组会

2020.01.14 周至公

Mask-Guided Attention Network for Occluded Pedestrian Detection

Yanwei Pang[1], Jin Xie[1], Muhammad Haris Khan[2], Rao Muhammad Anwer[2],
Fahad Shahbaz Khan[2,3], Ling Shao[2]
[1]Tianjin University [2]Inception Institute of Artificial Intelligence, UAE [3]CVL, Linköping University, Sweden
{pyw,jinxie}@tju.edu.cn, {muhammad.haris,rao.anwer,fahad.khan,ling.shao}@inceptioniai.org

15 Oct 2019

Mask-Guided Attention Network

- 解决行人检测在遮挡情况recall低的问题
- 思路：挖掘label信息，引入额外监督信息
- 方法：adapt visible-region bounding box annotation as an approximate alternative

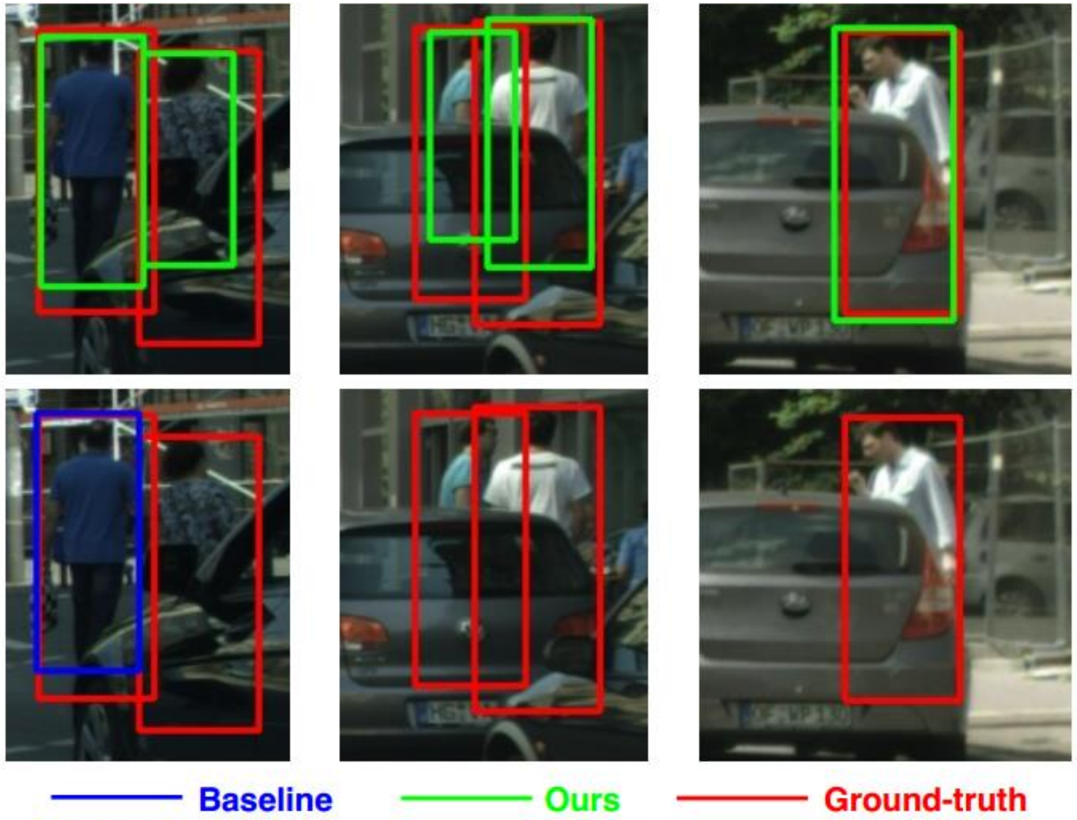


Figure 1. Detection examples using our approach (top row) and the baseline Faster R-CNN [24] (bottom row). For improved visualization, detection regions are cropped from images of CityPersons val. set [31]. All results are obtained using the same false positive per image (FPPI) criterion. Our approach robustly handles occlusions, yielding higher recall for occluded pedestrians.

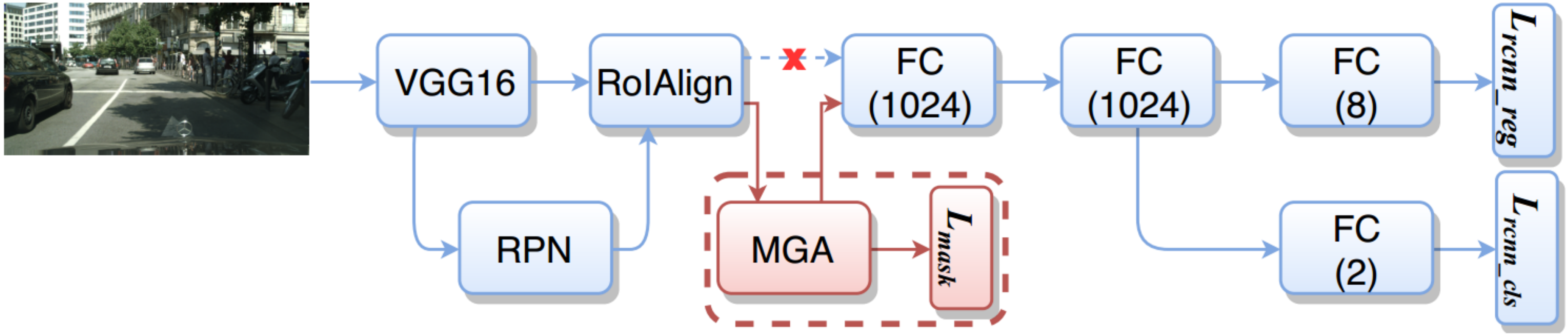


Figure 2. The overall network architecture of our Mask-Guided Attention Network (MGAN). It comprises a standard pedestrian detection (SPD) branch, whose components are shown in blue. It introduces a novel Mask-Guided Attention (MGA) module enclosed in red dashed box. Note, after RoI Align there is a classification stage in the SPD branch whose first layer is shown by FC (1024). In our architecture, standard full body features in SPD branch after RoI Align layer are modulated by MGA branch before getting scored by the classification stage. This is in contrast to baseline SPD where these features directly become the input to the classification stage without any modulation.

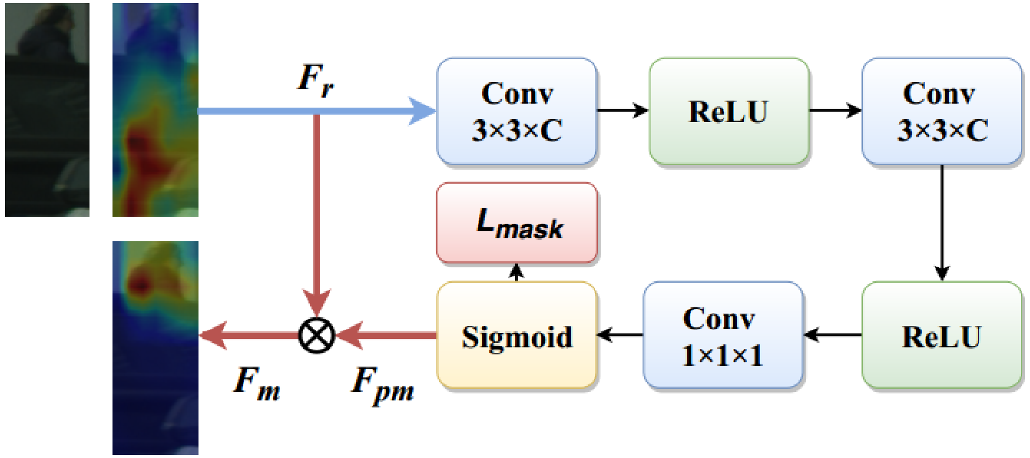


Figure 5. The network architecture of our Mask-Guided Attention (MGA) Branch. It takes RoI features and generates modulated features using a small stack of conv. operations, followed by ReLU nonlinearities.

$$L = L_0 + \alpha L_{mask} + \beta L_{occ},$$

$$L_0 = L_{rpn\_cls} + L_{rpn\_reg} + L_{rcnn\_cls} + L_{rcnn\_reg},$$

$$L_{mask} = BCELoss(p_n(x,y), \hat{p}_n(x,y)),$$

$$L_{occ} = \frac{1}{N}\sum_{n=1}^{N}\{[1 - \frac{1}{WH}\sum_{x}^{W}\sum_{y}^{H} p_n(x,y)]$$
$$CELoss(p_n^{rcnn\_cls}, \hat{p}_n^{rcnn\_cls})\}$$

make the classification loss aware of variable occlusion levels
与遮挡程度正相关

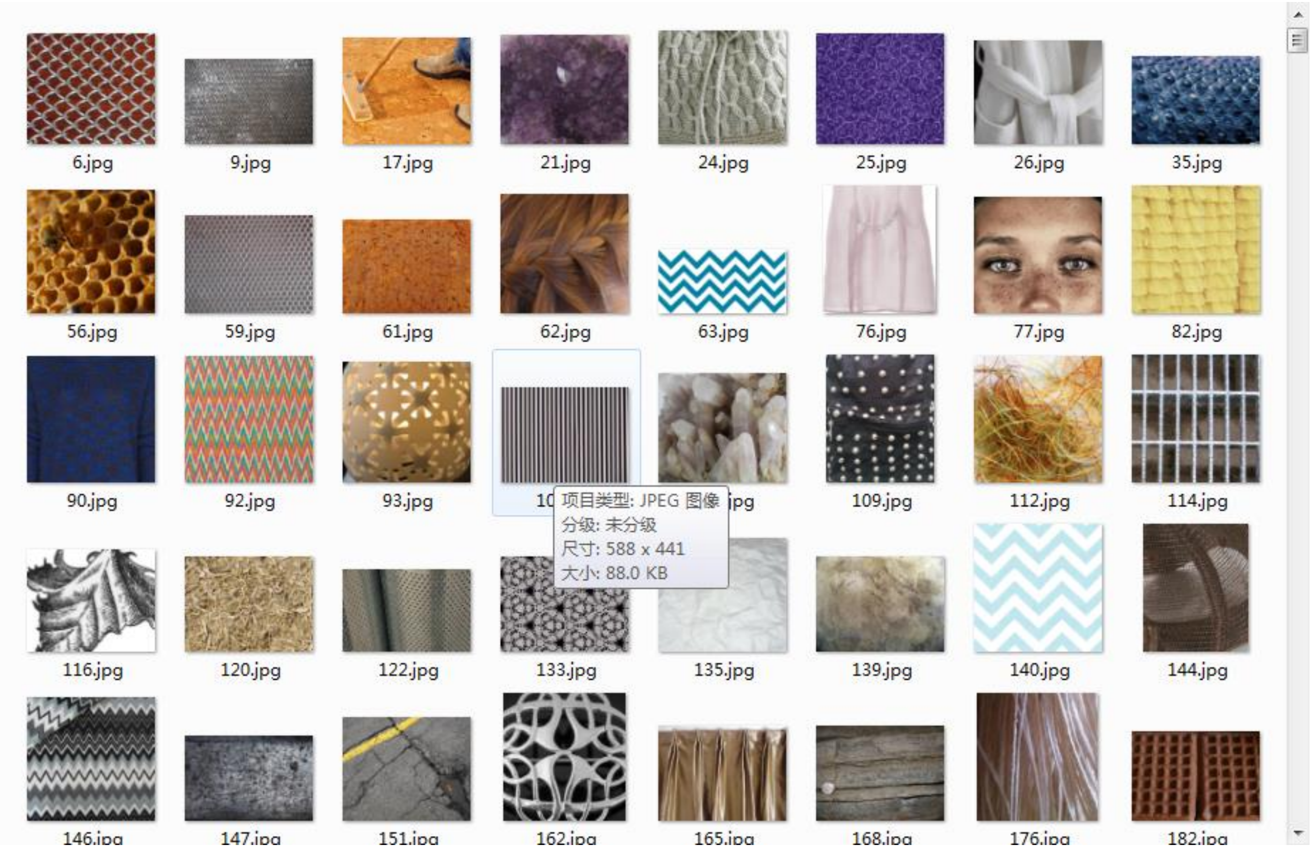
Method	**R**	**HO**
Baseline SPD (L_0 loss in Eq.(1))	13.8	57.0
Our MGAN (L_0 + L_{mask})	11.9	52.7
Our MGAN (L_0 + L_{occ})	13.2	55.6
Our Final MGAN (L_0 + L_{mask} + L_{occ})	**11.5**	**51.7**

Set	[50, 75]	[75, 125]	>125
Baseline SPD	66.3	59.7	43.1
Our MGAN	**61.7**	**52.3**	**37.6**

Table 3. Comparison (in log-average miss rates) by dividing pedestrians w.r.t. their height (pixels): small [50-75], medium [75-125] and large (>125) representing 28%, 37% and 35%, respectively of CityPersons **HO** set. Best results are boldfaced in each case.

Flyai

- 对图片的纹理进行分类
- 该数据集由五千多张图片组成，一共有47个纹理类别



提供者：Mr.Fire
通过在本地训练模型并对提交文件的实时评测，最终完成评分为90.07。
Keras SENet 2019-12-27 19:18:53
90.07

提供者：语言沉默
通过在本地训练模型并对提交文件的实时评测，最终完成评分为88.83。
PyTorch ResNet 2019-12-17 20:35:14
88.83

提供者：Jil
通过在本地训练模型并对提交文件的实时评测，最终完成评分为88.48。
PyTorch SENet 2019-12-19 01:30:26
88.48

4
提供者：FastAI可解释性差
batch数据为1，循环次数为1次，损失函数优化完，最终完成评分为81.47。
PyTorch SENet 2019-12-22 02:55:34
81.47

5
提供者：gboy
batch数据为16，循环次数为15次，损失函数优化完，最终完成评分为81.29。
PyTorch SENet 2019-12-24 18:39:24
81.29

6
提供者：Mr.sheep
通过在本地训练模型并对提交文件的实时评测，最终完成评分为80.5。
2020-01-06 00:59:31
80.50

7
提供者：梦幻济公
通过在本地训练模型并对提交文件的实时评测，最终完成评分为80.41。
2020-01-05 23:56:54
80.41

```
/resnet101/2019-12-23T15:57:15.420084/
no data argument
EPOCH = 100
MILESTONES = [30, 60, 80]
INIT_LR = 0.05
loss = CE
Training Epoch: 84: Average loss: 0.0000, Top 1 Accuracy: 1.0000, Top 5 Accuracy: 1.0000, LR: 0.000400
Test set: Average loss: 0.0730, Top 1 Accuracy: 0.5718, Top 5 Accuracy: 0.8378

/resnet101/2019-12-23T10:40:34.765905/
EPOCH = 100
MILESTONES = [30, 60, 80]
INIT_LR = 0.05
loss = CE
Training Epoch: 99: Average loss: 0.0015, Top 1 Accuracy: 0.9867, Top 5 Accuracy: 0.9997, LR: 0.000400
Test set: Average loss: 0.0755, Top 1 Accuracy: 0.6356, Top 5 Accuracy: 0.8528

/resnet101/2019-12-23T10:40:34.765905/
EPOCH = 100
MILESTONES = [30, 60, 80]
INIT_LR = 0.05
loss = Focalloss(gamma=2,alpha=None)
Training Epoch: 64: Average loss: 0.0013, Top 1 Accuracy: 0.9731, Top 5 Accuracy: 0.9994, LR: 0.002000
Test set: Average loss: 0.0418, Top 1 Accuracy: 0.6445, Top 5 Accuracy: 0.8759
```

```
/resnet101/2019-12-23T12:55:33.834223/
EPOCH = 100
MILESTONES = [30, 60, 80]
INIT_LR = 0.01
loss = CE
Training Epoch: 87: Average loss: 0.0005, Top 1 Accuracy: 0.9967, Top 5 Accuracy: 1.0000, LR: 0.000080
Test set: Average loss: 0.0617, Top 1 Accuracy: 0.6986, Top 5 Accuracy: 0.8945

/resnet101/2019-12-23T13:50:20.492098/
EPOCH = 100
MILESTONES = [30, 60, 80]
INIT_LR = 0.01
loss = Focalloss(gamma=2,alpha=None)
Training Epoch: 67: Average loss: 0.0005, Top 1 Accuracy: 0.9897, Top 5 Accuracy: 0.9994, LR: 0.000400
Test set: Average loss: 0.0325, Top 1 Accuracy: 0.7261, Top 5 Accuracy: 0.9113
```

```
2019-12-23T23:50:09.548667/
EPOCH = 100
MILESTONES = [30, 60, 80]
INIT_LR = 0.001
loss = Focalloss(gamma=2,alpha=None)
Training Epoch: 83: Average loss: 0.0003, Top 1 Accuracy: 0.9988, Top 5 Accuracy: 1.0000, LR: 0.000008
Test set: Average loss: 0.0209, Top 1 Accuracy: 0.7633, Top 5 Accuracy: 0.9424
```

Flyai

```
ricap
loss = ricap_criterion
Training Epoch: 63: Average loss: 0.0276, Top 1 Accuracy: 0.0544, Top 5 Accuracy: 0.1350, LR: 0.000040     0.7323
Test set: Average loss: 0.0316, Top 1 Accuracy: 0.7748, Top 5 Accuracy: 0.9282
```

- resnet152

```
MILESTONES = [30, 60, 80]
INIT_LR = 0.001
loss = Focalloss(gamma=2,alpha=None)
Epoch 83:    70,80 74.65
Training: Average loss: 0.0003, Top 1 Accuracy: 0.9979, Top 5 Accuracy: 1.0000, LR: 0.000008    0.7429
Test set: Average loss: 0.0210, Top 1 Accuracy: 0.7615, Top 5 Accuracy: 0.9406
```

Flyai

```
efficientnet-b3/2019-12-30T19:39:56.725418/
IMG_SIZE = 300
AutoAugment,RandomErasing
RAdam, Lookahead
ReduceLROnPlateau(optimizer, factor=0.5, patience=15, min_lr=1e-7)
INIT_LR = 0.001
ricap
loss = ricap_criterion
Epoch 40:
Training: Average loss: 0.0376, Top 1 Accuracy: 0.0464, Top 5 Accuracy: 0.1291, LR: 0.000125     76.42
val  set: Average loss: 0.0315, Top 1 Accuracy: 0.7535, Top 5 Accuracy: 0.9326
```

运行 代码

-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	3
0	25					1												4					3							1	1						
1		9																			1			1					1			1	1		2		
2			11									1									1																
3				14	1																								1			1					
4					15																1								1								
5						24																			1												
6							26															1														3	
7					1			15	1																				1				1				
8									17									1																			
9										29																											
10											18		1			1					1							5			4						
11		1										22								1					1												
12			1						1				10					1														2		1			
13														25																					1		
14		1			1										22										1												
15									1							15		1												1							
16					1												16		1							3											
17								1	1									15					1							1							
18																			24	1						3										1	
19			2														1			18																	
20			1										1								18			1													
21																				1		20						1									
22																1		4					14							1							
23		3						1																16								2					
24			1									5									1			1	21								2		1		
25							1										3		1							18											
26																				2		4					21										
27					2												1			1						1		20			3	1					
28					1																								13						1	1	
29																1		2										1		16							
30											1																	2			17						
31					1			2				1			1									1	1				5			17	1				
																													1			1	22				

```
[24, 11]
[16, 25, 18]
[10, 27, 30]
[32, 1, 34, 3, 36, 37, 4, 13, 23, 28, 31]
[41, 35]
[2, 8, 44, 45, 46, 15, 17, 20, 22, 29]
```

Labelsmooth 10,27,30

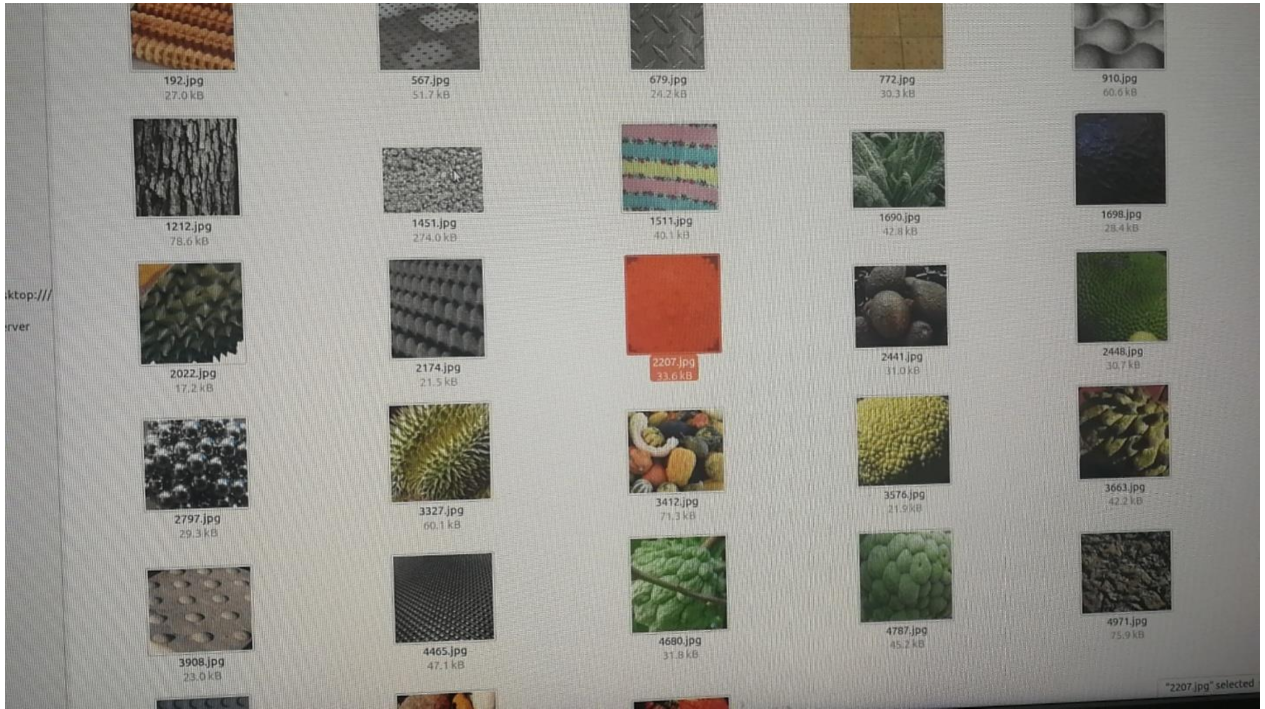
```
Epoch 54:
Training: Average loss: 0.0268, Top 1 Accuracy: 0.9952, Top 5 Accuracy: 1.0000, LR: 0.000125
val  set: Average loss: 0.0318, Top 1 Accuracy: 0.9178, Top 5 Accuracy: 1.0000
```

Labelsmooth 32,1,34,.....

```
class 47
Epoch 42:
Training: Average loss: 0.0238, Top 1 Accuracy: 0.9988, Top 3 Accuracy: 1.0000, LR: 0.000250
val  set: Average loss: 0.0383, Top 1 Accuracy: 0.7371, Top 3 Accuracy: 0.8964
```

Flyai

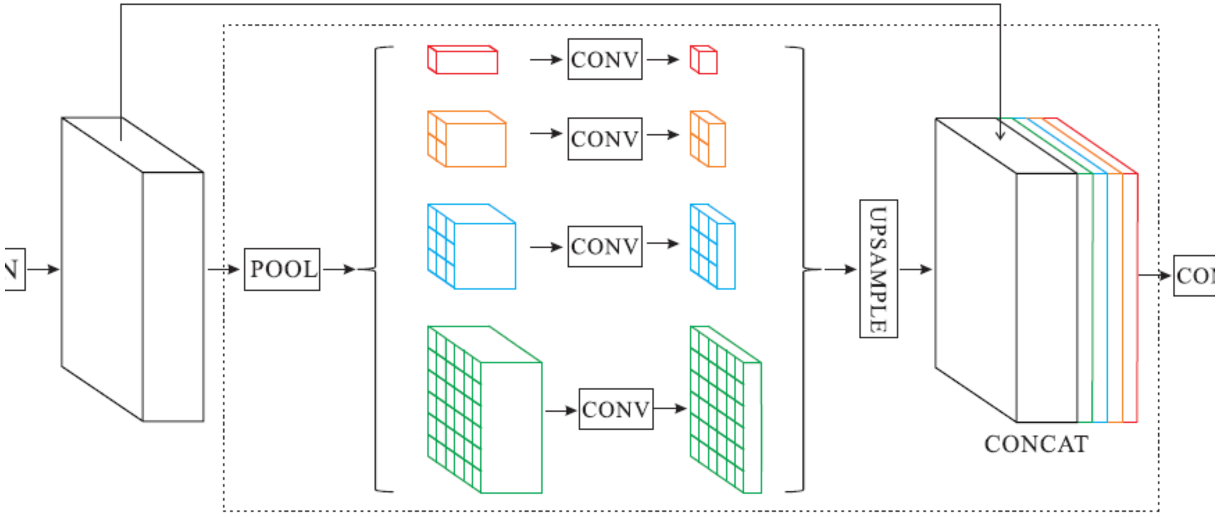

618.jpg
41.1 kB
963.jpg
29.9 kB
1522.jpg
171.7 kB
1692.jpg
45.4 kB
2420.jpg
43.2 kB
2520.jpg
254.7 kB
2807.jpg
38.4 kB
2990.jpg
96.6 kB
3169.jpg
56.2 kB
3604.jpg
170.0 kB
3688.jpg
93.5 kB
4188.jpg
348.2 kB
4768.jpg
158.1 kB
4907.jpg
16.0 kB
4923.jpg
101.0 kB
5202.jpg
320.8 kB


192.jpg
567.jpg
679.jpg
772.jpg
910.jpg
1212.jpg
1451.jpg
1511.jpg
1690.jpg
1698.jpg
2022.jpg
2174.jpg
2207.jpg
2441.jpg
2448.jpg
2797.jpg
3327.jpg
3412.jpg
3576.jpg
3663.jpg
3908.jpg
4465.jpg
4680.jpg
4787.jpg
4971.jpg

盲道分割

- 标注数据900+
- PSPnet50 训练　train/test 700/200
- 效果看起来非常好

Class_1 result: iou/accuracy **0.9383/0.9524**, name: blindroad.



(b) Feature Map　　(c) Pyramid Pooling Module

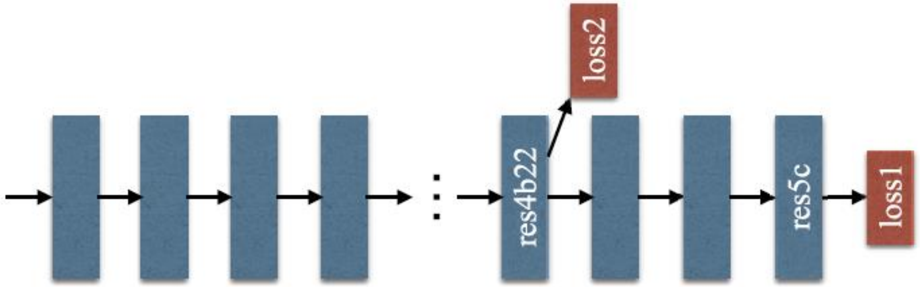


Figure 4. Illustration of auxiliary loss in ResNet101. Each blue box denotes a residue block. The auxiliary loss is added after the res4b22 residue block.

Top 20 Badcase on val   metric:iou

1577847044942	0.7979445373648355
1577232803897	0.8009551941177134
1578291863003	0.8208336882331608
1577847114286	0.8492020990798133
1577844270663	0.8848732919017531
1577429335613	0.939975460283409
1577848618954	0.9465578987388115
1577778671252	0.9551962480146075
1578294092068	0.957158835105404
1577846042819	0.9587025107211002
1577845232909	0.9647294304359669
1578294102241	0.9652424282344378
1577777929716	0.967580841516563
1577780728036	0.9679561765314393
1577844651070	0.9741004043688015
1578291971121	0.9741502090679909
1577778481516	0.9759523850362772
1577777714741	0.9766568418483876
1577846853138	0.9769348071997571
1578297367703	0.977773888444053