Report Title: **自动驾驶轨迹预测**

Author Name: **王小慧**

Student ID: **21307110415**

Institute: **信息科学与工程学院**

Report Date: **2024/12/8**

目录

1 引言

本实验使用了 Bench2Drive 数据集，旨在进行自动驾驶系统中的轨迹预测任务。该任务涉及在复杂的道路环境中根据车辆的历史轨迹和环境信息预测未来的车辆行驶路径，从而为自动驾驶系统提供必要的决策支持。

1.1 自动驾驶

1.1.1 自动驾驶核心任务

自动驾驶技术的核心目标是使车辆能够在没有人工干预的情况下自主完成驾驶任务。这一过程通常涉及到三个主要任务：感知（Perception）、预测（Prediction）和规划（Planning）。这三个任务是自动驾驶系统的核心模块，彼此协作，确保车辆能够安全、有效地应对复杂的道路环境。

- **感知(Perception):** 感知任务的目标是通过传感器（如摄像头、激光雷达（LiDAR）、雷达等）收集周围环境的信息，并将这些数据转化为有意义的结构化信息，如道路、障碍物、行人和交通标志等。
- **预测（Prediction）:** 预测任务基于感知信息，预测自动驾驶车辆在未来一段时间内的运动轨迹，包括其位置、速度和方向；也可以预测其他交通参与者在未来的运动轨迹。例如，预测前方车辆的加速、刹车或变道行为。
- **规划（Planning）:** 规划任务则是基于感知和预测结果，生成具体的控制指令，指引车辆的行驶路线和行为决策。规划不仅关注从起点到终点的全局路径选择，还需要在局部环境中实时做出决策，如避障、变道、加速或刹车等。

这三大任务在自动驾驶系统中紧密配合，每个模块都依赖其他模块的输出，共同确保车辆能够在复杂、动态的道路环境中作出准确、及时的决策。

1.1.2 端到端的预测任务

端到端自动驾驶模型的输出预测通常有两种形式：

- **轨迹/路径点（Trajectory/Waypoints）:** 在这种形式中，模型的输出是关于车辆未来路径的预测，即车辆应该遵循的轨迹或路径点（waypoints）。这些路径点是一系列预定的位置，指示车辆在特定时间应该处于哪里。这种输出形式侧重于规划车辆的行驶路线，而不是直接控制车辆的实际操作。轨迹或路径点的预测通常用于那些需要考虑车辆在未来一段时间内运动的场合，比如在复杂的交通环境中避障或规划最佳行驶路径。
- **控制动作（Control Actions）:** 这种形式的输出是直接的控制信号，如方向盘角度（steering angle）、油门（throttle）和刹车（braking）的指令。这些控制动作直

接应用于车辆的控制系统，以操纵车辆的实际运动。直接控制动作的预测更侧重于即时的反应和车辆的直接操纵，而不是预先规划的路径。

1.1.3 自动驾驶测试方法

自动驾驶测试通常分为两种主要类型：开环测试（Open Loop）和闭环测试（Closed Loop）。

- **开环测试（Open Loop）：**开环测试是基于预录制数据进行的测试，不需要动态环境，不依赖于系统的反馈信息，模型基于已有的输入数据进行预测。该测试能够有效评估模型在没有反馈调节的情况下的预测能力，与闭环测试相比更易于实现。
- **闭环测试（Closed Loop）：**与开环测试不同，闭环测试依赖于系统反馈。在这种测试中，模型不仅需要根据当前输入进行预测，还需要根据车辆的实时行为和环境信息进行动态调整。闭环测试更能模拟现实世界中的驾驶场景，能够更好地评估自动驾驶系统在实际驾驶中对环境变化的应对能力。闭环测试常使用 CARLA 模拟器。

基于算力和时间成本考量，本实验中我们主要选择开环测试方法，评价指标为 L2 误差，用于计算预测轨迹与真实轨迹的误差，较低的 L2 值表示预测轨迹与真实轨迹更加接近，模型的性能更好。

1.2 Bench2Drive 数据集

Bench2Drive 数据集 [1] 是一个包含多种不同驾驶场景的多模态数据集，适用于自动驾驶任务中的轨迹预测和驾驶行为建模。该数据集包含了 10000 条来自不同环境和天气条件下的驾驶轨迹，数据集的场景可分为五种主要的能力类别：**合并 (Merging)**、**超车 (Overtaking)**、**紧急刹车 (Emergency Brake)**、**让路 (Give Way)**、**交通标志 (Traffic Sign)**，每个类别代表了自动驾驶系统所面临的不同挑战。能力与场景对应图见图 1.

在本实验中，我们以这些能力类别为核心，进行模型训练和评估，旨在探索如何通过不同的算法和数据融合方法，提高自动驾驶系统在各种复杂场景中的轨迹预测精度。由于算力以及存储问题，我们无法实现全数据存储和训练，故基于在现实场景的重要度，选择交通标志 (Traffic Sign) 和紧急刹车 (Emergency Brake) 两种能力进行测试。其中交通标志 (Traffic Sign) 选用 **113** 条轨迹数据，紧急刹车 (Emergency Brake) 选择 **114** 条轨迹数据，每种能力中包含五种场景的数据，每种场景约 20 条数据，在每种场景中划分 20% 的数据作为测试集。

Skill	Scenario
Merging	CrossingBicycleFlow, EnterActorFlow, HighwayExit, InterurbanActorFlow, HighwayCutIn, InterurbanAdvancedActorFlow, MergerIntoSlowTrafficV2, MergeIntoSlowTraffic, NonSignalizedJunctionLeftTurn, NonSignalizedJunctionRightTurn, NonSignalizedJunctionLeftTurnEnterFlow, ParkingExit, LaneChange, SignalizedJunctionLeftTurn, SignalizedJunctionRightTurn, SignalizedJunctionLeftTurnEnterFlow
Overtaking	Accident, AccidentTwoWays, ConstructionObstacle, ConstructionObstacleTwoWays, HazardAtSideLaneTwoWays, HazardAtSideLane, ParkedObstacleTwoWays, ParkedObstacle, VehicleOpenDoorTwoWays
Emergency Brake	BlockedIntersection, DynamicObjectCrossing, HardBreakRoute, OppositeVehicleTakingPriority, OppositeVehicleRunningRedLight, ParkingCutIn, PedestrianCrossing, ParkingCrossingPedestrian, StaticCutIn, VehicleTurningRoute, VehicleTurningRoutePedestrian, ControlLoss
Give Way	InvadingTurn, YieldToEmergencyVehicle
Traffic Sign	EnterActorFlow, CrossingBicycleFlow, NonSignalizedJunctionLeftTurn, NonSignalizedJunctionRightTurn, NonSignalizedJunctionLeftTurnEnterFlow, OppositeVehicleTakingPriority, OppositeVehicleRunningRedLight, PedestrianCrossing, SignalizedJunctionLeftTurn, SignalizedJunctionRightTurn, SignalizedJunctionLeftTurnEnterFlow, TJunction, VanillaNonSignalizedTurn, VanillaSignalizedTurnEncounterGreenLight, VanillaSignalizedTurnEncounterRedLight, VanillaNonSignalizedTurnEncounterStopsign, VehicleTurningRoute, VehicleTurningRoutePedestrian

图 1 能力与场景对应图

1.3 注意力机制

1.3.1 交叉注意力机制

交叉注意力机制（Cross-Attention）是一种用于多模态数据融合的技术，它通过对来自不同模态的信息进行交互式的注意力计算，能够有效地捕捉不同模态之间的关联和互补信息。Cross -Attention 在自动驾驶的轨迹预测中具有重要作用，尤其是在融合多源传感器数据时。在自动驾驶系统中，传感器（如摄像头、激光雷达、雷达等）提供了环境信息，交叉注意力机制能够有效地将这些来自不同模态的数据进行融合，并动态地调整各个模态之间的权重，使得模型能够更好地捕捉到轨迹预测中的关键特征。

Cross-Attention 的核心思想是通过查询（query）、键（key）和值（value）的相互作用，结合来自不同模态的信息，增强对关键信息的关注。

1.3.2 卷积块注意力模块 (CBAM)

CBAM（Convolutional Block Attention Module）[2] 是一个轻量级的模块，旨在通过通道注意力机制（Channel Attention）和空间注意力机制（Spatial Attention）相结合，来增强卷积神经网络对重要特征的聚焦能力。该模块能够在不同层次上精确调节特征图，从而提高模型的表现，尤其在自动驾驶轨迹预测中，CBAM 可以有效帮助模型提取与预测轨迹相关的关键信息。CBAM 示意图见图 2.

- **通道注意力机制：**在轨迹预测任务中，车辆的未来运动轨迹受多种因素的影响，包括周围环境、其他交通参与者和自车的状态信息。通过通道注意力机制，CBAM 能够动态地调整各通道的权重，从而突出与轨迹预测相关的特征。例如，特征图

中的某些通道可能包含关于障碍物的重要信息，通道注意力机制将自动增强这些信息的表达，以便模型能够更好地做出决策。

- **空间注意力机制：**空间注意力机制则通过对空间维度上的特征进行加权，帮助模型聚焦于对轨迹预测至关重要的区域。例如，在预测未来轨迹时，前方道路的拓扑结构、交通流量以及障碍物的位置都可能对决策产生显著影响。空间注意力机制能够让模型集中关注这些区域，忽略不相关的背景信息，提升轨迹预测的精度。

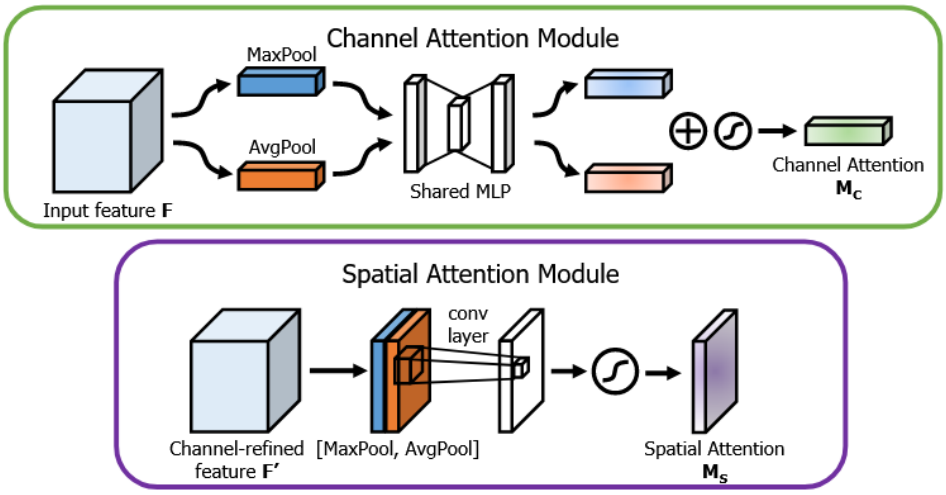


图 2 CBAM 主要工作流程

CBAM 通过同时融合通道和空间维度的注意力机制，能够有效捕捉图像或其他传感器数据中的关键特征，对于复杂的自动驾驶场景，尤其是在动态变化的道路环境中，CBAM 能够提高模型的鲁棒性和预测精度。通过这种机制，轨迹预测模型能够更好地应对不同的环境变化和交通状况，从而提高驾驶安全性和效率。

2 相关工作

2.1 模型综述

在自动驾驶领域，轨迹预测任务一直是一个重要的研究方向，近年来，随着深度学习技术的发展，多个模型应运而生，旨在提高轨迹预测的准确性和鲁棒性。以下是一些具有代表性的模型：

BevFormer[3] 是一个新型框架，它通过 temporal self-attention 和 spatial cross-attention 学习统一的鸟瞰（BEV）表示，以支持多种自动驾驶感知任务。BEVFormer 利用空间和时间信息，通过与空间和时间空间的交互来聚合信息。

TCP[4] 通过结合轨迹规划和直接控制来提高自动驾驶的性能。TCP 框架包含两个分支：轨迹规划分支和控制分支。轨迹分支负责预测未来的轨迹，而控制分支则采用多

步预测方案，推理当前动作与未来状态之间的关系。这两个分支相互连接，控制分支在每个时间步骤接收来自轨迹分支的相应指导。通过融合两个分支的输出，TCP 实现了互补优势。在 CARLA 中进行的评估表明，即使仅使用单目摄像头输入，TCP 方法在 CARLA Leaderboard 上也排名第一，大幅超越了其他使用多个传感器或融合机制的复杂模型。

UniAd[5] 是一个基于统一架构的端到端自动驾驶模型，它将感知 (Perception)、预测 (Prediction) 和规划 (Planning) 任务整合到一个统一的框架中，以规划 (Planning) 为导向，并将所有这些任务优先考虑以支持规划，充分利用各个模块之间的互补特性，提升了自动驾驶系统的整体性能和安全性。UniAD 在 nuScenes 基准测试中的表现显著超越了以往的技术水平，在所有方面都显示出优越性。

VAD[6] 是一种端到端的向量化自动驾驶范例，它将驾驶场景建模为完全向量化的表示。VAD 提出了两个主要优势：一方面，VAD 利用向量化的代理运动和地图元素作为明确的实例级规划约束，有效提高了规划的安全性；另一方面，VAD 比以往的端到端规划方法运行速度更快，摆脱了计算密集型的光栅化表示和手工设计的后处理步骤。VAD 在 nuScenes 数据集上实现了最先进的端到端规划性能，大幅超越了之前的最佳方法。

AD-MLP[7] 是一个基于 MLP 的方法，该方法直接从原始传感器数据输出自我车辆的未来轨迹，而不使用任何感知或预测信息，如摄像头图像或 LiDAR。这一简单的方法在 nuScenes 数据集上与其他基于感知的方法取得了相似的端到端规划性能，减少了平均 L2 误差约 20%。

2.2 TCP 模型详解

本实验我们主要基于 TCP 模型，以提升性能为目标，对模型进行修改优化。TCP 模型的整体框架图如图 3所示，主要包含编码器（Encoder）、轨迹分支（Trajectory Branch）、控制分支（Control Branch）。

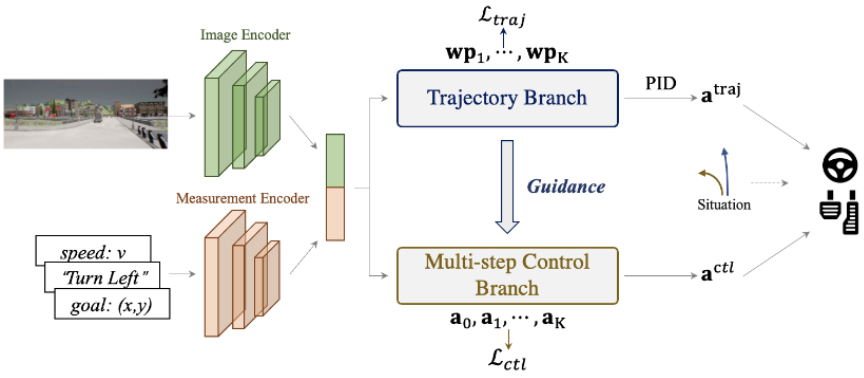


图 3 TCP 整体框架图

2.2.1 编码阶段（Image Encoder and Measurement Encoder）

图像编码和自车状态编码是两个关键的输入特征处理部分，它们分别负责从图像数据和自车的状态信息中提取有用的特征，并将这些特征传递到后续的分支。图像编码的目的是从输入图像中提取出能够有效表征环境信息的特征；自车状态编码的目的是将自车的当前状态信息（例如位置、速度、加速度等）转化为一个有意义的特征表示，供模型理解当前智能体（自车）在环境中的状态。

- 图像数据主要来自于前摄像头传感器的前图像（Front Image），图像编码使用 ResNet34.
- 自车状态主要包含导航信息 g、当前速度 v、控制信号 a，这些信息会被拼在一起形成测量输入（Measurement）m，状态编码使用 MLP.
- 图像编码得到的特征图和自车状态编码得到的特征会在后续的网络层中被拼接在一起，形成一个共享的特征表示。这些特征随后会被传递到两个分支：轨迹规划分支和控制预测分支。

2.2.2 轨迹分支（Trajectory Branch）

轨迹分支的主要任务是预测车辆在未来一段时间内的运动轨迹。这一分支通过分析历史和当前的传感器数据，使用深度学习模型来预测车辆在未来几个时间步长的位置和速度。轨迹分支主要包括以下部分：

- **GRU(门控递归单元)：**将输入的特征（Image Feature 和 Measurement Feature 拼接后的结果）输入到 GRU 中，通过自回归的方法逐步生成未来的路径点（Waypoint）。
- **PID 控制器：**生成的轨迹将被传递给 PID 控制器，计算出纵向和横向控制动作。

2.2.3 控制分支（Control Branch）

为了更好地应对复杂的驾驶场景，TCP 模型引入了多步预测方案，使得控制分支不仅关注当前时刻的控制信号，还考虑未来多个时间步长的控制信号，因此控制分支中也含有一个 GRU，用于预测控制信号（如转向、油门、刹车）。

然而，直接从当前的传感器输入推断未来时间步应该关注图像中的哪些区域是具有挑战性的。传统的方法通常仅依赖当前的图像信息来做出控制决策，但图像中的关键信息（如车道线、障碍物、交通标志等）往往分布在不同的区域。因此，如何在动态环境下有效选择图像中的关键区域变得至关重要。为了解决这一问题，TCP 模型设计了**轨迹引导注意力机制**，即通过轨迹规划分支的信息来帮助控制分支关注图像中的关键区域。具体实现见图 4.

- **轨迹分支输出：**通过轨迹分支的预测，模型获得未来多个时间步的轨迹预测。这个轨迹预测不仅告诉我们车辆目前的位置，还提供了预计的运动方向和可能的路线。

- **控制分支关注区域：**基于轨迹预测信息，控制分支能够确定哪些图像区域对于当前时刻的控制决策尤为重要。例如，车辆即将转弯时，车道线、交通标志和其他障碍物的位置信息会变得尤为关键。
- **计算注意力图：**轨迹分支生成的隐藏状态（Waypoint GRU Output）会与控制分支的隐藏状态（Temporal Module Output）结合，通过 MLP 计算得到注意力图 w_t。该注意力图反映了不同图像区域的关注程度，指示了哪些区域在当前时刻对控制决策有更大影响。
- **加权图像特征：**通过计算得到的注意力图 w_t，模型对输入图像的特征图 F 进行加权求和，提取出与控制决策相关的重要信息。这些加权后的图像特征与自车状态信息一起，作为控制分支的输入。

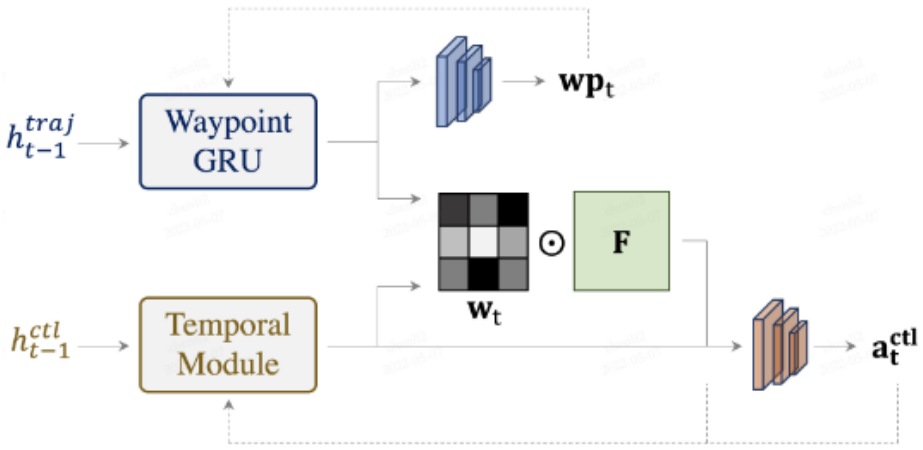


图 4 轨迹引导注意力机制示意图

3 实验步骤（个人工作）

3.1 模型复现

在本实验中，我们对开环的 TCP 模型及 AD-MLP 模型进行了复现，主要测试其在两个关键能力场景下的表现：Traffic Sign 和 Emergency Brake。Traffic Sign 共包含 113 条数据，其中 92 条数据为训练集（包含验证集），21 条数据为测试集；Emergency Brake 共包含 114 条数据，其中 92 条为训练集（包含验证集），22 条数据为测试集。所有数据应先通过 **tools/gen_tcp_data.py 以及 tools/gen_admlp_data.py** 生成数据索引方便模型训练与测试。两种能力各涵盖五种场景，详细见表 1.

- **TCP 模型复现：**我们首先复现了开环 TCP 模型，模型输入为图像（front image）和自车状态（包括速度、位置等），模型设置遵循原始设置。

- **AD-MLP 模型复现：**为了对比性能，我们还复现了 AD-MLP 模型。AD-MLP 仅使用自车状态作为输入，不涉及任何感知模块，使用一个 MLP 模型进行预测任务。

表 1 数据选取

能力	场景
`Traffic Sign`	CrossingBicycleFlow, OppositeVehicleRunningRedLight, SignalizedJunctionLeftTurn, SignalizedJunctionRightTurn, VanillaNonSignalizedTurnEncounterStopsign
`Emergency Brake`	BlockedIntersection, DynamicObjectCrossing, PedestrianCrossing, VehicleTurningRoute, VehicleTurningRoutePedestrian

3.2 图像特征提取网络优化

为了进一步提升 TCP 模型的图像特征提取能力，我们将原始的 ResNet34 替换为 **ResNet50**，ResNet50 的输出通道数从 512 增加到 2048，这使得网络在处理图像特征时能够捕捉到更加细致和丰富的空间信息。但由于通道数的增加，模型的参数量也大幅增加，这可能导致计算负担的增大。为了平衡性能与计算复杂度，我们在 ResNet50 的输出之后引入了一层**卷积模块**。该模块的作用是将 2048 维的输出通道降维为 512 维度，从而减少后续模块的参数量，确保模型能够高效运行。该降维操作通过卷积层实现，避免了直接全连接层带来的过多参数。

```
1 class CnnFeatureHead(nn.Module):
2     def __init__(self, in_channels=2048, out_channels=512):
3         super(CnnFeatureHead, self).__init__()
4         self.conv = nn.Conv2d(in_channels, out_channels, kernel_size=1, stride=1,
              padding=0)
5 
6     def forward(self, x):
7         x = self.conv(x)
8         return x
```

3.3 多模态数据融合

本部分我们尝试融入多模态的数据，观察数据集后发现，camera 传感器部分包含四种不同模态的图像，分别为：Depth、RGB、Instance、Semantic，每种模态图像侧重点不同，故考虑将不同模态的图像数据融合。我们选取 Depth、RGB、Instance 作为输入图像数据，在经过 ResNet 后，将提取出的特征通过 Cross-Attention 模块进行融合。

为了全面感知环境信息，我们设计了基于 Cross-Attention 的多模态融合模块，分别进行了多角度摄像头数据融合和多模态图像数据融合实验，通过统一的特征提取和融合策略提升模型性能。

- **多角度摄像头数据：**使用来自多个摄像头（例如 front, front_left, front_right, back, back_left, back_right）的图像数据，每个摄像头提供了不同角度下的环境信息。在具体操作时，由于侧边摄像头所附带的有用信息较少，我们将侧边摄像头的一部分与前后摄像头进行简单的拼接（front_left + front + front_right），最终输入模型的仅为两张图像。
- **多模态图像数据：**单个摄像头的图像数据包含多种模态，包括深度图（Depth）、RGB 图像（RGB）、实例分割图（Instance）等，每种模态提供不同层次的信息，例如深度图用于估计距离，RGB 图像用于捕捉纹理和颜色，实例分割用于区分场景中的不同物体。具体区别可见图 5.

图 5 摄像头数据的三种模态

以上两个实验采用了统一的处理范式，首先对各类图像数据分别通过共享的 ResNet 网络进行特征提取，得到 **rgb_feature, depth_feature, instance_feature**。随后，这些提取的特征先使用 **torch.stack()** 函数堆叠起来，再输入到 Cross-Attention 模块中进行融合，实现多角度和多模态信息的高效整合。以下是 Cross-Attention 的实现：

```
class CrossAttention(nn.Module):
    def __init__(self, in_dim, out_dim, device='cuda'):
        super(CrossAttention, self).__init__()
        self.query = nn.Linear(in_dim, out_dim).to(device)
        self.key = nn.Linear(in_dim, out_dim).to(device)
        self.value = nn.Linear(in_dim, out_dim).to(device)

    def forward(self, x):
        q = self.query(x)
        k = self.key(x)
        v = self.value(x)
        attn_weights = torch.einsum('bqi,bki->bqk', q, k) / (x.size(-1) ** 0.5)
        attn_weights = torch.softmax(attn_weights, dim=-1)
        out = torch.einsum('bqk,bki->bqi', attn_weights, v)
        return out.mean(dim=1)
```

3.4 卷积块注意力机制

在前文中提到，原模型在控制分支中采用了轨迹引导注意力机制，用于引导控制分支关注图像中的关键区域，显著提升了对目标区域的敏感性。借鉴这一思路，我们对图像特征提取模块进一步优化 (未加入多模态)。具体而言，我们考虑在模型中引入**空间注意力 (Spatial Attention)** 和**通道注意力 (Channel Attention)** 机制，以更细致地挖掘和聚焦图像特征中的重要信息。两种注意力机制代码如下：

```python
# 通道注意力
class ChannelAttention(nn.Module):
    def __init__(self, in_planes, ratio=16):
        super(ChannelAttention, self).__init__()
        self.avg_pool = nn.AdaptiveAvgPool2d(1)
        self.max_pool = nn.AdaptiveMaxPool2d(1)

        self.fc = nn.Sequential(nn.Conv2d(in_planes, in_planes // 16, 1, bias=False), nn.ReLU(), nn.Conv2d(in_planes // 16, in_planes, 1, bias=False))
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        avg_out = self.fc(self.avg_pool(x))
        max_out = self.fc(self.max_pool(x))
        out = avg_out + max_out
        return self.sigmoid(out)

# 空间注意力
class SpatialAttention(nn.Module):
    def __init__(self, kernel_size=7):
        super(SpatialAttention, self).__init__()
        self.conv1 = nn.Conv2d(2, 1, kernel_size, padding=kernel_size//2, bias=False)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        avg_out = torch.mean(x, dim=1, keepdim=True)
        max_out, _ = torch.max(x, dim=1, keepdim=True)
        x = torch.cat([avg_out, max_out], dim=1)
        x = self.conv1(x)
        return self.sigmoid(x)
```

尝试将其嵌入 ResNet50 的不同位置，包括：

- 浅层网络：第一层卷积层 Conv1 后。
- 深层网络：Layer4 后。
- 模块内部：BasicBlock 和 BottleNeck 中。

3.5 不同能力融合

在前述实验中，我们分别对 Traffic Sign（交通标志识别）和 Emergency Brake（紧急刹车）这两种能力进行了独立的测试。为了进一步提升模型的综合性能，我们将这两种能力的数据合并测试，观察它们在合并后是否能为单个能力带来性能提升。

4 结果与分析

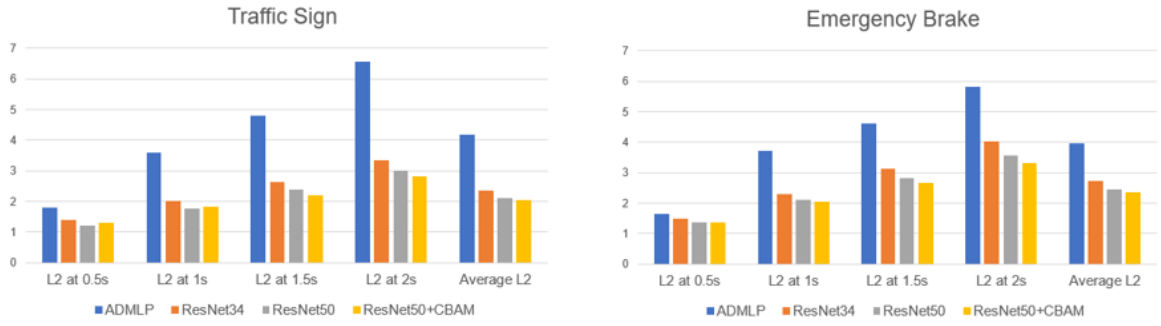
完整的测试结果可见表 2以及表 3，对比图见图 6.

表 2 Traffic Sign 复现结果

模型	L2(0.5s)	L2(1s)	L2(1.5s)	L2(2s)	L2(Avg)
TCP	1.389	2.0095	2.6369	3.3362	2.3429
ADMLP	1.8038	3.5797	4.805	6.5609	4.1874
TCP(ResNet50)	1.2227	1.7796	2.3904	3.0074	2.1
TCP(CBAM)	1.3186	1.8436	2.2006	2.8154	**2.0445**
TCP(CBAM) 能力融合	1.3367	1.9159	2.4805	3.0813	2.2036

表 3 Emergency Brake 复现结果

模型	L2(0.5s)	L2(1s)	L2(1.5s)	L2(2s)	L2(Avg)
TCP	1.4764	2.2912	3.1363	4.0387	2.7357
ADMLP	1.6572	3.7261	4.6119	5.8027	3.9495
TCP(ResNet50)	1.3586	2.0949	2.8158	3.5648	2.4585
TCP(CBAM)	1.3714	2.0326	2.6643	3.3236	2.3479
TCP(CBAM) 能力融合	1.2809	1.9468	2.6224	3.3382	**2.297**



(a) Traffic Sign 结果对比　　(b) Emergency Brake 结果对比

图 6 结果对比图

4.1 复现结果

在对原始模型进行复现时，我们分别对 Traffic Sign 和 Emergency Brake 能力进行测试。从整体结果来看，TCP 模型的性能显著优于 ADMLP 模型。但有趣的是，针对 TCP 模型，Traffic Sign 能力的表现优于 Emergency Brake，而在 ADMLP 模型中，Emergency Brake 能力的表现反而优于 Traffic Sign。

这一现象可能与任务的复杂性和特征表达方式有关。具体而言，Traffic Sign 任务通常具有较为固定和明确的目标，模型能够较快从图像中提取出有效特征；而 Emergency Brake 任务则需要模型对环境进行动态预测，并且对时序信息和快速变化的场景具有更高的要求。因此，ADMLP 模型可能更擅长处理与时序相关的特征，从而在紧急刹车任务上表现得更好。TCP 模型则可能在结合图像特征时具有优势，从而使得交通标志识别的表现优于紧急刹车。

4.2 图像特征提取网络优化结果

通过将 ResNet34 替换为 ResNet50，并进行通道降维操作，整体模型的性能得到了显著提升。尤其在长时间预测（例如 1.5 秒和 2 秒）的任务中，性能提升接近 **30%**。

这种改进可以归因于 ResNet50 具有更多的网络层数和更深的网络结构，相较于 ResNet34 能够提取更丰富的图像特征。通道降维操作则有助于减少冗余信息，保持关键特征，从而提高模型的推理效率和准确性。在长时间预测中，模型需要依赖更多的上下文信息和细节特征，ResNet50 的深层结构为这一任务提供了更强的特征表达能力，从而实现了更好的预测性能。但短时预测任务通常依赖于对即时图像特征的快速反应。ResNet50 相较于 ResNet34 虽然具有更深的网络结构，能够提取更丰富的特征，但在短时任务中，可能导致模型过度依赖深层特征和上下文信息，而忽略了浅层特征的细节。

4.3 多模态图像融合结果

在引入 Cross-Attention 后，我们观察到运行时长显著激增。因此，我们仅选取了一个场景进行验证性实验。实验结果见表 4.

表 4 Cross-Attention 对比实验

方法	L2(0.5s)	L2(1s)	L2(1.5s)	L2(2s)	L2(Avg)
原始模型	1.1782	1.5155	1.8687	2.2973	1.7149
多模态图像融合	1.3040	1.6489	2.0556	2.5345	1.8857
多角度图像融合	1.2507	1.8083	2.3836	2.8275	2.0675

实验结果表明，加入 Cross-Attention 后，模型的性能并未出现预期的提升。我们推测，可能的原因有两个：首先，数据量较少可能导致模型在训练过程中发生了欠拟合，

未能有效学习到多模态信息；其次，Cross-Attention 的融合方式可能过于直接，导致特征冗余度过高，从而影响了模型的学习效率。Cross-Attention 机制在处理多模态信息时，如果特征的互补性较弱，或者数据量不足以支撑复杂模型的训练，可能反而会造成性能的下降。

4.4 卷积块注意力机制结果

我们首先对比了在三种不同位置加入 CBAM 的效果，见图 7，可以观察到以下现象：

- 在 BottleNeck 和 BasicBlock 中引入 CBAM 时，模型性能较差。这是由于改变了 ResNet 原有的结构，导致 pre-trained 模型参数无法加载，从而使得模型无法有效学习到特征。
- 在 Conv1 后加入 CBAM 的总体性能优于在 Layer4 后加入 CBAM，这可能是因为在浅层网络中，模型更关注全局特征，而通过全局特征先定位出感兴趣的区域比在深层网络中关注局部特征更加高效。在深层网络中，模型更侧重于高维的特征和局部细节，因此引入注意力机制后可能会面临更高的特征冗余度，导致性能的提升幅度不如在浅层网络加入。

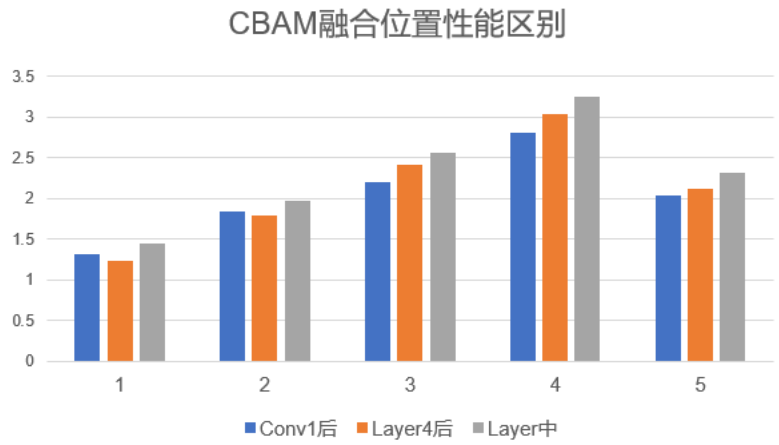
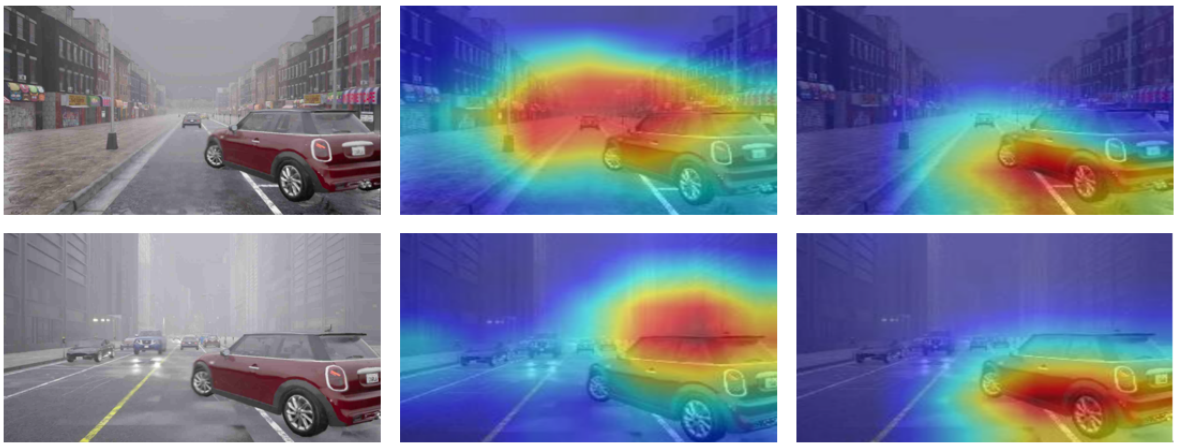


图 7 CBAM 融合位置性能对比

为了更加直观地观察模型的关注区域，我们对提取后的图像进行了类激活图绘制，可见图 8.

通过观察，使用 ResNet50 优化的模型提取到的特征主要集中在车顶或前方区域，而这时的场景涉及到前车变道和插入车道的行为。车顶或前方并不完全符合我们对这一行为的关注区域。相比之下，使用 CBAM 优化后的模型更倾向于关注插队车所在的路面区域，这与我们的直观认知相符。当前方车辆变道时，模型能够识别到前方车的偏向角度，从而作出更加合理的决策。



(a) 原始图像 (b) 类激活图(ResNet50优化模型) (c) 类激活图(CBAM优化模型)

图 8 类激活图对比

总体而言，实验结果表明，CBAM 的引入在一定程度上提升了模型的性能，特别是在长时预测中的表现更加准确，且平均误差较小。这表明，卷积块注意力机制能够有效提升模型对关键区域的关注，从而改善模型在复杂任务中的预测精度。

4.5 不同能力融合结果

我们使用 CBAM 优化后的模型（加入位置为 ResNet50 的 Conv1 后），将 Traffic Sign 和 Emergency Brake 数据合并进行训练并分别测试，可以观察到，Traffic Sign 的结果有所下降，而 Emergency Brake 的结果有所提升。该现象可能有以下的原因：

- **能力间的干扰：** Traffic Sign 和 Emergency Brake 是两个不同的任务，其中一个主要侧重于图像中的静态信息（交通标志识别），而另一个则更多依赖动态信息（紧急刹车预测）。在任务合并后，模型需要同时学习两个任务的特征，这可能导致任务之间相互干扰，尤其是当数据量有限时，模型可能难以平衡不同任务的特征学习，导致 Traffic Sign 的性能下降。

- **模型过度优化误差较大的任务：** 如果 Emergency Brake 的能力下损失值较大，模型可能会调整更多的参数来降低这个任务的误差，模型的学习目标偏向于优化 Emergency Brake，导致 Traffic Sign 的学习和优化受到压制。这可能会导致模型在 Emergency Brake 任务上过拟合，尽管这会提升 Emergency Brake 的准确性，但可能会牺牲其他任务的泛化能力，导致 Traffic Sign 任务的性能下降，特别是在数据量有限时，模型更容易出现这种过拟合现象。

5 讨论与总结

5.1 改进方向

在本研究中，我们探索了图像特征提取和多能力融合对自动驾驶决策系统的作用，在多模态融合和多能力融合方面仍有很大的改进空间。

首先，关于多模态融合，我们使用 Cross-Attention 在后期对不同模态的特征进行融合。然而，单纯地将 Cross-Attention 加到现有模型中不仅没有提升性能，而且增大了计算开销，影响了整体效率。因此，未来的研究不应简单地在特征提取后加上 Cross-Attention，而应考虑在特征提取的过程中就进行融合，改变简单的特征提取范式。例如，可以参考 **BevFormer** 的融合方式，在网络的初期阶段就融合这些信息，避免在后期融合时的冗余和信息损失。也可以考虑融合不同传感器的特征，例如 Camera 与 Lidar，可参考的模型有 **BevFusion**。

其次，关于多能力融合，在合并不同能力时，某些能力场景的性能有所下降。为了避免这种问题，或许可以借鉴 **Multi-Task Learning（MTL）** 的方法，通过共享底层网络的特征提取层，让不同能力任务从相同的特征空间中获取信息，再使用对每种能力独立的分类头。但由于能力种类较多，为每一种能力设定分类头可能会影响能力间的交互，因此可以考虑给能力再进行分类，例如分为动态能力和静态能力等，使得能力类内差异小、类间差异大。

最后，还可以考虑引入**目标检测**的任务，目标检测任务的目标是定位和分类图像中的特定对象（如车辆、行人、交通标志等），而预测任务则侧重于基于这些对象信息进行决策或行为预测（如判断是否需要紧急刹车、识别交通标志等），这两者之间是高度互补的。

5.2 感想与总结

在本次实验中，我们探讨了特征提取网络优化、多模态融合、多能力融合对自动驾驶任务性能的影响。通过实验，我们验证了不同模型架构和优化策略的有效性，并提出了研究改进方向。

在研究初期，我对自动驾驶领域并不了解，特别是目前很多自动驾驶模型都是端到端的复杂模型，并且都涉及到 CARLA，让刚接触的我感觉眼前一黑。但在不断阅读文献的过程中，终于逐渐了解了自动驾驶的任务分类、常见的技术方法和评测方法。

尽管理论上的积累比较丰富，但缺乏足够的实践经验使得很多想法难以转化为实际的实验成果，在实际的代码实现过程中，我遇到了许多预料之外的困难。许多实验设想与模型的实际实现存在不小的差距，尤其在集成多个模块时，模型的性能表现往往并不如预期。在实验过程中，我深刻认识到，提升模型性能并非单纯依赖堆砌更多的模块，而是需要合理设计模块的融合方式，精心选择合适的算法和架构。单一模块的性能提升无法直接映射为整体系统的性能提升，如何融合不同任务和模态的信息，如何处理各个

模块之间的交互关系，都是值得深入思考和探索的问题。

通过本次实验，不仅对自动驾驶领域的技术前沿有了更深的认识，还提升了自己的代码能力和模型设计能力。本次实验中也有不少值得深入研究的问题，这也激发了我进一步探索该领域的兴趣。但我也意识到，在今后的研究中，光有想法是不够的，关键在于如何将这些想法落实到具体的代码实现和实验验证上，每一个新的算法、每一次新的模块引入，都需要仔细的调试与调整。

参考文献

[1] Xiaosong Jia, Zhenjie Yang, Qifeng Li, Zhiyuan Zhang, and Junchi Yan. Bench2drive: Towards multi-ability benchmarking of closed-loop end-to-end autonomous driving. *ArXiv*, abs/2406.03877, 2024.

[2] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In-So Kweon. Cbam: Convolutional block attention module. *ArXiv*, abs/1807.06521, 2018.

[3] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *ArXiv*, abs/2203.17270, 2022.

[4] Peng Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. *ArXiv*, abs/2206.08129, 2022.

[5] Yi Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wen Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17853–17862, 2022.

[6] Bo Jiang, Shaoyu Chen, Qing Xu, Bencheng Liao, Jiajie Chen, Helong Zhou, Qian Zhang, Wenyu Liu, Chang Huang, and Xinggang Wang. Vad: Vectorized scene representation for efficient autonomous driving. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8306–8316, 2023.

[7] Jiang-Tian Zhai, Ze Feng, Jinhao Du, Yongqiang Mao, Jian Liu, Zichang Tan, Yifu Zhang, Xiaoqing Ye, and Jingdong Wang. Rethinking the open-loop evaluation of end-to-end autonomous driving in nuscenes. *ArXiv*, abs/2305.10430, 2023.