模型训练报告

I.引言

针对自动驾驶障碍物识别这个主题要求，首先需要对可能的障碍物进行准确的识别和跟踪。在本次报告中，我分别选择了YOLOv5以及基于Transformer的DETR算法，以探究它们在自动驾驶障碍物识别任务中的性能和优劣。

为了有效地进行障碍物识别，我选择了YOLOv5和DETR两种不同类型的算法。这两种算法在目标检测领域都有着显著的影响力，但采用了不同的方法来解决目标检测问题。简单来说，YOLO系列是一种基于深度卷积神经网络的实时目标检测算法，以其高效的检测速度和精准的检测结果而著称。相较于传统的目标检测算法，YOLO系列将目标检测问题转化为一个回归问题，通过将图像划分为网格并预测每网格中的目标的边界框和类别概率，从而实现了实时性和准确性的平衡。DETR算法这是基于Transformer架构的目标检测方法，通过将目标检测问题转化为一个序列到序列的转换任务，引入了注意力机制来捕捉目标之间的关系。能够在一次性推理中同时处理多个目标，具有良好的拓展性和准确性。之后会分别对两个算法进行更加详细的阐述。

II. 数据集质量评估

考虑到实现自动驾驶时可能出现的障碍物以及适用于多种交通场景的情况，我选取包括行人，其他车辆，摩托车，自行车，卡车，宠物，交通标志，警示桩，电线杆，垃圾桶等18种不同的障碍物类别，基于每个种类的训练数据量不少于一千，以及识别难度的情况，我使用边界框标注的标注规范构建了接近三万大小的数据集，其中训练集占23209张，剩下的平均分成测试集和验证集。

同时数据集的多样性和代表性也对模型的性能产生影响，多样性指的是数据集中包含不同的场景，天气，路况和环境，以及不同尺度和角度的障碍物。代表性则涉及到数据集是否能够准确地反映真实世界中驾驶场景的分布和特点。

考虑到多样性的影响，我提高了数据集的泛化能力，不仅包含了白天，晴朗天气下的图像，也引入了夜晚和雨天的情况。同时也对图像进行了数据增强操作，包括调整角度，分辨率，镜像，灰度调整等操作。

考虑到代表性的影响，我选取了一些真实驾驶场景中的数据，以提高模型在实际驾驶环境中的表现。

III. 算法实现

A.YOLOv5

1.算法原理

通过我对整个YOLO系列的学习，我对YOLO系列从v1到v5 的版本变化有了比较清晰的了解。

首先，YOLO系列不同于Faster-rcnn和Mask-rcnn经典的检测算法，YOLO系列是one-stage，用一个卷积神经网络就可以完成检测任务。优势是速度非常快，比较适合做一些像自动驾驶障碍物识别这种实时检测任务，但是缺点就是效果通常不会太好。在一些追求实时速度，不追求完全精确的任务下就非常适合YOLO算法。

从YOLOv1开始，把检测问题转化成了一个回归问题，利用整张图作为网络的输入，只需要经过一个卷积神经网络，得到bounding box的位置及其所属的类别。目标损失函数包括三部分，分别是坐标预测损失、置信度预测损失、类别预测损失。同时，YOLOv1的检测策略是将一张图片平均分成7×7的网格，每个网格分别负责预测中心点落在该网格内的目标。可是YOLOv1只提供了两种候选框，不能很好的适应物体，意味着v1对相互靠近重叠的物体和很小的物体的检测效果不好。

YOLOv2在YOLOv1的基础上进行了大量细节层面上的改进。首先YOLOv2舍弃了Dropout，卷积后全部加入Batch Normalization，同时对网络的每一层的输入都做了归一化，避免训练容易跑偏的情况，使训练收敛相对更容易。另外，YOLOv2采用了比YOLOv1更大的分辨率进行训练，提升了mAP。然后，在网络结构上，YOLOv2 采用Darknet-19 作为特征提取网络，借鉴了与VGG相似的结构，删除了全连接层，并且采用降维的思想，利用1×1的卷积来压缩特征。YOLOv2还采用了聚类提取先验框的机制，使网络更容易学习到准确的预测位置。并且YOLOv2引入了Anchor Box机制，通过提前筛选更有代表的anchor来使网络更容易收敛。YOLOv2的特征融合机制将之前的特征融合，来解决最后一层感受野太大而丢失一些小目标的问题。

YOLOv3利用draknet-53取代了draknet-19,使其更加时候进行一些小目标的检测。并且将特征做的更加细致，融入多持续特征图的信息来预测不同规格物体。YOLOv3还丰富了先验框，使用多scale的思想，设计了3个scale，一共九种候选框。同时v3中也使用的resnet的思想，使网络可以堆叠更多的层来进行特征提取。由于物体检测任务中可能一个物体有多个标签，因此在softmax层进行了改进，利用logistic激活函数来预测不同类别。

YOLOv4在数据层面的改进利用了BOF，只增加了训练成本，但是能显著提高精度，并不影响推理速度。v4用了一种Mosaic的数据增强策略，将四个图像拼接在一起成为一个新的图像进行训练。还有Random Erase，就是用随机值或训练集的平均像素值去替换图像的区域，以及Hide and Seek，可以根据概率设置随机隐藏一些补丁。YOLOv4中的损失函数使用的是CIOU的损失函数，同时考虑了重叠面积，中心点距离和长宽比三种几何因素。还使用了一种DIOU-NMS机制，在进行NMS时不仅要考虑IoU的值，还考虑了两个Box中心点之间的距离。YOLOv4在网络层面上的改进使用了BOS，增加了推断的代价，提高了模型精度。为了满足不同输入大小，利用SPPNet用最大池化来满足最终输入特征一致。v4中利用CSPNet将每一个block按照特征图的channel维度拆分成两部分，使得速度得到提升。值得提到的是YOLOv4加入了注意力机制，v4用到的是SAM，也就是空间的注意力机制。YOLOv4还使用了PAN机制，不仅有自顶向下的路径，还引入了一个自底向上的路径，使得底层信息更容易传到顶部。最后,v4的激活函数式Mish激活函数，相对于Relu激活函数更加符合实际，但是增加了计算量。

YOLOv5相当于是将v4进行了一个更好的实现，本质上是一个偏工程的项目。并且YOLOv5算法具有4个版本，具体包括：YOLOv5s、YOLOv5m、YOLOv5l、YOLOv5x四种。

2.代码框架和结构

YOLOv5的实现代码主要包括以下几个部分：

1. 输入端：Mosaic数据增强，自适应锚框计算，自适应图片缩放，归一化等操作。
2. 网络定义：定义了YOLOv5的网络结构，包括骨干网络（CSPDarknet53）、FPN+PAN的结构、多尺度检测头和损失函数(CIOU_LOSS)。
3. 损失计算：计算类别损失、位置损失和置信度损失，用于优化模型参数。
4. 前向传播：将图像数据通过网络前向传播，生成目标预测结果。
5. 后处理：对预测结果进行非极大值抑制（NMS）等后处理操作，得到最终的检测框和类别。

B.DETR

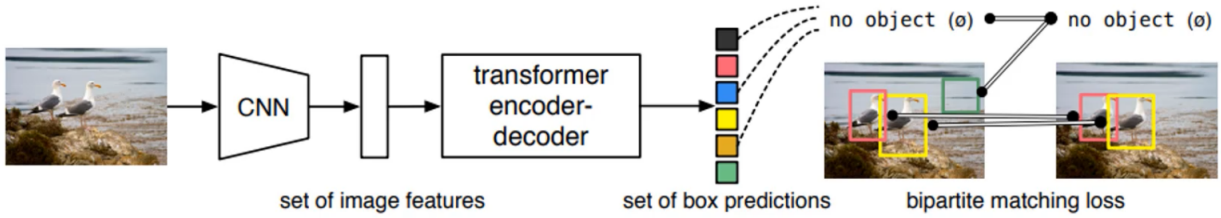
1.算法原理

DETR（DEtection Transformers）是一种基于Transformer架构的目标检测方法，与传统基于锚框的方法不同，DETR通过将目标检测问题转化为一个序列到序列的转换任务。它引入了注意力机制，同时对所有目标进行建模，从而实现端到端的目标检测。它不同于Faster-Rcnn，用各种proposal方法去做；也不同于YOLO系列基于anchor训练；DETR甚至不需要用NMS对输出结果进行过滤，它完全基于Transformer就可以实现目标检测算法。

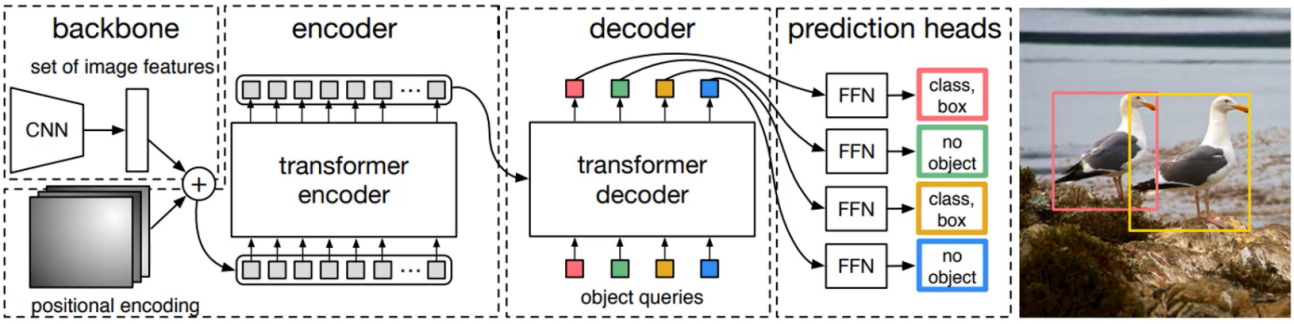
DETR进行目标检测一般是先全局范围内粗略搜，然后用放大精确锁定目标。因此其对小物体的检测效果不是很好，适用于大目标检测。

DETR的核心思想是，将图像中的目标框看作是一个序列，通过Transformer编码器将图像特征映射到一个特征序列。然后，通过Transformer解码器将特征序列转换为目标边界框和类别概率序列。在训练时，DETR采用了Hungarian匹配算法来将预测目标与真实目标进行匹配，计算匹配的损失。

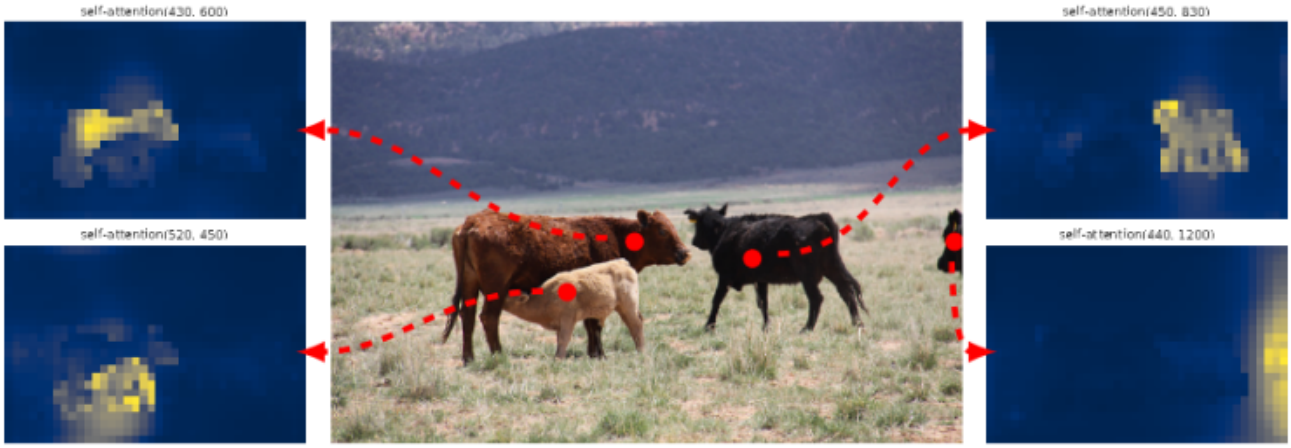
DETR的基本思想就是用CNN得到Patch作为输入，然后再做Transformer的encoder和decoder，encoder的方法和VIT基本一致，重点在decoder部分，DETR直接预测100个坐标框。



DETR网络架构的核心就是下图的object queries,就是要让这些query从原始特征找到是物体的位置。比较有特点的是在传统的transformer中decoder是串联进行的，但是在目标检测的过程中，DETR是并联进行的，100个坐标框同时生成。



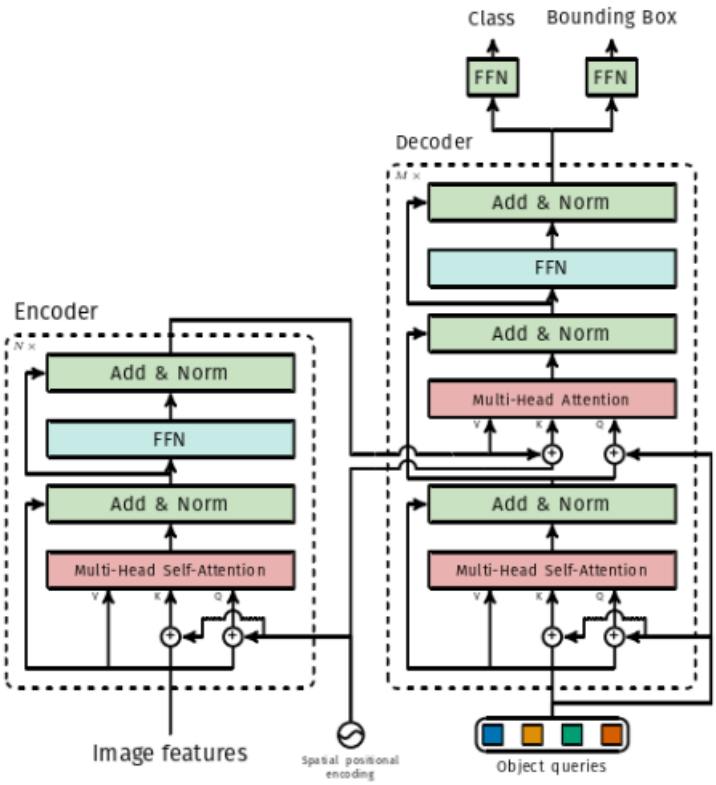
在Encoder过程中会得到各个目标的注意力结果，突出物体特征，减轻背景的影响，方便Ddcoder的过程。



解码器在一开始的时候会对100个object queries进行初始化(0+位置编码)，通过多层学习让其学习如何利用输入特征。

DETR的输出匹配，利用的是匈牙利算法，按照LOSS最小的组合，选出来n个目标，剩下100-n个都是背景。

2.代码框架和结构



DETR的实现代码框架主要包括以下几个部分：

1. 提取图像特征表示的CNN主干网
2. 编码器-解码器转换器：每个编码器层都有一个标准的体系结构，由一个多头自注意力模块和一个前馈网络(FFN)组成。解码器遵循转换器的标准架构，使用

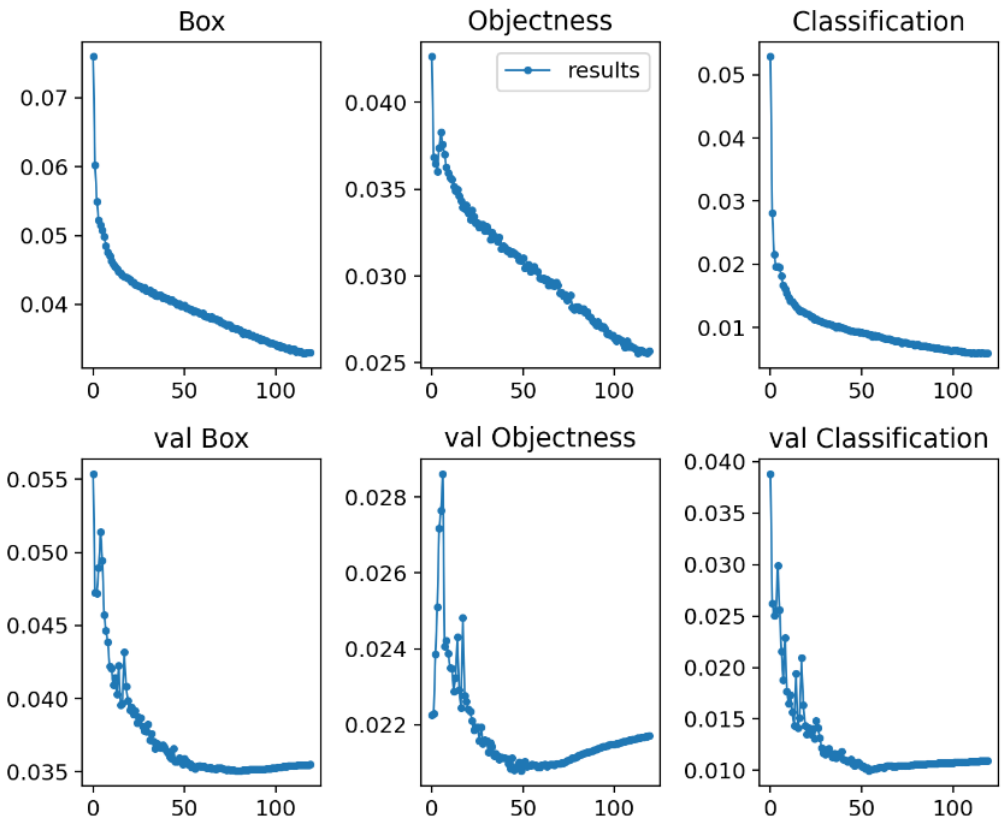
多头自注意力机制和编码器-解码器注意力机制转换 N 个大小为 d 的嵌入

3. 进行最终检测预测的简单前馈网络(FFN)

IV. 模型训练和评估

A.YOLOv5

1. Loss收敛曲线



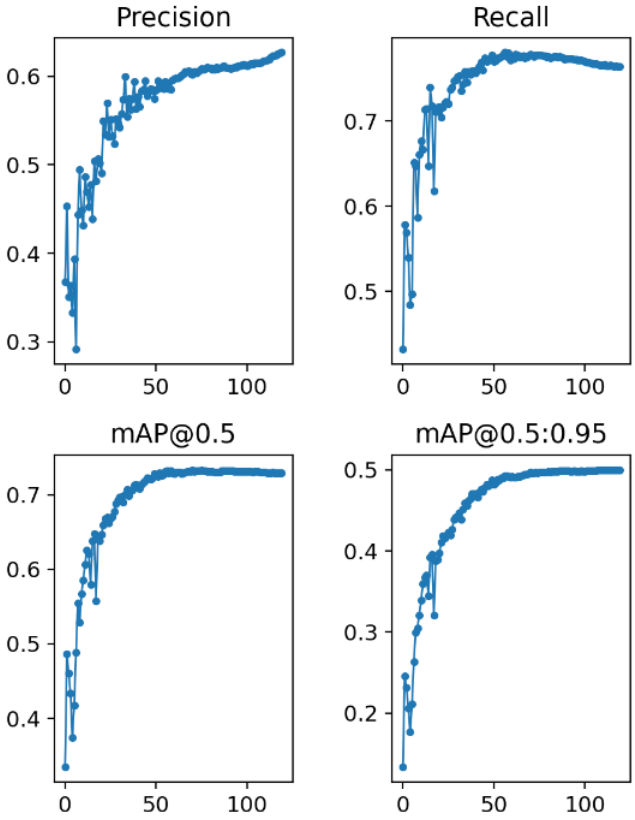
YOLOv5的训练由于显存的限制，我的batch-size达到了120，在我训练了120个epoch后得到上图的Loss曲线，通过曲线可以观察到，train_loss整体成下降的趋势，而val_loss在60个epoch之后整体呈现一个不减反增的情况，说明此时模型可能出现了过拟合。考虑到各种成本，我并没有进行改进并重新训练。

通过查阅资料可以得知导致过拟合的常见原因包括：数据不足，数据分布不均匀，模型复杂度过高，学习率过高，缺乏正则化，训练时间过长，数据预处理不当，验证集选择不当，特征工程不足等等。

针对我的模型进行分析：

1. 我首先要进一步提高我的数据集的质量，增加训练的数据量，重点是减小噪音数据的干扰，避免模型过分地记住噪音特征。
2. 虽然在我的数据集中每个类别的数量均超过1000。但是相对来说还是有些类别的数量过少，因此模型可能会在这些种类上出现过拟合，而对其他类别的泛化能力较弱。
3. 由于通过观察发现训练时的收敛速度还可以，所以我可能会通过进一步降低学习率，使用合适的优化算法的方式来避免模型太早出现过拟合现象。
4. 我还要优化输入特征以提高模型泛化能力。
5. 当然，我还可以通过Early Stopping的策略，在验证集上监控模型性能，一旦性能开始下降，就停止训练，避免过拟合。

2. mAP达成度

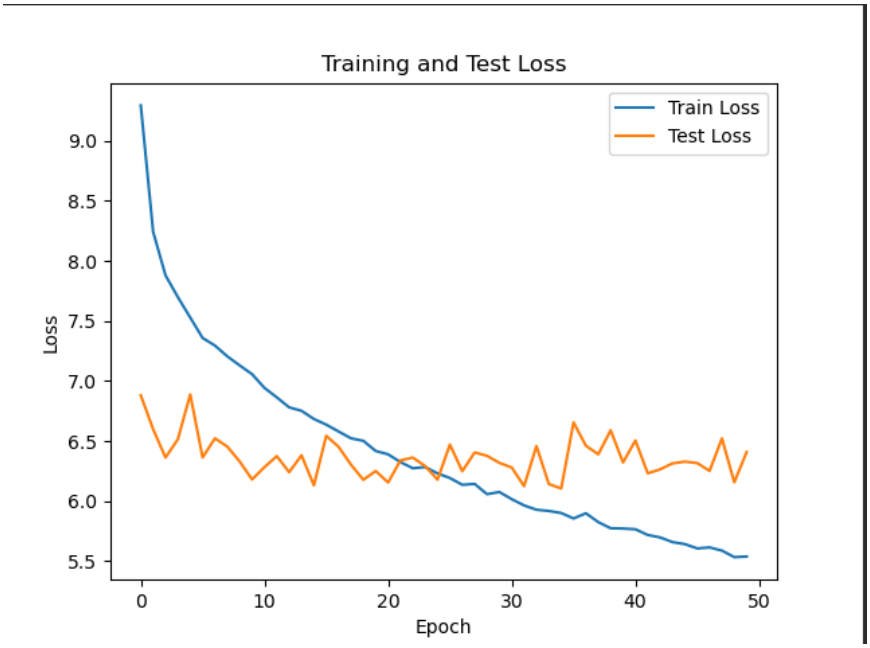


首先通过mAP的曲线可以看出来，我的模型在过拟合前呈现上升趋势并且达到了一个相对较高的mAP值(0.72+)，而在过拟合之后保持在相似的水平且平稳。这意味着模型的复杂度在初始阶段对训练数据的特征提取非常有效。同时平稳的mAP可能表明你的模型在训练数据上没有出现明显的过拟合，因此在验证集或测试集上的表现也能保持一定的稳定性。

在这种情况下，我应该在下次训练时持续健康模型在验证集或测试集上的性能，尤其是在未见过的数据上进行测试。同时，我应该尝试进一步优化模型，调整正则化参数，学习率，批次大小等超参数，并且进行清洗高质量的训练数据等等方法，来进一步提高我的mAP值。

B.DETR

1. Loss收敛曲线



由于DETR运行时需要大量的显存，DETR的作者在16个V100上也只能将batch-size设为64，并且训练了三天。所以考虑算力和时间成本，经过几次尝试，爆了几次显存，最终我把batch-size设为5，并且将epoch设为50。因此训练效果便一定不会很好，这次算一次尝试吧，经过这次的经历，我深刻意识到了算力对于模型训练的重要性。

从Loss收敛曲线可以看出，Train Loss曲线在持续下降，并且收敛速度还算可观。但是Test loss曲线出现了震荡的现象。通过查阅资料以及向老师请教，我归纳出的可能原因包括有batchsize更小；模型的泛化能力不够稳定，这可能是训练数据和测试数据的分布存在一些差异；过拟合，过拟合可能会导致模型在测试集上的泛化性能下降；学习率设置不当；数据噪声或不一致；优化器选择问题；数据量不足和不平衡等等问题。

针对我的模型进行分析：

1. 我认为造成Test Loss出现震荡现象最可能的原因是因为Batch Size太小，因为太小的batchsize会导致模型难以在训练中找到稳定的方向。这可能会影响模型在测试数据上的泛化能力。但是考虑到我的算力限制，我可以考虑尝试减小学

习率，并且使用一些正则化技术来减轻过拟合，我也准备尝试使用梯度累积的方式来间接增大批量的大小，当然这样时间成本也会增加。

2. 另外，DETR是一个比较复杂的模型，模型本身的复杂度可能比较高，需要更多的数据和更长的训练时间来达到好的泛化能力，下次我准备尝试简化模型的结构或者使用预训练模型来提升。同时我也应该增加一些类别的数据量，保持数据量的平衡。

2. mAP达成度

```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.453
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.704
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.476
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.106
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.384
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.635
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.381
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.570
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.606
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.204
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.549
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.788
```

通过对训练出的模型的mAP值的计算，可以看到mAP@0.50已经达到了0.704，表明模型现在已经有了一定的目标检测的能力，考虑到资源的限制，这已经是一个良好的起点，但仍然有优化空间。我会考虑进一步尝试不同的数据增强策略，优化超参数等等来提高模型在自动驾驶障碍物识别任务上的表现。

V. 模型比较与分析

A. 性能指标对比

1.模型架构和推理速度

YOLOv5

- 采用单阶段检测的方法，将目标检测任务转化为回归问题。
- 通过引入CSPDarknet53作为主干网络，有效地平衡了速度和精度。

- 预测过程中的多尺度特征融合和锚框选择有助于处理不同尺寸的目标。
- 推理速度相对较快，适用于实时性要求高的场景。

DETR

- 采用Transformers结构，以编码器-解码器架构进行目标检测。
- 注意力机制使得模型能够对全局信息进行建模，有助于处理比较复杂的场景。
- 但是由于注意力机制的引入，使得推理速度相对较慢，适用于更注重精确度的任务。

简单来说，YOLOv5的单阶段检测和CSPDarknet53主干网络使其在速度和精度之间取得了平衡，因此在需要实时性能和较高准确度的应用中具有优势。DETR通过注意力机制建模全局信息，能够理解场景的整体结构，但由于注意力机制的复杂性，其推理速度较慢，更适合那些更关注准确度的任务。

2.目标检测精度

YOLOv5

- YOLOv5在多个基准数据集上取得了不错的检测精度，尤其在小目标上表现优越。
- 通过数据增强等技术，能够提升模型的鲁棒性和泛化能力。

DETR

- DETR通过注意力机制实现对目标的整体建模，理论上可以达到很高的精度。
- 在少量目标的情况下，DETR表现出色，但在目标密集的场景中可能存在误差。

简单来说，YOLOv5通过单阶段检测的方式能够高效地定位和识别目标，尤其在小目标上具有明显的优势。DETR利用注意力机制在理论上可以实现更高的精度，但在处理密集目标或者需要精确定位的场景中，其全局注意力可能导致一些定位误差。

3. 对目标数量和尺寸的适应性

YOLOv5

- YOLOv5采用多尺度预测和锚框机制，适应不同尺寸的目标。
- 随着输入尺寸的变化，模型能够在不同目标数量和尺寸下都保持相对较好的表现。

DETR

- DETR通过全局注意力机制不受目标数量限制，能够处理任意数量的目标。
- 在稀疏目标的情况下表现良好，但在密集目标场景可能存在性能下降。

综合看来， YOLOv5通过锚框和多尺度策略，能够较好地适应不同尺寸和密度的目标。DETR利用全局注意力机制实现了任意数量目标的检测，但在密集目标情况下可能由于全局关注而出现性能下降。

4. 数据需求和训练效率

YOLOv5

- YOLOv5通常需要大量的标注数据进行训练，以达到较高的检测精度。
- 数据增强等技术有助于提升模型的泛化能力和鲁棒性。
- 训练相对较快，因为它是一个单阶段的检测模型。

DETR

- DETR在训练时不需要具体的锚框或标注框，通过端到端的方式学习目标检测。
- 在某些情况下，DETR可能需要更多的训练数据来适应其注意力机制的复杂性。
- 由于Transformer结构的引入，训练速度较慢，可能需要更长的时间来收敛。

简单来说，YOLOv5的训练过程相对较快，但需要大量的标注数据。DETR则通过Transformer实现端到端的学习，可能在训练过程中需要更多的数据来适应其注意力机制，且训练速度较慢。

5.多任务处理和定制性能

YOLOv5

- YOLOv5相对容易进行多任务的扩展，可以结合其他任务如分割、跟踪等。
- 可以通过修改模型结构和损失函数来适应不同任务需求。

DETR

- DETR的架构和注意力机制主要设计用于目标检测，不太容易进行多任务扩展。
- 难以进行简单的架构调整，以满足其他目标检测相关任务。

综合看来，由于YOLOv5的设计较为灵活，可以相对容易地进行多任务扩展，而DETR的架构较为专注于目标检测，难以进行简单的任务扩展。

6. 鲁棒性和泛化能力

YOLOv5

- YOLOv5通过数据增强等技术提升鲁棒性，但在某些情况下可能对噪声较敏感。
- 对于目标形状和尺寸的变化相对较为敏感。

DETR

- DETR通过全局注意力机制理论上具有较好的鲁棒性，可以适应不同形状和尺寸的目标。
- 在某些情况下，DETR的注意力机制可能使其更具稳定性，对于一些噪声较少的数据集表现出色。

简单来说，YOLOv5通过数据增强提升鲁棒性，但在面对噪声等情况时可能较为敏感。DETR的全局注意力机制使其在处理不同形状和尺寸的目标时更具优势，更适合在数据噪声较少的情况下表现出色。

7.多尺度和多层次特征融合

YOLOv5

- YOLOv5通过多尺度预测和不同层级的特征融合，能够处理不同大小的目标。
- 通过不同大小的锚框和特征图，实现多尺度的目标检测。

DETR

- DETR的全局注意力机制使其能够对整个场景进行建模，处理多尺度目标较为自然。
- 由于注意力机制的引入，DETR在不同尺度的目标上表现较为均衡。

YOLOv5通过多尺度预测和特征融合，能够适应不同大小的目标，但对于全局信息的建模可能相对较弱。DETR通过全局注意力机制实现了对整个场景的建模，能够较自然地处理多尺度的目标。

8.模型预训练和微调

YOLOv5

- YOLOv5通常使用在大规模数据集上预训练的模型，并通过微调来适应特定任务。
- 可以从预训练模型中迅速获得良好的特征表示。

DETR

- DETR通过自监督预训练的方式进行初始化，然后进行有监督微调。
- 自监督预训练有助于提取更具语义信息的特征，但微调过程可能需要更多数据。

简单来说，YOLOv5通过在大规模数据集上预训练的方式，可以从初始模型中迅速获得特征表示。DETR通过自监督预训练可以获得更具语义信息的特征，但微调过程可能需要更多的数据来适应目标检测任务。

9.模型大小和部署

YOLOv5

- YOLOv5采用了轻量级结构，模型参数较少，适合嵌入式设备和移动端部署。
- 可以在较小的设备上保持较好的性能。

DETR

- DETR使用了Transformer等较大的结构，模型相对较大，不太适合资源有限的设备。
- 部署时需要考虑模型的大小和计算资源需求。

简单来说，YOLOv5通过采用轻量级结构，可以在计算资源受限的设备上保持较好的性能，更适合嵌入式设备和移动端部署。DETR由于较大的模型结构，可能在资源有限的设备上部署相对困难。

B. 优劣势分析

1. YOLOv5的优势与劣势

YOLOv5因其设计上的回归特性和轻量化网络结构，在实时性方面表现出色。其单次前向传递实现快速的目标检测，适用于需要快速响应的自动驾驶场景，例如交通拥堵或紧急情况下的避障。

YOLOv5采用的多尺度检测头使得它能够准确地捕捉不同尺寸目标，从而在复杂场景下取得优异的检测性能。较小感受野和回归设计，使其在小尺寸障碍物的检测方面表现出色。这使得它在城市交通等场景中能够准确地检测到行人和自行车等小型目标。然而，YOLOv5可能在大尺寸目标和遮挡情况下表现不如DETR，在处理遮挡较多的复杂场景时，YOLOv5可能会出现一些性能下降。

2. DETR的优势与劣势

DETR则通过Transformer的自注意机制使得DETR能够全局建模目标之间的关系，特别在处理大尺寸目标、复杂场景和遮挡目标的检测上表现优异。然而，DETR可能在小尺寸目标上的检测精度稍低于YOLOv5，这可能导致其在城市交通中需要对行人和自行车等小型障碍物进行更多努力，影响其在拥挤场景下的性能。

DETR的速度相对较慢，主要受限于Transformer架构的计算复杂性。尽管DETR通过一些优化措施（如并行计算）尝试提高速度，但在某些实时性要求较高的场景中，可能难以达到与YOLOv5相同的实时性。此外，DETR可能需要更多的计算资源来实现与YOLOv5相同的性能。

C. 适用场景分析

1. YOLOv5适用情况

综合考虑，YOLOv5适用于对实时性要求较高的场景，特别是城市交通等需要快速响应的环境。它在小尺寸障碍物的检测、轻量级和实时性方面表现出色。

2. DETR适用情况

DETR适用于对检测精度和全局上下文建模要求较高的场景。它在大尺寸目标、遮挡场景和复杂环境中表现出色，适用于开放道路、高速公路等开放性场景。

VI. 总结

通过这次的大作业，我收获了很多知识以及能力，我在前期学会了各种有意思的小项目，像是使用OpenCV实现的数字识别，全景图像拼接，文档扫描等等有意思的小项目，学会了在flash上部署自己的模型，在做大作业的时候也学会了如何一步一步地创建不同类型的数据集，学会了怎样根据框架来进行模型的训练，学会了各种调优的方式，尽管在过程中一直会遇到各种各样的困难，有的麻烦的也能解决个两三天，但是通过各种方式也都克服了。在最后我还了解到现在有一种YOLO和DETR算法结合的一种方法，号称地表最强目标检测器的DEYO，我相信这会更好地解决YOLO和DETR的劣势，接下来我要去再了解一下这个算法。

我真诚地想加入软件创新实验室这个大家庭，特别喜欢这个氛围和环境，我也非常清楚自己还有很多不足之处，但是我想做的，我一定会去努力做好。不论今后发展的方向如何，希望能够跟各方面的大佬一起学习，一起进步，感谢您的阅读！