AVS\_嵌入式立体感知可行性报告V1.0.0

姜传民

浙江省北大信息技术高等研究院

AVS中心AIOT实验室

2021年3月8日

**目 录**

[1. 引言 1](#_Toc66292240)

[1.1 调研目标 1](#_Toc66292241)

[1.2 适用范围 1](#_Toc66292242)

[1.3 相关术语 1](#_Toc66292243)

[1.4 参考资料 1](#_Toc66292244)

[2. 3D目标检测综述 2](#_Toc66292245)

[2.1 基于候选区域方法 2](#_Toc66292246)

[2.2 基于单次检测方法 3](#_Toc66292247)

[3. 候选方案 5](#_Toc66292248)

[4. 方案对比 6](#_Toc66292249)

[4.1 推理速度 6](#_Toc66292250)

[4.2 性能 7](#_Toc66292251)

[4.3 可实现性 7](#_Toc66292252)

[4.3.1 3DSSD[26] 8](#_Toc66292253)

[4.3.2 SECOND[22] 9](#_Toc66292254)

[4.3.3 SA-SSD[24] 11](#_Toc66292255)

[4.3.4 CIA-SSD[25] 12](#_Toc66292256)

[4.3.5 Voxel-RCNN[19] 13](#_Toc66292257)

[4.3.6 PointPillars[23] 14](#_Toc66292258)

[4.3.7 PV-RCNN[15] 15](#_Toc66292259)

[4.4 可扩展空间 16](#_Toc66292260)

[5. 结论 17](#_Toc66292261)

[5.1 结论 17](#_Toc66292262)

[6. 修订记录 18](#_Toc66292263)

[附录1 文献 19](#_Toc66292264)

[附录2 源码 22](#_Toc66292265)

[附录3 算子 23](#_Toc66292266)

[Caffe 23](#_Toc66292267)

[TensorFlow 23](#_Toc66292268)

[ONNX 24](#_Toc66292269)

1. 引言
   1. 调研目标

无人驾驶已经从研究阶段逐渐转化到商用，全球头部公司Waymo, Baidu先后推出无人驾驶的出租车业务。针对当下热潮，调研近几年基于点云的3D目标检测的科研进展，分析不同算法之间的原理、3D点云处理方式，从中选择合理的候选方案，并从推理速度、性能、可实现性和可扩展空间等维度分析候选方案，确定最终方案。

* 1. 适用范围

本文档目标范围是AVS中心AIOT实验室相关算法人员和软件开发人员。

* 1. 相关术语

|  |  |  |
| --- | --- | --- |
| 术语 | 说明 | 备注 |
| 3DSSD | Radar Info PrePost Processing | 雷达信号预处理后处理库 |
| SECOND | Sparsely Embedded  Convolutional Detection | 一种模型的简称 |
| CIA-SSD | Confifident IoU-Aware Single-Stage Object Detector | 一种模型的简称 |
| SA-SSD | Structure aware single-stage 3D object detection | 一种模型的简称 |
| PV-RCNN | Point-voxel feature RCNN | 一种模型的简称 |
| KITTI | 由德国卡尔斯鲁厄理工学院（KIT）和丰田工业大学芝加哥分校（TTIC）共同创立。 | 用于自动驾驶场景下的计算机视觉算法测评数据集 |
| SSFA | Spatial-Semantic Feature Aggregation的缩写 | CIA-SSD中的一个模块，用于处理体素特征 |
| DI-NMS | Distance-Variant IoU-weighted NMS的简称 | 一种对NMS优化的方法 |

* 1. 参考资料

参见附录1，附录2和附录3

1. 3D目标检测综述

3D目标检测将一个场景的点云作为输入，输出针对每个检测目标的3D边界框。3D目标检测的方法大致可以分为两大类：基于候选区域的方法和基于单次检测的方法。

* 1. 基于候选区域方法

基于候选区域的方法首先提出一些包含目标的候选区域，之后提取区域特征来决定每个区域的类别。根据区域生成的方式不同，又可以划分为三类：基于多视角的、基于分割的和基于锥状体的。

基于多视角方法[1] [2] [3] [4] [5] [6] 融合多个视角的映射（激光雷达前视图、鸟瞰图以及图像），获得3D旋转边界框。这类方法的计算量较大。MV3D[1] 从鸟瞰图中生成一组高精度的3D候选框，并将它们投影到多个视图的特征地图上（例如，LiDAR前视图、RGB图像）。然后它们结合了这些来自不同视图的区域级特征，以预测定向的3D边界框。尽管这个方法在交并比(IoU)为0.25，而且只取300个候选区域的条件下，召回率达到99.1%，但是推理速度太慢。后续方法从两个角度进行改进。[2] [3] [4] 从提升融合方法的效率和有效性上进行优化。[5] [6] 从提升输入数据展示的鲁棒性进行研究。

基于分割的方法[7] [8] [9] 首先利用现有分割技术去除大部分背景点，之后再前景点上生成高质量的候选区域，进而解决了计算量。Point RCNN[7] 直接利用分割后的3D点云获取前景点，之后融合语义特征和空间特征，产生高质量的3D边界框。PointRGCN[8] 在Point RCNN[7] 的基础上，利用图卷积网络（Graph Convolution Network）进行3D目标检测。PointPainting[9] 将点云投影到基于图像的分割网络的输出中，并将语义预测分数附加到点上。再将这些绘制的点送入到现存的检测器中。

基于锥状体的方法[10] [11] [12] 通常为每个2D区域生成一个锥状体的候选区域，之后为评估3D边界框，利用PointNet[13] 或者PointNet++[14] 学习每个3D锥状体的点云特征。F-PointNet[10] 首先提出了这类方法。这类方法的表现取决于先验的2D候选区域提取效果。

其他方法如PV-RCNN[15] 利用3D卷积网络和基于PointNet[13] 的集抽象学习点云特征。VoteNet[16] 直接为从点云目标的虚拟中心点投票，并通过聚合投票特性生成一组高质量的3D目标候选区域。ImVoteNet[17] 将2D检测信息融入到3D投票流程中。Part-A2 Net[18] 由两个阶段构成，在部分感知阶段利用稀疏卷积和稀疏反卷积的UNet类网络学习特征得到粗略预测位置，之后在部分聚合阶段融合了交并比感知池化方法优化边界框位置。

* 1. 基于单次检测方法

这类方法直接预测目标的分类置信度和3D目标检测框。它们不需要生成候选区域，因此相对较快。根据输入数据的类型不同，分为三类：基于鸟瞰图、基于离散体和基于点。

基于鸟瞰图的方法将鸟瞰图作为输入，PIXOR[20] 用等间距单元离散场景的点云，并以类似的方式编码反射率，产生规则表示。 然后应用全卷积网络(FCN)来估计物体的位置和航向角。该方法在当时较为突出。

基于离散体的方法通常将点云转化为规则的离散表示，之后利用卷积网络进行预测分类和目标的3D边界框。VoxelNet[21] 将点云划分为等间距的体素，并将每个体素内的特征编码为4D张量。然后连接RPN以产生检测结果。虽然它的性能很强，但由于体素和三维卷积的稀疏性，这种方法非常缓慢。SECOND[22] 利用稀疏卷积网络提高VoxelNet[21] 的推理效率。PointPillars利用PointNet[13] 学习由垂直柱状构成的点云特征，将这些学习的特征编码为伪图。并将一个2D目标检测流程应用于3D边界框。PointPillars[23] 比大多数融合方法的mAP都高。而且PointPillars[23] 在3D和BEV 两种视角的KITTI数据集上能够以62FPS的速度，有较强的实际应用。SA-SSD[24] 利用细粒度结构信息来提升位置的准确度。具体来说，他们首先将点云转换为张量，并将其馈送到骨干网络中，以提取多级特征。此外，还利用具有点等级监督的辅助网络来指导特征学习点云的结构。实验表明该方法当时在KITTI的鸟瞰图检测上在Car类中排名第一。CIA-SSD[25] 在SECOND[22] 的基础上增加了语义-空间特征聚合模块，并优化了NMS。它在KITTI上的表现也非常突出。

基于点的方法直接采用原始的点云作为输入。3DSSD[26] 介绍了一种用于距离-FPS(D-FPS)和特征-FPS(F-FPS)的融合采样策略，以去除PointRCNN[7] 中耗时的特征传播(FP)层和细化模块。 然后，使用候选生成(CG)层充分利用有代表性的点，这些点进一步被馈送到一个带有3D中心标签的无锚定检测头中，以预测3D对象框。实验表明该方法比两阶段基于点的PointRCNN[7] 表现更好，同时维持在25FPS以上。

其他方法，如Point-GNN[27] 提出图神经网络从激光雷达中检测3D目标。它们先将输入点云编码成具有固定半径的近邻图，之后将图送入Point-GNN[27] ，以预测分类和目标框。

1. 候选方案

从近三年端到端的3D目标检测算法中对比7个候选方案，其类别分别是基于点云的3DSSD[26] 、基于体素的SECOND[22] , CIA-SSD[25] , Voxel-RCNN[19] , PointPillars[23] 以及其他方法的SA-SSD[24] 和PV-RCNN[15] 。

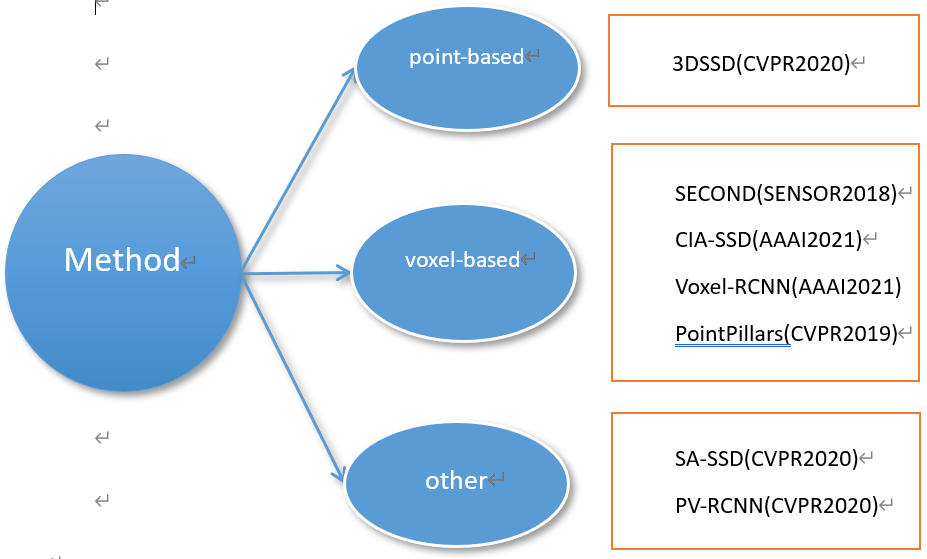
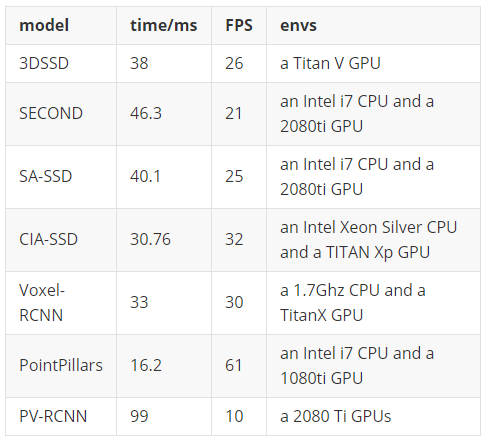


图1 候选方案

1. 方案对比
   1. 推理速度

模型配置环境大同小异，这里将候选方案的运行环境罗列出来，进行对比。

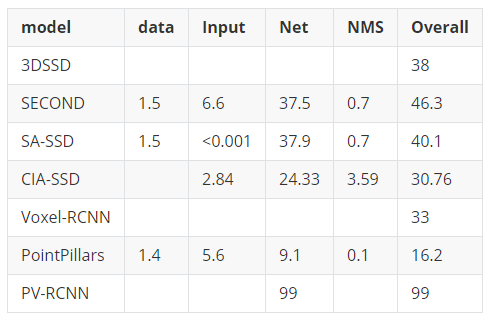
表1 模型环境对比



由表1可知，用于推理的环境配置主要是一块Inter i7处理器和一块2080Ti。

不同模型的推理时间也会由于其模型结构不同而产生差异，后续会详细分析不同算子。针对不同模型的推理时间进行对比，如表2所示。

表2 模型推理时间对比(ms)

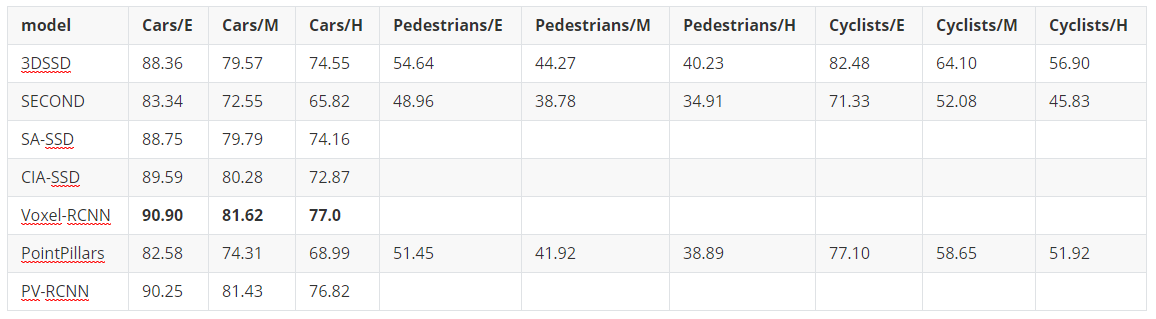


由表2可知：

* PointPillars[23] 的推理速度最快，是其他模型的2~3倍；
* PV-RCNN[15] 的推理速度相对较慢，推理时间是99ms，约是其他方法的2~3倍；
* SECOND[22] , SA-SSD[24] , CIA-SSD[25] , Voxel-RCNN[19] , Pointpillars[23] 五种模型的推理时间都在50ms以内，相差不大；
* 模型推理过程中，在CPU上数据的加载和前处理，以及数据从CPU转移到GPU上的过程，耗时较少，在GPU上的运算时间占用较大比例，NMS等后处理过程的耗时与是否采用多锚定、多检测头强相关。
  1. 性能

本报告针对数据集KITTI进行分析，KITTI数据集根据遮挡情况将目标分为easy, medium,hard三个等级。具体检测结果如表格3所示。

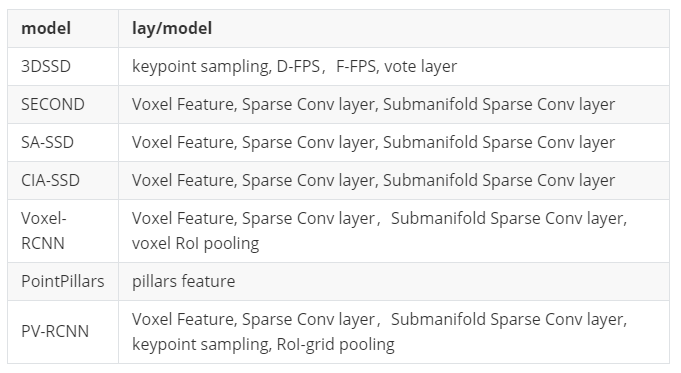
表3 不同模型的AP(%)



由表3可知，在候选方案中，Voxel-RCNN[19] 和PV-RCNN[15] 在汽车上的AP表现比较突出，之后的CIA-SSD[25] , SA-SSD[24] , 3DSSD[26] 相对比较接近，PointPillars[23] 和SECOND[22] 相对较弱一些。

* 1. 可实现性

表4 不同模型自定义算子或前后处理



各个模型在部署上都会有涉及自定义算子或者前后处理的部分，具体如表4所示。

总的来说，PointPillars的复现和后续的部署难度最小。其次是SECOND[22] 和CIA-SSD[25] ，3DSSD[26] 和PV-RCNN[15] 的难度相对较大。下面对各个网络的算法部署进行详细分析。

* + 1. 3DSSD[26]

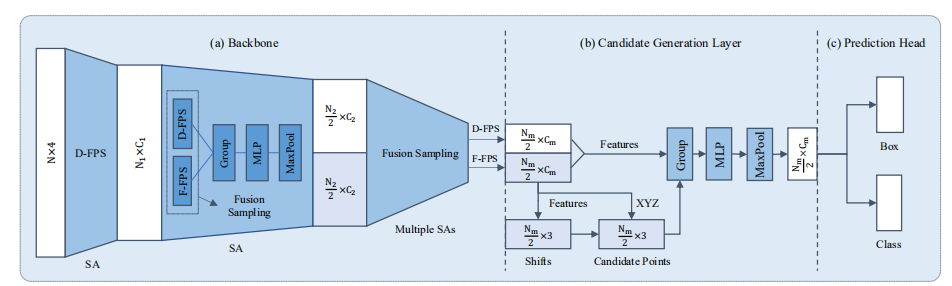


图1 3DSSD结构图

由图1可知，3DSSD[26] 由Backbone、Candidate Generation Layer和Prediction Head三部分构成。其中Backbone是由两个Set Abstraction Layer构成，Candidate Generation Layer是由一个Vote Layer和一个Set Abstraction Layer构成。Prediction Head部分是由卷积层构成。接下来通过代码详细分析下Set Abstraction Layer和Vote Layer。

Set Abstraction Layer操作如代码1所示，有三种方法：FS，F-FPS和D-FPS。其中FS是F-FPS和D-FPS的融合，D-FPS是PointNet中的方法，其采用的是利用最远点采样法，F-FPS是作者针对前者的不足而提出的计算特征距离的方法。

代码1：Set Abstraction Layer中的FS，F-FPS和D-FPS的实现方法A



Vote Layer如代码2所示，首先对点进行卷积，提取特征，再平移，最后获取中心点。

代码2：Vote Layer A



Set Abstraction Layer和Vote layer算子在部署到NX平台都需要自行构建算子。

* + 1. SECOND[22]

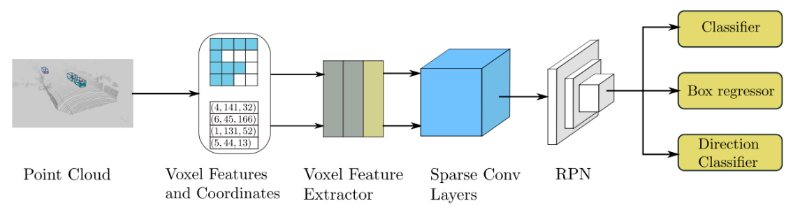


图2 SECOND结构图

SECOND[22] 主要体素特征提取、稀疏卷积层和RPN由三部分构成。下面利用代码详述描述体素特征提取和稀疏卷积两个部分。

代码3：体素特征提取B



体素特征提取层如代码3所示。先是线性相乘，之后再归一化和激活函数，再沿着特征方向提取最大值，最后合并。整个过程可以在NX上实现。

稀疏卷积层的实现方式如代码4所示。其对激活状态的点和非激活状态点的处理状态不一样。其实现方式在NX上无法直接实现，需要自定义算子。

代码4：稀疏卷积层B



* + 1. SA-SSD[24]

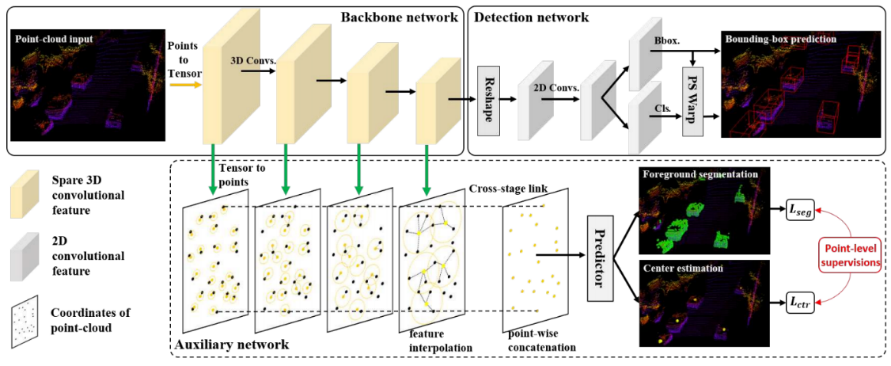


图3 SA-SSD结构图

SA-SSD[24] 的结构如图3所示，由Backbone Network, Auxiliary Network和Detection Network三部分构成。在训练过程中，三个部分都参与训练，在推理过程中会去除Auxiliary Network。因此考虑其可行性时，只考虑Backbone Network和Detection Network。

文中转化为张量的方法是基于体素或基于柱状体的。这部分已经在SECOND[22] 和PointPillars[23] 中讨论过了。除此之外，文章在后处理汇总添加了part-sensivite warping操作，用于缓解预测框和对应分类不一致现象。后处理时需要考虑。

* + 1. CIA-SSD[25]

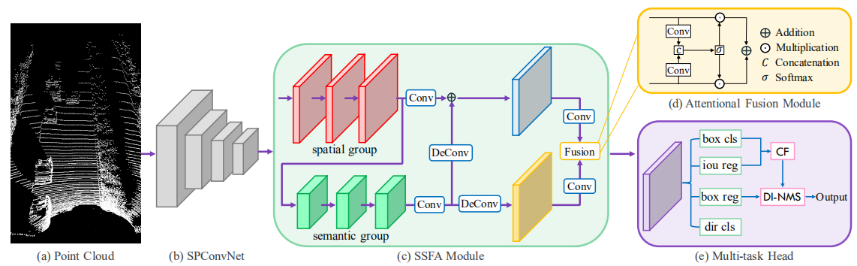


图4 CIA-SSD结构图

CIA-SSD[25] 的结构如图4所示，由体素特征提取和稀疏卷积块，SSFA模块和多任务检测头三部分构成。其中体素特征和稀疏卷积网络已经在上述SECOND[22] 中讨论了。接下来谈论SSFA模块和多任务检测头这两部分。

Spatial-Semantic Feature Aggregation

* 由两组卷积层和一个注意力融合模块构成。两组卷积层分别称为空间组合语义组，输出的特征分别是空间特征和语义特征。
* 空间特征的尺寸和输入保持一致，避免空间特征信息的损失
* 将空间特征作为输入，首先用卷积，长宽下降一倍，通道增加一倍。之后再利用Decon使得其恢复与空间特征一致的尺寸，便于求和，同时利用另一个Decon恢复其原有尺寸，便于之后的注意力融合。
* 融合的过程如图4中的(d)所示，首先将每个特征的通道压缩成一个，并合并结果。之后利用softmax函数对正则化两个合并后的结果，再分离成两个BEV注意力映射。最后我们将上述BEV注意力映射作为各个特征的权重，并对加权后的特征求和。进而完成了语义特征和空间特征的融合。

在多任务检测头中，对后处理的NMS进行优化，提出了DI-NMS。这里不是直接使用IoU进行对比，而是计算预测边框和锚定之间的距离作为参数计算的距离相关参数作为对比的指标，如图5中step1和step2。进而对分类置信度进行了优化。并利用高斯公式优化了边界框，使其更加归一化，如图5中step5-step8。这部分在NX上的推理过程，也需要随之优化和改进。

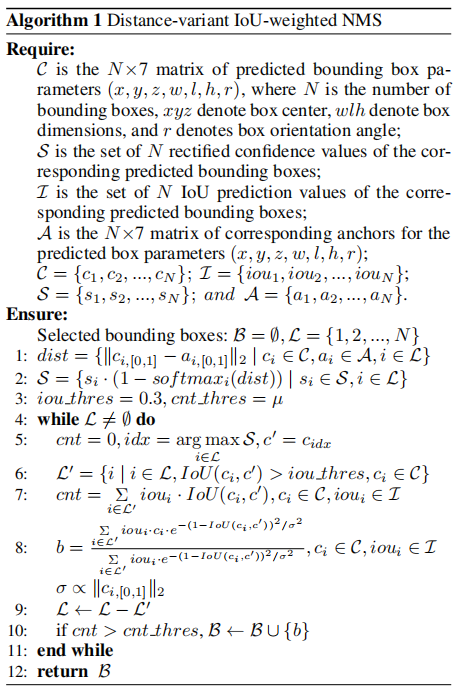


图5 DI-NMS

* + 1. Voxel-RCNN[19]

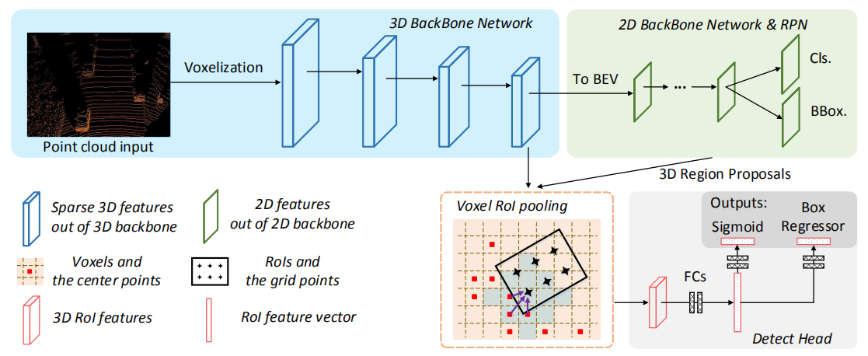


图6 Voxel-RCNN结构

Voxel-RCNN[19] 网络结构如图6所示，由3D Backbone Network, 2D Backbone Network & RPN, Voxel RoI Pooling和Detect Head四个部分构成。其中3D Backebone Network在 SECOND[22] 中已经讨论过了，2D Backbone Network & RPN和Detect Head比较熟悉。详细讨论下Voxel RoI pooling的实现方式。

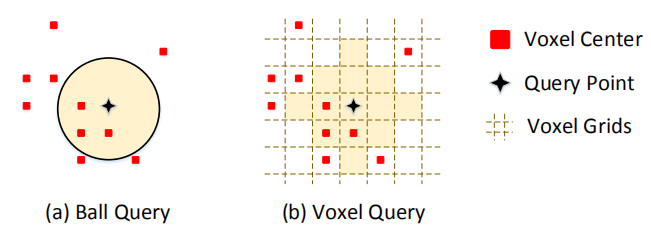


图7 voxel query

如图7所示，首先query point量化为一个体素，之后通过索引平移能够高效的得到临近的体素。我们在voxel query中利用曼哈顿距离，并且在一个距离阈值上采样最多K个voxels。

我们将近邻体素的特征集成到网格点进行特征提取。具体来说，给定一个网格点，我们首先利用voxel query来分组一系列的近邻体素。之后我们利用Point net模块聚合近邻体素特征。具体来说就是计算每个计算体素块中心到Query Point之间的距离，在对齐使用全连接，得到聚合后的临近体素特征。

这部分后处理工作在NX平台也是需要在之前的RoI Pooling的基础上进行优化和改进。

* + 1. PointPillars[23]

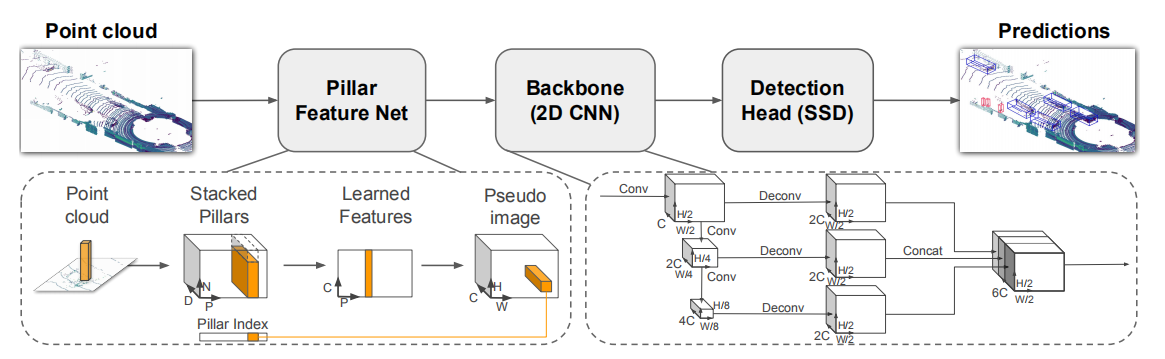


图8 PointPillars结构图

PointPillars[23] 结构相对简单，如图8所示，由Pillar Feature Net, 2D Backbone, Detection Head三个部分构成。其中Backbone是由2D CNN构成，Detection Head采用的是SSD的检测头。Pillar Feature Net类似于Voxel表示。只是将z轴方向转化为无限高。

* + 1. PV-RCNN[15]

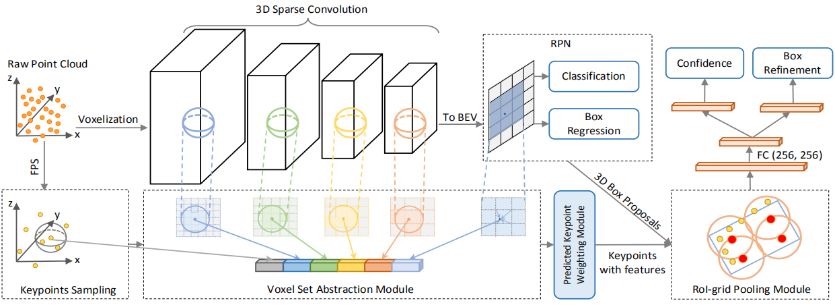


图9 PV-RCNN结构图

PV-RCNN[15] 的结构如图9所示。结构由基于体素和基于点云两个分支构成。其中基于体素的分支由Voxelization, 3D Sparse Conv, RPN构成，基于点云分支由Keypoints Sampling, Voxel Set Abstraction Models，Predicted Keypoint Weight Models构成，最后两个分支合并进行RoI-grid Pooling Module和Detection Head构成。其中Voxelization, 3D 3D Sparse Conv和RPN在SECOND[22] 中有所讨论。基于点云的分支中，Keypoints Sampling采用的是最远点采样法(Furthest-Point-Sampling algorithm)。下面详细讨论下Voxel Set Abstraction Models，Predicted Keypoint Weight Models和RoI-grid Pooling Module三个部分的实现方式。

Voxel Set Abstraction Models针对不同层级的体素特征以及采样关键点和BEV特征进行以下两步操作：

* 首先获取某个半径以内的体素集；
* 之后再对上述体素集随机采样，利用多层感知机对采样后的体素集进行编码，最后进行最大池化；

Predicted Keypoint Weight Models结构如图10所示，其中Keypoint Features经过三层全连接和Sigmoid激活函数后，作为参数与原有的Keypoint Features相乘。

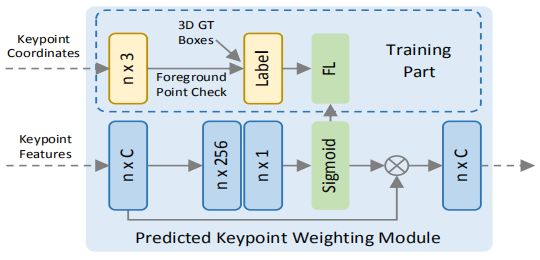


图10 Predicted Keypoint Weight Models

RoI-grid Pooling Module的结构如图11所示，类似于Voxel Set Abstraction Models。

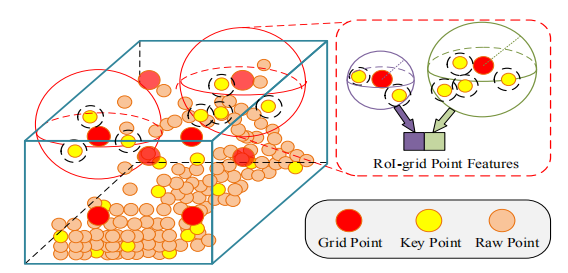


图10 RoI-grid Pooling Module

* 1. 可扩展空间
* 从图8和图2中可以看出，PointPillars[23] 和SECOND[22] 结构相对简单，扩展空间比较大。但是PointPillars[23] 采用的pillar-based，是对体素的改进，从三维降成二维，但是基于该方法的改进和优化较少，后续切换不同方法时还需要考虑重新构建算子，SECOND[22] 是基于体素的模型。 如果基于这个方案，在之后的改进中，可以直接基于体素的基础上进一步优化和改进。
* 3DSSD[26] , SA-SSD[24] , CIA-SSD[25] , Voxel-RCNN[19] , PV-RCNN[15] 在前两者的基础上进行改编和优化，相对复杂。
* 3DSSD[26] 是建立在point-based的基础上，近两年除了pointnet++奠定了point-based的方法基础外，很少看到这类方法，从工程的角度来说，PointNet系列使用的是SA和FP层，在部署上会有较大难度，不建议使用该方法。
* PV-RCNN[15] 是point-based和pillar-based的结合，后续优化空间很大，因为推理时间较长，从各个阶段的推理时间进行分析和总结，进而优化推理时间。
* SA-SSD[24] 中的辅助网络进行优化也是一个很好的思路。

总结：

从可扩展性分析得出，建议使用目前主流的模型框架，即Voxel-based方法，便于后期的优化和改进。

1. 结论
   1. 结论

从上述分析可知，PV-RCNN[15] 速度较慢，不满足实时性。3D-SSD[26] 在部署时考虑较多新的算子。SA-SSD[24] , Voxel-RCNN[19] 也需要在SECOND[22] 的基础上增加算子。目前PointPillers[23] 最为简单，但是不便于后续的voxel-based扩展。结合目前的技术积累和模型部署难度，得到以下结论。

综合考虑，目前推荐方法如下：

PointPillars[23]

1. 修订记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 修改日期 | 修改人 | 修改内容 | 备注 |
| 1.0.0 | 2021-03-08 | 姜传民 | 新建 |  |
| 1.1.0 | 2021-03-10 | 任亦立 | 修改格式，新建批注 |  |
| 1.2.0 |  |  |  |  |

附录1 文献

1. X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3D object detection network for autonomous driving,” in CVPR, 2017.
2. J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, “Joint 3D proposal generation and object detection from view aggregation,” in IROS, 2018.
3. M. Liang, B. Yang, S. Wang, and R. Urtasun, “Deep continuous fusion for multi-sensor 3D object detection,” in ECCV, 2018.
4. M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, “Multi-task multi-sensor fusion for 3D object detection,” in CVPR, 2019.
5. H. Lu, X. Chen, G. Zhang, Q. Zhou, Y. Ma, and Y. Zhao, “SCANet: Spatial-channel attention network for 3D object detection,” in ICASSP, 2019.
6. Y. Zeng, Y. Hu, S. Liu, J. Ye, Y. Han, X. Li, and N. Sun,“RT3D: Real-time 3D vehicle detection in lidar point cloud for autonomous driving,” IEEE RAL, 2018.
7. S. Shi, X. Wang, and H. Li, “PointRCNN: 3D object proposal generation and detection from point cloud,” in CVPR, 2019.
8. Z. Jesus, G. Silvio, and G. Bernard, “PointRGCN: Graph convolution networks for 3D vehicles detection refinement,” arXivpreprint arXiv:1911.12236, 2019.
9. V. Sourabh, L. Alex H., H. Bassam, and B. Oscar, “PointPainting: Sequential fusion for 3D object detection,” in CVPR, 2020.
10. C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, “Frustum PointNets for 3D object detection from RGB-D data,” in CVPR, 2018.
11. X. Zhao, Z. Liu, R. Hu, and K. Huang, “3D object detection using scale invariant and feature reweighting networks,” in AAAI, 2019.
12. D. Xu, D. Anguelov, and A. Jain, “PointFusion: Deep sensor fusion for 3D bounding box estimation,” in CVPR, 2018.
13. C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in CVPR, 2017.
14. C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” in NeurIPS, 2017.
15. S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, “PV-RCNN: Point-voxel feature set abstraction for 3D objectdetection,” in CVPR, 2020.
16. C. R. Qi, O. Litany, K. He, and L. J. Guibas, “Deep hough voting for 3D object detection in point clouds,” ICCV, 2019.
17. C. R. Qi, X. Chen, O. Litany, and L. J. Guibas, “ImVoteNet: Boosting 3D object detection in point clouds with image votes,”in CVPR, 2020.
18. S. Shi, Z. Wang, X. Wang, and H. Li, “From points to parts: 3D object detection from point cloud with part-aware and part-aggregation network,” TPAMI, 2020.
19. J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, H. Li, "Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection," in AAAI, 2021.
20. B. Yang, W. Luo, and R. Urtasun, “PIXOR: Real-time 3D object detection from point clouds,” in CVPR, 2018.
21. Y. Zhou and O. Tuzel, “VoxelNet: End-to-end learning for point cloud based 3D object detection,” in CVPR, 2018.
22. Y. Yan, Y. Mao, and B. Li, “SECOND: Sparsely embedded convolutional detection,” Sensors, 2018.
23. A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “PointPillars: Fast encoders for object detection from point clouds,” in CVPR, 2019.
24. C. He, H. Zeng, J. Huang, X.-S. Hua, and L. Zhang, “Structure aware single-stage 3D object detection from point cloud,” in CVPR, 2020.
25. W. Zheng, W. Tang, S. Chen, L. Jiang, C. Fu, "CIA-SSD: Confident IoU-Aware Single-Stage Object Detector From Point Cloud," in AAAI, 2021.
26. Z. Yang, Y. Sun, S. Liu, and J. Jia, “3DSSD: Point-based 3D single stage object detector,” in CVPR, 2020.
27. W. Shi and R. Rajkumar, “Point-GNN: Graph neural network for 3D object detection in a point cloud,” in CVPR, 2020.

附录2 源码

1. https:// github.com /Jia-Research-Lab/3DSSD
2. https:// github.com /traveller59/second.pytorch
3. https:// github.com /skyhehe123/SA-SSD
4. https:// github.com /Vegeta2020/CIA-SSD
5. https:// github.com /djiajunustc/Voxel-R-CNN
6. https://github.com/nutonomy/second.pytorch?utm\_source=catalyzex.com
7. <https://github.com/open-mmlab/OpenPCDet?utm_source=catalyzex>

附录3 算子

Caffe

These are the operations that are supported in a Caffe framework:

* BatchNormalization
* BNLL
* Clip
* Concatenation
* Convolution
* Crop
* Deconvolution
* Dropout
* ElementWise
* ELU
* InnerProduct
* Input
* LeakyReLU
* LRN
* Permute
* Pooling
* Power
* Reduction
* ReLU, TanH, and Sigmoid
* Reshape
* SoftMax
* Scale

TensorFlow

These are the operations that are supported in a TensorFlow framework:

* Add, AddV2, AddN, Sub, Mul, Div, FloorDiv, RealDiv, Minimum, Maximum
* AvgPool, AvgPool3D[13](https://docs.nvidia.com/deeplearning/tensorrt/support-matrix/index.html" \l "fntarg_13)
* ArgMin
* AvgPool
* BiasAdd
* Cast13
* Clip
* CombinedNonMaxSuppression
* ConcatV2
* Const
* Conv2D, Conv3D
* Conv2DBackpropInput, Conv3DBackpropInputV2
* ConvTranspose2D
* DepthToSpace
* DepthwiseConv2dNative
* Elu
* ExpandDims
* FusedBatchNorm, FusedBatchNormV2, FusedBatchNormV3
* FusedConv2DBiasActivation
* GatherV2
* Identity
* LeakyReLU
* MatMul, BatchMatMul, BatchMatMulV2
* MaxPool, MaxPool3D13
* Mean
* Negative, Abs, Sqrt, Recip, Rsqrt, Pow, Exp, Log, Square
* Pad is supported if followed by one of these TensorFlow layers: Conv2D, DepthwiseConv2dNative, MaxPool, and AvgPool.
* Pack, Unpack
* ReLU, TanH, Sigmoid
* Relu6
* Reshape
* ResizeBilinear, ResizeNearestNeighbor
* Sin, Cos, Tan, Asin, Acos, Atan, Sinh, Cosh, Asinh, Acosh, Atanh, Ceil, Floor
* Selu
* Shape13
* Slice, StridedSlice
* SoftMax

**Note:** If the input to a TensorFlow SoftMax op is not NHWC, TensorFlow will automatically insert a transpose layer with a non-constant permutation, causing the UFF converter to fail. It is therefore advisable to manually transpose SoftMax inputs to NHWC using a constant permutation.

* Softplus
* Softsign
* SpaceToDepth
* Split
* SquaredDifference
* Squeeze
* TopKV2
* Transpose

ONNX

These are the operations that are supported in the ONNX framework:

* Abs
* Acos
* Acosh
* And
* Asin
* Asinh
* Atan
* Atanh
* Add
* ArgMax
* ArgMin
* AveragePool
* BatchNormalization
* Cast
* Ceil
* Clip
* Concat
* Constant
* ConstantOfShape
* Conv
* ConvTranspose
* Cos
* Cosh
* DepthToSpace
* DequantizeLinear
* Div
* Dropout
* Elu
* Equal
* Erf
* Exp
* Expand
* Flatten
* Floor
* Gather
* Gemm
* GlobalAveragePool
* GlobalMaxPool
* Greater
* GRU
* HardSigmoid
* Identity
* ImageScaler
* InstanceNormalization
* LRN
* LeakyRelU
* Less
* Log
* LogSoftmax
* Loop
* LRN
* LSTM
* MatMul
* Max
* MaxPool
* Mean
* Min
* Mul
* Neg
* Not
* Or
* Pad
* ParametricSoftplus
* Pow
* PRelu
* QuantizeLinear
* RandomUniform
* RandomUniformLike
* Range
* Reciprocal
* ReduceL1
* ReduceL2
* ReduceLogSum
* ReduceLogSumExp
* ReduceMax
* ReduceMean
* ReduceMin
* ReduceProd
* ReduceSum
* ReduceSumSquare
* Relu
* Reshape
* Resize
* RNN
* ScaledTanh
* Scan
* Selu
* Shape
* Sigmoid
* Sin
* Sinh
* Size
* Slice
* Softmax
* Softplus
* Softsign
* SpaceToDepth
* Split
* Sqrt
* Squeeze
* Sub
* Sum
* Tan
* Tanh
* ThresholdedRelu
* Tile
* TopK
* Transpose
* Unsqueeze
* Upsample
* Where

https://docs.nvidia.com/deeplearning/tensorrt/support-matrix/index.html