深度估计

- 要做一件什么事呢?
  - 输入彩色图像
  - 输出深度估计
  - 深浅即为远近
  - 离相机距离估计

深度估计

- 有啥应用场景呢?
  - AR,VR这些好玩的家伙:
  - 可以得到每个像素点的深度值
  - 辅助驾驶场景:
  - 获取各目标距离信息

深度估计

- 有啥应用场景呢?
  - 机器人以后都用得上
  - 驾驶，追踪，分析等场景
  - 可以当作视觉领域的一个基础
  - 下游应用还是蛮多的

深度估计

- 有啥应用场景呢?
  - 很多应用项目的基础，例如三维重建：
  - 深度信息在三维世界中相当重要（坐标转换）
  - 那为啥不用激光雷达直接获取深度信息和点云呢?
  - 因为穷！所以咱们来唠最省钱的单目深度估计(End2End这种)

深度估计

- KITTI数据集
  - 一个字：广（深度,分割,检测,3D等）
  - 两个字：准确（详情见宣传片）
  - 三个字：经典啊（研究论文都会用）
  - 四个字：用就得了（后面任务也会用）

深度估计

- KITTI数据集

  - 用的相当广泛，大部分算法都以此为训练数据

Depth Evaluation

This benchmark is related to our work published in Sparsity Invariant CNNs (THREEDV 2017). It contains over 93 thousand depth maps with corresponding raw LiDaR scans and RGB images, aligned with the "raw data" of the KITTI dataset. Given the large amount of training data, this dataset shall allow a training of complex deep learning models for the tasks of depth completion and single image depth prediction. Also, we provide manually selected images with unpublished depth maps to serve as a benchmark for those two challenging tasks.

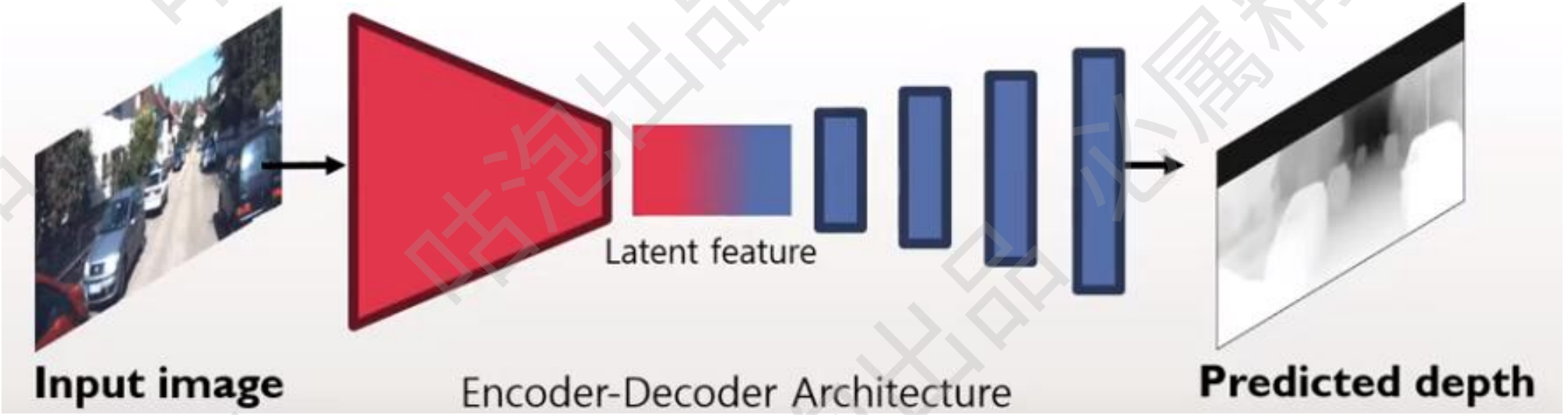
Make sure to unzip annotated depth maps and raw LiDaR scans into the same directory so that all corresponding files end up in the same folder structure. The structure of all provided depth maps is aligned with the structure of our raw data to easily find corresponding left and right images, or other provided information.

- Download annotated depth maps data set (14 GB)
- Download projected raw LiDaR scans data set (5 GB)
- Download manually selected validation and test data sets (2 GB)
- Download development kit (48 K)

深度估计

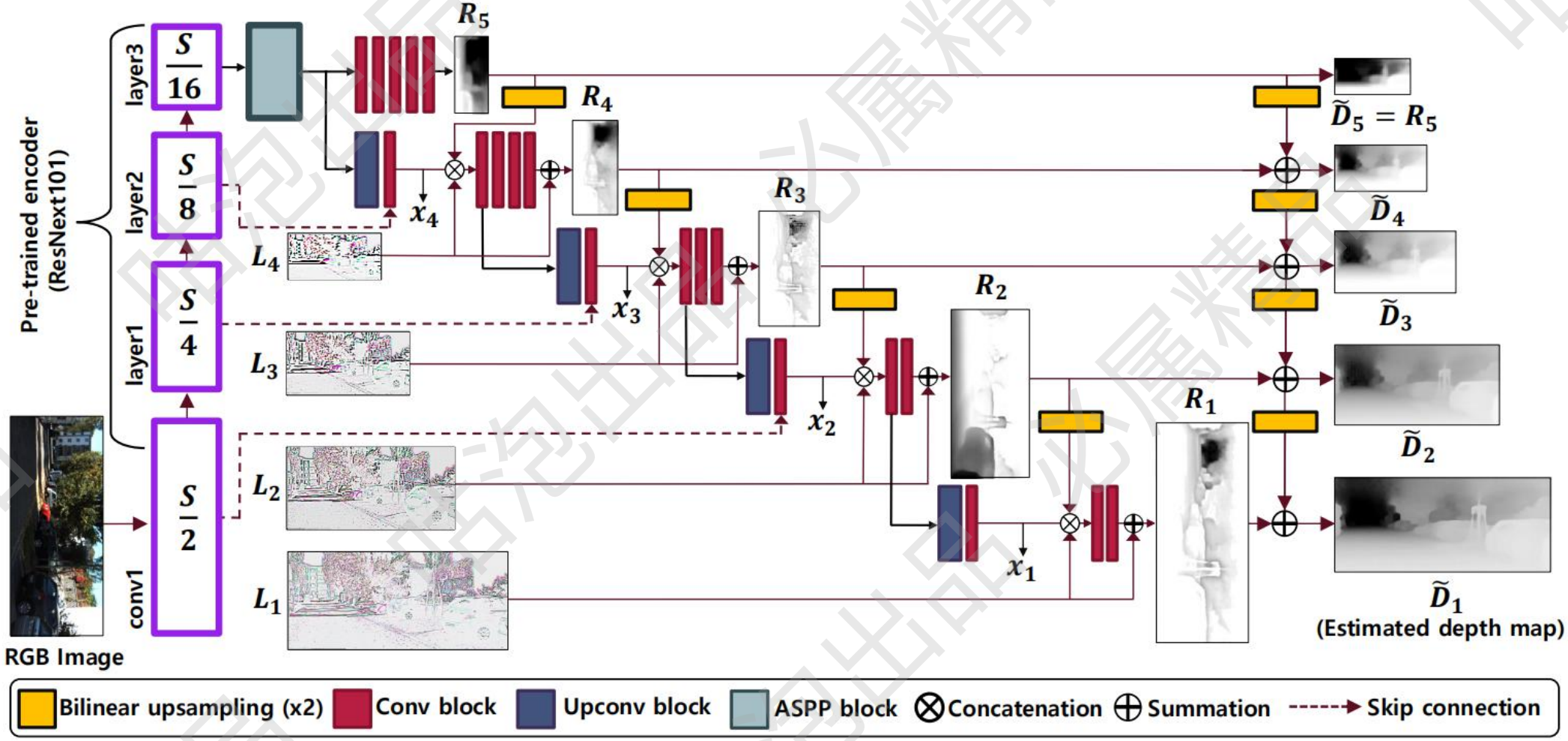
- 输入和输出

- 输入就是单张图像
- 然后估计你猜到了过程就是编码解码结构（就跟那个分割挺像的）



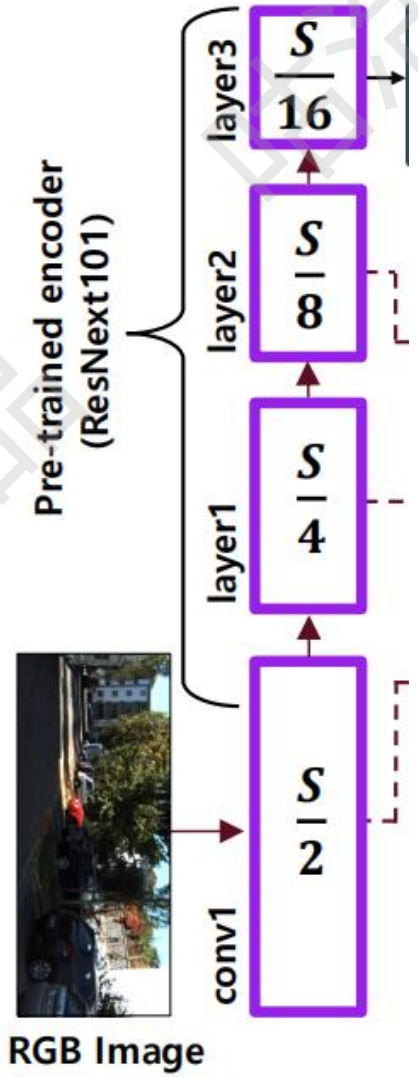
深度估计

- 深度估计模型整体架构



深度估计

- 第一招：层级
  - 对于输入的backbone选择，基本啥都行，看家里啥条件
  - 为啥要整出来这么多层？(横看成岭侧成峰，远近高低各不同)
  - 这套路也是老演员了，基本现在模型都是特征越丰富越好
  - 4个层，尺度大小各不相同，先准备好backbone输出



深度估计

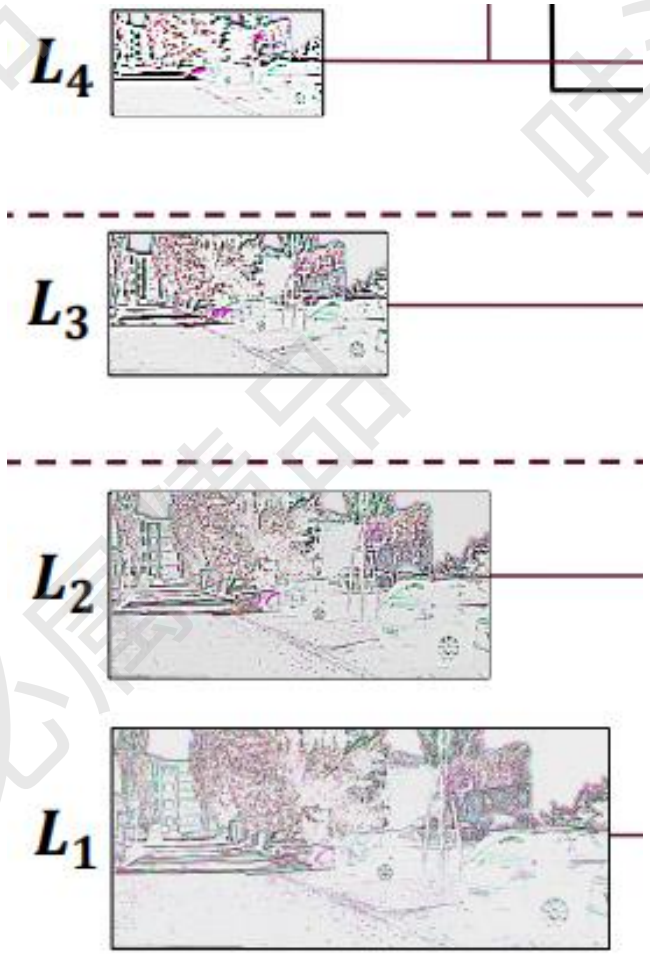
- 难点是啥呢?

  - 深度估计中挑战在哪呢？啥地方容易整的不准呢？是物体的边界还是内部呢?

  - 下图就是不断调整最终得到比较好的边界（轮廓）信息，那这些特征哪来的呢?

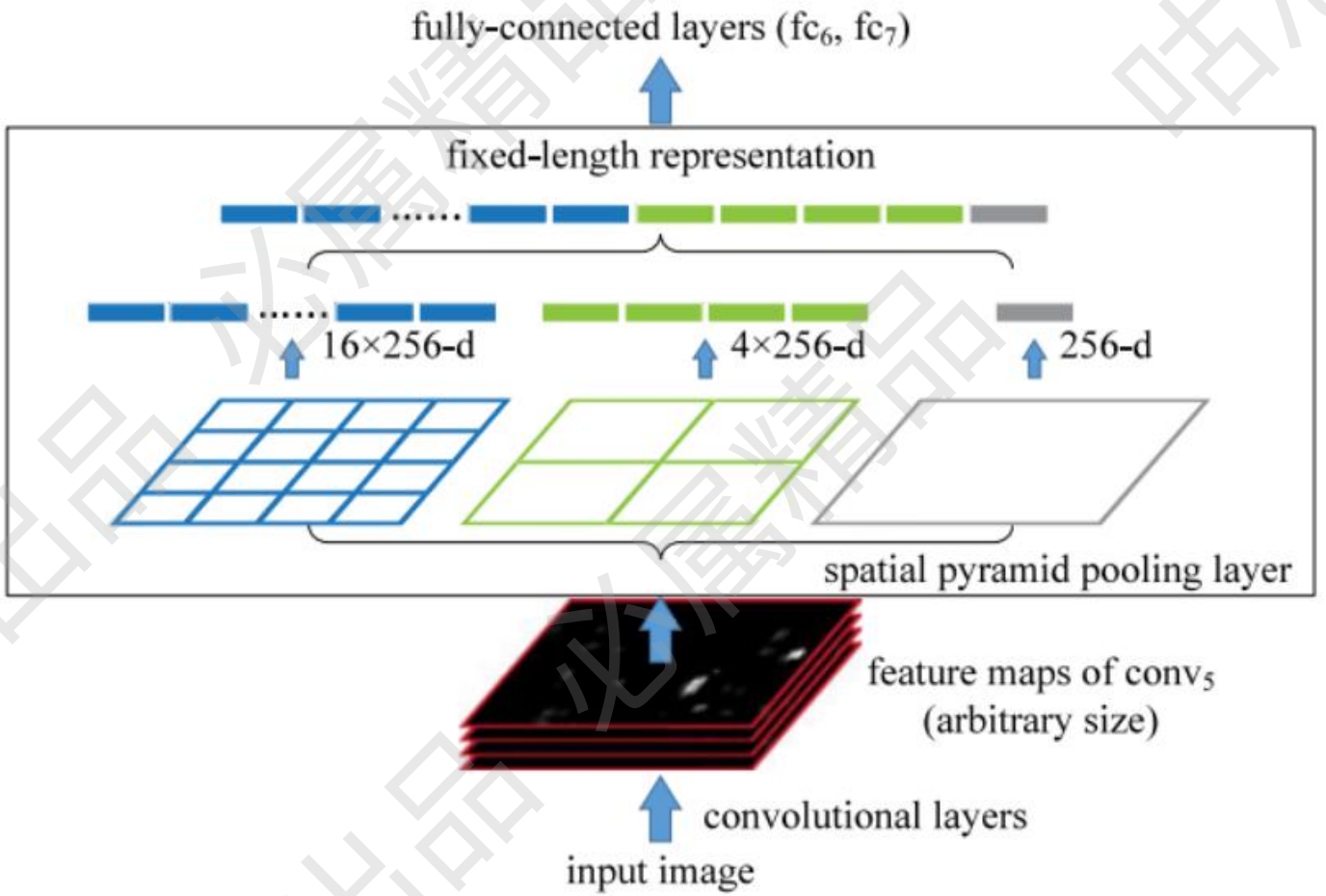
深度估计

- 第二招：差异
  - 如何得到边界信息呢？（类似局部特征）
  - 两个特征图做减法来提取差异特征
  - 分别对各个层级都做这个事，提取不同尺度的差异
  - 实际做法类似unet（对输入做上下采样后算差异）



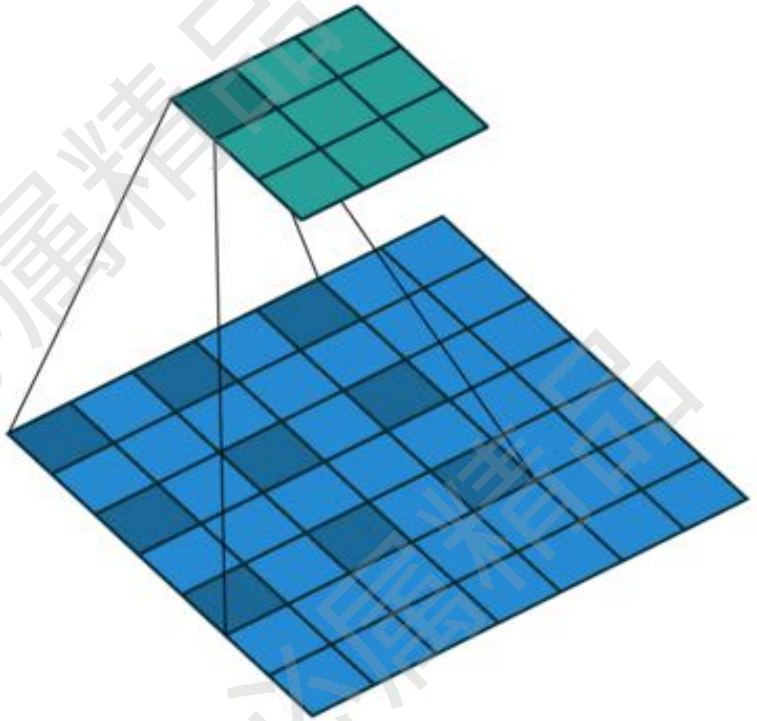
深度估计

- SPP是干啥的?
  - 为了得到固定的特征
  - 通常得要求输入大小固定
  - 但是resize会丢失信息
  - SPP它就来解决这个问题的



深度估计

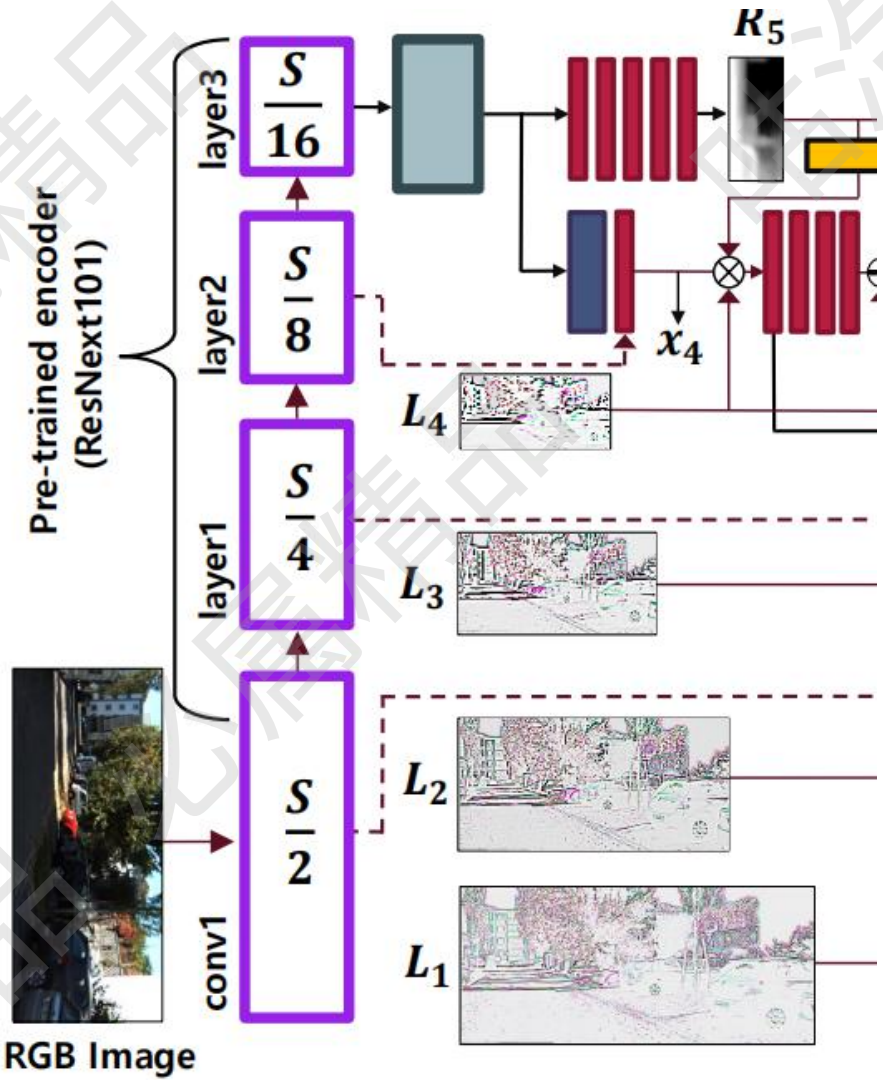
- 空洞卷积是干啥的?
  - 如何才能让感受野更大?
  - 并不用每个特征点都计算，岔开
  - 可以设置空洞倍率来得到不同的特征图



```
self.atrous_block1 = nn.Conv2d(in_channel, depth, 1, 1)
self.atrous_block6 = nn.Conv2d(in_channel, depth, 3, 1, padding=6, dilation=6)
self.atrous_block12 = nn.Conv2d(in_channel, depth, 3, 1, padding=12, dilation=12)
self.atrous_block18 = nn.Conv2d(in_channel, depth, 3, 1, padding=18, dilation=18)
```

深度估计

- 第三招：ASPP
  - atrous spatial pyramid pooling重出江湖
  - 主要集成了SPP的思想和空洞卷积
  - 文中只在backbone最后特征图做了一次
  - 其实就是希望特征多样性能体现出来

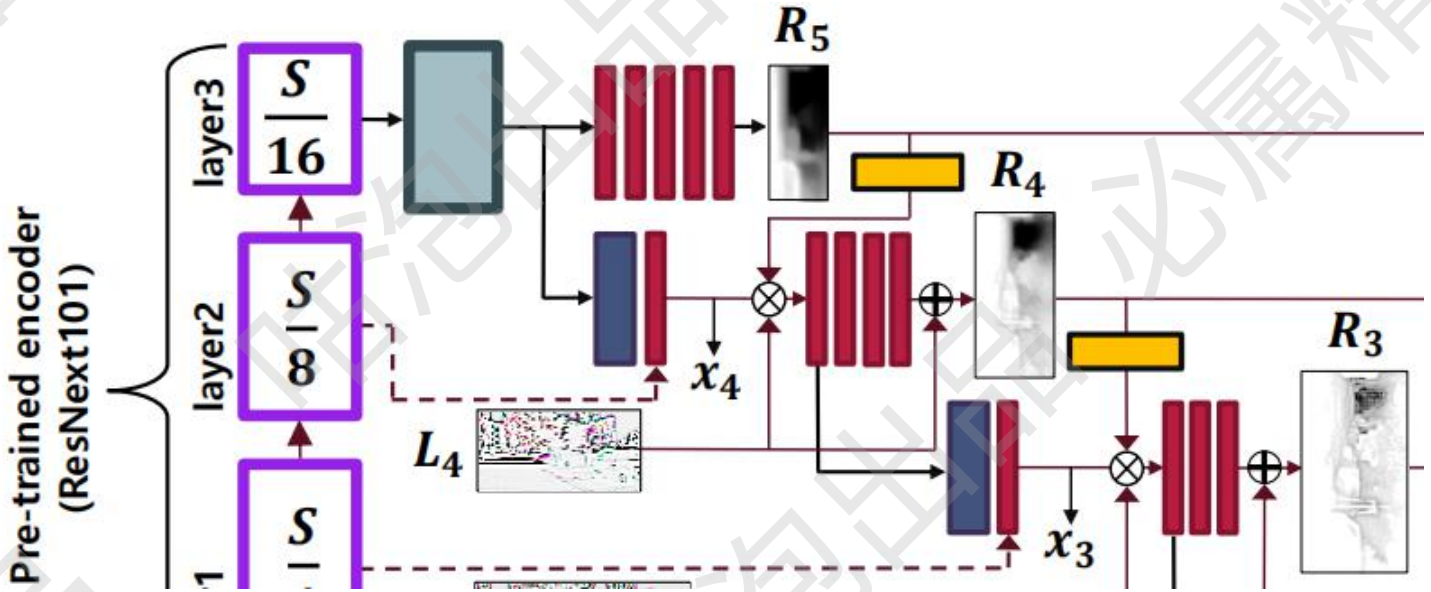


深度估计

- 第四招：一顿叠加

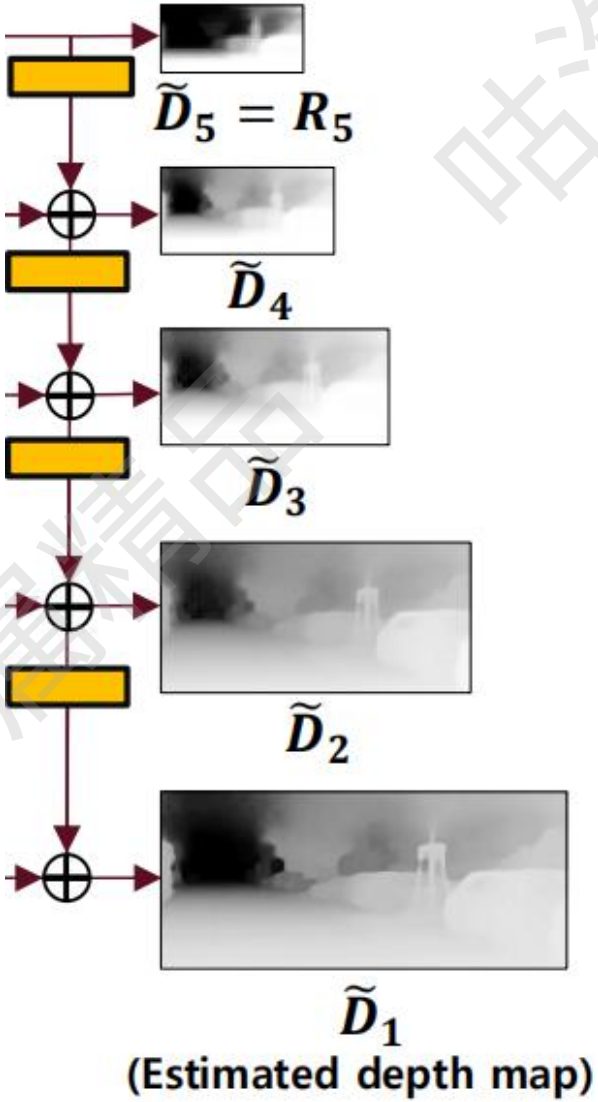
  - X表示拼接（特征图预测结果R5,差异结果L4,中间特征X4拼起来）

  - 特征融合过程中，越来越好，一会还要经过卷积（处对象也得磨合）



深度估计

- 第五招：Coarse-to-Fine
  - 活干的越来越细，先把整体做好，再还原细节
  - 逐层提特征，拼接，再不断加入到下一层的输出中
  - 这个模块挺通用的，很多算法现在都这套路
  - 轮廓，细节会越来越突出，最终还原到实际图像中

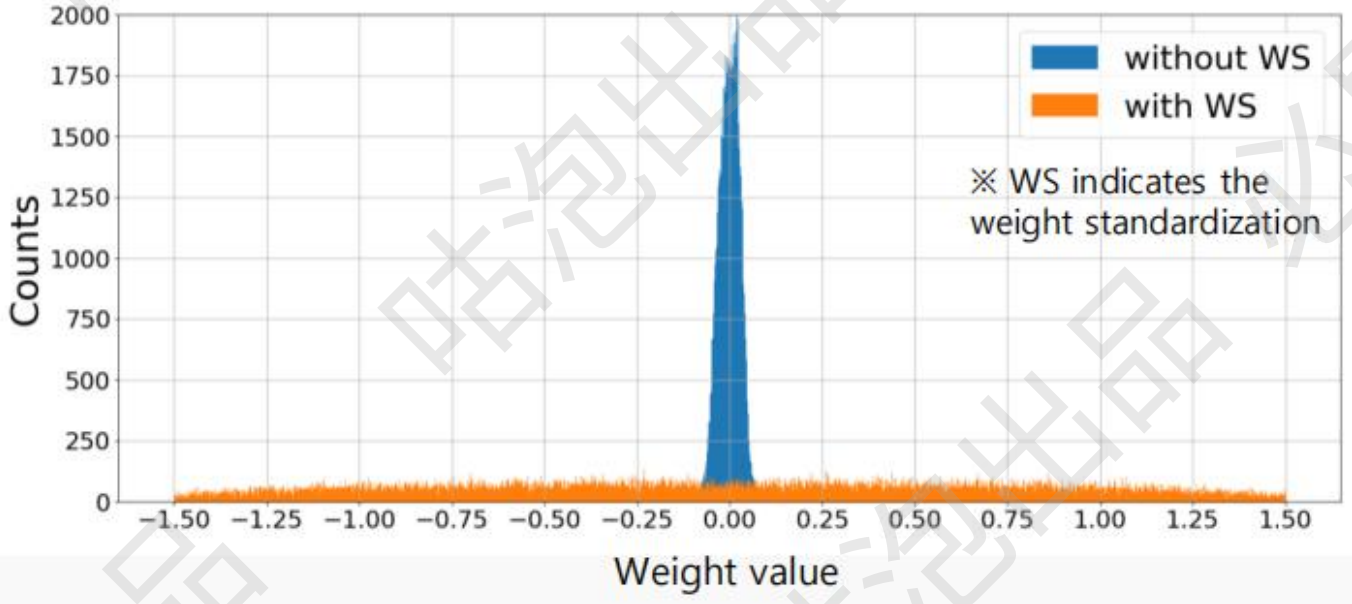


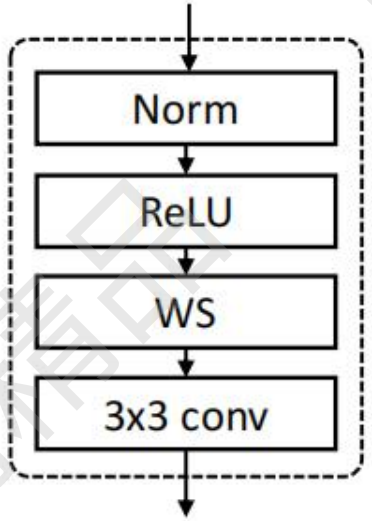
深度估计

- 第六招：权重参数预处理

- 不处理会怎样呢？那经过RELU就死一大片呗！

- 传统套路是CONV+BN+RELU，现在是多一个预处理：

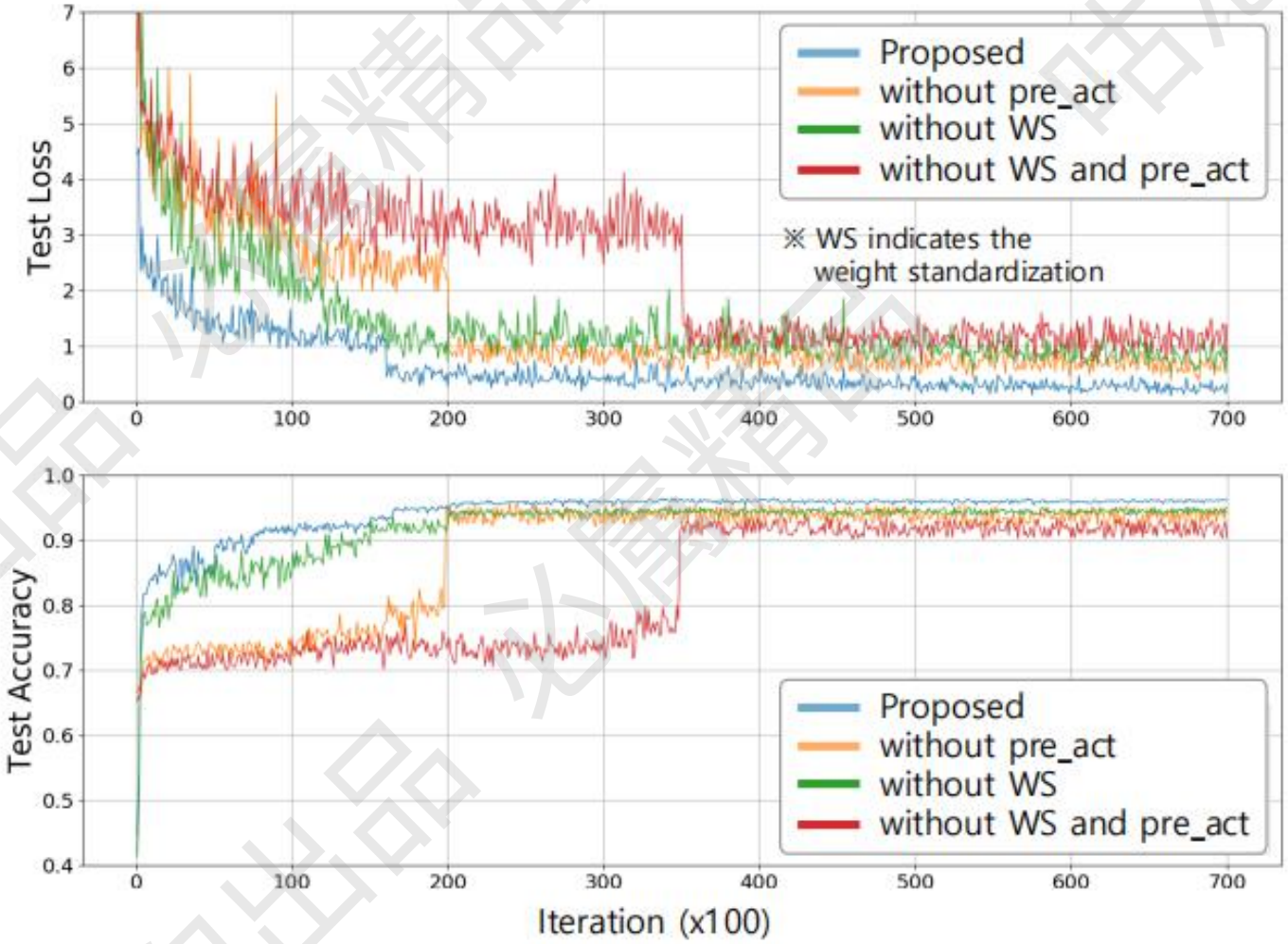




$$W\sigma(x)+b \Rightarrow \sigma(Wx+b)$$

深度估计

- 效果
  - pre-activation感觉简化计算
  - 那这招是不是通用的呢?
  - 还是自己设计的网络没预训练
  - 任务中也可以酌情来进行考虑



深度估计

- 第七招：损失函数

  - 深度估计损失函数：$D(y,y^*) = \frac{1}{2n}\sum_{i=1}^{n}(\log y_i - \log y_i^* + \alpha(y,y^*))^2$

  - 前面还好理解，关键是α这项是要干啥呢？

$$D(y,y^*) = \frac{1}{2n^2}\sum_{i,j}\left((\log y_i - \log y_j) - (\log y_i^* - \log y_j^*)\right)^2$$

$$= \frac{1}{n}\sum_i d_i^2 - \frac{1}{n^2}\sum_{i,j} d_i d_j = \frac{1}{n}\sum_i d_i^2 - \frac{1}{n^2}\left(\sum_i d_i\right)^2$$

  - 化简后相当于正则项了，惩罚这种家伙（大家都没考好，就他考好了）

深度估计

- 效果与总结

  - 网络设计技巧相对通用，粗到细的过程已成经典

  - 1：输入数据；2：GT；3.模型输出结果（KITTI与 NYU数据集）