KAIST Korea Advanced Institute of Science and Technology

LAB 2: PSMent

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

한국과학기술원

시각지능연구실

박사과정 김태우

prerequisite

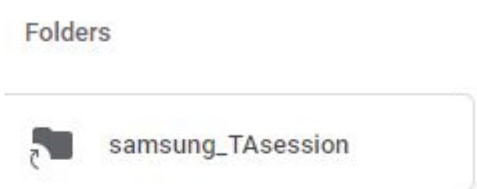
Linking with google drive for colab

1. Google 계정을 로그인 하세요

2. 다음 링크 클릭
https://drive.google.com/drive/folders/1ZB4kTn8fXbsshdX1u-Nzz6Xm5-4cb0rq?usp=sharing

3. 우측 상단의 Samsung_TAsession -> Add shortcut to drive -> My drive

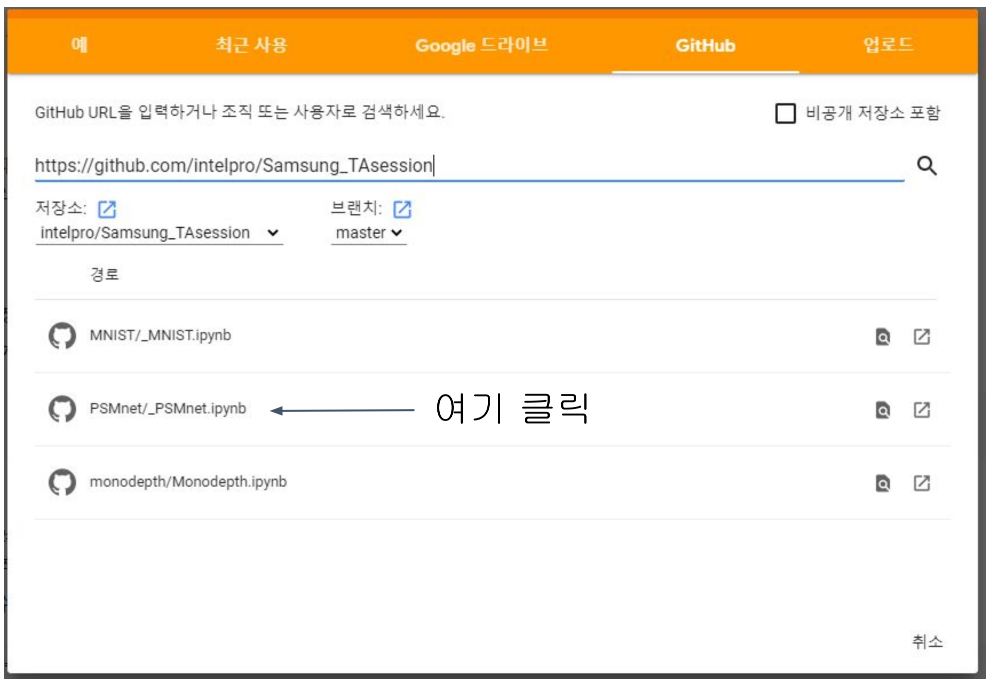
4. My Drive에서 공유된 폴더 확인



Github repository를 Colab 환경에서 열기

1. Colab으로 들어가기
https://colab.research.google.com/notebooks/intro.ipynb

2. 파일 -> 노트 열기 -> Github 탭 클릭 -> 아래 링크 복사 붙여넣기 -> _PSMnet.ipynb 열기
https://github.com/intelpro/Samsung_TAsession



Github repository를 Colab 환경에서 열기

3. 런타임 변경
런타임 탭 -> 런타임 유형변경 -> 하드웨어 가속기: GPU

4. 맨 위의 google drive 공유 폴더를 colab 환경에 docking 시키는 코드 실행

google drive의 공유 폴더를 colab docker 환경으로 drive를 mount 시키는 코드

만약, mount 된 드라이브가 /contet/gdrive/My Drive/samsung_TAsession과 다를경우

해당경로를 확인하여 datapath에 붙여넣을것.

```
[ ] %cd /content/
    !git clone https://github.com/intelpro/Samsung_TAsession/
    %cd /contet/Samsung_TAsession/PSMnet/
    !ls
    !mkdir saved_model
    from google.colab import drive
    drive.mount('/content/gdrive/')
    datapath = '/content/gdrive/My Drive/samsung_TAsession/KITTI_2015/training/'
    savemodel = './saved_model'
```

KAIST

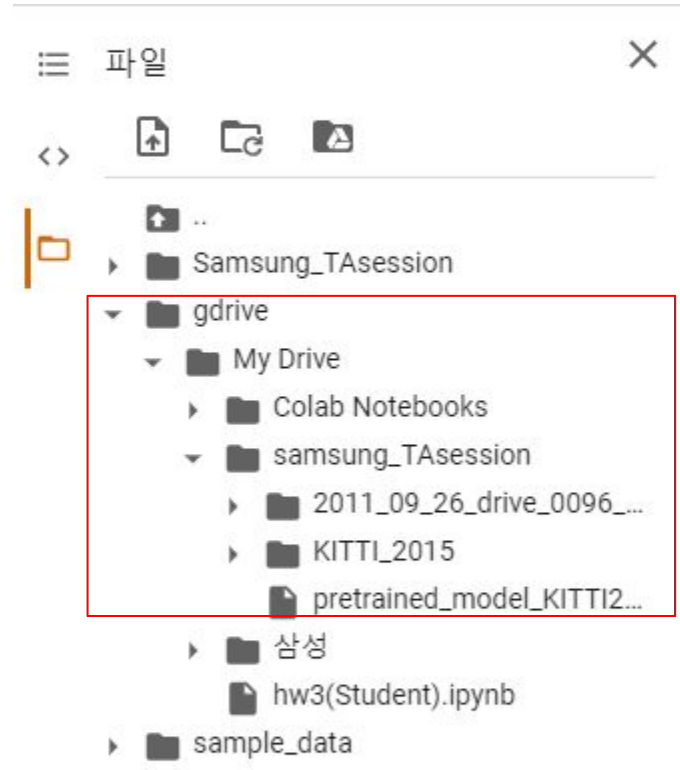
Google drive mount 확인

5. 정상적으로 google drive가 colab환경에서 mount 됨을 확인

```
/content
Cloning into 'Samsung_TAsession'...
remote: Enumerating objects: 167, done.
remote: Counting objects: 100% (167/167), done.
remote: Compressing objects: 100% (131/131), done.
remote: Total 167 (delta 69), reused 107 (delta 29), pack-reused 0
Receiving objects: 100% (167/167), 6.08 MiB | 4.77 MiB/s, done.
Resolving deltas: 100% (69/69), done.
/content/Samsung_TAsession/PSMnet
dataloader  preprocess.py  _PSMnet.ipynb  __pycache__  submodule.py
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_

Enter your authorization code:

Mounted at /content/gdrive/
```

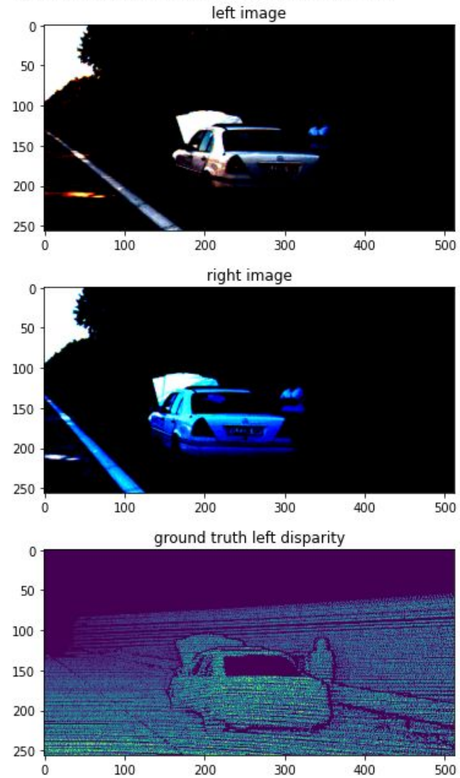


< 결과 이미지 >

Dataloader 정상 동작 확인

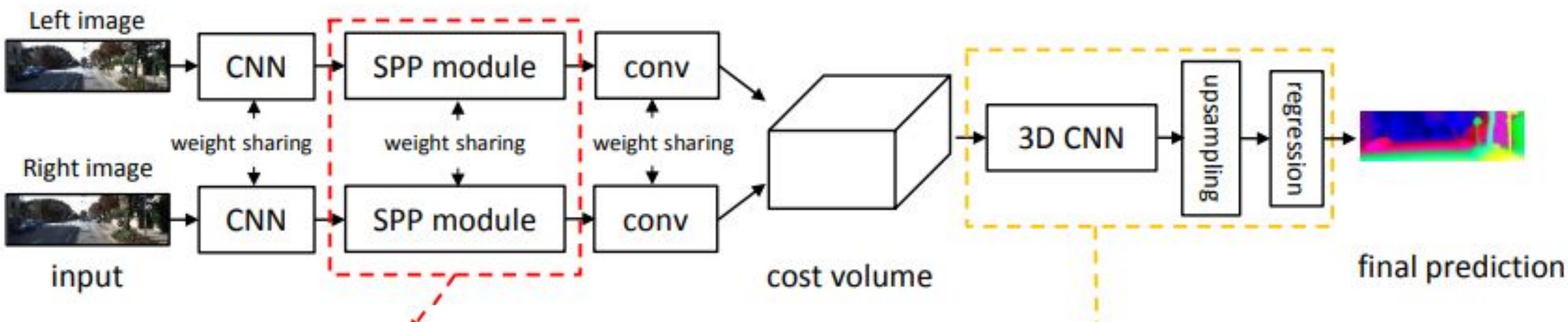
6. import로 모듈 가져오기 -> Get dataset string
-> Define dataloader -> Check KITTI dataset data 순차적으로 실행시킨다.

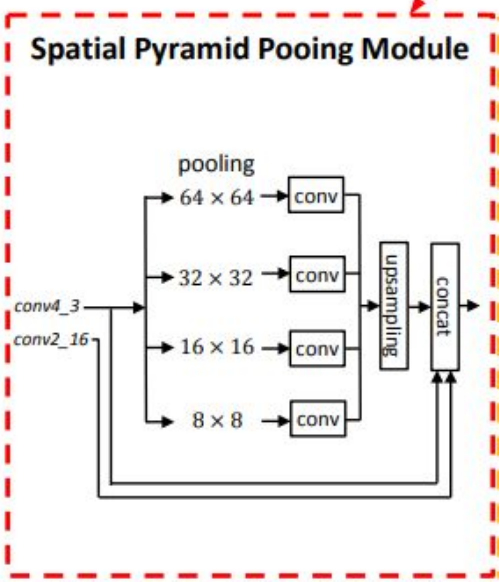
```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
<matplotlib.image.AxesImage at 0x7fca0bf63320>
```



Assignment

Spatial pyramid pooling module





Note

- Average pooling window size를 64, 32, 16, 8 의 여러 레벨로 pooling을 하면서 CNN의 receptive field를 늘리기 위한 방법
- Stereo matching 뿐만이 아니라 object detection, semantic segmentation, image classification 등 여러 분야에 광범위하게 사용됨.

Assignment

Spatial pyramid pooling module - Feature extraction 정의하기

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	$3 \times 3, 128$ $1 \times 1, 32$	$\frac{1}{4}H \times \frac{1}{4}W \times 32$

```python
## Your implementation Here
self.branch1 = nn.Sequential(nn.AvgPool2d((None, None), stride=(None,None)),
                             convbn(None, None, 1, 1, 0, 1),
                             nn.ReLU(inplace=True))

self.branch2 = nn.Sequential(nn.AvgPool2d((None, None), stride=(None,None)),
                             convbn(None, None, 1, 1, 0, 1),
                             nn.ReLU(inplace=True))

self.branch3 = nn.Sequential(nn.AvgPool2d((None, None), stride=(None,None)),
                             convbn(None, None, 1, 1, 0, 1),
                             nn.ReLU(inplace=True))

self.branch4 = nn.Sequential(nn.AvgPool2d((None, None), stride=(None,None)),
                             convbn(None, None, 1, 1, 0, 1),
                             nn.ReLU(inplace=True))
## Your implementation end
```

Assignment

- None으로 되어 있는 부분을 채워 넣을 것
- Pooling level은 64, 32, 16, 8
- pooling window size에 맞게 stride도 잘 채워 넣어야 함.
- conv4_x에서 뽑은 feature map이 각 branch를 지나고나면 채널이 128에서 32채널로 바뀜에 유의
- covbn의 정의를 알고싶으면 def convbn(...) 이라고 된 부분을 참고

Assignment

만약 Spatial pooling module을 제대로 구현하였다면..

Training with KITTI dataset 실행시..

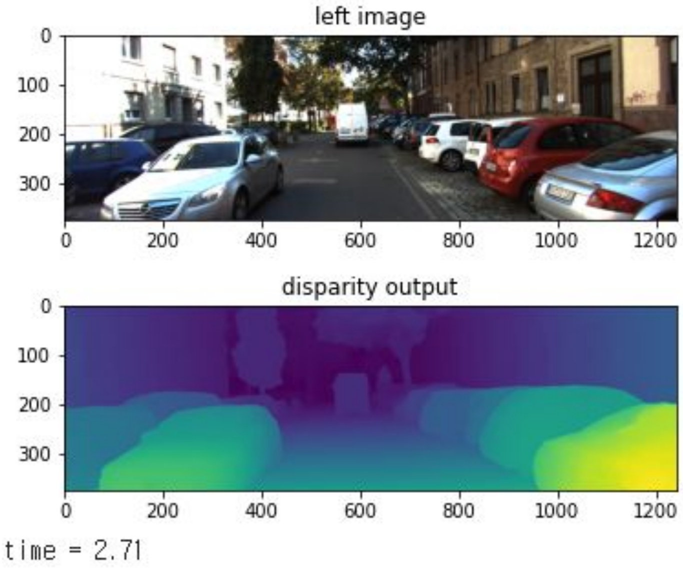
```
••• This is 0-th epoch
    /usr/local/lib/python3.6/dist-packages/torch/nn/functional.py:2796:
      warnings.warn("nn.functional.upsample is deprecated. Use nn.functi
    /usr/local/lib/python3.6/dist-packages/torch/nn/functional.py:2973:
      "See the documentation of nn.Upsample for details.".format(mode))
    /usr/local/lib/python3.6/dist-packages/torch/nn/functional.py:2973:
      "See the documentation of nn.Upsample for details.".format(mode))
    /usr/local/lib/python3.6/dist-packages/torch/nn/_reduction.py:43: Us
      warnings.warn(warning.format(ret))
    Iter 0 training loss = 76.164 , time = 2.01
    Iter 3 training loss = 9.776 , time = 1.76
    Iter 6 training loss = 7.997 , time = 1.77
    Iter 9 training loss = 13.371 , time = 1.78
    Iter 12 training loss = 11.346 , time = 1.77
    Iter 15 training loss = 10.733 , time = 1.75
    Iter 18 training loss = 7.748 , time = 1.75
    Iter 21 training loss = 7.496 , time = 1.74
    Iter 24 training loss = 6.336 , time = 1.73
    Iter 27 training loss = 6.175 , time = 1.74
    Iter 30 training loss = 7.019 , time = 1.74
    Iter 33 training loss = 5.445 , time = 1.74
    Iter 36 training loss = 7.691 , time = 1.74
    Iter 39 training loss = 3.301 , time = 1.73
    Iter 42 training loss = 9.453 , time = 1.72
    Iter 45 training loss = 6.158 , time = 1.72
    Iter 48 training loss = 5.845 , time = 1.74
    Iter 51 training loss = 6.859 , time = 1.74
```

정상적으로 loss가
줄어듬을 확인 할 수 있음.

Assignment

만약 Spatial pooling module을 제대로 구현하였다면..

Test with KITTI test sample 실행시..



정상적으로 disparity output이 나옴을 확인할 수 있음

For your information

nn.AvgPool2d

https://pytorch.org/docs/master/generated/torch.nn.AvgPool2d.html

AVGPOOL2D

CLASS `torch.nn.AvgPool2d(kernel_size: Union[T, Tuple[T, T]], stride: Optional[Union[T, Tuple[T, T]]] = None, padding: Union[T, Tuple[T, T]] = 0, ceil_mode: bool = False, count_include_pad: bool = True, divisor_override: bool = None)` [SOURCE]

Parameters

- **kernel_size** – the size of the window
- **stride** – the stride of the window. Default value is `kernel_size`
- **padding** – implicit zero padding to be added on both sides
- **ceil_mode** – when True, will use *ceil* instead of *floor* to compute the output shape
- **count_include_pad** – when True, will include the zero-padding in the averaging calculation
- **divisor_override** – if specified, it will be used as divisor, otherwise `kernel_size` will be used