一、 论文分类

1. point cloud
   - ✧ PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation (CVPR2017) <13297>
     网络结构简单（仅含 mlp 和最大池化），点云数据作为输入，完成物体分类，部分分割，场景语义分析等任务
   - ✧ PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space (ICRA2017) <9275>
   - ✧ PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud (CVPR2018) <2044>
   - ✧ PointDistiller: Structured Knowledge Distillation Towards Efficient and Compact 3D Detection (CVPR2022) <18>
     提出了针对基于点云和基于体素模型的知识蒸馏方法，提出了局部蒸馏，以提取局部几何信息，提出重新加权蒸馏策略，蒸馏过程注重更为关键的点
2. Voxel
   - ✧ VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection (CVPR2018) <3415>
3. Point-Voxel (One-Stage)
   - ✧ Point-Voxel CNN for Efficient 3D Deep Learning (Neurips2019) <557>
   - ✧ Structure Aware Single-stage 3D Object Detection from Point Cloud (CVPR2020) <475>
   - ✧ Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution (ECCV2020) <445>
4. Point-Voxel (Two-Stage)
   - ✧ LiDAR R-CNN: An Efficient and Universal 3D Object Detector (CVPR2021) <166>
   - ✧ Pyramid R-CNN: Towards Better Performance and Adaptability for 3D Object Detection (ICCV2021) <122>
   - ✧ Improving 3D Object Detection with Channel-wise Transformer (ICCV2021) <169>
   - ✧ PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection (CVPR2022) <1404>
5. Pillar
   - ✧ PointPillars: Fast Encoders for Object Detection from Point Clouds (CVPR2019) <2655>
   - ✧ Pillar-based object detection for autonomous driving (ECCV2020) <174>
   - ✧ Embracing Single Stride 3D Object Detector with Sparse Transformer (CVPR2022) <134>
6. BEV
   - ✧ Multi-View 3D Object Detection Network for Autonomous Driving (CVPR2017) <2950>
   - ✧ Hdnet: Exploiting hd maps for 3d object detection (PMLR2018) <353>

- ✧ PIXOR: Real-time 3D Object Detection from Point Clouds (CVPR2018) <1154>
- ✧ BirdNet: a 3D Object Detection Framework from LiDAR Information (ITSC2018) <318>
- ✧ YOLO3D: End-to-end real-time 3D Oriented Object Bounding Box Detection from LiDAR Point Cloud (ECCV2018) <158>
- ✧ Complex-YOLO: An Euler-Region-Proposal for Real-time 3D Object Detection on Point Clouds (ECCV2018) <362>
- ✧ Lift, Splat, Shoot: Encoding Images from Arbitrary Camera Rigs by Implicitly Unprojecting to 3D (ECCV2020) <445>
  提出了基于显式深度估计将环视感知转换到 BEV 空间的方案，将多个传感器感知融合后再进行检测，省去了后处理步骤
- ✧ BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers (ECCV2022) <413>
  提出了 SCA 和 TSA，其中 SCA 运用了 Deformable Attention 的方法，减小了 attention 操作的计算开销，TSA 在计算 t 时刻 BEV 特征时使用了 t-1 时刻的 BEV 特征，更好地利用了时间信息
- ✧ BEVDepth: Acquisition of Reliable Depth for Multi-View 3D Object Detectionpoint cloud (AAAI2023) <193>
  训练阶段利用点云数据对预测的深度进行显式监督，能够获得更为准确的预测深度，进而获得更为准确的 BEV 特征，提升检测性能，使用点云数据增强 camera-only BEV 方法
- ✧ Temporal Enhanced Training of Multi-view 3D Object Detector via Historical Object Prediction (ICCV2023) <3>
  提出了一种即插即用的 BEV 特征增强方法，可以很容易地集成到经典的 BEV 检测框架中去，同时仅在训练阶段进行 HoP 操作，不会增加推理成本
- ✧ DistillBEV: Boosting Multi-Camera 3D Object Detection with Cross-Modal Knowledge Distillation (ICCV2023) <0>
  3d 检测的知识蒸馏，用 lidar 或 fusion 模型的 BEV 特征监督 camera-only 模型的 BEV 特征，以增强 camera-only 模型的性能
- ✧ SparseBEV: High-Performance Sparse 3D Object Detection from Multi-Camera Videos (ICCV2023) <4>
- ✧ Sparse4D: Multi-view 3D Object Detection with Sparse Spatial-Temporal Fusion (2023) <21>

7. Fusion
   - ✧ BEVFusion: A Simple and Robust LiDAR-Camera Fusion Framework (NeurIPS2022) <101>
   - ✧ FUTR3D: A Unified Sensor Fusion Framework for 3D Detection (CVPR2023) <102>
     首次提出端到端的传感器融合框架，可用于几乎任何传感器配置，nuScenes 数据集
   - ✧ SupFusion: Supervised LiDAR-Camera Fusion for 3D Object Detection (ICCV2023) <1>

- ✧ BEVFusion4D: Learning LiDAR-Camera Fusion Under Bird's-Eye-View via Cross-Modality Guidance and Temporal Aggregation (2023) <5>

8. 2d detector to 3d detector
   - ✧ Object as Query: Lifting any 2D Object Detector to 3D Detection (ICCV2023) <1>
9. Detr
   - ✧ End-to-End Object Detection with Transformers (ECCV2020) <8681>
   - ✧ PETR: Position Embedding Transformation for Multi-View 3D Object Detection (ECCV2022) <218>
   - ✧ DETR3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries (PMLR2022) <321>

     将 Transformer 引入 3d 检测任务，一次性生成 N 个 bbox，提出了 set-to-set 损失函数计算 bbox 和 ground truth 之间损失
   - ✧ Cascade-DETR: Delving into High-Quality Universal Object Detection (ICCV2023) <0>
   - ✧ Exploring Object-Centric Temporal Modeling for Efficient Multi-View 3D Object Detection (ICCV2023) <18>
10. Others
    - ✧ AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE (ICLR2021)

PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation

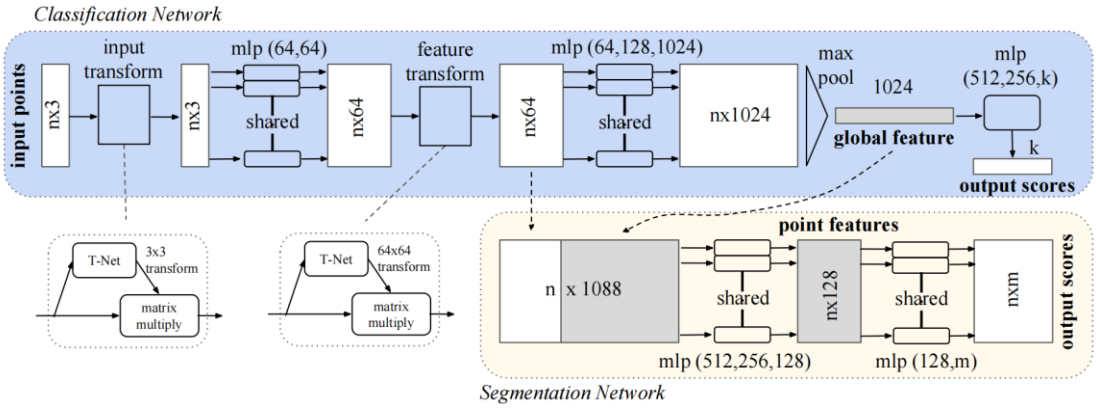
一、 本文贡献

1. 提出了能够直接使用点云数据的 PointNet
2. 能够实现物体分类、部分分割和场景语义理解等功能

二、 研究动机

1. 点云数据格式不规范，因此通常需要先将点云数据转换为体素，然后再输入到网络中，但这么做会带来额外的计算开销，同时会引起量化问题
2. 点云具有无序性，以及刚体运动的其他不变性（如平移旋转），因此网络应满足输入顺序不同，点云发生平移或旋转这些情况下，网络输出不变

三、 方法



- 分类网络

1. 网络以无序点集作为输入，每个 3D 点表示为(x, y, z)，输入形状为$(n, 3)$
2. 经过输入变换，即乘一个3×3矩阵，形状仍为$(n, 3)$
3. 将变换后的点集输入到 mlp(64,64)网络中，隐藏层维度为 64，输出层维度为 64，此时特征的形状变为$(n, 64)$
4. 将特征输入进行特征变换，即乘一个64×64矩阵，得到形状为(64,64)的矩阵
5. 将变换后的特征输入到 mlp(64,128,1024)网络中，含两个隐藏层，第一个隐藏层维度为 64，第二个隐藏层维度为 128，输出层维度为 1024
6. 对特征进行最大池化，得到形状为1024的全局特征
7. 将全局特征输入到 mlp(512,256,k)网络中，将输出（形状为$(1, k)$）作为 k 种物体的输出分数，选择输出分数最高的作为最终分类结果

- 分割网络

1. 将全局特征(1,1024)维度扩展后$(n, 1024)$与变换后的特征$(n, 64)$拼接，得到具有全局信息的点云特征
2. 将具有全局信息的点云特征输入到 mlp(512,256,128)，对点云信息进行降维，方便后续分割
3. 将降维后的点云信息输入到 mlp(128,m)，将输出（形状为(n, m)），作为每个

点的 m 类分数，对任意一个点，选择最高的分数作为该点的类别

四、 设计原因

1. 为什么使用了最大池化？

   为了使网络的输出不受输入顺序的影响，有三种方案，分别是（1）将输入进行排序（2）将输入作为序列训练 RNN，将输入进行各种排列以增强数据（3）使用对称函数对各个点的信息进行聚合，聚合函数指以 n 个向量作为输入，输出一个与输入顺序无关的向量

   对于方案（1），在高维空间难以找到一种稳定的排序算法（主要是点扰动引起的）；对于方案（2），RNN 在输入序列长度较长时不鲁棒

   因此选择方案（3），以点集为输入的一般函数表示为$f(\{x_1, x_2, \dots, x_n\})$，用 $g(h(x_1), h(x_2), \dots, h(x_n))$ 对其进行近似，其中 $f: 2^{\mathbb{R}^N} \to \mathbb{R}, h: \mathbb{R}^N \to \mathbb{R}^K, g: \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n}$，本文的 mlp 相当于$h$，最大池化相当于$g$

五、 训练 trick

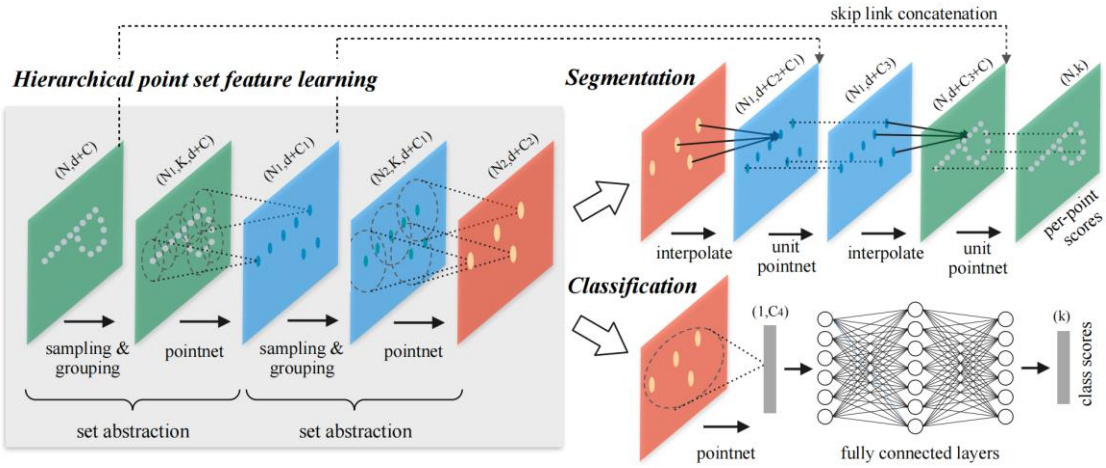
1. 激活函数选择 ReLU，分类网络的最后一个 mlp 使用 dropout 以防止过拟合
2. 特征变换网络形状为$(64,64)$，训练过程很难收敛，通过增加正则化项 $L_{reg} = \| I - AA^T \|_F^2$ 以加快收敛速度

PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric (PointNet++)

一、 本文贡献

1. 引入分层的神经网络，递归地将 PointNet 用于对点集的嵌套划分
2. 通过利用空间距离，网络能学习到局部特征
3. 提出集学习层以自适应结合不同尺度的特征

二、 方法



PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud (PointRCNN)

一、 本文贡献

二、 研究动机

三、 方法

PointDistiller: Structured Knowledge Distillation Towards Efficient and Compact 3D Detection (PointDistiller)

一、 本文贡献

1. 提出了一种结构化的知识蒸馏框架，能够用于基于点云的知识蒸馏
2. 提出了局部蒸馏方法，首先利用动态图卷积对点云的局部几何结构进行编码，然后将其提取给学生
3. 提出重新加权策略处理点云的稀疏性及噪声，突出学生对多点体素的学习，在知识蒸馏中给予多点体素更高的学习权重

二、 研究动机

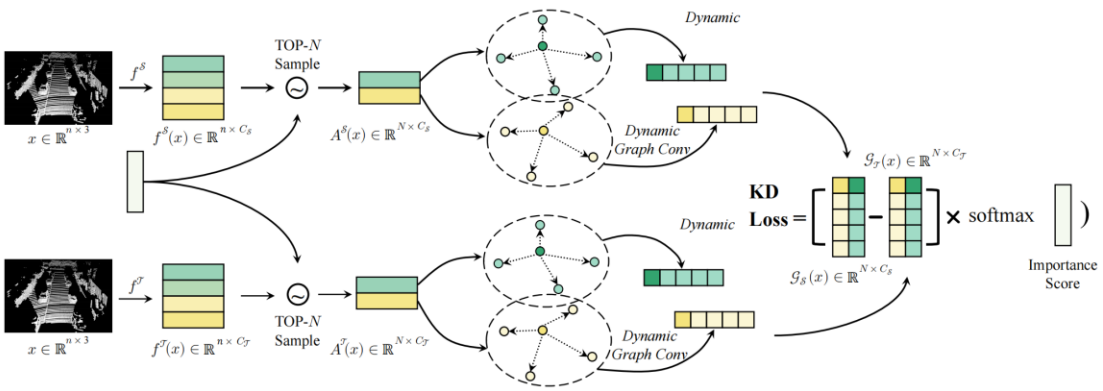
大型 3D 检测器和轻量化 3D 检测器的性能差异越来越大，能取得 SOTA 效果的一般是大型 3D 检测器，而轻量化 3D 检测器一般用于实时场景，为了减小这种差异，提出了很多种模型压缩技术，如网络剪枝，量化，轻量级模型设计以及知识蒸馏，本文的方法就属于知识蒸馏

点云不同于图像，（1）缺乏拓扑信息（2）在度量空间不规则且稀疏分布，因此基于图像的知识蒸馏方法不能直接用于点云模型

在点云的局部几何结构中捕获和利用几何信息对点云的表示学习有重要的影响；因此本文使用局部蒸馏，首先使用 knn 集中局部相邻的体素或点云，然后使用动态图卷积层编码局部几何结构的语义信息，再将语义信息从 teacher 蒸馏到 student，而不采用直接将 teacher 的 backbone 特征蒸馏到 student 的方法；使用这种方法，student 检测器能够继承 teacher 理解点云局部几何信息的能力

处理点云数据的一种主流方法是将点云转换为体素，然后将它们编码为结构化数据，但是由于点云的稀疏性和噪声，绝大多数体素只包含一个点，这种单点体素很大可能是噪声，因此这些单点体素的特征在知识蒸馏中的重要性应该较低；因此本文提出重新加权策略，通过给 student 对多点体素更大的学习权重以强调 student 对多点体素的学习，本质思想是突出对预测影响大的特征的知识蒸馏

三、 方法



PointDistiller 共有两个模块:(1)用于从点云的局部几何结构蒸馏 teacher

知识的**局部蒸馏**（2）通过突出 student 对相对更关键的体素和点的学习以处理点云稀疏性的**重新加权策略**

符号约定如下：点云表示为 $\mathcal{X}=\{x_1,x_2,...,x_n\}$，对应的标签表示为 $\mathcal{Y}=\{y_1,y_2,...,y_m\}$，目标检测器表示为 $\mathcal{F}=f^{\circ}g$，其中 f 表示从输入提取特征的特征编码层，g 表示用于预测的检测头，样本 x 的特征表示为 $f(x)\in\mathbb{R}^{n\times C}$，其中 n 表示基于体素的检测器的体素数或基于点云的检测器的点数，C 表示通道数，除此之外，如果第 i 个点属于第 j 个体素，则定义 $v_{ij}(x)=1$，否则定义 $v_{ij}(x)=0$，第 j 个体素中点的个数表示为 $\sum_j v_{ij}(x)$，$\mathcal{S}$ 表示 student 检测器，$\mathcal{T}$ 表示 teacher 检测器

流程具体如下：

1. 采样顶部待蒸馏体素（点）

   针对基于体素的检测器，我们定义第 i 个体素的重要性分数为 $\sum_j v_{ij}(x)$，针对基于点的检测器，受以前工作的启发（使用 attention 定位图片的关键像素），我们定义第 i 个点的重要性分数为它的特征在 channels 上的排列不变最大值，表示为 $\max(f(x)[i])$，我们根据重要性分数，采样分数最高的 N 个体素或点，定义分数最高的 N 个体素或点的 student 和 teacher 特征为 $A^{\mathcal{T}}(x)\in\mathbb{R}^{N\times C_{\mathcal{T}}}$ 和 $A^{\mathcal{S}}(x)\in\mathbb{R}^{N\times C_{\mathcal{S}}}$

2. 提取局部几何信息

   定义 $z_i=A(x)[i]$ 为第 i 个待蒸馏的体素或点的特征，通过 knn 算法对体素或点进行聚集，然后根据这个体素（点）以及和他的 K 个邻居体素（点）来构建一个图，将这个体素和它的 $K-1$ 个邻居体素的特征表示为 $z_{i,1}$ 和 $\mathcal{N}_i=\{z_{i,2},z_{i,3},...,z_{i,K}\}$；

   首先通过将每个点的特征与全局质心体素进行拼接以进行更新，可以表述为 $\hat{z}_{i,j}=\mathrm{cat}([z_{i,1},z_{i,j}])$，然后用动态图卷积进行聚合计算，表述为 $\mathcal{G}_i=\gamma(\hat{z}_{i,1},...,\hat{z}_{i,K})$，其中 γ 是一个具有 ReLU 激活和批处理归一化的非线性层，局部蒸馏的训练目标表示为 $\arg\min_{\theta_S,\theta_\gamma}\mathbb{E}_x\left[\frac{1}{N}\sum_{i=1}^{N}\left\|\mathcal{G}_i^{\mathcal{S}}(x)-\mathcal{G}_i^{\mathcal{T}}(x)\right\|\right]$，其中 θ_S 表示 student 编码层的参数，$\theta_\gamma=[\theta_\gamma s,\theta_\gamma\tau]$ 表示 student 和 teacher 检测器动态图卷积层的参数

3. 重新加权知识蒸馏损失

根据重要性分数，重新加权每个体素的学习权重，定义N个待学习的权重为$\phi \in \mathbb{R}^N$，将每个图的学习权重定义为图特征取最大值再经过 softmax 函数，表述为$\phi = \text{softmax}\ (\max(G^T(x))/\tau)$，针对基于体素的方法，学习权重定义为$\phi_i = \text{softmax}\left(\sum_j v_{i,j}/\tau\right)$，知识蒸馏的训练目标表述为$\arg\min_{\theta_S,\theta_\gamma} \mathbb{E}_x\left[\frac{1}{N}\sum_{i=1}^{N}\phi_i \cdot \left\|\mathcal{G}_i^S(x) - \mathcal{G}_i^T(x)\right\|\right]$，通过这种加权方法，student 检测器能够将更大的权重放在更关键的图上

VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection (VoxelNet)

一、 本文贡献

二、 研究动机

三、 方法

Point-Voxel CNN for Efficient 3D Deep Learning (PVCNN)

一、 本文贡献

二、 研究动机

三、 方法

Structure Aware Single-stage 3D Object Detection from Point Cloud

一、 本文贡献

二、 研究动机

三、 方法

Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution

一、 本文贡献

二、 研究动机

三、 方法

Lift, Splat, Shoot: Encoding Images from Arbitrary Camera Rigs by Implicitly Unprojecting to 3D (LSS)

一、 本文贡献

1. 提出了基于显式深度估计将环视感知转换到 BEV 空间的方法
2. 解决了多个传感器融合的问题，省去了多个传感器单独检测再进行后处理的操作

二、 研究动机

1. 为什么要提出 BEV?
   输入为来自多个传感器的信息，这些传感器的坐标系不同，我们需要一个全新的坐标系中进行目标检测。
2. 传统做法和 BEV 方法区别
   传统方法是使用多个传感器单独检测，然后将每个传感器单独检测的结果经行融合，但是这种方法无法将对单独检测进行融合这一步骤的损失反向传播到单独检测去；
   而 BEV 方法是将多个传感器的特征信息转化到 BEV 空间中去，再使用 BEV 特征进行目标检测，目标检测的损失可以反向传播到单个传感器的输入去。

三、 方法

1. 问题描述
   $\{X_k \in \mathbb{R}^{3\times H\times W}\}_n$表示来自$n$个相机的图片，每个相机图片对应一个外参矩阵$E_k \in \mathbb{R}^{3\times 4}$和一个内参矩阵$I_k \in \mathbb{R}^{3\times 3}$，其中内外参矩阵定义了从世界坐标系$(x, y, z)$到像素坐标系$(h, w, d)$的映射，我们的目标是根据$\{X_k\}_n$以及$E_k, I_k$得到 BEV 特征$y \in \mathbb{R}^{C\times X\times Y}$。
2. 第一阶段
   独立地处理单个图像，将单个相机图片对应的二维坐标系转换到所有相机共享的三维坐标系中（世界坐标系），坐标系转换的一个难点是：每个像素对应的“深度”是未知的。
   我们用$X \in \mathbb{R}^{3\times H\times W}$表示一张图片，$E$和$I$分别表示这张图片对应相机的外参矩阵和内参矩阵，$p$表示图片坐标$(h, w)$的像素，我们将$|D|$个点$\{(h, w, d) \in \mathbb{R}^3 | d \in D\}$与每个像素关联起来（其中$D$是一组离散的深度），可以得到$H \times W \times |D|$个 3d 点，再根据内外参矩阵，将这一系列点转换到世界坐标系中去。
   具体的实验设定为$|\mathrm{D}| = 41$，表示距离图像特征平面 4m到 44m，间隔 1m 的 41 个特征平面。
3. 第二阶段
   模型预测每个图像特征点的C维语义特征c以及D维深度分布概率α，然后将α和c做外积，得到$H \times W \times D \times C$的特征图（可以理解为$H \times W \times D$个 3 维点的$C$维特征）
4. 第三阶段
   根据 3 维点的空间位置(x, y, z)将每个 3 维点的语义特征c放到 BEV 网格的对应位置（直接垂直投影），得到 BEV 特征图

四、 实验细节

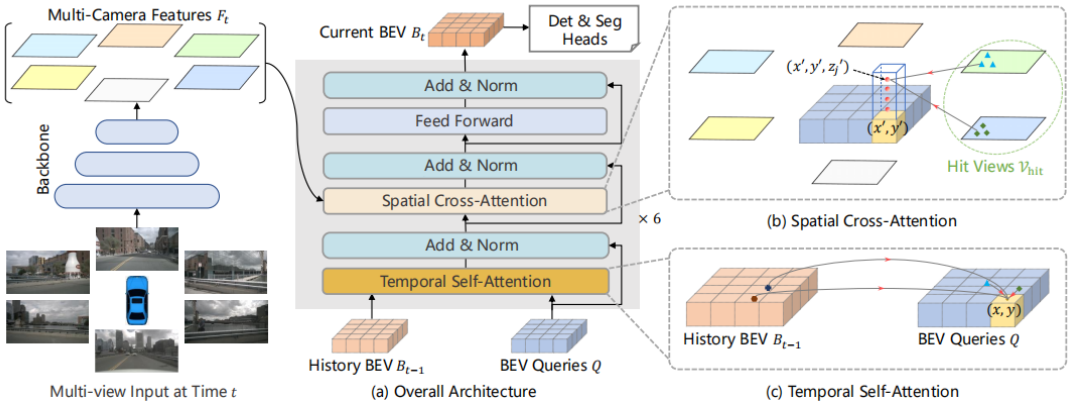
1. BEV 空间共有200×200个网格。每个网格对应现实世界物理尺寸$0.5m \times 0.5m$，整个 BEV 空间对应车辆前后左右各 50m，且高度不限的 3 维空间
2. $H \times W \times D$个 3 维点有可能在 BEV 空间（车辆前后左右各 50m）外，BEV 空间范围外的 3 维点直接删除，只将有效的 3 维点分配到 BEV pillar 中
3. 由于单张 2D 图像的多个像素点可能投影在俯视图一个位置，相邻两相机可能有部分成像区域重叠两个原因，可能存在多个 3 维点分配到同一个 BEV pillar 的情况，这种情况使用 sum-pooling，将同一 BEV pillar 的 3 维点特征相加，最后得到了$200 \times 200 \times C$的特征图，本文中的C取 64

BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers (ECCV2022)

一、 文章贡献

1. 使用 Spatio Temporal Transformer 得到 BEV features（基于 Transformer，将多个视角图片特征转化得到 BEV 特征；
2. Spatial Cross-Attention 模仿 Deformable Attention，减小了 Attention 计算开销
3. Temporal Self-Attention，获取时刻t的 BEV 特征时，运用了时刻$t-1$的 BEV 特征，更好地利用了时间信息

二、 网络架构



1. 网络概述

网络共有 6 层 encoder，除了 BEV Queries，Spatial Cross-Attention，Temporal Self-Attention，其余部分和 Transformer 架构一致

其中，BEV Queries 是预定义的网格，通过 Attention 在多视角图片中查询 BEV 空间的特征。Temporal Cross-Attention 和 Spatial Self-Attention 是具体的 Attention 计算层，根据 BEV Queries 查找和聚合时空特征以及历史 BEV 特征

2. 网络细节

首先将多个视角图片（时刻t）先输入到 backbone，得到特征$F_t = \{F_t^i\}_{i=1}^{N_{view}}$

将时刻$t-1$的 BEV 特征B_{t-1}和时刻t的查询 Q（$B_{t-1} \epsilon \mathbb{R}^{200\times200\times256}, Q \epsilon \mathbb{R}^{200\times200\times256}$）输入到 Temporal Self-Attention 中，其中 Query 为B_{t-1}或 Q，Key 为B_{t-1}或 Q，Value 为B_{t-1}或 Q 的线性映射，然后将B_{t-1}作自注意力和 Q 作自注意力得到的输出取平均值作为 Value 输入到 Spatial Cross-Attention 中

然后将特征$F_t = \{F_t^i\}_{i=1}^{N_{view}}$作线性变换得到 Spatial Cross-Attention 的 Key 和 Value，Temporal Self-Attention 的输出线性变换后作为 Query 进行 attention 操作，描述如下：

$$O = Softmax(\Phi W_q * (XW_K)^T) * XW_V = W_{transformer}(X, \Phi) * XW_V$$

如此经过 6 层 encoder，生成时刻t的 BEV 特征B_t

输入时刻t的 BEV 特征B_t，使用 3D detection head 和 map segmentation head 预测三维检测框和语义图

三、 模型细节

1. Deformable Attention

由于多视图图片的输入规模很大（$N_{view} \times h \times w \times c$），所以 BEV Queries 与图片的 Attention 计算开销很大，因此采用 Deformable Attention 的思想，让Q_p只与图片中感兴趣区域进行 Attention 计算以减小开销

首先将 BEV Queries 提升到柱状的 Queries，在柱状上采样N_{ref}个 3D 参考点，然后将这些参考点投影到 2D 视图图片上（根据相机内外参投影，有可能只能投影到部分视角的图片），我们将能被投影到的视角定义为v_{hit}，然后，我们将这些视角上的投影点视为Q_p的参考点，并且从v_{hit}的参考点附近采样特征

Spatial Cross-Attention 描述如下：

$$SCA(Q_p, F_t) = \frac{1}{|v_{hit}|} \sum_{i \in v_{hit}} \sum_{j=1}^{N_{ref}} DeformAttn(Q_p, P(p,i,j), F_t^i)$$

其中，$P(p,i,j)$为投影函数（用于得到Q_p的第i个视角图片的第j个投影点）

下面简述一下$P(p,i,j)$的实现原理：

先计算$p = (x, y)$对应的真实世界位置$x' = (x - \frac{W}{2}) \times s;\ y' = (y - \frac{H}{2}) \times s$，$(x', y')$为车身为原点时$p$对应真实世界的坐标

在 3D 空间中，位于(x', y')的对象将出现在高度z'上，因此我们预设了一组 anchor 高度$\{z'\}_{j=1}^{N_{ref}}$以捕获不同高度的信息

对于Q_p，我们可以得到$(x', y', z')_{j=1}^{N_{ref}}$，然后我们通过摄影机的投影矩阵将 3 维参考点投影到不同的图像视图上

2. Temporal self-attention

给定时刻t的Q和历史 BEV 特征B_{t-1}，首先根据车辆自身的运动将B_{t-1}和Q对齐，得到对齐的历史 BEV 特征B'_{t-1}，除此之外，从$t-1$到t，现实世界还存在着很多可移动的物体发生移动，因此准确对应同一对象在不同时间的 BEV 特征十分重要，使用 Temporal Self-Attention 实现这一点，描述如下：

$$TSA(Q_p, \{Q, B'_{t-1}\}) = \sum_{V \in \{Q, B'_{t-1}\}} DeformAttn(Q_p, p, V)$$

其中Q_p表示位于(x, y)的 BEV Query，不同于普通的 Deformable Attention，

偏移量Δp可以由Q和B'_{t-1}的拼接预测，对于第一个时间戳，我们用$\{Q, Q\}$代替$\{Q, B'_{t-1}\}$

BEVDepth: Acquisition of Reliable Depth for Multi-View 3D Object Detection (BEVDepth)

五、 贡献

1. 解决了 LSS 深度估计不准确的问题
2. 提出了高效计算 voxel-pooling 的方法（主要通过 CUDA 并行操作）
3. 引入了多帧融合

二、 研究动机

Lift-splat 方法虽然获得了不错的检测性能，但是该方法学习得到的深度是很糟糕的，而准确的深度预测对于检测性能的提升有着重要意义（见下图），所以有理由相信如果能让 LSS 方法进行更为准确的深度预测，其性能会有很大提升

D^{pred}	mAP↑	mATE↓	NDS↑
learned	0.282	0.768	0.327
random soft	0.245	0.838	0.290
random hard	0.176	0.922	0.224
ground truth	0.470	0.393	0.515

Table 1: Evaluation of depth prediction on the nuScenes *val* set. “soft” and “hard” denote gaussian and one-hot randomization along depth dimension, respectively.

三、 LSS

✧ 本文使用的基于 Lift-splat 的检测器由四个部分组成

1. Image encoder (ResNet)

   提取图像特征$\mathrm{F^{2d}} = \{\mathrm{F_i^{2d}} \epsilon \mathbb{R}^{\mathrm{C_F} \times \mathrm{H} \times \mathrm{W}}, \mathrm{i} = 1,2, \dots, \mathrm{N}\}$，N表示视图数

2. 估计图像深度的 DepthNet

   根据$\mathrm{F^{2d}}$预测深度$\mathrm{D^{Pred}} = \{\mathrm{D_i^{pred}} \epsilon \mathrm{F_i^{2d}} \epsilon \mathbb{R}^{\mathrm{C_D} \times \mathrm{H} \times \mathrm{W}}, \mathrm{i} = 1,2, \dots, \mathrm{N}\}$，$\mathrm{C_D}$表示离散深度个数

3. 将 2d 特征投影为 3d 特征的 View transformer

   根据 2d 特征和预测的图像深度投影得到$\mathrm{F_i^{3d}} = \mathrm{F_i^{2d}} \otimes \mathrm{D_i^{pred}}$，然后池化得到 BEV 表示$\mathrm{F^{bev}}$，其中$\mathrm{F_i^{3d}} \epsilon \mathbb{R}^{\mathrm{C_F} \times \mathrm{C_D} \times \mathrm{H} \times \mathrm{W}}$

4. 用于预测类别，3d bbox 的检测头

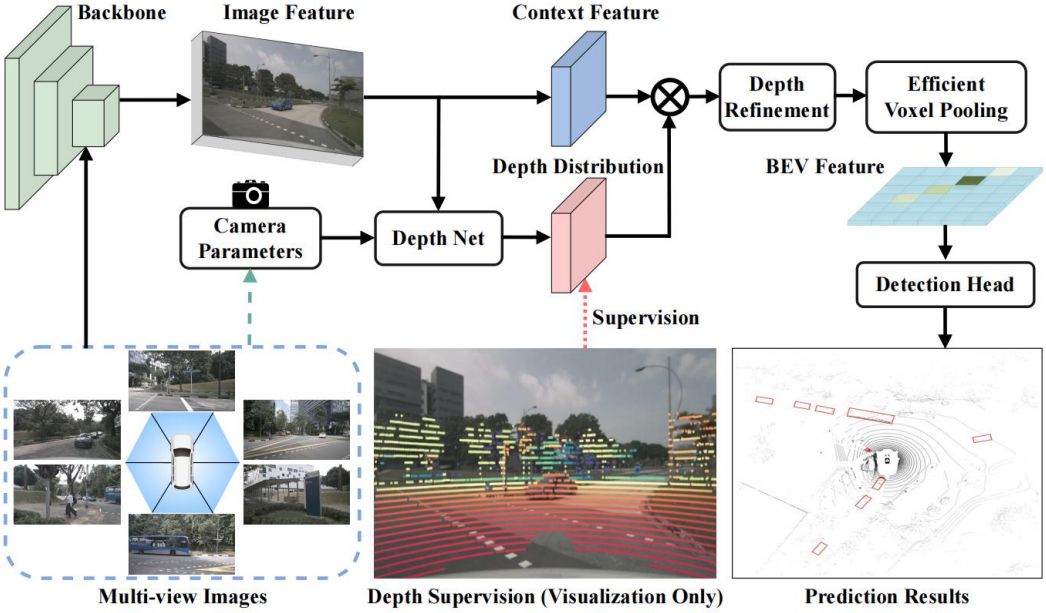
✧ 对 LSS 预测的深度的可视化，发现虽然在一些关键物体附近的深度预测较为准确，但绝大部分区域的深度预测是不准确的

四、 方法

✧ 使用显式深度监督优化深度估计（其实就是在训练阶段利用点云数据的深度

信息对深度估计进行监督）

- ✧ 提出了能够高效计算 voxel-pooling 的方法，以进行加速
- ✧ 采用多帧融合，以提高目标检测的性能



- ✧ 显式深度监督细节

1. 上一节中，对深度预测的监督来自于目标检测 head 的 loss（ground truth 和 predict bbox 之间的 loss），然而事实上，仅使用检测头进行监督是不够的，所以使用点云数据的深度真值D^{gt}来监督预测得到的深度值D^{pred}
2. 定义$R_i \epsilon \mathbb{R}^{3\times3}$和$t_i \epsilon \mathbb{R}^{3\times3}$为自车坐标系到第i个视图坐标系的旋转和平移矩阵，$K_i$为第i个相机的内参，我们使用如下方法获取实际深度

$$P_i^{img}(ud, vd, d) = K_i(R_i P + t_i)$$

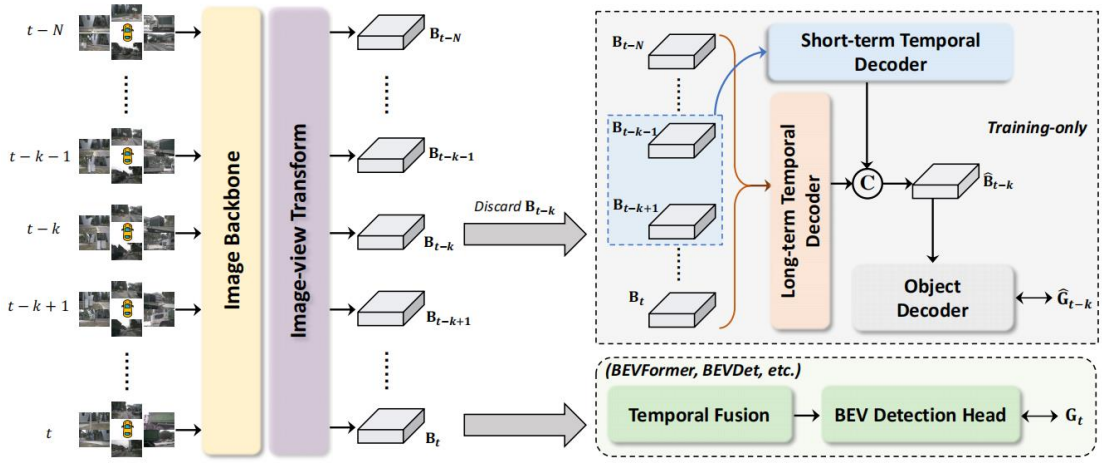
u和v表示像素位置，这样我们就获得了根据点云数据计算得到的每个像素点的深度真值

Temporal Enhanced Training of Multi-view 3D Object Detector via Historical Object Prediction (HoP)

一、 贡献

1. 提出了一种即插即用的增强 BEV 特征的方法，能够很容易地集成到经典的 BEV 检测框架中去（如 BEVFormer，BEVDet 等）
2. 仅在训练阶段进行 HoP 操作，不会额外增加推理开销
3. 显著提高经典 BEV 检测方法的性能

二、 方法



✧ Pipeline

1. 首先利用经典的 BEV 检测框架（如 BEVFormer，BEVDet 等）得到 BEV 特征序列$\{B_{t-N},\dots,B_t\}$，并且我们将相应时间戳的 ground truth 表示为$\{G_{t-N},\dots,G_t\}$
2. 然后使用时间戳$t-k$以外的 BEV 特征预测时间戳$t-k$的目标，$\hat{P}_{t-k}=\hat{\mathcal{D}}(\mathcal{T}(\{B_{t-N},\dots,B_t\}-\{B_{t-k}\}))$，其中$\mathcal{T}(\{B_{t-N},\dots,B_t\}-\{B_{t-k}\})$代表时间戳$t-k$的伪 BEV 特征，$\hat{\mathcal{D}}$代表使用伪 BEV 特征进行目标检测的检测器

✧ 关于$\mathcal{T}$

输入到$\mathcal{T}$的剩余 BEV 特征分为两部分，short-term 和 long-term，short-term BEV 特征表示为$\{B_{t-k-1},B_{t-k+1}\}$，long-term BEV 特征表示为$\{B_{t-N},\dots,B_{t-k-1},B_{t-k+1},\dots,B_t\}$，首先分别在 short-term 和 long-term BEV 特征添加 time embeddings，然后将处理过的 BEV 特征输入到$\mathcal{T}$，得到$\hat{B}_{t-k}$

✧ 关于$\hat{\mathcal{D}}$

将$\hat{B}_{t-k}$输入到$\hat{\mathcal{D}}$，得到$\hat{P}_{t-k}$

✧ 关于 short-term temporal decoder

1. 只操作B_{t-k-1},B_{t-k+1}，记为B^{adj}

2. 首先定义一个 learnable 网格化的 short-term BEV queries，记为 $Q_{t-k}^{short} \epsilon \mathbb{R}^{H \times W \times C}$

3. 根据如下公式求$t-k$时间戳的 short-term 伪 BEV 特征，$\widehat{B}_{t-k}^{short} = \sum_{V \epsilon B^{adj}} DeformAttn(Q_{t-k}^{short}, p, V)$，其中$p$表示 BEV 平面上的参考点

4. 然后将$\widehat{B}_{t-k}^{short}$输入到前馈网络，得到 short-term 分支的输出

✧ 关于 long-term temporal decoder

1. 操作$B^{rem} = \{B_{t-N}, \dots, B_t\} - \{B_{t-k}\}$，首先对$B^{rem}$进行 channel reduction 操作以获得更好的训练效率
2. 首先定义一个 learn 网格化的 long-term BEV queries，记为 $Q_{t-k}^{long} \epsilon \mathbb{R}^{H \times W \times C/r}$，以及还原参数$W_r \epsilon \mathbb{R}^{C \times C/r}$

3. 根据如下公式求$t-k$时间戳的 long-term 伪 BEV 特征，$\widehat{B}_{t-k}^{long} = \sum_{V \epsilon B^{rem}} DeformAttn(Q_{t-k}^{long}, p, VW_r)$

✧ 融合 short-term BEV 特征和 long-term BEV 特征

1. 首先将$\widehat{B}_{t-k}^{short}$和$\widehat{B}_{t-k}^{long}$连接起来
2. 然后通过3×3卷积提取特征

✧ 关于 Object decoder

1. 使用 Object decoder，利用$\widehat{B}_{t-k}$预测$\widehat{P}_{t-k}$
2. 为了对齐 ground truthG_{t-k}和要学习的目标$\widehat{G}_{t-k}$，我们用$e(t)$表示时间戳t的自我坐标系，$\widehat{G}_{t-k} = T_{e(t-k)}^{e(t)} G_{t-k}$

3. G_{t-k}表示$t-k$时刻坐标系下的 Ground truth，$\widehat{P}_{t-k}$表示t时刻坐标系下对$t-k$时刻 Ground truth 的预测，因此需要对G_{t-k}进行坐标变换以对齐

✧ 其他增强手段——历史时刻 Query 融合

1. 对于时间戳$t-k$，预定义一个 object query，记为O，$\overline{O}_{t-k} = \mathcal{D}(B_{t-k}, O)$，其中$\mathcal{D}$代表目标检测器，将 object query 和 BEV 特征作 attention 得到目标（通常是 3d bbox）
2. 使用上一时间戳的$\overline{O}_{t-k-1}$得到$O_{t-k}^{his} = MLP(\overline{O}_{t-k-1})$
3. 融合预定义的 object query 和根据上一时间戳求得的该时间戳 object query，再次作 attention，求该时间戳的目标，$\overline{O}_{t-k} = \mathcal{D}(B_{t-k}, O + O_{t-k}^{his})$

DistillBEV: Boosting Multi-Camera 3D Object Detection with Cross-Modal Knowledge Distillation (DistillBEV)

一、 贡献

1. 提出了跨模态知识蒸馏方法 DistillBEV，通过让多相机 BEV 模型模仿 Lidar 模型的特征，实现了多相机 BEV 模型性能的提升
2. 方法具有通用性，student 可为 CNN 范式，也可为 Transformer 范式

二、 研究动机

基于多相机 BEV 的三维目标检测方法与基于激光雷达的方法还存在明显的性能差距

三、 方法

本文中的方法可以分如下几个部分：

- ✧ 区域分解
- ✧ 自适应缩放
- ✧ 空间注意力
- ✧ 多尺度蒸馏
- ✧ 蒸馏损失
- ✧ 带时序融合的蒸馏

1. 区域分解

   区域分解的目的是引导 student 将注意力更多地放在关键区域，具体操作是将 feature map 分为 4 种类型：TP、FP、TN、FN，然后定义一个区域分解掩模 M：

$$M_{i,j} = \begin{cases} 1, & if(i,j) \epsilon TP \ or \ FN \\ \eta, & if(i,j) \epsilon FP \\ 0, & if(i,j) \epsilon TN \end{cases}$$

   当 teacher 在一些区域处生成 high activations，尽管该区域是 FP（被误判断为前景物体的背景物体），我们仍鼓励 student 模仿 teacher 认为这些区域是有价值的，FP 区域通过如下方式寻找：

$$FP = (H^t > \gamma) \& (H^g < \gamma)$$

2. 自适应缩放

   在 BEV 空间从 teacher 到 student 进行知识蒸馏的另一个可能影响因素是各种物体之间巨大的尺度差异，我们应该让不同尺度和类别的物体对蒸馏损失的贡献程度相近：

$$S_{i,j} = \begin{cases} \frac{1}{\sqrt{H_k W_k}}, & \text{if(i,j)} \epsilon O_k \\ \frac{1}{N_{FP}}, & \text{if(i,j)} \epsilon \text{FP} \\ \frac{1}{N_{TN}}, & \text{if(i,j)} \epsilon \text{TN} \end{cases}$$

3. 空间注意力

使用基于从 student 和 teacher 中提取特征的 spatial attention map 进一步提取特征，具体描述如下：

$$\mathcal{P}(F)_{i,j}=\frac{1}{C}\sum_{c=1}^{C}|F_{c,i,j}|$$

$$\mathcal{N}(F)=HW\text{softmax}\left(\mathcal{P}(F)/\tau\right)$$

其中 F 为 feature map，$\mathcal{P}(F)$ 表示在 channels 上的 average pooling，$\mathcal{N}(F)\in\mathbb{R}^{H\times W}$ 是对空间注意力的归一化，最后的空间注意力同时考虑了 student 和 teacher 的特征图：

$$A=(\mathcal{N}(F^t)+\mathcal{N}(\tilde{F}^s))/2,\quad \tilde{F}^s=\mathcal{G}(F^s)$$

其中$\mathcal{G}$用于将F^s的形状转化为F^t一样

4. 蒸馏损失

定义 student 和 teacher 网络在第 n 个蒸馏层的损失如下：

$$L_{\text{feat}}=\alpha\sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W}M_{i,j}S_{i,j}A_{i,j}\left(F_{c,i,j}^{t(n)}-\tilde{F}_{c,i,j}^{s(n)}\right)^2+\beta\sum_{c=1}^{C}\sum_{i=1}^{H}\sum_{j=1}^{W}\bar{M}_{i,j}S_{i,j}A_{i,j}\left(F_{c,i,j}^{t(n)}-\tilde{F}_{c,i,j}^{s(n)}\right)^2$$

其中 M 是上文提到的区域分解掩模，S 是上文提到的自适应缩放因子，A 是上文提到的空间注意力图；

定义注意力模仿损失如下：

$$L_{\text{attn}}=\sum_{i=1}^{H}\sum_{j=1}^{W}\left|\mathcal{P}(F^{t(n)})_{i,j}-\mathcal{P}(\tilde{F}^{s(n)})_{i,j}\right|$$

最终的蒸馏损失定义为：

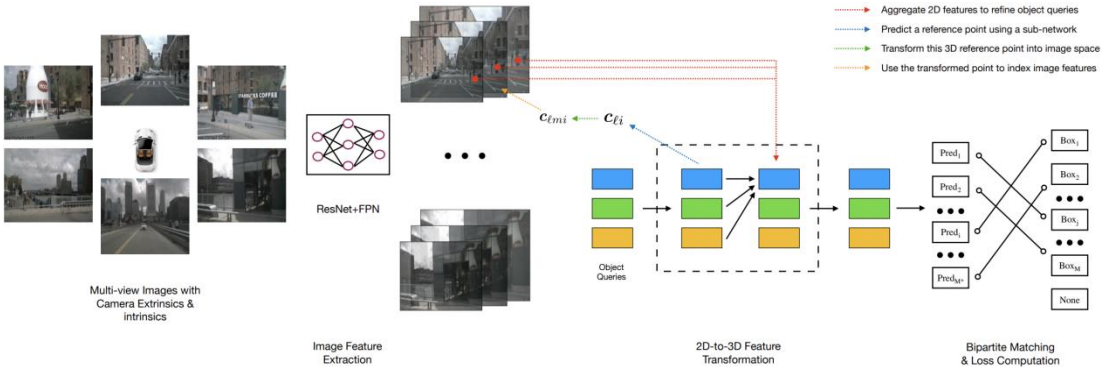
$$L_{\text{dist}}=\sum_{n=1}^{N}L_{\text{feat}}(F^{t(n)},F^{s(n)})+\lambda L_{\text{attn}}(F^{t(n)},F^{s(n)})$$

DETR3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries (DETR3D)

一、 文章贡献

4. 受 DETR 启发，将 Transformer 引入 3D 检测任务；模仿 DETR，一次性生成 N 个 bbox（bounding box），采用 set-to-set 损失函数计算预测 bbox 和 ground truth 之间的二分图匹配损失
5. 端到端，不同于 proposal-based 和 anchor-based 的方法，不需要后处理操作（如 nms 非极大值抑制等）

二、 网络架构



1. 输入：环视的 6 张图片，$I = \{im_1, \ldots, im_6\} \in \mathbb{R}^{H_{im} \times W_{im} \times 3}$
2. 特征提取：将 6 张环视图片输入到 ResNet 或 VoVNet 提取特征，再通过 FPN 进行特征增强，得到 4 个尺寸的特征，$F_k = \{f_{k1}, \ldots, f_{k6} \in \mathbb{R}^{\mathrm{H} \times \mathrm{W} \times \mathrm{C}}\}$
3. 参考点生成：预设 object query 数目M^*，decoder 的第$l \in \{0, \ldots, L-1\}$层处理$Q_l = \{q_{l1}, \ldots, q_{lM^*}\} \in \mathbb{R}^C$，并得到$Q_{l+1}$，根据$q_{li}$生成$M^*$个参考点$c_{li} \in \mathbb{R}^3$，$c_{li} = \Phi^{ref}(q_{li})$
4. 视角转换：首先将c_{li}用齐次坐标表示$c_{li}^* \in \mathbb{R}^4$，根据相机内外参，将 3d 齐次坐标转换为 2d 齐次坐标，$c_{lmi} = T_m c_{li}^*, T_m \in \mathbb{R}^{3 \times 4}$
5. 特征采样：使用双线性插值从 2d 特征图采样，$f_{lkmi} = f^{bilinear}(F_{km}, c_{lmi})$，下标表示第$i$个点在第$l$层，对第$k$个特征图的第$m$个视图进行特征采样，然后将同一个点在同一层的采样特征进行聚合，$f_{li} = \frac{1}{\sum_k \sum_m f_{lkmi} \sigma_{lkmi}}$
6. 迭代更新：迭代 query 进入下一层，$q_{(l+1)i} = f_{li} + q_{li}$
7. output-head：将Q_l输入分类头和检测头，$\hat{c}_{li} = \Phi^{cls}(q_{li})$，$\hat{b}_{li} = \Phi^{reg}(q_{li})$，对每个 object query 都会预测 9 维回归结果（bbox 坐标+长宽高+航向角+速度+分类），训练的时候，使用所有L层进行监督，测试的时候，只使用最后一层进行预测

三、 损失函数

采用 DETR 的 set-to-set 计算方法，对预测的(bbox,class)和 ground truth 的

(bbox,class)进行匹配，回归损失采用 L1_loss，分类损失使用 focal loss

四、 其他细节

2. 关于 object query 的解释

anchor-based 方法将检测描述为对预定义的 anchors 进行类别分类和边框系数回归，而 DETR 将检测视为集合预测问题，可以视为从图像序列转换为集合序列，而 object query 就是可学习的位置编码集合

3. set-to-set 计算方法

预测产生 N 个 bbox（N 为模型设定），这些 bbox 与 ground truth 的 bbox 对应关系尚未确定，存在一种匹配，使得 loss 最小化，我们先找到这种最优匹配，然后再计算 loss 并对 loss 进行优化，DETR 寻求最优匹配的方法为 hungarian algorithm