

# 单台设备上的 2 级自动驾驶:潜入 Openpilot 的恶魔

陈黎<sup>1, \*, †</sup>, 唐图田<sup>2, \*</sup>, 蔡志田<sup>1, \*</sup>, 李洪阳<sup>1, \*</sup>, 吴鹏浩<sup>3</sup>, 李洪阳<sup>1, \*, 2</sup>,  
†, 石建平<sup>4</sup>, 闫俊池<sup>1, 2</sup>, 乔宇<sup>1\*</sup> 等贡献<sup>†</sup>项目负责人

<sup>1</sup> 上海人工智能实验室 <sup>2</sup> 上海交通大学 <sup>3</sup> UCSD <sup>4</sup> 商汤科技

chenli1@pjlab.org.cn ttatang@sjtu.edu.cn

## 摘要

由于配备了广泛的传感器，主流的自动驾驶解决方案在安全系统设计方面正变得更加模块化。虽然这些传感器已经奠定了坚实的基础，但目前大多数量产解决方案仍处于 2 级 (L2，部分驾驶自动化) 阶段。这其中，逗号.ai 出现在我们的视线中，声称一款 999 美元的售后设备内置一个摄像头和电路板，拥有处理 L2 场景的能力。再加上逗号发布的整个系统的开源软件。这个项目被命名为“Openpilot”。有可能吗？如果有，是如何实现的？带着好奇心，我们深入研究了 Openpilot，并得出结论，它成功的关键是端到端系统设计，而不是传统的模块化框架。Openpilot 内部的模型被称为 Supercombo，它可以从单目输入实时预测自我车辆的未来轨迹和其他道路语义。遗憾的是，实现所有这些工作的训练过程和海量数据并没有向公众公开。为了实现深入的调查，我们试图重新实现训练细节，并在公共基准上测试管道。在这项工作中提出的重构网络被称为 **OP-Deeptide**。为了将我们的版本与原始的 Supercombo 进行公平的比较，我们引入了双模型部署方案来测试现实世界中的驾驶性能。OP-Deeptide 在 nuScenes、Comma2k19、CARLA 和内部现实场景(在上海收集)上的实验结果验证了低成本设备确实可以实现大多数 L2 功能，并与原始 Supercombo 模型相当。在这份报告中，我们希望与观众分享我们的最新发现，从工业产品层面阐明端到端自动驾驶的新视角，并潜在地激励社区根据这项工作提供的环境继续提高性能。我们的代码，数据集，基准可以在 <https://github.com/OpenPerceptionX/Openpilot-Deeptide> 上获得。

内 容	
1 介绍	3
2 预赛	4
2.1 Openpilot 概述	4
2.2 Supercombo 模式	-
3 的方法	6
3.1 一个端到端的规划模型	6
3.2 在设备部署	8
4 实验	9
4.1 Openpilot: 一个真实的测试	9
4.2 OP-Deepdive: 我们对 Supercombo 的重新实现	9
4.2.1 数据准备	9
4.2.2 指标	10
4.2.3 实现细节	10
4.2.4 定量结果	11
4.3 问题的对偶模型实验	12
4.3.1 卡拉模拟试验	12
4.3.2 真实世界的测试	13
5 讨论	14
5.1 开环闭环系统 vs	14
5.2 模仿度规	14
5.3 “调试” 一个端到端的模式	14
5.4 双模部署方案的原因	15
6 结论和未来工作	15
Openpilot 分析	17
. Openpilot: 一个真实的测试	17
A.1.1 定性结果	17
A.1.2 定量结果	18
A.1.3 失败案例	19





图 1:左 :Openpilot 通过单台设备实现 L2。右 :鸟瞰图预测轨迹(pred0-4) vs 人类驾驶员(gt)。

## 1 介绍

自动驾驶见证了解决方案的爆炸式增长，其能力和容量从未停止增长。搭载最新的硬件技术，一辆消费级汽车可以具备执行 176 tera operations per second (teraOPs)[1]的能力。它比阿波罗制导计算机所能做的要多 100 亿倍，阿波罗 11 号和宇航员阿姆斯特朗就是用制导计算机登上月球的。计算能力上的这一大步，使自动驾驶车辆能够应对各种复杂传感器产生的数据，包括多视角相机、长短距离雷达、激光雷达、精确惯性测量单元等。超能力计算芯片和传感器的组合通常要花费 1 万美元以上。

这种对昂贵设备的渴望成功推动了许多自动驾驶解决方案的发展，如特斯拉的 AutoPilot[2]、Waymo 的 Waymo One[3]、通用汽车的 Super Cruise[4]和百度的阿波罗 Go[5]。遗憾的是，无论汽车公司如何精美地宣传，在官方文件中，大部分量产产品仍然属于 2 级(部分驾驶自动化)[6]驾驶性能。除了 L2 之外，没有人有足够的信心做出进一步的宣称。

不过，在将自动驾驶技术商业化的路上，给日常生活中的客户，逗号。ai<sup>1</sup> build 以自己的方式构建。他们的售后产品如图 1 所示，逗号二/三，据称具有 L2 的能力，单个设备的成本仅为 999 美元。这可能是目前市场上最实惠的自动驾驶设备了。为了与许多不同品牌的车辆兼容，他们的软件 Openpilot 是开源的，托管在 Github 上[7]。更令人惊讶的是，在 2020 年，《消费者报告》将逗号二评为 18 个竞争对手中整体评级<sup>2</sup>最高的，击败了特斯拉、凯迪拉克、福特等竞争对手。

等等，这有可能吗？如果有，怎么可能呢？它在什么样的情况下工作？为了回答这些问题，我们对逗号装置进行了细致的测试，并从头开始重新实现了他们的部分系统。经过几个月的深入研究，我们得出结论，Openpilot 简单而有效，因为它可以在许多简单的 L2 情况下工作。在这份报告中，我们希望与观众分享我们的最新发现，并激励社区深入了解 Openpilot 的恶魔。大多数传统自动驾驶解决方案的感知、预测和规划单元都是基于模块的，而 Openpilot 采用端到端系统级设计理念，直接从摄像头图像中预测轨迹。二者的区别如图 2 所示。Openpilot 中的这个端到端网络被命名为 Supercombo，它是我们想在这项工作中重新实现和讨论的主要目标。

虽然 Openpilot 被认为是第一个将端到端精神应用于大规模消费产品的自动驾驶，但学术界也有很多开创性的工作试图实现端到端

<sup>1</sup> <https://comma.ai/>

<sup>2</sup> <https://data.consumerreports.org/reports/cr-active-driving-assistance-systems/>

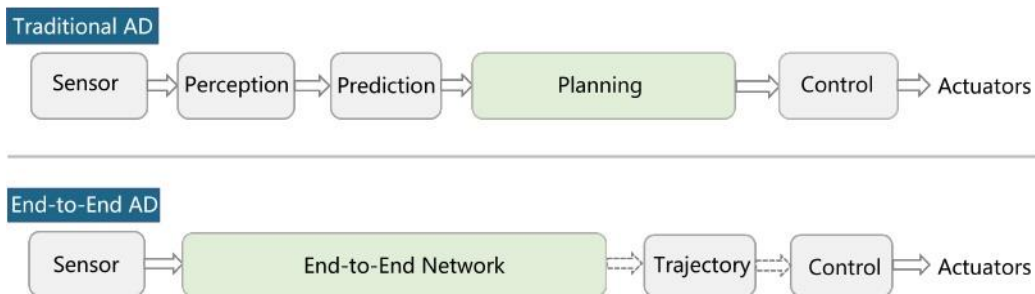


图 2:传统 AD 解决方案与 Openpilot 的比较。

无显式感知模块的框架设计[8,9,10,11,12]。这些工作在某种意义上激励了端到端设计将通过协同训练优化和多任务学习来利用更好的特征表示。学术界的文献在模拟环境(如 CARLA[13])上取得了令人印象深刻的结果,然而,当我们将算法迁移到现实案例时,驾驶性能和安全保障仍不清楚。

为此, Openpilot 的重要性在于在产品层面实现整个自动驾驶系统的端到端设计,并满足日常场景的基本 L2 功能。Openpilot 的成功将鼓励学术界和工业界进行更多的调查。遗憾的是,训练细节以及大量的数据并没有向公众公布。我们认为,这些对于以端到端的精神实现 L2 目标非常关键。这促使我们构建自己的端到端模型,即 OP-Deeptide, 其中我们保持大多数预处理和后处理与 Openpilot 一致。

在公开数据集和真实场景上进行了广泛的实验,以评估重构模型的性能。在真实世界的测试中,我们精巧地分析了底层逻辑,并提出了一个双模型部署框架,以成功地在设备上运行我们自己的模型。数值结果证明,原始的 Supercombo 和我们重新实现的模型 OP-Deeptide 都可以在高速公路场景中表现出色,但在街道场景中并不适用。在定量指标方面,我们的模型与最初的 Openpilot 模型是一致的。车载性能测试进一步证明, Openpilot 适用于封闭道路(高速、普通道路等),在 ADAS 等驾驶辅助系统中表现优异。

在本报告中,我们首先概述了 Openpilot 系统,以及我们在第 2 节中使用的测试场景。然后,我们将在第 3 节中提供 OP-Deeptide 重新实现版本的详细信息。结果,以及与原始模型的比较,见第 4 节。第 5 节提到了进一步的讨论。我们将这些贡献总结如下:

- 1.我们在真实场景中测试了 Openpilot 系统,并得出结论,它确实可以在单个视觉设备上实现 L2 自动驾驶。
- 2.我们从头开始对 Openpilot 的训练阶段进行了恢复,在 Openpilot 中设计了 Supercombo 的网络结构,并在公共数据集上测试了我们的重构模型。OP-Deeptide 与原始版本相当。
- 3.本技术报告是研究界在工业大规模对消费者产品意义上,为规划和控制的最终驱动任务进行端到端系统设计工作的起点。

2.1 Openpilot 概述

Openpilot 是 Comma 推出的一个相对成熟的 L2 辅助驾驶系统开源项目。该系统实现了传统的辅助驾驶功能,包括自适应巡航控制(ACC)、自动车道定心(ALC)、前向碰撞警告(FCW)和车道保持辅助(LKA),这些功能基于一种名为 Supercombo 的端到端模型。用户界面如图 3 所示。它已经提供给了大量的消费者,并且设计上是兼容的



图 3:逗号设备上的 Openpilot。图片改编自 Comma.ai。

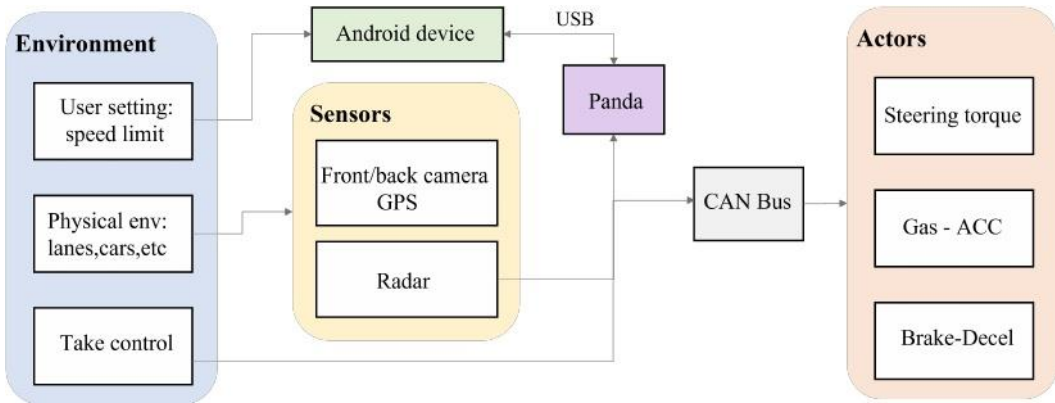


图 4:Openpilot 系统概述。传感器捕获的环境信息和通过 Panda 设备交换的 ego 车辆状态被发送到定制的安卓设备以运行算法。执行器信号通过 CAN 总线传输来控制车辆。

拥有超过 150 种车型。用户只需几步，就可以在自己的车上安装售后服务套件，享受辅助驾驶的体  
验。

Openpilot 主要由软件和硬件两部分组成。软件部分包含各种算法和中间件。这个名为 EON 的硬  
件设备是整套设备的大脑，负责运行 NEO 系统(一个高度定制的安卓系统)和软件算法。整体系统  
架构如图 4 所示。EON 设备上的不同单元捕获相应环境信息，例如，摄像头拍摄物理世界的正面图  
像，名为 Panda 的汽车接口从 CAN 总线提取车辆状态。在这一步中，车辆上的 stock 雷达也  
被用到。然后，利用这些数据，Openpilot 运行 Supercombo 模型，并通过后处理软件和中间件验  
证其输出。最后通过 Panda 接口向车辆控制器发送控制信号。

我们在真实世界中对系统进行了定性和定量的测试，以揭示其优点和缺点。我们参考读者  
对详细的原始系统级测试到附录。A 和我们的网页 <https://sites.google.com/view/openpilot-deepdive/home>。

2.2Supercombo 模型

整个系统的核心是运行在 EON 器件上的软件算法，它负责感知、规划、定位和控制。在这些算  
法中，感知算法(即 Supercombo 模型)是值得深入研究的关键第一步。它接收来自摄像机的图像，并  
预测车道、道路边缘、领先车辆的位置和速度，以及最重要的是，自我代理在未来应该遵循的轨  
迹。因此，它也被称为轨迹规划模型(trjectory planning model)。我们在图 5 中给出了 Supercombo  
的简化说明。具体来说，模型执行的流水线包括以下部分。

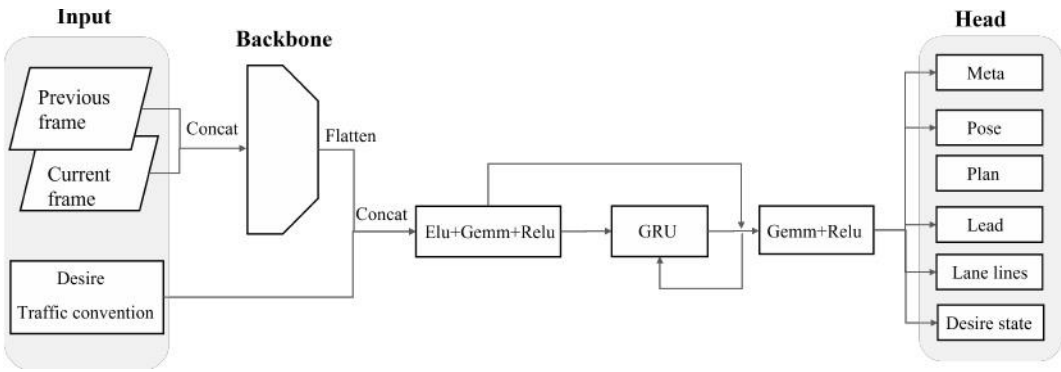


图 5:Supercombo 的管道。它需要两个连续的帧，一个代表所需的高级命令的向量，一个指示右/左交通约定的布尔值作为输入。最终的预测包括一个计划的轨迹，自我姿态，车道线，领导代理的状态等。

**预处理。**首先，将相机拍摄的原始单帧 3 通道 RGB 图像( $3 \times 256 \times 512$ )，转换为 6 通道 YUV 格式( $6 \times 128 \times 256$ )。然后，将连续的两帧拼接在一起作为模型输入，得到( $12 \times 128 \times 256$ )的输入尺寸。

**主要的网络。**主干网采用谷歌的 Efficientnet-B2[14]，性能好，效率高。采用分组卷积，减少骨干网的参数量。为了捕获时间信息，一个 GRU(门控循环单元)连接到主干网。

**预测的头。**几个完全连接的层连接到 GRU 上，作为预测头。输出包括 5 条可能的轨迹，其中选择置信度最高的轨迹作为规划轨迹。每条轨迹包含了 ego 车辆坐标系下的 33 个 3D 点的坐标。此外，Supercombo 还可以预测车道线、道路边缘、前方物体的位置和速度，以及车辆的其他一些信息。

需要注意的是，虽然网络结构以及前后处理方法都是开源的，但训练管道和数据仍然是内部的。因此，我们需要从头开始重新实现它。另外，对于训练数据，逗号.ai 声称，他们的 Supercombo 模型是通过收集 100 万分钟的驾驶<sup>3</sup>视频进行训练的。然而，他们只向公众提供了其中的一小部分。

### 3 的方法

在本节中，我们提出了一个受 Supercombo 模型启发的端到端轨迹规划模型，以及一个基于 EON 器件设计的部署框架，如图 6 和图 9 所示。

#### 3.1 端到端规划模型

记住，我们的目标是重新实现 Supercombo 模式。理想情况下，我们的模型应该具有与 Supercombo 完全相同的结构，并且需要相同的输入和输出格式。遗憾的是，这并不可行，因为一些训练所需的数据并不容易获取，比如领头车的位置和速度。因此，我们在输入和输出格式上做了一些改变，这些改变对我们认为的主要结论没有重大影响。如图 6 所示，我们重新实现的模型的主要结构与原始的 Supercombo 非常相似。它从单一的、面向前方的摄像头连续取下两帧，并直接预测未来的轨迹。输入帧通过主干并被平展成长度为 1024 的特征向量。然后，将向量输入到宽度为 512 的 GRU 模块中，使模型能够记住时间

<sup>3</sup><https://blog.comma.ai/0810release/>



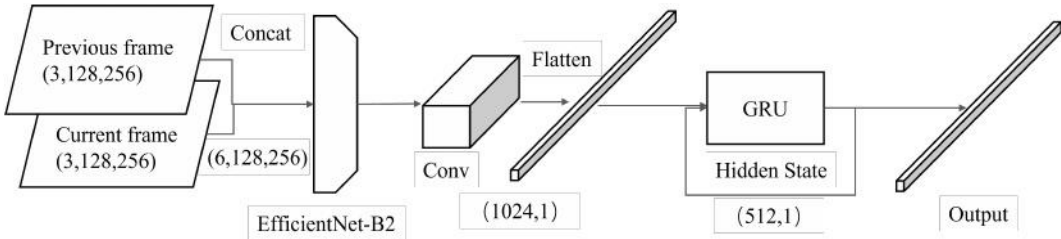


图 6:我们重新实施规划模型的结构。

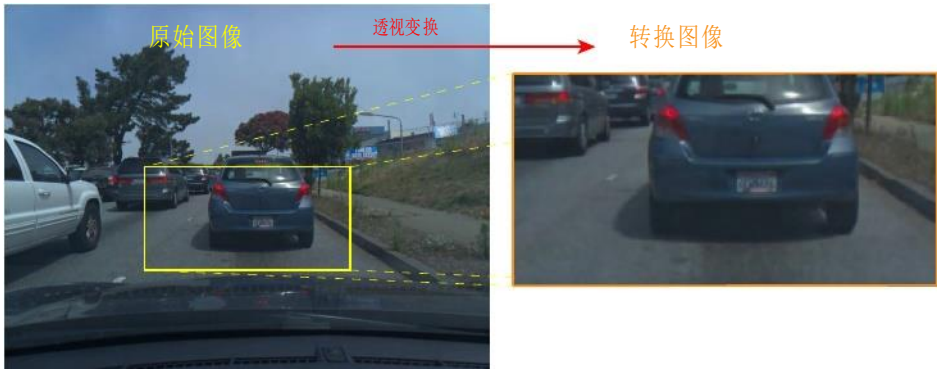


图 7:透视转换。

信息。最后，几个全连接层将网络输出组织成所需的格式。更多细节将在下文讨论。

**预处理-透视变换。**Openpilot 设计用于各种车辆。一个直接的问题是，不同的用户可能会将设备安装在不同的位置。这也被称为 *mount bias*。要消除它，关键是要知道相机是如何安装的(即相机外参)。Openpilot 需要用户在安装后进行摄像头校准过程，在此过程中，驾驶员需要手动驾驶车辆沿着长而直的道路行驶一段时间，并保持相对固定的速度。然后就推导出了相机的 extrinsics。此外，还有一个 *在线校准*过程，在设备工作时持续运行，以保持外部参数的更新，以防来自悬架系统的振动。一旦知道了外部参数，我们就可以做一个简单的图像变形操作，使图像看起来像从标准相机上拍摄的图像。这个标准相机在某些材料中也被称为 *虚拟相机*。图 7 展示了透视变换的结果以及从相机中获取的原始图像。我们可以清楚地看到，Openpilot 只关注了一个非常狭窄的区域，因为图像中的大部分异常区域都被裁剪掉了。在这一步中，图片被调整为  $128 \times 256$ 。

**预处理-颜色变换。**Openpilot 直接从 CMOS(互补金属氧化物半导体，一种图像传感器)读取原始图像。要将原始图像转换为常用的 RGB-888 格式，我们需要一个辩论过程，这将带来额外的开销。Openpilot 使用 YUV-422 格式作为模型输入，以减少开销。但是，由于所有公开可用的数据集已经将 RGB-888 识别为默认图像格式，因此将它们转换回 YUV-422 是没有意义的。因此，我们坚持使用 RGB-888 格式。换句话说，颜色变换在我们的工作中被忽略了。

**骨干。**Openpilot 采用了 EfficientNet-B2[14]作为主干。由于最新的 AutoML(自动机器学习)和模型扩展技术，它具有准确性和效率。它包含 920 万个参数，一次需要 6.5 亿次浮点运算。相比之下，常用的 ResNet-50[15]骨干网包含 2500 万个参数，一次需要 27 亿次浮点运算。给定一个形状为 (6,128,256) 的输入张量，骨干输出一个形状为 (148,4,8) 的张量，然后，一个  $3 \times 3$  核的卷积层将通道数量减少到 32 个。最后， $\text{shape}(32,4,8)$  的张量被平展成一个长度 1024 的特征向量。



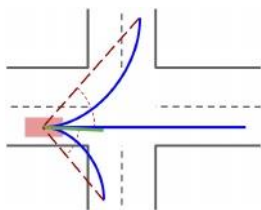


图 8:MTP 损失的工作原理 [17]。在这幅鸟瞰图 (BEV) 中，粉色矩形代表 ego 载载体。地面真实轨迹显示为绿色。预测轨迹显示为蓝色。很明显，我们应该选择直线为 P 的轨迹。

**GRU 模块。**特征向量后面附加一个宽度为 512 的 GRU[16] 模块。这里使用 GRU 的目的是捕获时间信息。

**预测的头。**在 GRU 模块之后，几个完全连接的层负责根据需要组织输出维度。超级组合模型将产生一个长度为 6609 的张量作为最终输出，这是巨大的。预测包括规划的轨迹、预测的车道线、道路边缘、领先车辆的位置以及一些其他信息。然而，由于目前还没有这样的数据集提供所有这些注释，我们在重新实现时只关注计划的轨迹。设  $M$  为可能的轨迹数量。每条轨迹由 ego 车辆坐标系下  $N$  个 3D 点的坐标组成，一个数字为置信度值。那么，输出维度  $D$  为，

$$D = M \times (N \times 3 + 1). \quad (1)$$

通常，我们有  $M = 5$  和  $N = 33$ 。考虑到国际单位下坐标的原始值比较大，我们在所有  $x$  坐标上添加一个指数函数，在所有  $y$  坐标上添加一个  $\sinh$  函数。(Openpilot 从来没有被设计成在倒车时工作!)

**的损失。**首先，我们认为地面真实轨迹是人类司机实际驾驶的轨迹。然后，我们跟随信息性逗号。ai 的博客<sup>4</sup>使用 MTP (Multimodal Trajectory Prediction, 多模态轨迹预测) 损失 [17]，其中包括一个回归损失  $\mathcal{L}_{reg}$  和一个分类损失  $\mathcal{L}_{cls}$ 。总的损失描述为：

$$\mathcal{L} = \mathcal{L}_{reg} + \alpha \mathcal{L}_{cls}. \quad (2)$$

具体来说，我们计算 ground truth 和  $M$  个预测轨迹之间的余弦相似度。将  $P$  表示为相似度最高的预测轨迹。然后我们计算  $P$  和真实值之间的回归损失 (smooth- $L_1$ )。对于  $\mathcal{L}_{cls}$ ，我们给  $P$  置信度 1，而其他的保持为 0。分类任务采用 BCE (binary cross entropy) 损失。这个过程如图 8 所示。在推理过程中，将置信度最高的轨迹作为预测结果。

### 3.2 设备上的部署

为了进一步验证实际应用程序的性能，我们探索了基于现有 Openpilot 软件和硬件系统的设备上部署管道。考虑到我们的模型只预测要跟随的轨迹，而原始的超级组合预测一些其他信息，我们设计了一个双模型部署方案。在这个方案中，两个模型都部署在板上，并交替进行预测。具体地说，我们用我们的模型预测的轨迹取代了超级组合预测的轨迹，并保持其他信息不变。由于设备端部署与特定硬件平台紧密耦合，我们仔细分析了 Openpilot 的摄像头数据流和底层模型框架。然后，我们在考虑模块化的思想下添加了额外部署的模型。整体的模型部署框架如图 9 所示。蓝色框是我们的模型添加的部分，并替换超级组合中的计划节点。

除了总体部署框架的设计之外，软件开发过程中的以下步骤也是必不可少的：(a) 将模型应用到 Openpilot 仿真环境中，验证基于 ONNX 模型的双模型算法的逻辑。(b) 将 ONNX 转换为 DLC 格式 (高通平台上用于神经网络加速的特定模型格式)，并将 CARLA 仿真代码迁移到设备端。(c) 将 Openpilot 装置连接到汽车上，并进行道路试验，以验证双模型框架的性能。(d) 将重新实现的模型的输入和输出与原模型对齐。

<sup>4</sup> <https://blog.comma.ai/end-to-end-lateral-planning/>

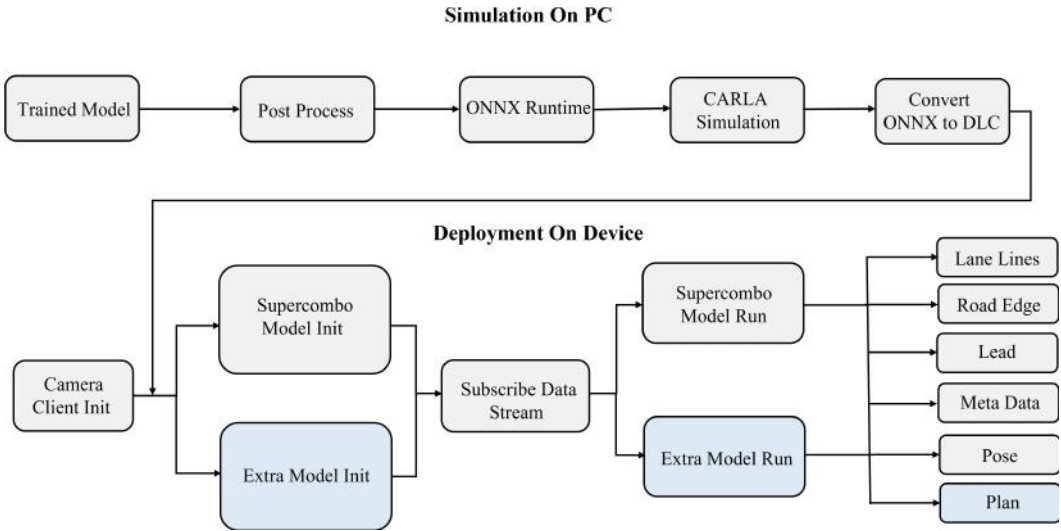


图 9:我们的双模型部署框架的流水线。我们训练的模型在 CARLA 仿真中进行了验证，并在部署到设备上之前转换为 DLC 格式。两个模型同时运行，每个模型都贡献了最终输出的一部分。

## 4 实验

### 4.1Openpilot:一个现实世界的测试

为了测试包括 ACC、ALC、LKA 和 DM(驾驶员监控)在内的功能，我们精心设计了一些测试场景，从定性和定量两方面研究其实际表现，并尝试发现故障案例。

我们测试的主要结论如下。具体内容请参见附录。一个。

1. Openpilot 能够在封闭道路(高速、普通道路等)上行驶，并能发挥出色的驾驶辅助功能。
- 2.它不能处理一些复杂的城市道路，比如没有车道的双向道路。在这些情况下，需要人工干预，以避免碰撞。

### 4.2 OP-Deeptide:我们对 Supercombo 的重新实现

#### 4.2.1 数据准备

我们在两个数据集上训练和评估我们的模型，nuScenes[18]和 Comma2k19[19]。表 1 显示了它们的一些关键特征。

表 1:本报告中使用的两个数据集之间的比较。

数据集	生 FPS(赫兹)	对齐 FPS(赫兹)	长度每 序列 (帧/秒)	完全 长度 (分钟)	场景	位置
nuScenes [18]	12	2	40 / 20	330	街	美国 新加坡
Comma2k19 [19]	20.	20.	1000 / 60	2000	高速公路	美国

**nuScenes.** nuScenes[18]是由 motion 策划的公共自动驾驶大规模数据集。它是多用途的，支持各种类型的任务，如检测、跟踪、预测和分割。它每秒提供 12 帧原始帧，但是，只有 2 帧是对齐的，可以在我们的设置中使用。我们从中提取 556 分钟的有效视频片段，并随机将其分成 80%/20%分别作为训练集和验证集。



为了迎合相对较低的 2hz 帧率，我们对轨迹的地面真实值进行了更改。而不是 33，每条地面真值轨迹只包含 10 个点，这些点记录了未来 5 秒内 ego vehicle 的位置。

**Comma2k19。**Comma2k19[19]是加利福尼亚州 280 高速公路上超过 33 小时的通勤数据集，由 Comma 提供。Ai 在 2018 年。这个数据集是使用他们的早期产品收集的。我们相信这是训练数据的一个小子集，逗号就在上面。ai 训练 Supercombo，在他们的博客中说的是 100 万分钟。因此，这个数据集是训练我们模型的理想选择。同样，我们选择 80%作为训练集，剩下的 20%作为验证集。

我们将地面真相与 Openpilot 的 Comma2k19 对齐。在当前 ego 飞行器的坐标系下，每条地面真值轨迹包含未来 10 秒内 33 个规划位置。这些点不是均匀分布的。相反，它们遵循一种称为*基于时间的锚点*的机制。

在不定时间  $t = 0$  时，轨迹包含了汽车在某个时间  $t$  出现的位置， $t$  正是基于时间的锚点。 $T$  的值是固定的，从集合  $\{0$  中选择。 $\}$ ，0.00976562, 0.0390625, 0.08789062, 0.15625, 0.24414062, 0.3515625, 0.47851562, 0.625, 0.7910151562, 0.9765625, 1.18164062, 1.40625, 1.9140625, 2.19726562, 2.5, 2.82226562, 3.1640625, 3.52539062, 3.90625, 4.30664062, 4.7265625, 5.16601562, 5.625, 6.10351562, 6.6015625, 7.11914062, 7.65625, 8.21289062, 8.7890625, 9.38476562, 10.}。注意，它们不是均匀分布在 0 和 10 之间的。相反，它们在不久的将来是稠密的，随着时间的推移会变得稀疏，这表明模型应该更多地关注不久的将来。坐标是由 30 Hz 的数据线性插值得到的。

#### 4.2.2 指标

我们使用两种指标来评估方法，即模仿指标和舒适度指标。

**模仿指标。**模仿指标旨在显示模型从人类驾驶员那里学习的程度。受 KITTI 检测挑战[20]中使用的 3D 检测指标的启发，我们首先根据 x 轴上的距离将轨迹点分为 5 组，即 0 ~ 10 米、10 ~ 20 米、20 ~ 30 米、30 ~ 50 米和 50+米。在每个范围内，我们计算两个度量，平均欧几里德距离和平均精度。

- 平均欧几里德距离误差。**顾名思义，我们计算相应的预测点和地面真值点在三维空间中的欧几里德距离。然后，我们对它们进行平均。此外，我们还计算了点在 x 轴和 y 轴上投影时的欧几里德距离。越低越好。单位是米。
- 平均精度。**如果两个对应点之间的距离小于一个阈值，我们将其标记为“命中”。否则，我们标记为“未命中”。然后，我们就可以计算命中率，并取平均值。例如，AP@0.5 表示低于阈值 0.5 的平均命中率。越高越好。

**舒适指标。**舒适度指标可以反映出可能的轨迹是否会让乘客感到舒适。具体来说，我们使用加速度和横向加速度来测量舒适度。

- 混蛋。**跃变是物体加速度随时间变化的速率。需要注意的是，由于 jerk 是一个有向量，我们实际上报告的是它的振幅。单位是  $m/s^3$ 。越低越好。
- 横向加速度。**横向加速度是指与汽车行驶方向横向的加速度。此外，它是一个有向量，所以我们实际上报告了它的振幅。单位是  $m/s^2$ 。越低越好。

#### 4.2.3 实施细节

我们的实现与 PyTorch[21] 1.8 兼容。默认情况下，我们使用 AdamW 作为优化器。批量大小和学习率分别设置为 48 和  $10^{-4}$ 。应用了一个值为 1.0 的梯度剪辑。在训练过程中，我们使用了 8 块 NVIDIA V100 gpu。由于有一个 GRU 模块，我们需要通过填充零来初始化它的隐藏状态。参数每 40 步更新一次，

表 2:nuScenes 数据集的结果。

(a)我们重新实现的模型 OP-Deepdive 的模仿指标。欧几里德距离误差。**d.e.**-  $\rightarrow$ x:投影到 x 轴上的欧几里德距离误差。**AP@0.5**:平均精度低于阈值 0.5 米。其余的可以推断。

范围	d.e.	d.e. $\rightarrow$ -	d.e. $\rightarrow$ -			
		x	y			
(米)	(米)	(米)	(米)	AP@0.5	AP@1	AP@2
清廉	2.02	1.92	<b>0.25</b>	0.28	0.51	<b>0.73</b>
10 -						
20	4.39	3.91	1.01	0.045	0.14	0.33
20 -						
30	5.25	4.63	1.24	0.016	0.067	0.18
30 -						
50	6.51	5.93	1.26	0.010	0.038	0.12

(b)原始超级连击模型的模仿指标。

范围	DE.(仪	DE. $\rightarrow$ - x(仪	DE. $\rightarrow$ - y(仪			
(米)	表)	表)	表)	AP@0.5	AP@1	AP@2
010 10	1.96	1.85 5.42	<b>0.32</b> 1.04	0.237	0.408	<b>0.692</b>
20 20	5.80	8.11 11.05	1.26 1.23	0.025	0.064	0.170
30 30	8.54			0.013	0.038	0.090
50	11.36			0.008	0.023	0.053

(c)微调超级组合模型的模仿指标。

范围	D.E.	De $\rightarrow$ -	De $\rightarrow$ -			
(米)	(米)	x	y	AP@0.5	AP@1	AP@2
		(米)	(米)			
清廉	1.39	1.31	<b>0.22</b>	0.305	0.568	<b>0.809</b>
10 -						
20	3.57	3.16	0.84	0.054	0.162	0.387
20 -						
30	4.78	4.28	1.06	0.028	0.088	0.235
30 -						
50	6.25	5.90	0.93	0.015	0.050	0.149

梯度在其上累积。在 Comma2k19 数据集上训练 100 个 epoch 大约需要 120 个小时。在单个 NVIDIA GTX 1080 GPU上，该网络可以以 100 FPS 的速度进行推理。

4.2.4 量化结果

在 nuScenes数据集上。由于每个地面真值轨迹仅持续 5 秒，且大部分采集场景都在繁忙的市中心，因此只有少数样本落在“50+米”范围内。因此，我们不报告这个范围内的结果。

表 2a 显示了我们在 nuScenes 数据集上重新实现的模型的结果。老实说，我们没有取得一个合理的结果，因为距离误差很大，而 APs 很低。

为了理解背后的魔力，我们在 nuScenes 数据集上测试了原始的 Supercombo 模型。如表 2b 所示，结果更加糟糕。这是可以理解的，因为 nuScenes 的帧率不是 30 FPS。我们使用我们的管道进一步微调 Supercombo 模型。如表 2c 所示，它的性能比我们的重新实现要好一些。我们得出结论，在 nuScenes 数据集上，更好的预训练权值可以带来更好的性能，但好处并不显著。

由于基于如此低的帧率(2 Hz)计算横向加速度和抖动是不合理的，因此我们没有在 nuScenes 数据集上报告舒适度指标。

**关于 Comma2k19 数据集。**在模仿指标方面，我们的模型在 Comma2k19 数据集上取得了很好的效果。原始的 Supercombo 模型显示了可比较的结果，如表 3a 和表 3b 所示。平均而言，我们的模型比最初的 Supercombo 表现得更好。

至于舒适度指标，然而，事情变得不同。如表 3c 所示，我们的模型产生的轨迹比 Supercombo 产生的轨迹具有更高的痉挛和横向加速度。与此同时，来自人类演示的轨迹(即地面真实值)显示

表 3:Comma2k19 数据集上的结果。(a)我们重新实现的模型  
OP-Deepdive 的模仿指标。

范围 (米)	D.E. (米)	De→ - x (米)	De→ - y (米)	AP@0.5	AP@1	AP@2
清 廉	0.451	0.426	<b>0.067</b>	0.909	0.951	<b>0.971</b>
10 - 20	1.161	1.093	0.180	0.589	0.808	0.914
20 - 30	1.624	1.529	0.248	0.384	0.651	0.850
30 - 50	2.335	2.192	0.367	0.225	0.465	0.729
50 +	7.373	6.475	2.04	0.030	0.100	0.239

(b)原始模型 Supercombo 的模仿指标。

范围 (米)	D.E. (米)	De→ - x (米)	De→ - y (米)	AP@0.5	AP@1	AP@2
清 廉	0.4396	0.2679	<b>0.1998</b>	0.7966	0.9510	<b>0.9829</b>
10 - 20	1.4824	1.2157	0.2928	0.1782	0.6170	0.8617
20 - 30	2.2831	1.8652	0.4483	0.0263	0.2661	0.7368
30 - 50	3.4265	2.7790	0.6818	0.0026	0.0889	0.4922
50 +	11.1124	9.0153	2.4796	0.0001	0.0004	0.0062

(c)舒适度指标。

方法	平均混蛋	马克斯混蛋	平均横向 Acc。	最大横向 Acc。
Supercombo	2.2243	10.8209	0.3750	1.0505
OP-Deepdive(我们的)	4.7959	24.007	0.4342	1.7931
人类的演示	0.3232	2.2764	0.3723	0.7592

比两种模型低得多的抽动和横向加速度。这表明，与人类驾驶员相比，预测的轨迹不够平滑。然而，这并不意味着 Openpilot 不如人类驾驶员，因为在 Openpilot 向汽车发送命令之前，预测的轨迹需要进一步处理。

4.3 双模型实验

在我们成功地重新实现 Supercombo 模型后，该模型在一些公共数据集上实现了合理的性能，我们希望将其部署在逗号二板上，并查看其实际性能。

4.3.1 CARLA 模拟 试验

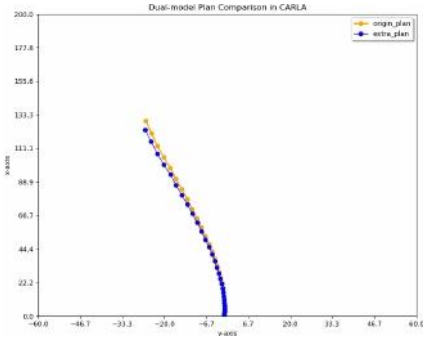
我们首先在自动驾驶仿真平台 CARLA[13]中对双模型框架进行了测试。模拟测试的主要目的是在我们进行真正的车载测试之前，验证双模型框架在逻辑上是否正确。但是，请注意，仿真与现实世界存在一定差距。



- 数据来源不同。CARLA 中的数据流由模拟器渲染。渲染的图像与真实世界的图像之间存在一定的领域差距。
  - 硬件平台不同。在 PC 上进行仿真，PC 有足够的计算能力。然而，在实际测试中，逗号二板基于高通 SOC 平台，其计算能力有限，需要显式神经网络加速。
- 模型格式不同。仿真平台采用 ONNX，这是一种常用的模型格式，而逗号二板需要 DLC，这是高通移动平台特有的模型格式。



(a)我们在 CARLA 中运行的双模型部署框架。

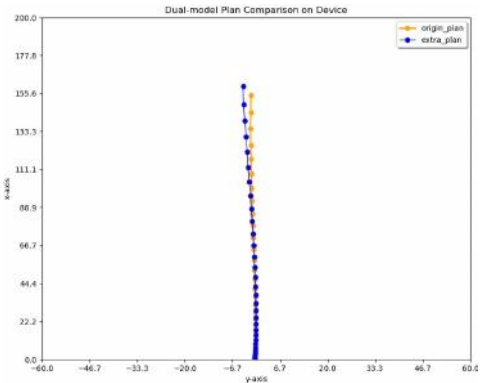


(b)轨迹比较。

图 10:在 CARLA 中验证的双模型部署框架的定性结果。



(a)我们的双模型部署框架在机上测试。



(b)轨迹比较。图 11:机载验证的

双模型部署框架定性结果。

图 10a 显示了在 CARLA 仿真中运行的双模型框架的示例。请注意，规划的轨迹来自我们重新实现的模型，而车道线是原始的 Supercombo 模型。两个模型是同时运行的。

我们在图 10b 中进一步比较了两个模型的预测轨迹。总的来说，这些轨迹彼此接近。但与原来的 Openpilot 相比，执行时间和丢帧率明显增加。原因可能包括两个方面。两个型号同时运行，从而占用更多的硬件资源。此外，它们订阅了相同的数据流节点，导致丢帧率增加。我们有理由怀疑，当部署到移动设备上时，问题可能会被进一步放大。

#### 4.3.2 真实测试

在仿真环境下对双模型框架进行验证后，最终可以实现机载部署。我们使用的板为逗号二，车辆为雷克萨斯 ES 2020。图 11a 是我们的双车型部署框架的结果。出于安全考虑，该软件在 debug 模式下运行，在这种模式下，它进行预测，但不控制车辆。总体上符合预期，这证明了我们在单台设备上的部署方案是有效的。与上一节类似，我们也在图 11b 中展示了这两种轨迹。可以观察到，输出曲线是相对一致的。我们进一步分析了两种模型在设备上的运行状态。运行时间和丢帧率均高于仿真环境，与预期一致。由于设备上的计算能力和带宽有限，丢帧率和运行时间将不可避免地增加。

## 5 讨论

### 5.1 开环系统与闭环系统

在本文中，我们的模型以开环的方式进行训练和测试。虽然结果似乎可以接受，但这种方式有一定的缺陷。

首先，作为逗号。人工智能已经发现<sup>5</sup>，该模型只是学习和预测人类司机最有可能的行为。在从真实世界收集的数据集中，人类司机只是在大多数时间里让车辆保持直线行驶，并避免任何可能的不良行为，如危险驾驶和交通违法。这样，模型甚至没有机会学习如何从错误中恢复。例如，如果我们手动输入一个车辆正在越过中心线的视频序列，这当然是一种危险驾驶，模型很可能预测出一个持续直线前进的轨迹，而不是回到正确的直线。

其次，我们注意到它可能会引入一种特征泄露。在某些情况下，如果车辆当前正在加速，模型会预测出更长的、更快的轨迹，反之亦然。此外，如果车辆正在左转或右转，模型会预测出相应的左转或右转轨迹。这表明，该模型是根据驾驶员的驾驶意图而不是道路特征来学习的。揭示这一问题的典型场景是根据交通灯启停。模型可能会在看到红灯或停车线之前很久就决定减速。然后，当汽车在红灯变绿之前慢慢开始移动时，该车型可能会决定快速加速。这让我们相信存在某种时间上的特征泄露。

闭环训练和测试的障碍在于我们根本无法做到这一点。让一辆未经训练的车辆根据其预测轨迹在道路上行驶太危险，因此不实用。有两种解决方案。第一种是用逗号介绍的。艾未未的博客。我们可以通过 WARP 机制模拟闭环训练和测试，使从一个角度收集的图像看起来像是从另一个角度收集的。第二个是在模拟器中对模型进行完全的训练和测试。然而，这两种解决方案都不是完美的，因为 WARP 机制会引入图像伪影，而现有的模拟器[13]无法呈现真实的图像。

### 5.2 模仿指标

虽然我们在本文中提出了两个模仿指标(平均欧氏距离误差和平均精度)，但我们认为，模仿学习的指标，特别是在自动驾驶领域，仍然是开放的。有两个主要的因素需要关注：

首先，模型应该学习的理想轨迹，没有明确定义。目前我们采用人类驾驶员的轨迹作为“地面真实值”。这是个不错的选择。然而，人类司机可能会犯错误。模型可能会学习这些不好的例子，这可能会导致进一步的问题。此外，假设前面的车辆移动缓慢。对于人类驾驶员来说，我们可以选择超车或不超车，这就导致了至少两种可能的轨迹。那么，我们如何决定哪条轨迹更好，并迫使模型去学习它呢？

第二，距离这么远的轨迹点，有必要去关心吗？考虑到 Openpilot 以 30 FPS 的速度推断，那么理论上只有未来 0.033 秒内的轨迹点是重要的，因为后面的会被下一次预测覆盖。如果一辆车以 120 公里/小时的速度移动，那么它在 0.033 秒内只移动 1.1 米。这是否意味着我们只需要关注几米以内的轨迹？

### 5.3 “调试”一个端到端的模型

众所周知，算法设计者应该注意边界情况和失效情况。传统上，在感知和规划模块分离的自动驾驶系统中，我们可能会在故障上得出哪个部分是故障的结论。然而，对于像 Supercombo 这样的端到端模型，其感知和规划模块融合在一起，我们如何在失败时调试它？比如，如果模型在红灯时拒绝停车，我们如何知道它是否成功检测到红灯和停车线？

---

<sup>5</sup> <https://blog.comma.ai/end-to-end-lateral-planning/>

坦率地说，我们也不知道答案。当然，有一些基因解决方案可以让模型变得鲁棒，比如增加更多的数据，让模型变得更深更大等等，但它们并不能帮助解决一个具体的问题。

## 5.4 双模型部署方案的原因

在我们的工作中，双模型部署方案主要基于两个方面的考虑：一方面，由于数据集标注不足，我们重新实现的模型只预测了轨迹，而原始的 Supercombo 还预测了其他信息，包括引线、姿态、车道、道路、边缘等。Openpilot 系统中的其他中间件模块也需要这些额外的预测。另一方面，作为一个批量生产的售后市场产品，Openpilot 不仅仅是一个纯粹的端到端神经网络。相反，它是一个系统工程，涉及软件算法、通信中间件、硬件驱动。各个模块之间有一定程度的耦合。因此，我们以模块化的思想将模型部署到现有的 Openpilot 系统中，而不是从头开始。

当然，抛弃双模型框架在技术上是可行的。如果重新实现的模型可以与 Supercombo 完美地对齐，那么我们就可以简单地替换它。实现这一目标的一种可能方法是重新标记现有的数据集，并确保所有所需的信息都被正确标记。

## 6 结论和未来工作

在本报告中，来发现逗号是如何。ai 设法在单个设备上实现 L2 辅助驾驶，我们从头开始重新实现基本的 Supercombo 模型，并在公共数据集上对其进行测试。实验表明，原始的 Openpilot 和我们重新实现的模型都可以在高速公路场景中表现良好。为了在现实世界中测试整个系统，我们设计了一个双模型部署框架。我们在 CARLA 仿真环境中进行验证，并在机上进行部署，证明了我们的方案是适用的。我们的工作证实了一个简单而有效的 L2 辅助驾驶系统可以集成到一个单板上，并且在大多数情况下可以很好地工作。当然，还有很多需要探索和讨论的地方。我们希望这篇报道能给观众一些启发。

我们相信，在这一领域仍然存在许多值得探索的开放性问题。举几个例子：

- 1.由于端到端学习是数据饥渴的，我们可能会探索生成和标记高质量数据的新方法，如众包和自动标记。
- 2.两个数据集之间性能差距背后的原因目前还不清楚。

## 参考文献

- [1]英特尔。新款 Mobileye EyeQ ultra 将实现消费类自动驾驶。  
<https://www.intel.com/content/www/us/en/newsroom/news/mobileye-ces-2022-tech-news.html>, 2017。
- [2]山塔努·英格尔和马杜里·普乌特。特斯拉 Autopilot:半自动驾驶，未来自动驾驶的提升。《国际工程技术研究学报》，3(9):369-372,2016。
- Phil LeBeau。Waymo 启动商业拼车服务。<https://www.cnbc.com/2018/12/05/waymo-starts-commercial-约车-service.html>, 2018。
- [4]通用汽车。《超级巡航》取材于真实世界的交通场景。<https://media.gm.com/media/me/en/cadillac/news.detail.html/content/Pages/news/me/en/2013/凯迪拉克/super-cruise-taking-on-real-world-traffic-scenes.html>, 2013。
- [5]范浩阳、范主、刘长春、张亮亮、庄丽、李东、朱伟成、胡江涛、李鸿业、孔琦。百度 apollo em 运动规划器。arXiv 预印本 arXiv:1807.08048, 2018。
- [6]韶山 Liu 杰唐、张哲,jean-luc Gaudiot。自动驾驶的计算机架构。《计算机》，50(8):18-25,2017。

- [7] Comma.ai. Openpilot. <https://github.com/commaai/openpilot>, 2017。
- [8] Sergio Casas、Abbas Sadat、Raquel Urtasun。MP3:地图、感知、预测和计划的统一模型。在 *CVPR*, 2021 年。
- Kashyap Chitta, Aditya Prakash 和 Andreas Geiger。NEAT:端到端自动驾驶的神经注意力领域。在 *ICCV*, 2021 年。
- Aditya Prakash, Kashyap Chitta 和 Andreas Geiger。用于端到端自动驾驶的多模态融合 transformer。在 *CVPR*, 2021 年。
- [11] 陈典和 Philipp Krähenbühl。向所有车辆学习。在 *CVPR*, 2022。
- 胡圣超, 陈利, 吴鹏浩, 李洪阳, 严俊池, 陶大成。ST-P3:基于时空特征学习的端到端视觉自动驾驶。*arXiv 预印本 arXiv:*, 2022。
- [13] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, Vladlen Koltun。开放的城市驾驶模拟器。在 *CoRL*, 2017 年。
- [14] 谭明星和 Quoc Le。Efficientnet:重新思考卷积神经网络的模型扩展。在 *ICML*, 2019 年。
- [15] 何开明、张翔宇、任少卿、孙坚。深度残差学习用于图像识别。在 *CVPR*, 2016。
- [16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio。使用 rnn 编码器-解码器学习短语表示, 用于统计机器翻译。*arXiv 预印本 arXiv:1406.1078*, 2014。
- [17] 崔恒刚, 周方杰, Lin, 阮氏, 黄子国, Jeff Schneider, Nemanja Djuric。基于深度卷积网络的自动驾驶多模态轨迹预测。在 *ICRA*, 2019 年。
- [18] 霍尔格·凯撒、Varun Bankiti、Alex H Lang、Sourabh Vora、威尼斯 Erin Liong、徐强、Anush Krishnan、Yu Pan、Giancarlo Baldan、Oscar Beijbom。nuScenes:用于自动驾驶的多模态数据集。在 *CVPR*, 2020 年。
- [19] 哈拉尔德·谢弗、埃德尔·桑塔纳、安德鲁·哈登和里卡多·比亚西尼。通勤数据:comma2k19 数据集。*arXiv:1812.05752*, 2018。
- [20] Andreas Geiger、Philip Lenz、Raquel Urtasun。我们为自动驾驶做好准备了吗?kitti vision 基准套件。在 *CVPR*, 2012 年。
- [21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, 等。Pytorch:一个命令式、高性能的深度学习库。在 2019 年的 *NeurIPS* 中。



(一) 一份。



(b) 断路器。



(c)。



(d) 后车。

图 12:ACC 测试。

## Openpilot 分析

### A.1 Openpilot: 一个现实世界的测试

在本节中，我们将详细介绍如何在现实世界中测试 Openpilot。这些案例收集于中国上海的城市、高速公路和人行横道。定性和定量的结果，以及典型的失效案例如下。完整的视频可以在这个网站 <https://sites.google.com/view/openpilot-deepdive/home> 找到。

#### A.1.1 定性结果

**ACC。** 对于 ACC 功能，我们测试了三种不同的情况，包括汽车切入和切断，AEB/FCW 和汽车跟随。图 12a 和图 12b 为车辆插断典型场景。结果表明，当车辆在直线道路上行驶，车速保持在 30 ~ 90kph 范围内时，Openpilot 能够正确识别插断车辆并及时改变车速。此外，我们还在低速 (< 40 公里/小时) 下测试了 AEB/FCW 功能。如图 12c 所示，当前车突然刹车时，Openpilot 会立即将车减速。图 12d 展示了汽车跟驰测试的典型场景。我们在全速度范围 (从 20 到 120 公里/小时) 和包括直道、弯道和十字路口在内的多种场景下测试这个功能。在大多数情况下，Openpilot 可以成功地跟随领先的车辆。

**ALC。** 在 ALC 方面，我们在有可见车道线和没有可见车道线的两种情况下进行了测试。结果表明，当车速保持在 50kph 以上时，只要驾驶员打开转向灯并在方向盘上施加较小的力，Openpilot 就能自动改变行驶车道。但是，Openpilot 不能区分车道线的类型，例如虚线和实线。换句话说，当有实线时，它仍然会尝试改变车道线。图 13a 显示了 Openpilot 在辅助变道过程中



(a)有可见的线路。



(b)没有可见的车道线。

图 13:ALC 的测试。



(a)在交通拥挤的直道上。



(b)在十字路口左转。图 14:对走

走走停停功能的测试。

可见的车道线。图 13b 显示，当弯曲坡道中没有可见的车道线时，Openpilot 仍然可以保持直线。

**走走停停的。**当交通拥挤时，车辆可能需要频繁地启动和停止，这可能会让人类司机感到厌烦。在这种情况下，我们测试 Openpilot 是否能很好地辅助人类驾驶员。从图 14a 可以看出，在交通流量较大的直线道路上，Openpilot 可以在连续启停的情况下紧紧跟随前车。当遇到有红绿灯的路口时，也需要走走停停的功能。如图 14b 所示，在十字路口红灯熄灭时，Openpilot 可以跟随前车启动并左转。

### A.1.2 量化结果

我们还进行了一些定量实验来测试 Openpilot 测量到行驶车道的距离、到前车的距离以及前车的速度的准确性。

**与车道线的距离。**为了测量到车道线距离的误差，我们在直线道路上停车，与左侧车道线的距离不同。然后，我们可以将模型预测结果与逗号二装置与车道线之间的实际投影距离进行比较。结果如表 4a 所示。我们可以得出结论，Openpilot 可以准确地预测车道线的位置。

**与前车的距离。**为了测量这个距离，我们让车辆在路上停下来。演员车被放置在与装置不同的距离上。然后，我们可以在几秒钟内测量平均预测距离和标准推导，并与实际值进行比较。结果如下：



表 4:定量结果。(a)到左侧车道线的距离。

实际距离(m)	0	-0.12	-0.32	-0.49	-0.78	-1.02	-1.20
预测距离(m)	-0.05	0.10	-0.30	-0.56	-0.79	-1.14	-1.32

(b)与前车的距离。

实际距离(m/s)	10	20.	30.	40	50	60	70	80
预测距离(m/s)std. (m/s)	10.41 - 19.61		29.41	34.28	51.75	56.63	58.0	63.88
	0.27 - 0.58		- 1.04	- 1.81	- 3.21	- 4.09	- 8.07	- 5.12

(c)前车的速度。

实际速度 (m/s)	5.556	11.111	16.667
预测速度(m/s)	5.45	10.13	12.86
性病 (米/秒)。	1.01	2.72	4.42

报告于表 4b。我们可以得出结论，Openpilot 能够在 50米内精确测量到领先车辆的距离。

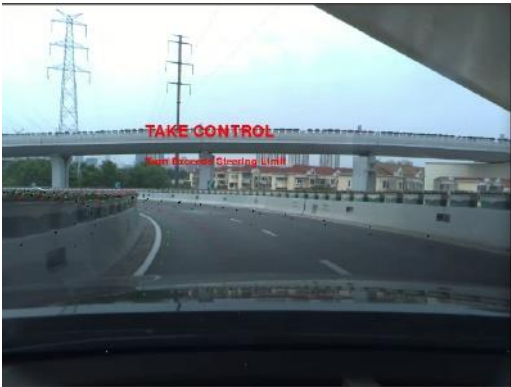
**前车的速度。**为了测量这一点，我们让车辆跟随一辆以恒定速度移动的演员车。同样，我们可以在几秒钟内测量平均速度和标准导数，并与实际值进行比较。结果报告在表 4c。我们可以得出结论，Openpilot 可以准确测量 60 km/h (16.667 m/s)以下的领先车的速度。

A.1.3 故障案例

虽然 Openpilot 在大多数情况下表现出相对较好的性能，但在处理复杂情况时可能不太好。我们总结了一些典型的失败案例。以图 15a 为例，Openpilot 无法识别道路上的圆锥形桶、行人、自行车等物体。图 15b 显示，当车辆快速通过弯曲道路时，Openpilot 可能无法很好地处理，可能会提醒人类驾驶员控制车辆。从图 15c 可以看出，即使在低速状态下，当另一辆车近距离插队时，Openpilot 也可能无法及时做出快速反应。图 15d 显示，Openpilot 在跟随车辆转弯时失去了领先目标，因为领先车辆很快消失。图 15e 显示 Openpilot 在夜间无法探测到领先车辆。



(a)未检测到锥形桶。



(b)速度太快，不能适应弯道。



(c)其他汽车插口紧密。



前车转弯太快。



(e)夜间检测不到车辆。图 15:故障案例。