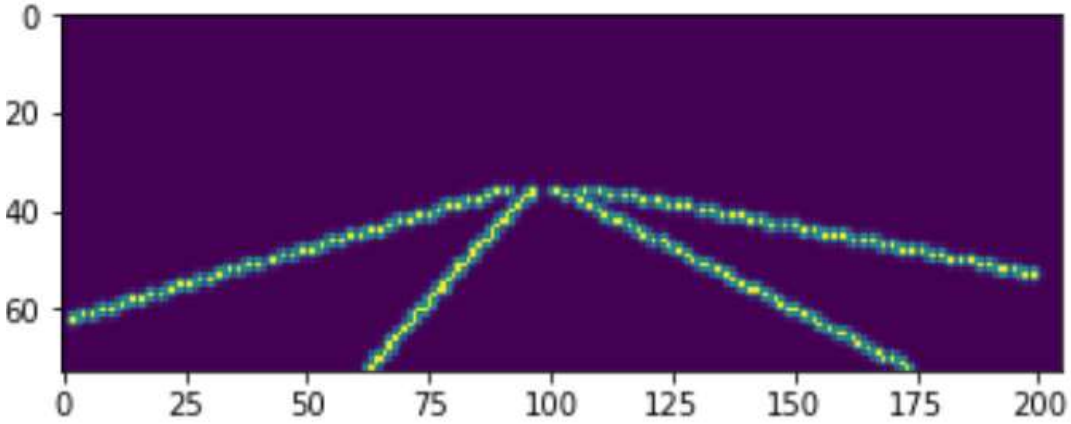
A Keypoint-based Global Association Network for Lane Detection

简称GANet

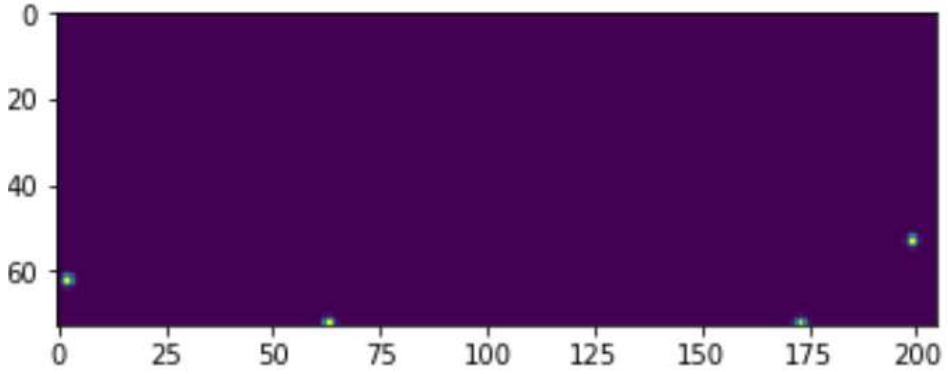
By zhangyunzhi

GANet训练标签

1、使用三次B样条回归出车道线
2、均匀取45个点，并以它们为中心做**高斯掩码（kernel=5）**

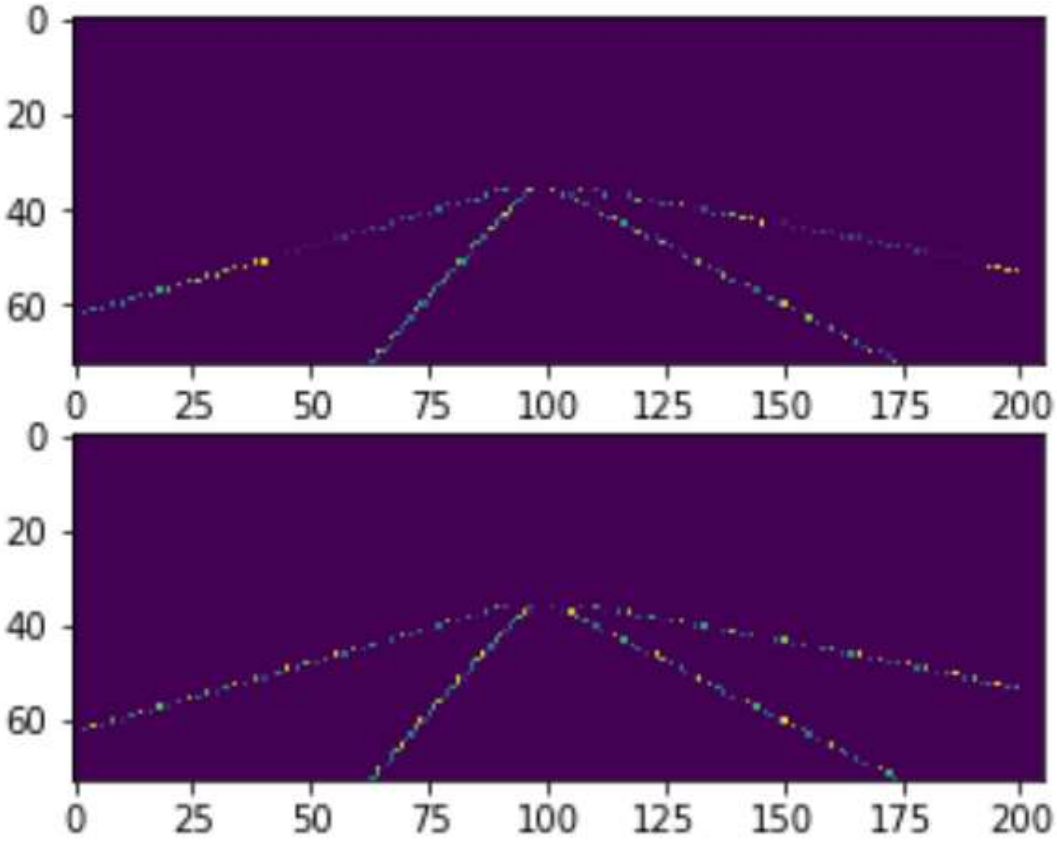


左图：关键点的heat map



右图：车道线起始点的heat map， 用于聚类

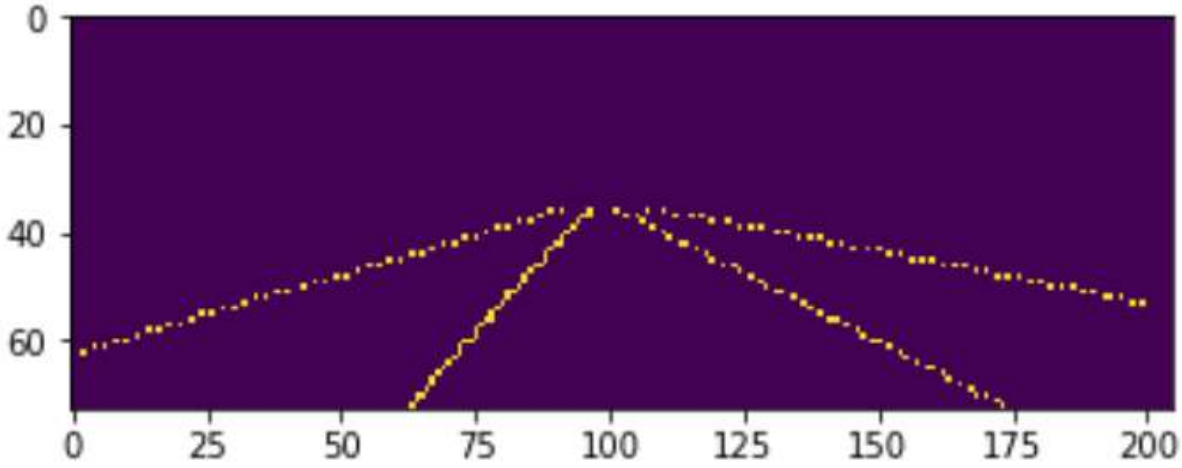
Error：误差图



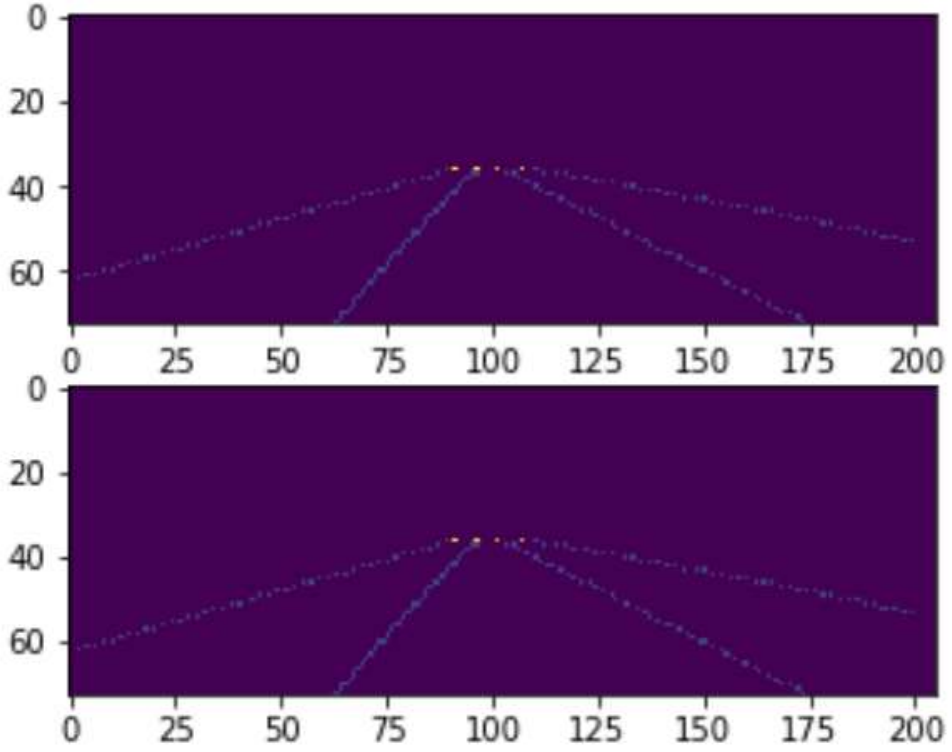
Label给定关键点坐标都是float类型
这个热力图用于对齐整数和float类型的坐标存在误差

```
1  240.573 590 257.848 580 275.127 570 292.409 560 309.699 550 327.126 540 344
   467.012 460 484.586 450 502.935 440 520.431 430 538.821 420 557.229 410 575
   704.416 330 722.879 320 741.337 310 759.788 300 778.228 290
2  1146.04 590 1133.33 580 1120.99 570 1108.67 560 1097.02 550 1084.72 540 107
   992.112 460 980.928 450 969.781 440 958.669 430 947.589 420 936.959 410 925
   850.519 330 839.503 320 828.468 310 818.147 300 807.161 290
3  1660.47 470 1616.64 460 1573.83 450 1532 440 1490.15 430 1447.33 420 1404.4
   1096.34 340 1050.83 330 1002.88 320 953.887 310 901.854 300 847.714 290
4
```

车道线掩码和消失点加权掩码(training)

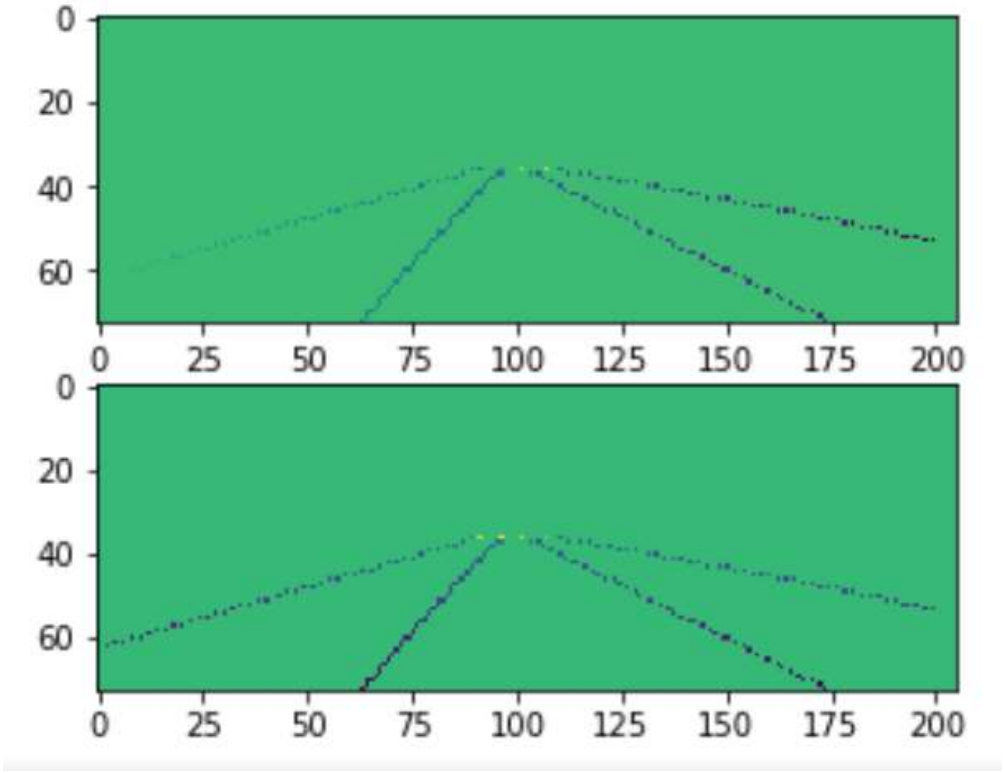


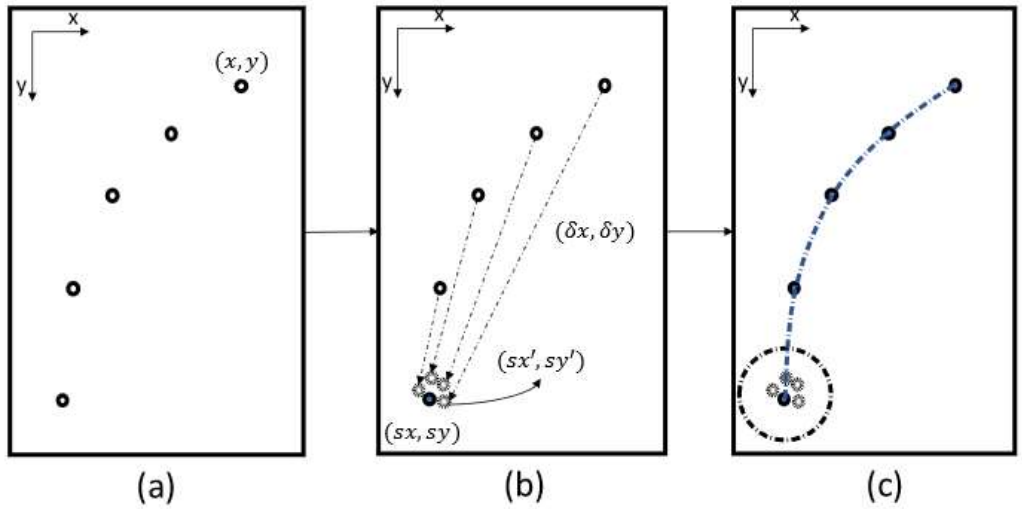
普通掩码：整数关键点处为1



加权掩码：给消失点加权

车道线起始点的偏移量





使用起始点聚类

掩码L1loss: 用于预测offset和error

```
def forward(self, output, target, mask):
    loss = F.l1_loss(output * mask, target * mask, size_average=False)
    mask = mask.bool().float()
    loss = loss / (mask.sum() + 1e-4)
    return loss
```

使用掩码加权的l1 loss

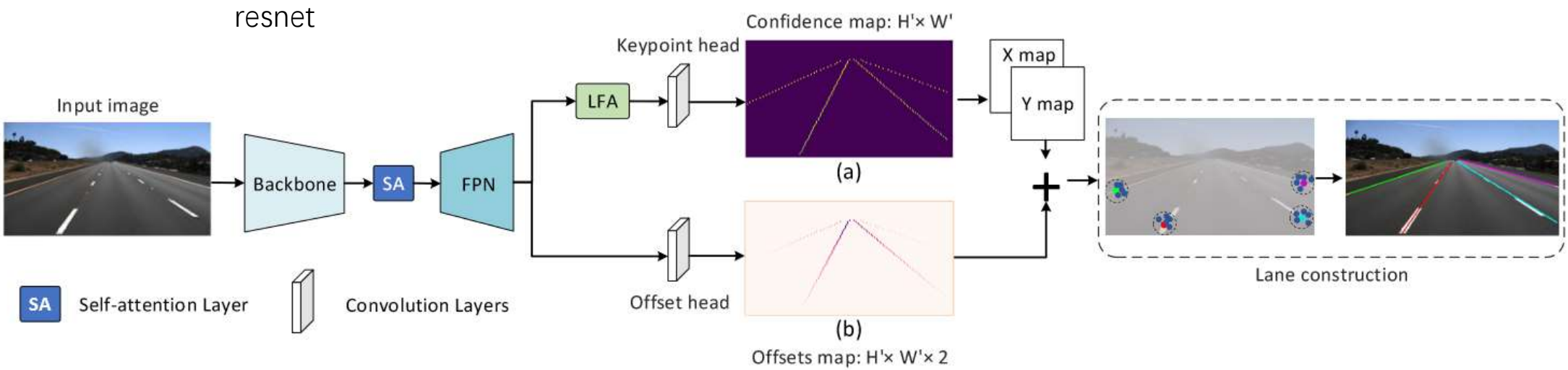
用于**error map**(掩码为正常mask)和**offset map**(掩码为加权掩码)

使用mask会使loss更加稳定，防止前景背景分布不均

而key point heat map和start point heat map（权重为0？）用focal loss

网络结构:

- 多个**解耦的头**部分别预测热力图、偏移量图和误差图
- FPN结构，但是，最终输出时，每一层过一遍**dcn**
- Self-attention：仅用于FPN**下采样最多**的输出
- 推理时，abs(offset) < 1是**候选起始点**



LanePointConv:

B: batch_size
DCN: deformable CNN 1D
Kernel_sz: in [(1,7),(1,5),(1,3)]
Detach: to detach grad by 90%

integrated with a deformable convolution to aggregate context of the i-th keypoint as:

$$\hat{\mathcal{F}}(p_i) = \sum_{m=1}^{M} w_m \cdot \mathcal{F}(p_i + \Delta p_i^m), \tag{9}$$

where $w_m, m = 1, ..., M$ is the weight of the convolution and $(\cdot)$ means multiplication.

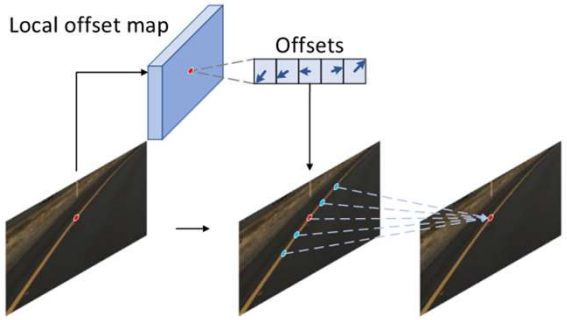
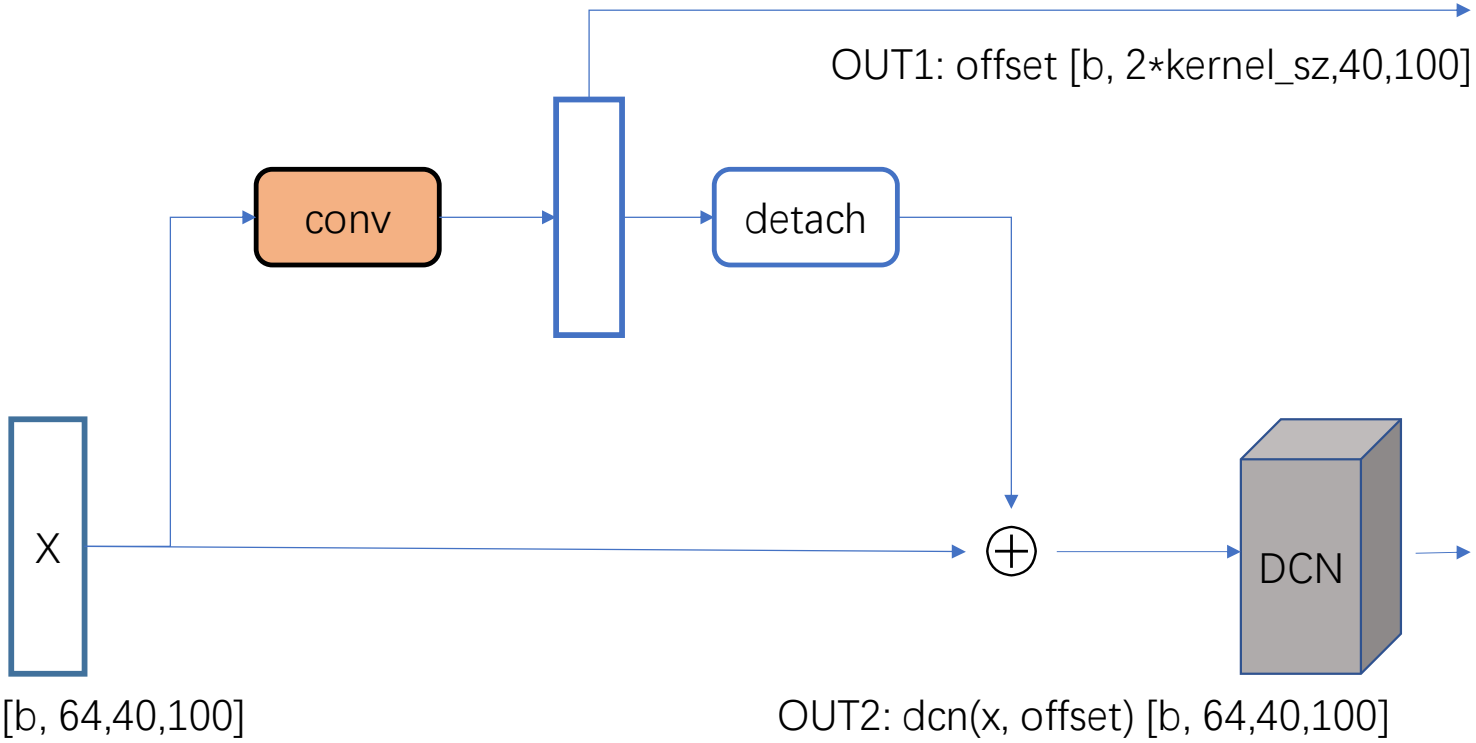
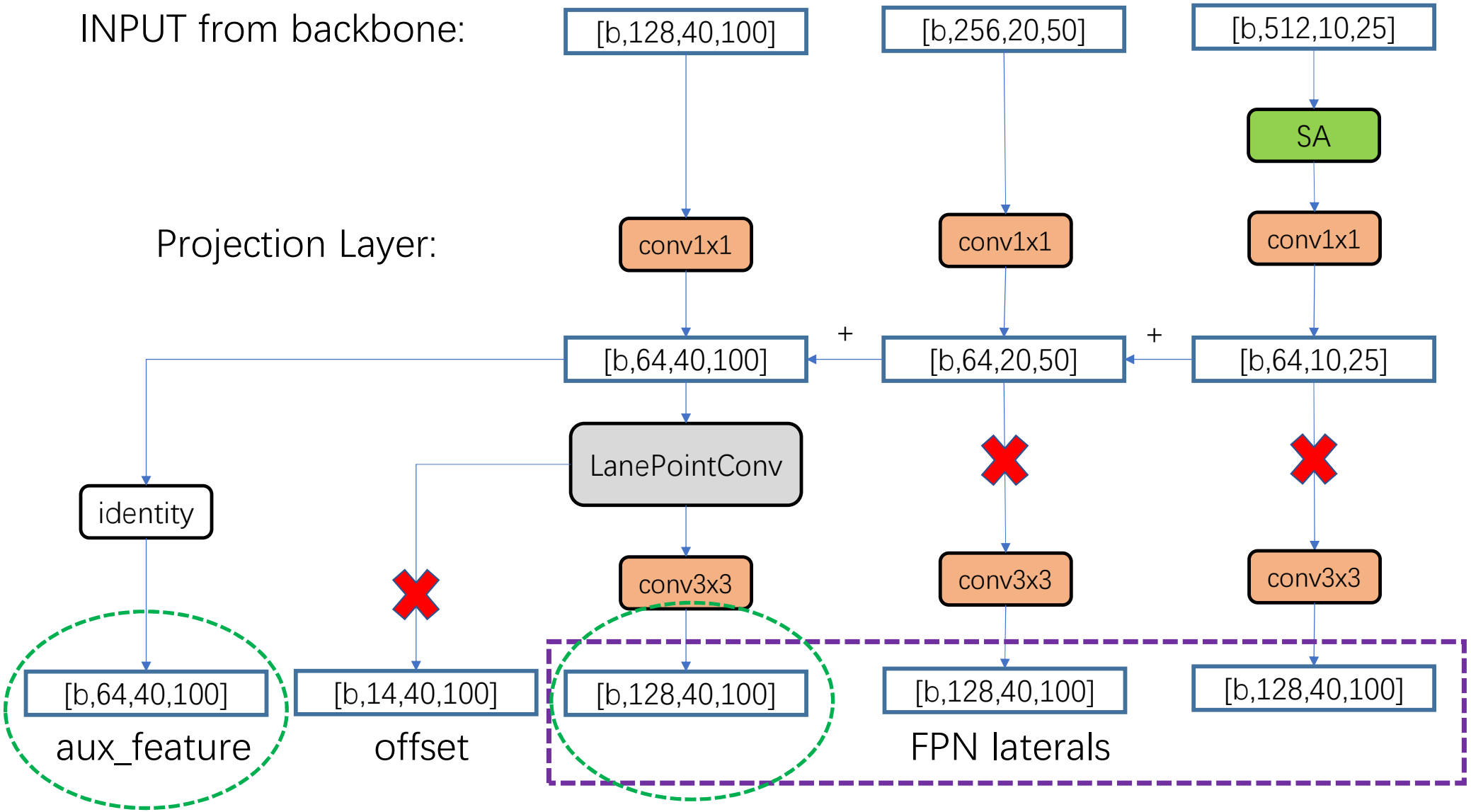
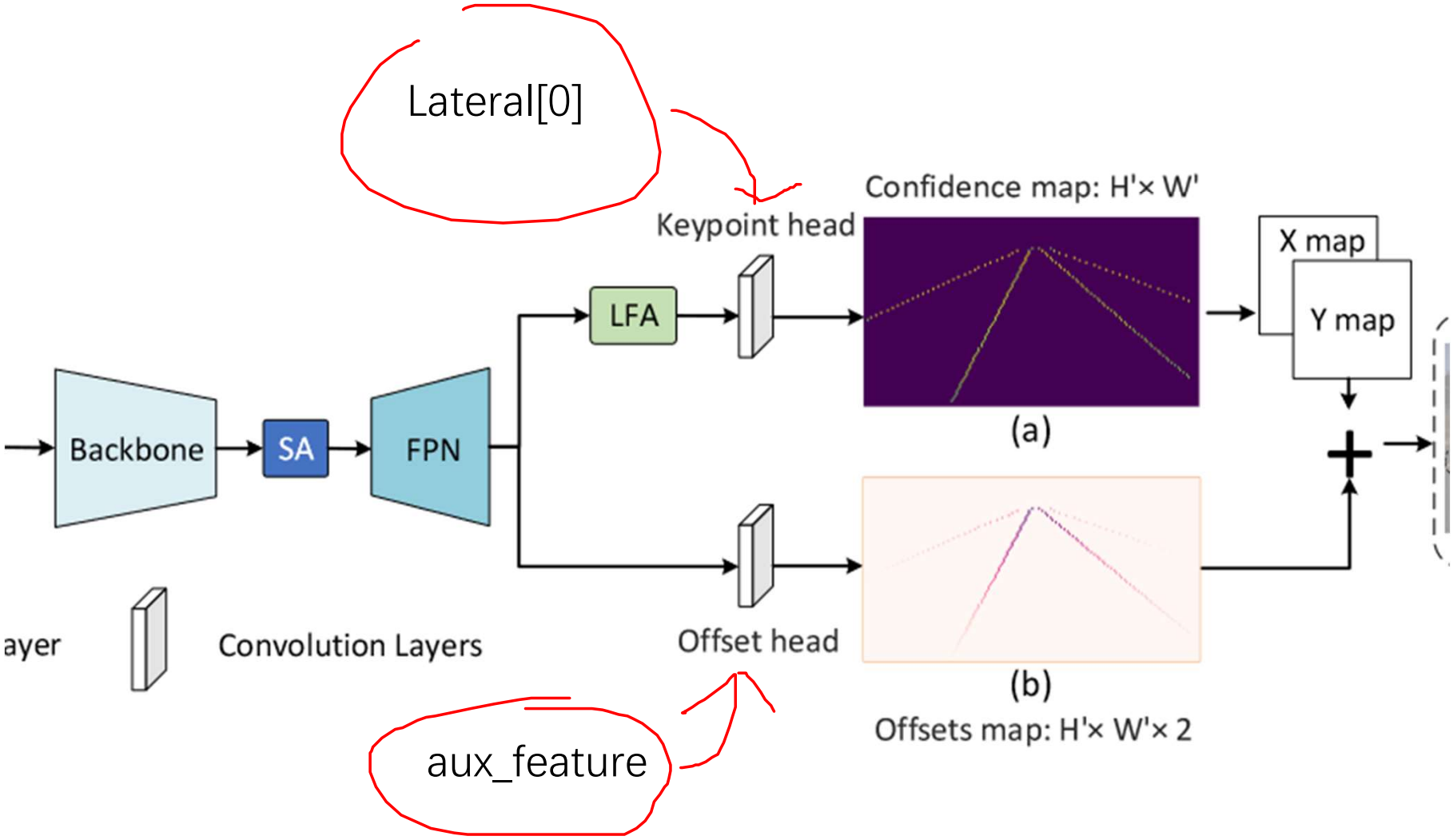


Figure 4. Illustration of LFA module. The red dot denotes the observed keypoint. We first predict offsets between the red dot and its adjacent keypoints (in blue) and then gather features of these keypoints to enhance the context of the red keypoint.




INPUT from backbone:
[b,128,40,100]
[b,256,20,50]
[b,512,10,25]
SA
Projection Layer:
conv1x1
conv1x1
conv1x1
+
+
[b,64,40,100]
[b,64,20,50]
[b,64,10,25]
LanePointConv
identity
conv3x3
conv3x3
conv3x3
[b,64,40,100]
[b,14,40,100]
[b,128,40,100]
[b,128,40,100]
[b,128,40,100]
aux_feature
offset
FPN laterals


Lateral[0]
Keypoint head
Confidence map: H'× W'
LFA
X map
Y map
Backbone
SA
FPN
(a)
ayer
Convolution Layers
Offset head
(b)
Offsets map: H'× W'× 2
aux_feature

推理

- 起始点

  (heat_map > 0.3) * (offset < 1)取均值

- 关键点:

  heat_map[offset+meshgrid2d] where heat_map>0.3

  然后计算和起始点的距离使用KNN算法聚类

$$(sx', sy') = (x, y) + (\delta x, \delta y), \tag{7}$$

where (x, y) is the coordinate of the observed keypoint and $(\delta x, \delta y) = O_{yx}$ denotes the corresponding offset obtained in Section 3.1.2. The keypoint (x, y) is associated to the i-th lane only if the distance between (sx', sy') and (sx, sy) is less than a predefined threshold θ_{dis}. As shown in Fig-

=5

hm:[1, 1, 40, 100] from key point head 热力图
offset:[1,2,40,100] from offset head 坐标偏移量且为y offset
error: [1,2,40,100] from error head 把关键点从整数值修正到浮点数

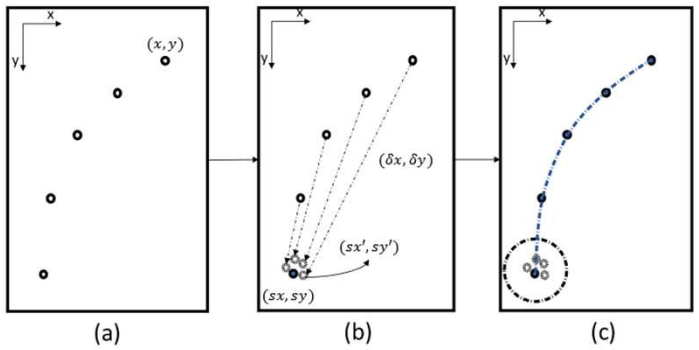


Figure 3. Illustration of lane construction. (a) Valid keypoints are selected from the confidence map. (x, y) is taken as an example. (b) Starting point (sx, sy) (blue dot) is sampled first. The rest keypoints point to the starting point with the predicted offset $(\delta x, \delta y)$ and estimate the coordinate of the starting points as $(sx', sy') = (x, y) + (\delta x, \delta y)$ (hollow dots). (c) Keypoints that point to the neighbourhood of starting point (sx, sy) are grouped as a whole lane.

```
key_points = (meshgrid+error)[where(nms(hm)) > 0.3]
start_points = (meshgrid+offset)[where(nms(hm) > 0.3)]

id(x,y): which group the point (x, y) belonging to
base case: group = [ [(x0, y0)] ]

for (x, y) in start_points:
    for (cx, cy) in group.flatten():
        if |(x,y) - (cx, cy)| < 5    # 欧式距离小于5
            id(x,y) = id(cx,cy)      # (x,y) 的group_id为(cx,cy)的group_id
            group[id] += (x,y)       # id对应的group添加新的点
            break inner loop
    if GROUP_NOT_FOUND:
        group += [(x,y)]     # 产生一个新的group

# 注意到key_points和start_points是对齐的，因此:
[id(x,y) for (x,y) in key_points] ==  [id(sx,sy) for (sx,sy) in start_points]
```