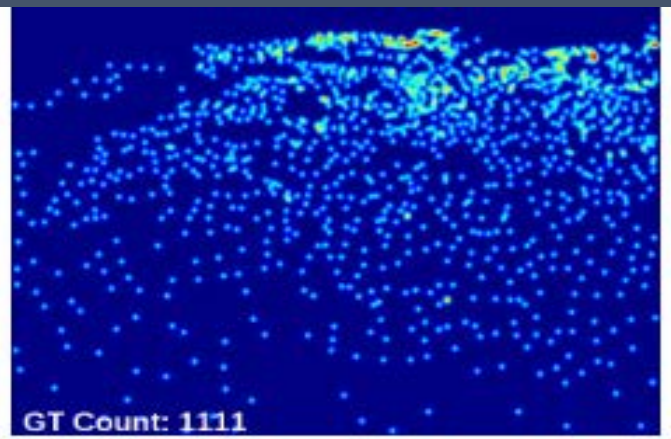
我会不定期翻译一些论文，不定期更新上传到这里
https://github.com/xr0927/chapter8-translate-papers

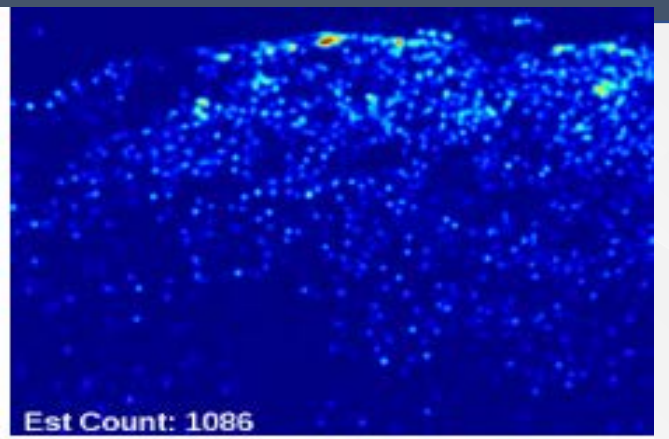
CNN-based Cascaded Multi-task Learning of High-level Prior and Density Estimation for Crowd Counting

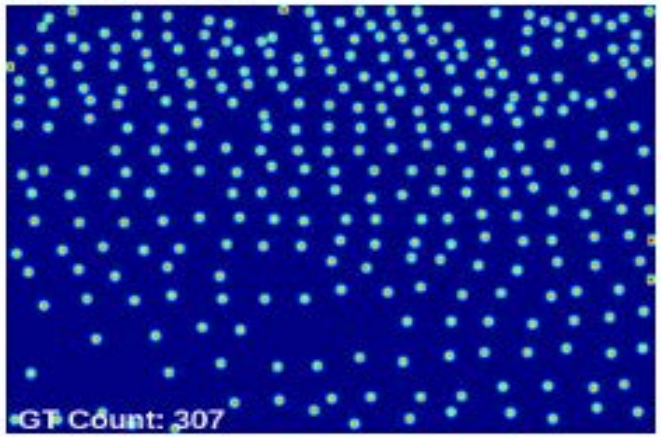
报告人：徐如如

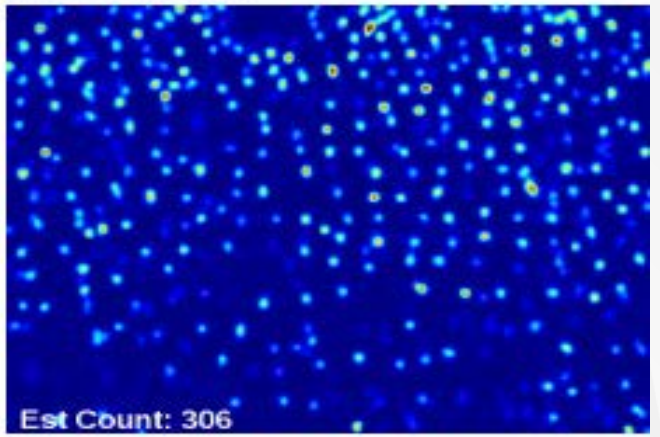
20180331

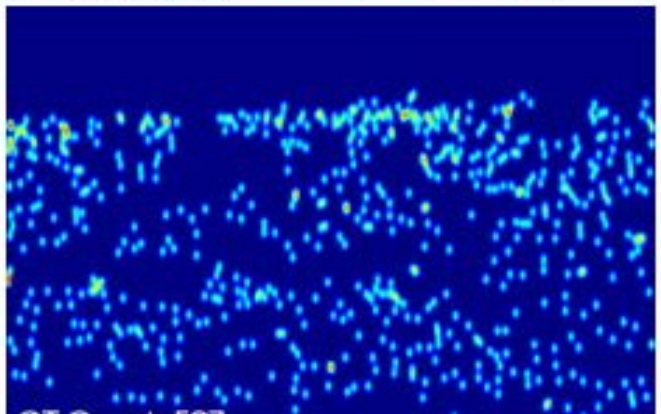
Result:

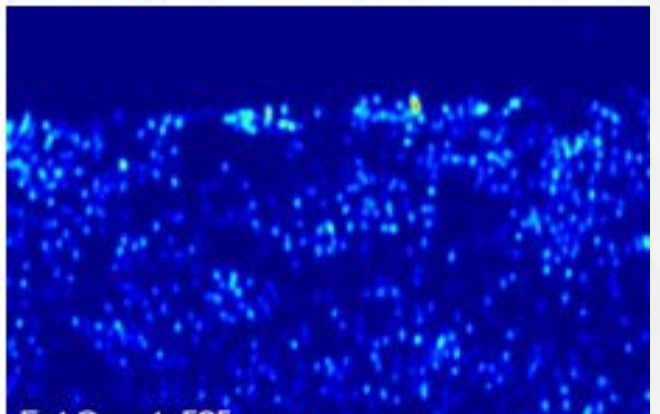






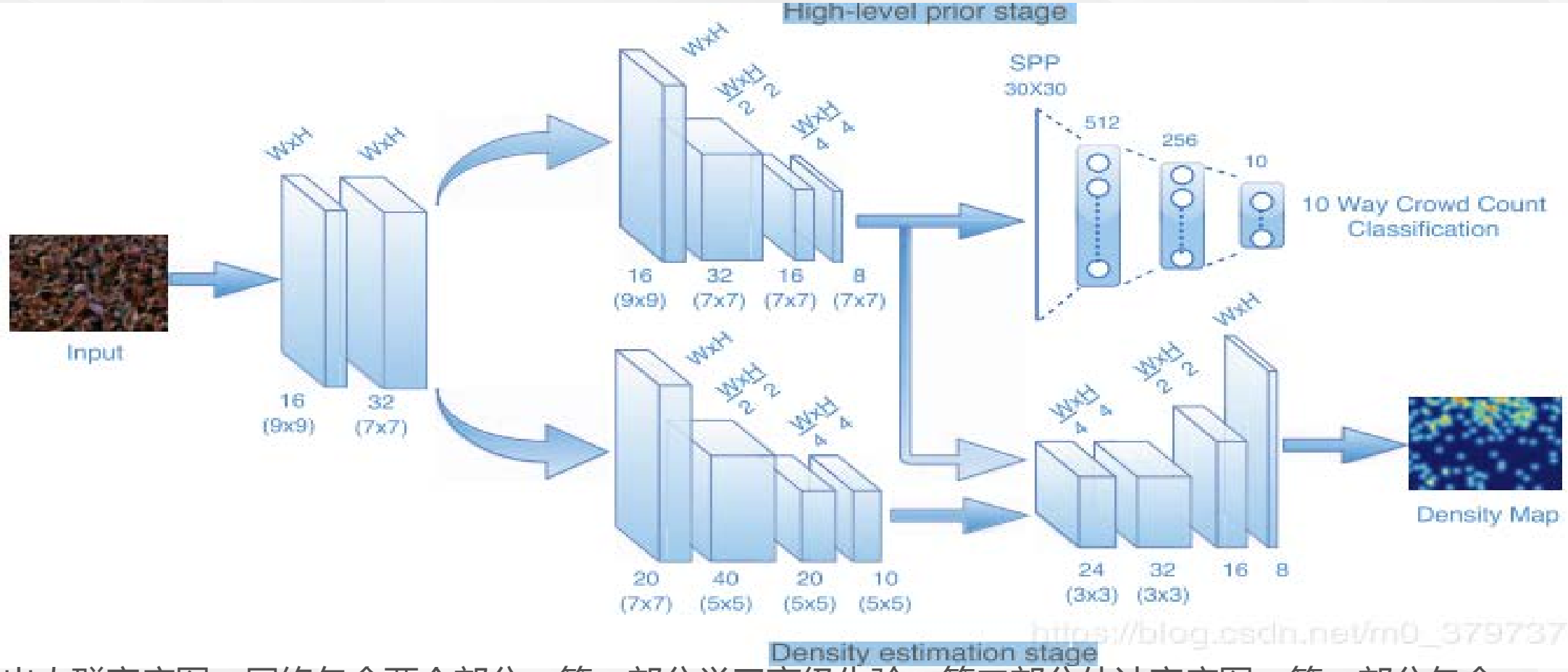






method

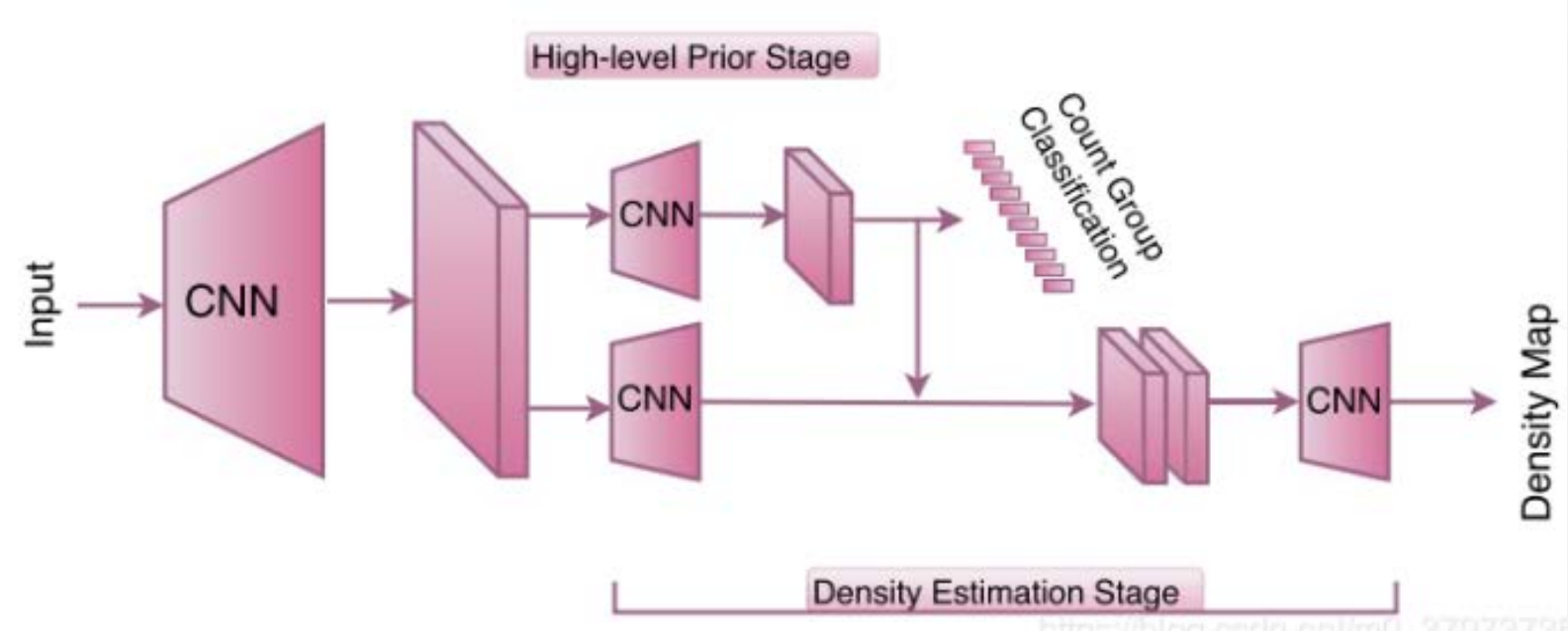
提出了学习两个相关的子任务：高级先验和密度图估计。网络如下所示：



输入图像，输出人群密度图。网络包含两个部分，第一部分学习高级先验，第二部分估计密度图。第一部分包含一组卷积层，金字塔形的池化层和全连接层。第二部分由一组卷积层组成，然后是小步长卷积层，用于对前一层的输出进行向上采样，以弥补早期池化层造成的细节损失。

Approach:

本论文的目的是将一个高级先验与网络合并，学习出一个满足数据集中各种密度等级的模型。高级先验先根据图中人的数量分为不同的带标签的组。利用标签，这个高级先验能大致估计整个图片中的人数，而不受尺度变化的影响，从而使网络能学到更多的判别全局特征。利用高级先验和CNN网络共同进行密度图的估计



这两个任务（人群密度估计和高级先验）共享一个卷积层，然后分为两个网络。将高阶先验学习到的全局特征与第二组卷积层获得的特征图连接起来，再由一组卷积层进一步处理，得到高分辨率密度图。

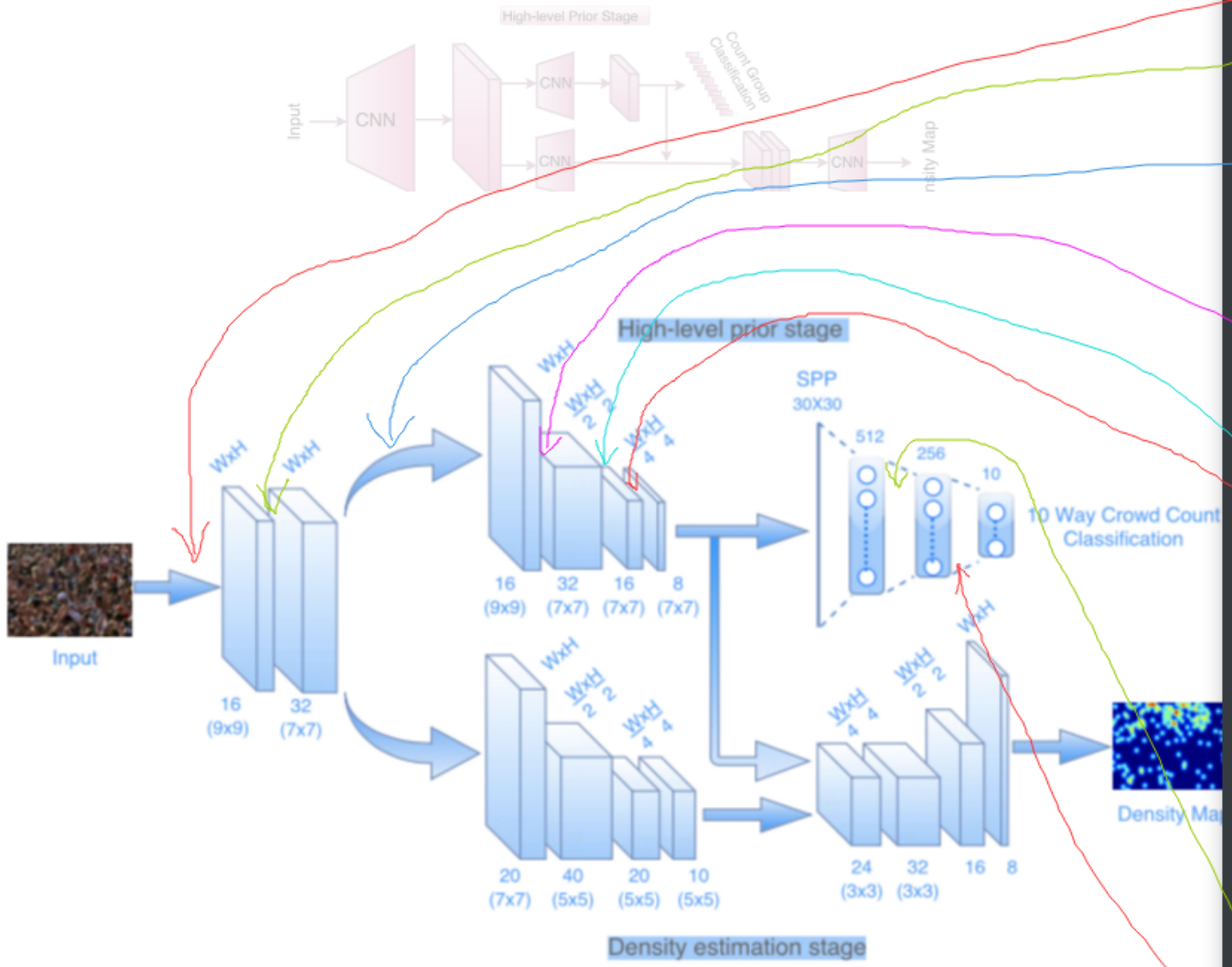
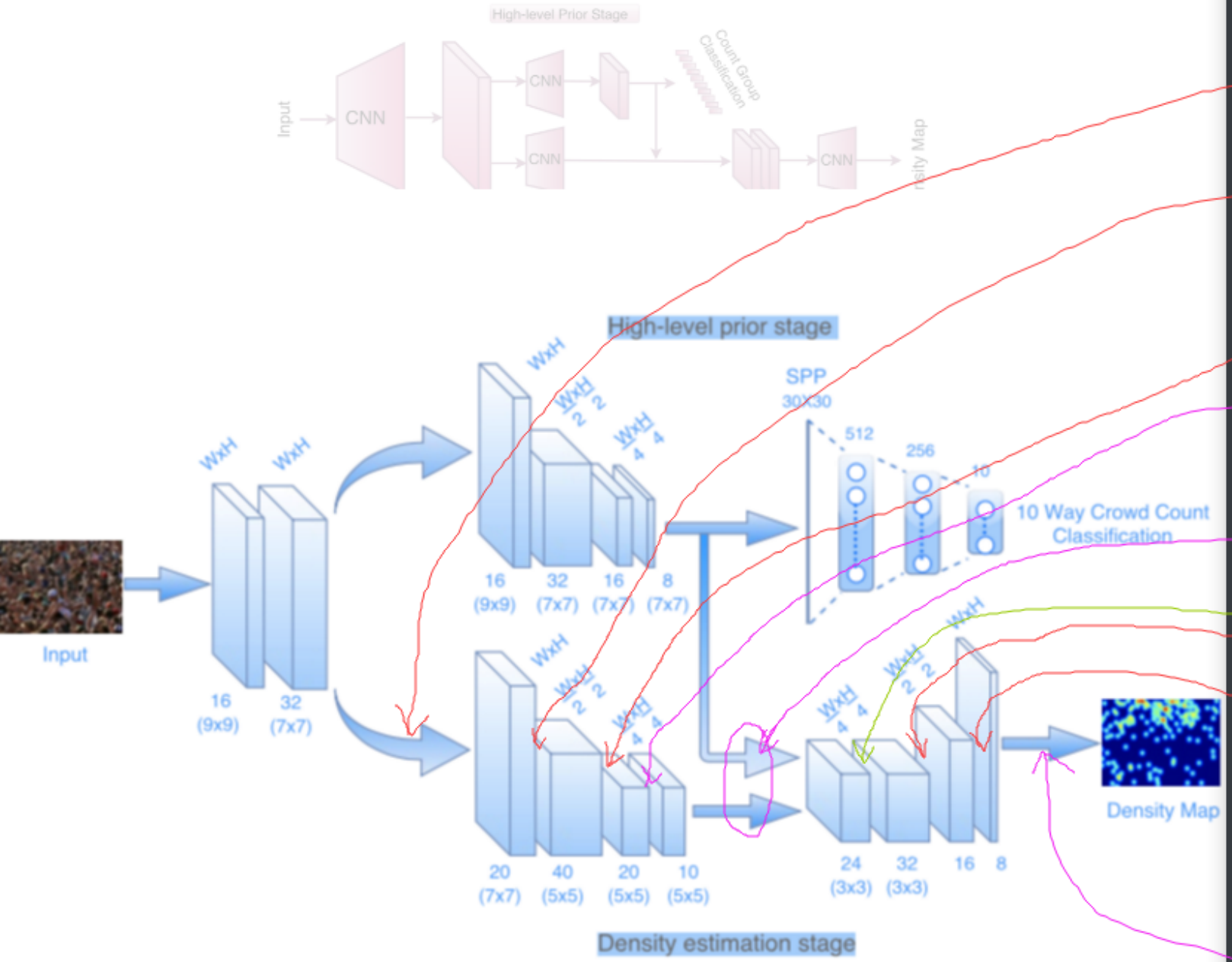


Figure 2: Overview of the proposed cascaded architectu
for jointly learning high-level prior and density estimatio

Figure 2: Overview of the proposed cascaded architecture

```
    (0): Conv2d(
      (conv): Conv2d(1, 16, kernel_size=(9, 9), stride=(1, 1), padding=(4, 4))
      (relu): PReLU(num_parameters=1)
    )
    (1): Conv2d(
      (conv): Conv2d(16, 32, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
      (relu): PReLU(num_parameters=1)
    )
  )
  (h1_prior_1): Sequential(
    (0): Conv2d(
      (conv): Conv2d(32, 16, kernel_size=(9, 9), stride=(1, 1), padding=(4, 4))
      (relu): PReLU(num_parameters=1)
    )
    (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (2): Conv2d(
      (conv): Conv2d(16, 32, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
      (relu): PReLU(num_parameters=1)
    )
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(
      (conv): Conv2d(32, 16, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
      (relu): PReLU(num_parameters=1)
    )
    (5): Conv2d(
      (conv): Conv2d(16, 8, kernel_size=(7, 7), stride=(1, 1), padding=(3, 3))
      (relu): PReLU(num_parameters=1)
    )
  )
  (h1_prior_2): Sequential(
    (0): AdaptiveMaxPool2d(output_size=(32, 32))
    (1): Conv2d(
      (conv): Conv2d(8, 4, kernel_size=(1, 1), stride=(1, 1))
      (relu): PReLU(num_parameters=1)
    )
  )
  (h1_prior_fc1): FC(
    (fc): Linear(in_features=4096, out_features=512, bias=True)
    (relu): PReLU(num_parameters=1)
  )
  (h1_prior_fc2): FC(
    (fc): Linear(in_features=512, out_features=256, bias=True)
    (relu): PReLU(num_parameters=1)
  )
  (h1_prior_fc3): FC(
    (fc): Linear(in_features=256, out_features=10, bias=True)
```

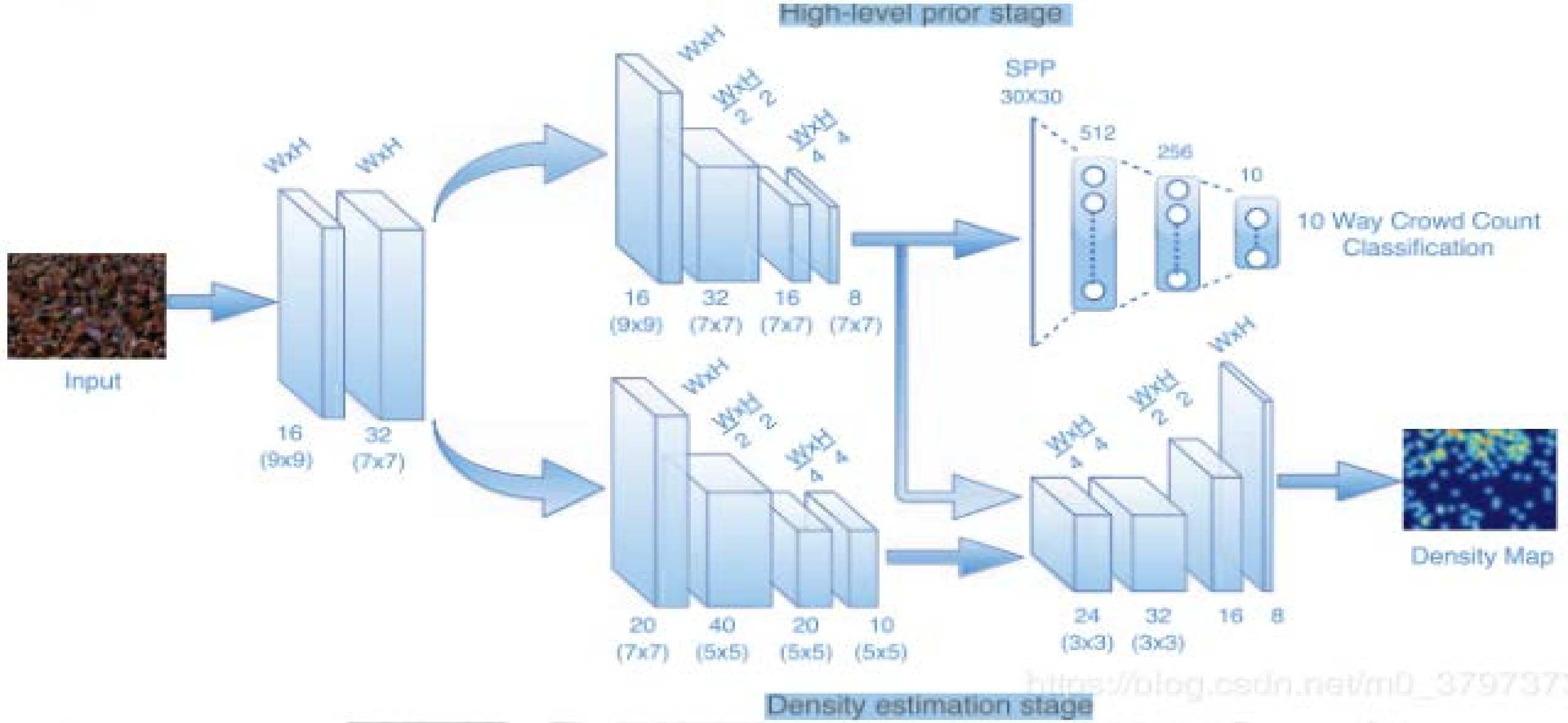


gure 2: Overview of the proposed cascaded architectu

```
    (0): Conv2d(
      (conv): Conv2d(32, 20, kernel_size=(7, 7), stride=(1, 1), padding=(3
      (relu): PReLU(num_parameters=1)
    )
    (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mo
    (2): Conv2d(
      (conv): Conv2d(20, 40, kernel_size=(5, 5), stride=(1, 1), padding=(2
      (relu): PReLU(num_parameters=1)
    )
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mo
    (4): Conv2d(
      (conv): Conv2d(40, 20, kernel_size=(5, 5), stride=(1, 1), padding=(2
      (relu): PReLU(num_parameters=1)
    )
    (5): Conv2d(
      (conv): Conv2d(20, 10, kernel_size=(5, 5), stride=(1, 1), padding=(2
      (relu): PReLU(num_parameters=1)
    )
  )
  (de_stage_2): Sequential(
    (0): Conv2d(
      (conv): Conv2d(18, 24, kernel_size=(3, 3), stride=(1, 1), padding=(1
      (relu): PReLU(num_parameters=1)
    )
    (1): Conv2d(
      (conv): Conv2d(24, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1
      (relu): PReLU(num_parameters=1)
    )
    (2): ConvTranspose2d(32, 16, kernel_size=(4, 4), stride=(2, 2), paddin
    (3): PReLU(num_parameters=1)
    (4): ConvTranspose2d(16, 8, kernel_size=(4, 4), stride=(2, 2), padding
    (5): PReLU(num_parameters=1)
    (6): Conv2d(
      (conv): Conv2d(8, 1, kernel_size=(1, 1), stride=(1, 1))
      (relu): ReLU(inplace)
    )
  )
)
(loss_mse_fn): MSELoss()
```

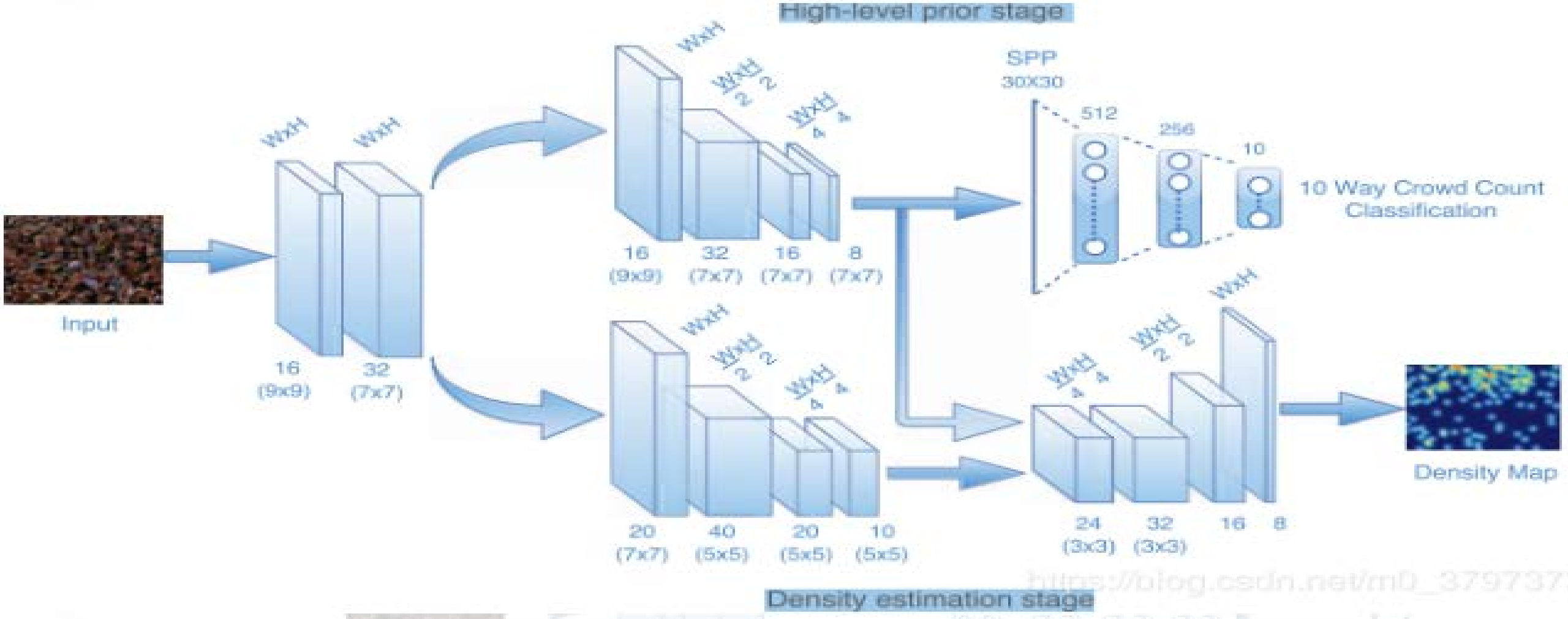
Shared convolutional layers

这部分由两个卷积层组成，每个卷积层后接PReLU激活函数。第一个卷积层有16张特征图，卷积核大小为9x9；第二个卷积层有32张特征图，卷积核大小为7x7。这一浅层网络生成的特征图将会被高水平先验阶段和密度图估计阶段进行共享。



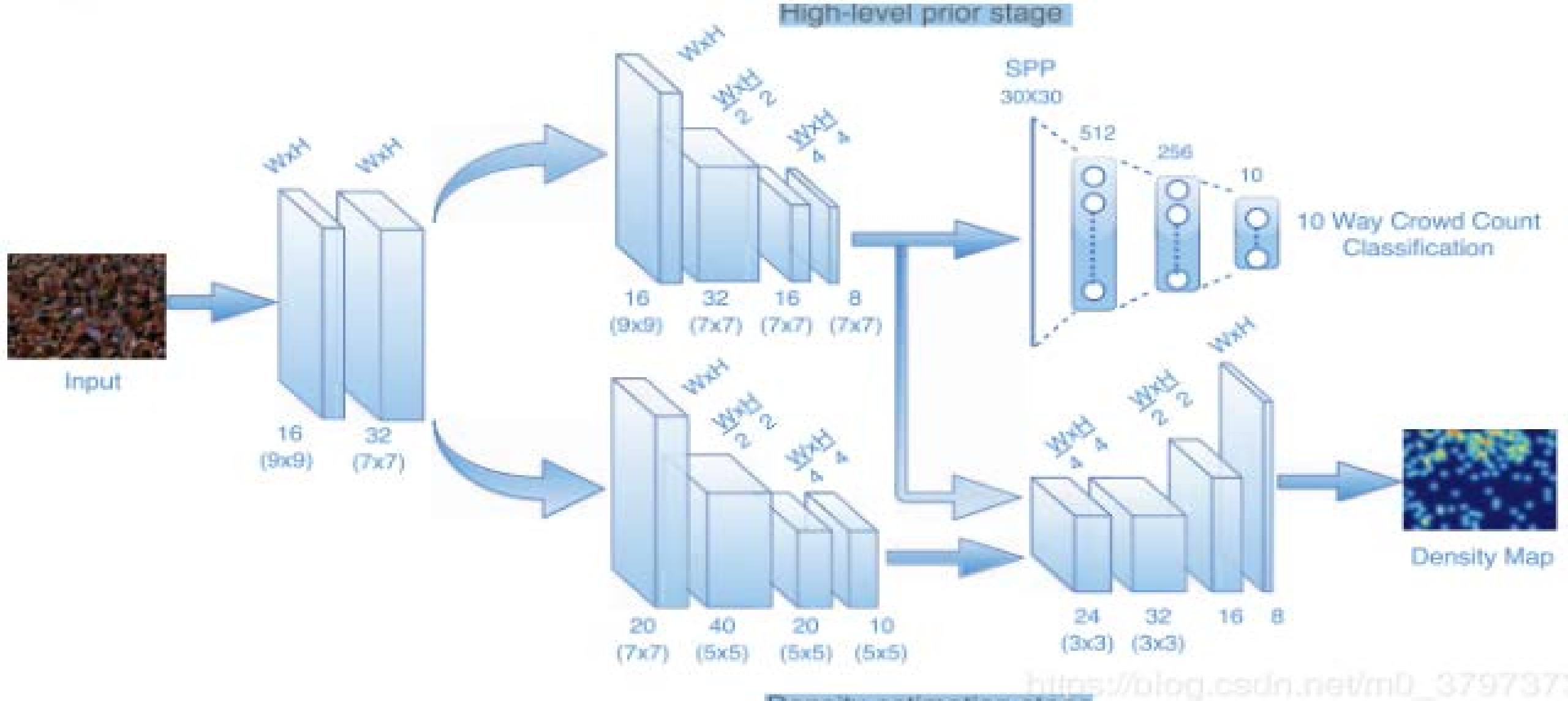
Hight-level prior stage

将人群分为几组比起直接对整个图像进行分类或回归简单，因为不需要太多的训练数据。因此，我们将人群计数量化为10组，并学习了一个人群计数组分类器，该分类器还执行将高层先验合并到网络中的任务。先验部分接受之前得到的特征图作为输入。这个部分包含4个卷积层，每个卷积层之后都用PReLU作为激活函数。开始两个卷积层后有步长为2的最大池化层。最后包含3个全连接层，激活函数仍然是PReLU，神经元数量分别为512、256、10。为了能够使用任意大小的图像进行训练，使用空间金字塔池(SPP)，因为它消除了包含完全连接层的深度网络的固定大小约束。SPP层将卷积层的特征集合起来，产生固定大小的输出，并可以将其提供给完全连接的层。交叉熵误差作为这一阶段的损失层



Density estimation

这部分网络仍包含4个卷积层，每层之后都有PReLU函数作为激活函数，开始两层后有步长为2的最大池化层。第一个卷积层为7×7，20个通道，第二个卷积层为5×5，40个通道，第三个为5×5，20个通道，第四层为5×5，10个通道。这个网络的输出与高级先验的输出通过2个卷积层和2个小步长卷积层结合起来。头两个卷积层是3×3，24通道和32通道，小步长卷积层是16通道和18通道。这些小步长卷积层除了能整合先验外，还可以将特征图提升到原始输入尺寸，从而恢复之前最大池化层丢失的细节。这些图层的使用使CNN输出的上采样率提高了4倍，从而使我们能够在全分辨率密度图上回归。标准欧几里得损耗作为损失层。



Loss function

高级先验部分的交叉熵损失函数为：

$$L_c = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}[(y^i = j)F_c(X_i, \Theta)],$$

N表示训练例子数量，theta是一组网络参数，X_i是第i个训练例子，F_C（X_i，Theta）表示输出的分类，yi是真值分类。M是类别的数量。

密度估计损失函数为：

$$L_d = \frac{1}{N}\sum_{i=1}^{N}\|F_d(X_i, C_i, \Theta) - D_i\|_2,$$

Fd是估计的密度图，Di是真实的密度图，Ci是高级先验阶段最后一个卷积层得出的特征图。则总的损失函数为：

$$L = \lambda L_c + L_d,$$

这种损失函数不同于传统的多任务学习，因为最后一阶段的损失项取决于前一阶段的输出。

Training and implementation details

为了创建数据集，先在原始图像上随机找100个位置，每个位置裁剪出1/4尺寸的局部图像。然后用水平翻转和加入噪声的方法创建另外200个局部图像。需要说明的是，裁剪只是为了数据扩充，数据输入可以是任意尺寸。真值密度图的计算用了一个简单的方法：

$$D_i(x) = \sum_{x_g \in S} \mathcal{N}(x - x_g, \sigma)$$

D_i对应第i个训练patch，通过对每个人位置x_g为中心进行2维高斯核求和得到。
sigma表示2维高斯核的尺度参数（标准差），S表示所有人所在位置点的集合。