import drive

drive.mount('/content/gdrive/')
datapath = '/content/gdrive/MyDrive/PSMNet/data/KITTI_2015/training/'
savemodel = './saved_model'
```

```
Cloning into 'PSMNet_Colab'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 41 (delta 13), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (41/41), done.
/content/PSMNet_Colab/PSMnet
Mounted at /content/gdrive/
```

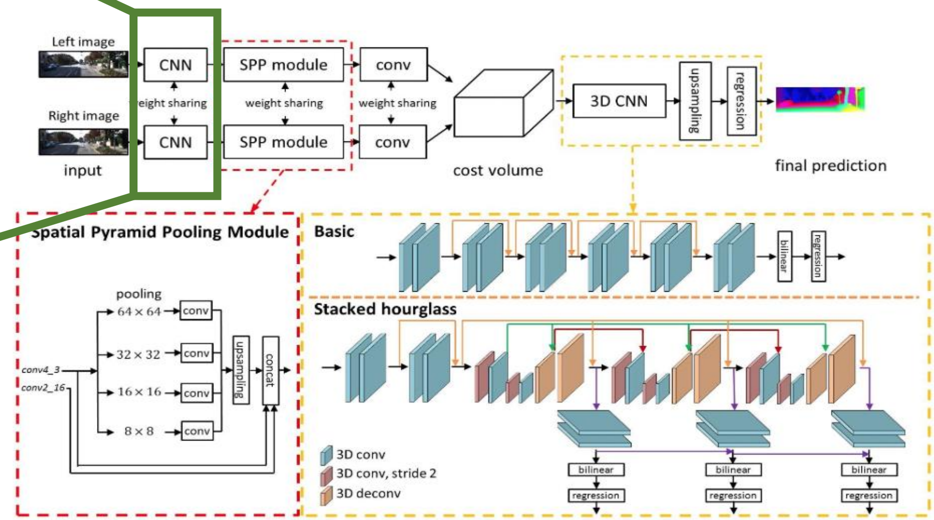


- Mount를 위해 colab에서 제일 상단의 코드를 실행
- 실행 후에, 좌측 상단의 파일 부분을 눌러, 정상적으로 mount 되었는지 확인
- 정상적으로 mount 되면 우측의 이미지처럼 되어야 함

Pyramid stereo Matching Network.

Network layers – feature extraction

Name	Layer setting	Output dimension
input		$H \times W \times 3$
CNN		
conv0_1	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_2	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_3	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv1_x	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 16$	$\frac{1}{4}H \times \frac{1}{4}W \times 64$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila = 2	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
conv4_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila= 4	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
SPP module		
branch_1	64×64 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_2	32×32 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_3	16×16 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_4	8×8 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
concat[conv2_16, conv4_3, branch_1, branch_2, branch_3, branch_4]		$\frac{1}{4}H \times \frac{1}{4}W \times 320$
fusion	$3 \times 3, 128$ $1 \times 1, 32$	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
Cost volume		
Concat left and shifted right		$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 64$



Shape:

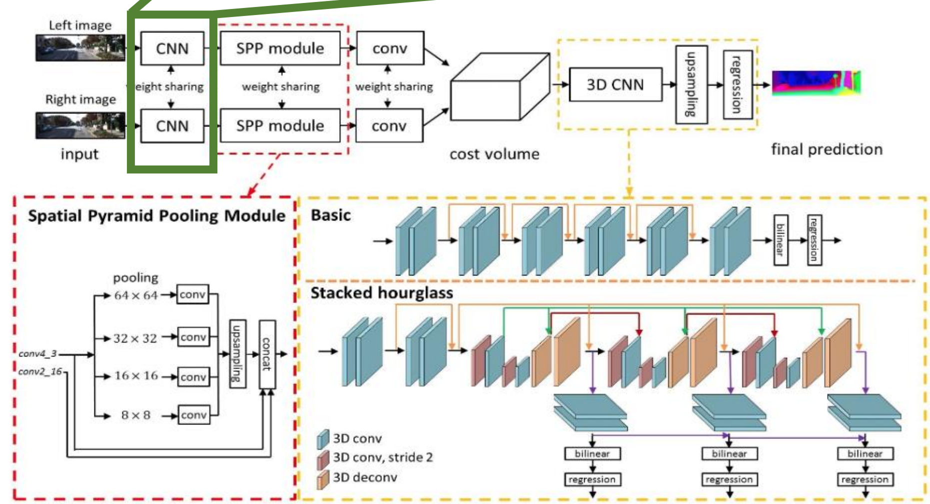
- Input: $(N, C_{in}, H_{in}, W_{in})$
- Output: $(N, C_{out}, H_{out}, W_{out})$ where

$$H_{out} = \left\lfloor \frac{H_{in} + 2 \times \text{padding}[0] - \text{dilation}[0] \times (\text{kernel\_size}[0] - 1) - 1}{\text{stride}[0]} + 1 \right\rfloor$$

$$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel\_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$$

Name	Layer setting	Output dimension
input		$H \times W \times 3$
CNN		
conv0_1	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_2	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_3	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv1_x	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 16$	$\frac{1}{4}H \times \frac{1}{4}W \times 64$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila = 2	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
conv4_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila= 4	$\frac{1}{4}H \times \frac{1}{4}W \times 128$

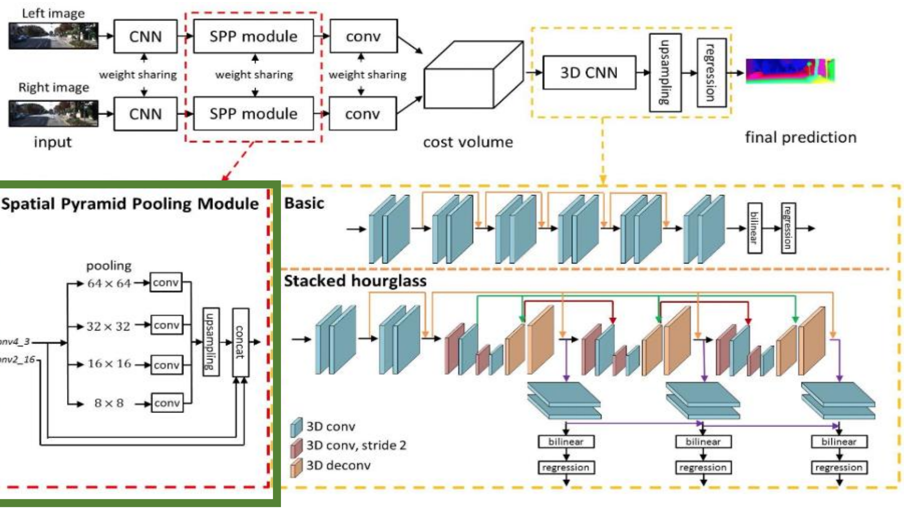
```python
class BasicBlock(nn.Module):
    expansion = 1
    def __init__(self, inplanes, planes, stride, downsample, pad, dilation):
        super(BasicBlock, self).__init__()
        self.conv1 = nn.Sequential(convbn(inplanes, planes, 3, stride, pad, dilation),
                                   nn.ReLU(inplace=True))
        self.conv2 = convbn(planes, planes, 3, 1, pad, dilation)
        self.downsample = downsample
        self.stride = stride

    def forward(self, x):
        out = self.conv1(x)
        out = self.conv2(out)
        if self.downsample is not None:
            x = self.downsample(x)
        out += x

        return out
```



Pyramid stereo Matching Network.

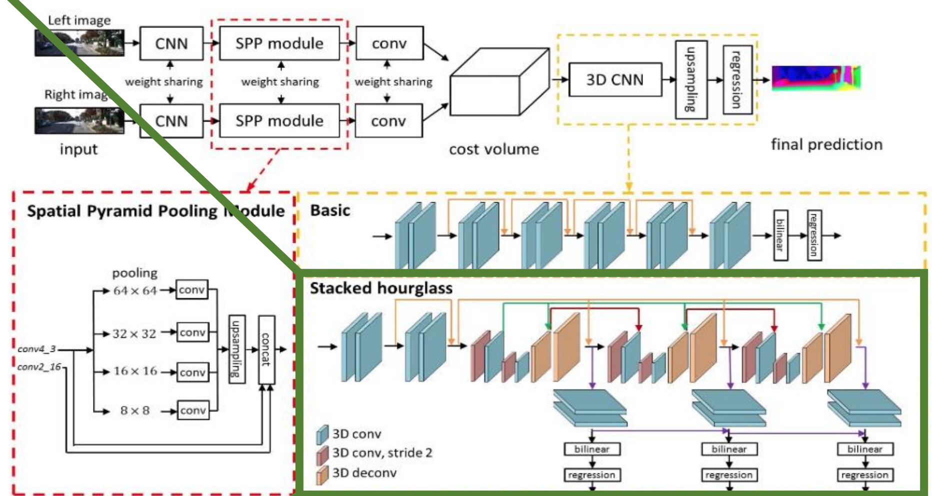
Network layers – SPP module

Name	Layer setting	Output dimension
input		$H \times W \times 3$
CNN		
conv0_1	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_2	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv0_3	$3 \times 3, 32$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv1_x	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$	$\frac{1}{2}H \times \frac{1}{2}W \times 32$
conv2_x	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 16$	$\frac{1}{4}H \times \frac{1}{4}W \times 64$
conv3_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila = 2	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
conv4_x	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, dila= 4	$\frac{1}{4}H \times \frac{1}{4}W \times 128$
SPP module		
branch_1	64×64 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_2	32×32 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_3	16×16 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_4	8×8 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
concat[conv2_16, conv4_3, branch_1, branch_2, branch_3, branch_4]		$\frac{1}{4}H \times \frac{1}{4}W \times 320$
fusion	$3 \times 3, 128$ $1 \times 1, 32$	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
Cost volume		
Concat left and shifted right		$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 64$



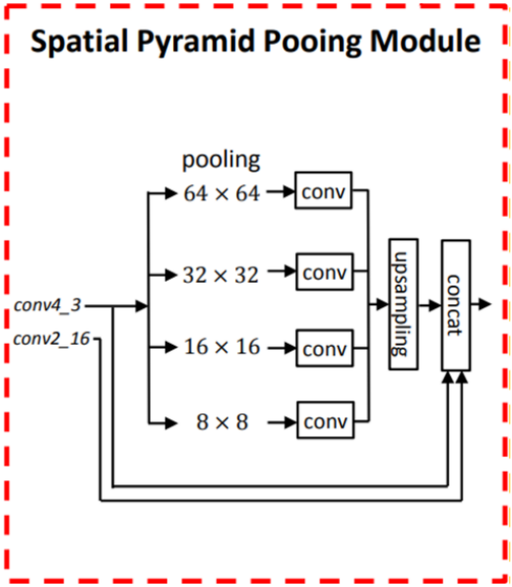
Network layers – 3D CNN

3D CNN (stacked hourglass)		
3Dconv0	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 32$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dconv1	$\begin{bmatrix} 3 \times 3 \times 3, 32 \\ 3 \times 3 \times 3, 32 \end{bmatrix}$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dstack1_1	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack1_2	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 64$
3Dstack1_3	deconv $3 \times 3 \times 3, 64$ add **3Dstack1_1**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack1_4	deconv $3 \times 3 \times 3, 32$ add **3Dconv1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dstack2_1	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$ add **3Dstack1_3**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack2_2	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 64$
3Dstack2_3	deconv $3 \times 3 \times 3, 64$ add **3Dstack1_1**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack2_4	deconv $3 \times 3 \times 3, 32$ add **3Dconv1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
3Dstack3_1	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$ add **3Dstack2_3**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack3_2	$3 \times 3 \times 3, 64$ $3 \times 3 \times 3, 64$	$\frac{1}{16}D \times \frac{1}{16}H \times \frac{1}{16}W \times 64$
3Dstack3_3	deconv $3 \times 3 \times 3, 64$ add **3Dstack1_1**	$\frac{1}{8}D \times \frac{1}{8}H \times \frac{1}{8}W \times 64$
3Dstack3_4	deconv $3 \times 3 \times 3, 32$ add **3Dconv1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 32$
output_1	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 1$	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 1$
output_2	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 1$ add **output_1**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 1$
output_3	$3 \times 3 \times 3, 32$ $3 \times 3 \times 3, 1$ add **output_2**	$\frac{1}{4}D \times \frac{1}{4}H \times \frac{1}{4}W \times 1$



HW1 Implementation

- **Implement the SPP module-based feature extraction.**



SPP module		
branch_1	64×64 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_2	32×32 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_3	16×16 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
branch_4	8×8 avg. pool $3 \times 3, 32$ bilinear interpolation	$\frac{1}{4}H \times \frac{1}{4}W \times 32$
concat[conv2_16, conv4_3, branch_1, branch_2, branch_3, branch_4]		$\frac{1}{4}H \times \frac{1}{4}W \times 320$
fusion	$3 \times 3, 128$ $1 \times 1, 32$	$\frac{1}{4}H \times \frac{1}{4}W \times 32$

HW1 Implementation

- **Implement hourglass module**

Define hourglass module

Let`s define hourglass module using 3d convolution and batchnormalization3D

Please refer to convbn_3d function

```
[ ]  def convbn_3d(in_planes, out_planes, kernel_size, stride, pad):
         return nn.Sequential(nn.Conv3d(in_planes, out_planes, kernel_size=kernel_size, padding=pad, stride=stride,bias=False),
                              nn.BatchNorm3d(out_planes))
```

3D CNN (stacked hourglass)		
3Dconv0	$3\times3\times3,32$ $3\times3\times3,32$	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times32$
3Dconv1	$\begin{bmatrix}3\times3\times3,32\\3\times3\times3,32\end{bmatrix}$	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times32$
3Dstack1_1	$3\times3\times3,64$ $3\times3\times3,64$	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack1_2	$3\times3\times3,64$ $3\times3\times3,64$	$\frac{1}{16}D\times\frac{1}{16}H\times\frac{1}{16}W\times64$
3Dstack1_3	deconv $3\times3\times3,64$ add **3Dstack1_1**	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack1_4	deconv $3\times3\times3,32$ add **3Dconv1**	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times32$
3Dstack2_1	$3\times3\times3,64$ $3\times3\times3,64$ add **3Dstack1_3**	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack2_2	$3\times3\times3,64$ $3\times3\times3,64$	$\frac{1}{16}D\times\frac{1}{16}H\times\frac{1}{16}W\times64$
3Dstack2_3	deconv $3\times3\times3,64$ add **3Dstack1_1**	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack2_4	deconv $3\times3\times3,32$ add **3Dconv1**	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times32$
3Dstack3_1	$3\times3\times3,64$ $3\times3\times3,64$ add **3Dstack2_3**	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack3_2	$3\times3\times3,64$ $3\times3\times3,64$	$\frac{1}{16}D\times\frac{1}{16}H\times\frac{1}{16}W\times64$
3Dstack3_3	deconv $3\times3\times3,64$ add **3Dstack1_1**	$\frac{1}{8}D\times\frac{1}{8}H\times\frac{1}{8}W\times64$
3Dstack3_4	deconv $3\times3\times3,32$ add **3Dconv1**	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times32$
output_1	$3\times3\times3,32$ $3\times3\times3,1$	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times1$
output_2	$3\times3\times3,32$ $3\times3\times3,1$ add **output_1**	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times1$
output_3	$3\times3\times3,32$ $3\times3\times3,1$ add **output_2**	$\frac{1}{4}D\times\frac{1}{4}H\times\frac{1}{4}W\times1$

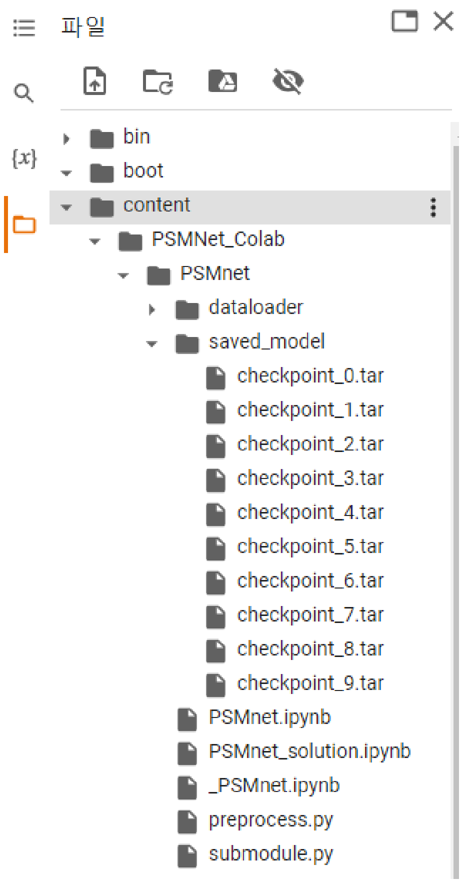
HW1 Implementation

- **Run the training code with evaluation**

```
torch.save({
    'epoch': epoch,
    'state_dict': model.state_dict(),
    'train_loss': total_train_loss/len(TrainImgLoader),
 }, savefilename)
#------------ TEST -----------------------------------------------------
total_test_loss = 0
for batch_idx, (imgL, imgR, disp_L) in enumerate(TestImgLoader):
    test_loss, disp = test(model, imgL,imgR, disp_L)
    print('Iter %d test loss = %.3f' %(batch_idx, test_loss))
    total_test_loss += test_loss
print('total test loss = %.3f' %(total_test_loss/len(TestImgLoader)))
```

HW1 Implementation

- **Make sure the checkpoint is saved.**



HW1 Requirements

- To KLMS, the report in pdf format (~2 page) and the ipynb code and checkpoint must be submitted together. Both Korean and English are available. The report must include the following information.

1. Tune the model network with several hyper-parameters to design so that test loss (end-point-error) < 13.0 . And write down what factors helped improve performance.

2. Describe the strengths and limitations of PSMNet.

- Score Criteria
  - Code implementation
  - Achieve the test loss (end-point-error)
  - Report Question 1
  - Report Question 2