




实现自驾车Level 2 的端到端模型！一起来了解OpenPilot 是如何运作的



Gi-Luen Huang · Follow

Oct 27, 2023

 53









OpenPilot 是由Comma.ai 开发的一款端到端模型，目前已经成功进入产品阶段，而且还通过实车测试，证实其性能完全符合预期。这款能达到Level 2 自动驾驶水平的模型，推出后在自动驾驶界引起了广泛关注。

文献参考

虽然Openpilot是开源程式码，但在官方的GitHub repo. 中，并未公开他们的模型训练方法或所使用的大型训练资料集。因此，有研究者尝试重现Openpilot的训练细节，并在公开的benchmark资料集上进行测试。他们对自己实现的模型（OP-Deepdive）和原始的Openpilot（Supercombo）进行了实机测试比较。实验数据显示，在nuScenes和Comma2k19这两个公开数据集上，OP-Deepdive的表现与Supercombo相当，甚至在某些方面表现更佳。

 Web page: <https://sites.google.com/view/openpilot-deepdive/home>

 Paper link: <https://arxiv.org/abs/2206.08176>

 官方github link: <https://github.com/OpenDriveLab/Openpilot-Deepdive>

OpenPilot要解决的问题

传统的自动驾驶模型通常采用多个不同模块的组合方式，这些模块分别训练后再整合在一起以实现自动驾驶功能。相反地，OpenPilot采用了端到端（end-to-end）的方法，它直接从相机影像中预测未来的轨迹，并基于这些预测来控制车辆。在Figure 2中，可以清楚地看到传统自驾模型与端到端自驾模型之间的差异：传统模型依赖于各个分开训练的模块来处理不同的驾驶任务（如感知、决策、控制等），而端到端模型则是直接从输入到输出，将这些任务合并为一个整体的学习过程。

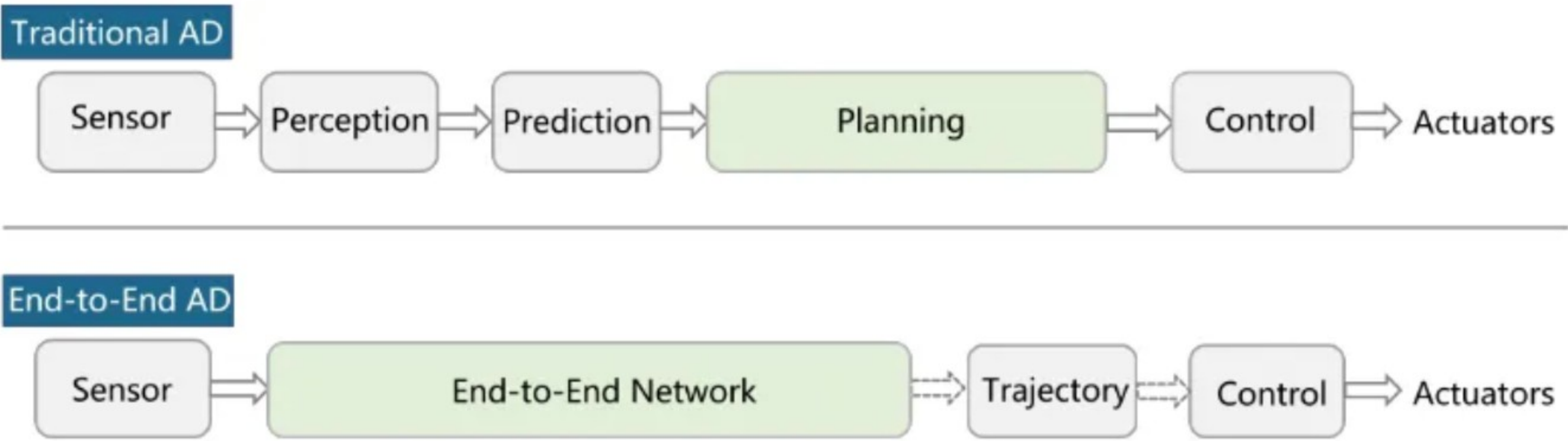


Figure 2: A Comparison of the traditional AD solutions and Openpilot.

OpenPilot 模型— Supercombo

如Figure 5所示，该模型的整体架构相对简单，主要输入包括两个frames ——过去和当前的frame，以及车辆是左驾还是右驾的相关资讯。模型的Backbone采用的是EfficientNet-v2，这一部分用于提取图像特征。此外，模型还使用了GRU（门控递归单元）来捕捉时序上的资讯，增强模型对时间连续性的理解能力。

模型的最终输出包括Meta、Pose、Trajectory planning等多个方面的数据，这些输出共同参与最终的驾驶决策过程。

由于Comma.ai公开的数据集Comma2k19并未提供Supercombo所需的全部输入和输出资料，因此研究者开发的OP-Deepdive模型只实现了其中的Plan部分。这意味着OP-Deepdive主要集中于路径规划的功能，而没有完全涵盖原始Supercombo模型的所有功能。

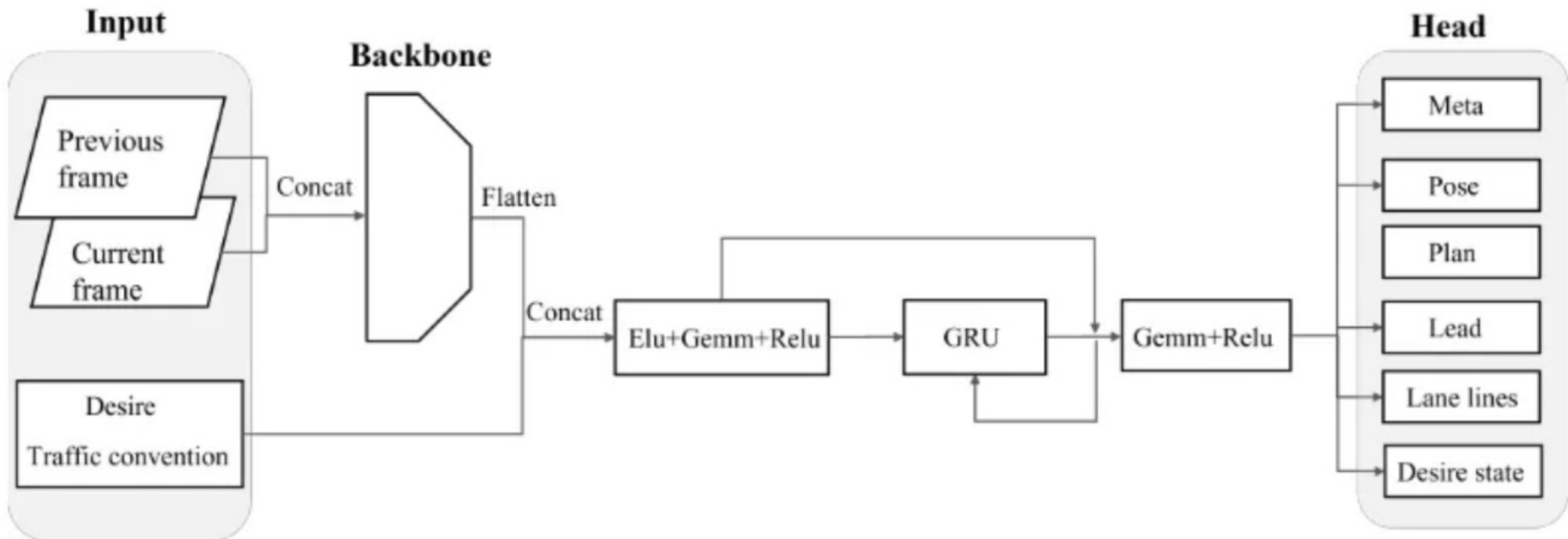


Figure 5: The pipeline of Supercombo. It takes two consecutive frames, a vector representing the desired high level command, and a boolean indicating right/left traffic convention as inputs. Final predictions include a planned trajectory, ego pose, lane lines, state of lead agents, etc.

OP-deepdive 模型

如Figure 6所示，OP-Deepdive模型在基本架构上与Supercombo相似，但在输入和输出方面有所不同。这种差异主要体现在它使用的输入和输出是Supercombo模型的一个子集。在输入方面，OP-Deepdive模型仅使用先前的frame和当前frame作为输入信息。这意味着模型专注于处理这两个时间点的图像数据，以从中提取对驾驶决策有用的信息。在输出方面，OP-Deepdive模型专注于未来轨迹预测（trajectory planning）。这表明模型的主要目标是根据输入的frame数据，预测车辆在未来一段时间内的行驶轨迹。

总的来说，尽管OP-Deepdive在某些方面简化了Supercombo的功能，它依然聚焦于核心的自动驾驶任务——路径规划，这对于实现有效的自动驾驶至关重要。

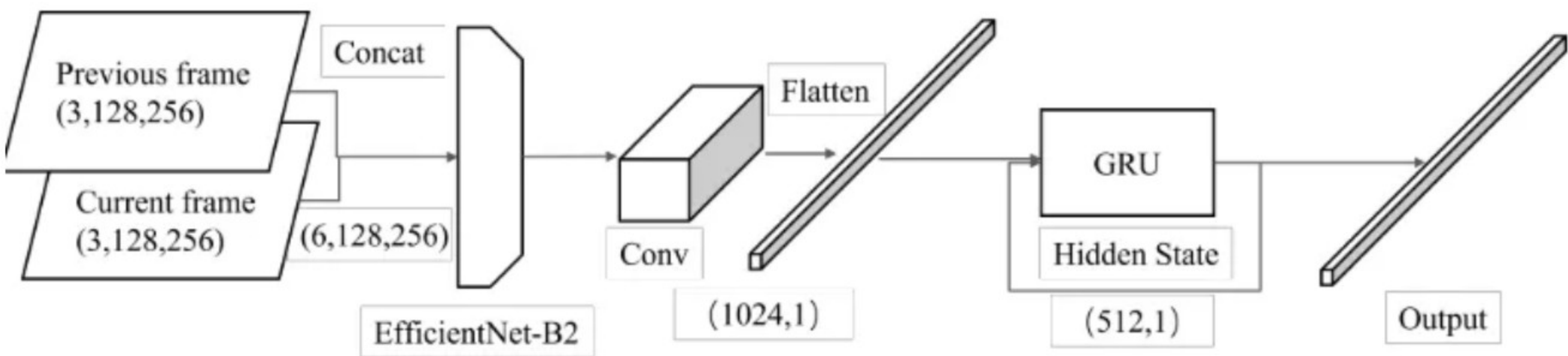


Figure 6: The structure of our reimplemented planning model.

前处理(Preprocessing)

Openpilot为了实现在不同车款及不同地点的通用性，引入了“虚拟相机”（virtual camera）的概念。这一概念的主要作用是减少不同相机的内部和外部参数（例如镜头特性和安装位置）对模型性能的影响。

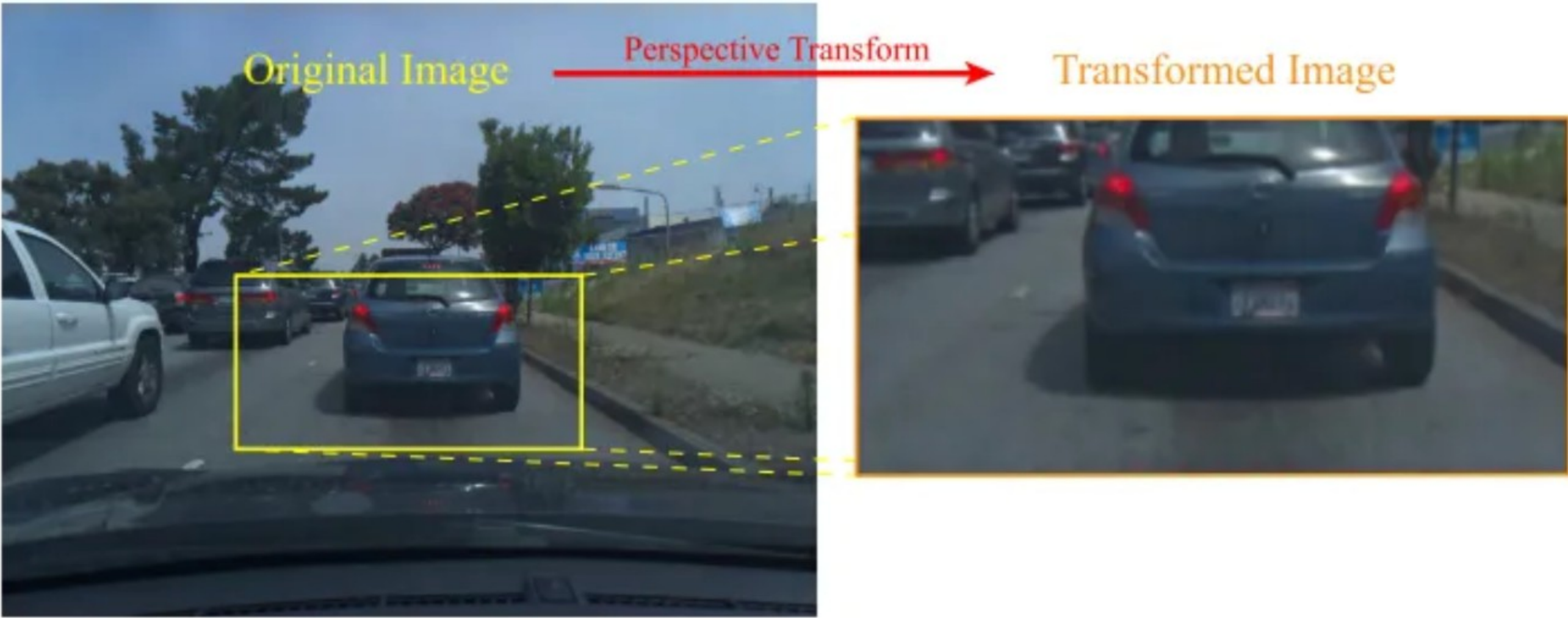


Figure 7: The perspective transformation.

虚拟相机投射的概念涉及几个关键步骤，主要是建立现实相机和虚拟相机之间的几何关系，从而能够将现实相机捕捉的影像“转换”到虚拟相机的视角。这个过程可以分为以下几个主要步骤：

1. 假设相机平放在地面上（ $z=0$ ）：这是一个初始假设，用于建立一个基本的参考平面。在这个平面上，相机被假设为水平放置，这样可以简化计算过程。
2. 设置虚拟相机的高度：确定虚拟相机相对于参考平面的高度。这个高度设置是根据虚拟相机的预期位置和角度来决定的，以符合模型的需求。
3. 确定虚拟相机与地面的关系（外参）：基于虚拟相机的位置和姿态，建立它与地面之间的外部参数。这些参数描述了虚拟相机与地面的空间关系。
4. 了解当前相机与地面的关系：这需要假设或已经知道实际相机的外部参数，即相机与地面之间的实际空间关系。
5. 推导warp matrix：利用上述信息，计算一个变换矩阵（warp matrix），它能够将从实际相机捕捉的影像映射到虚拟相机的视角。这使得从不同实体相机获得的影像能够被转换为一个统一格式，从而使模型能够更有效地处理来自不同车型和安装配置的数据。

通过这个过程，Openpilot能够更好地适应不同车型和相机设置，从而提高其在多种车辆上的通用性和性能，相关的程式码可参考[这里](#)。

损失函数(loss function)

在Openpilot的模型中，为了预测未来轨迹，其核心任务是预测一系列的点，这些点合在一起构成轨迹。具体来说，模型会预测出N个点（在此文章中为33个点），这些预测点会与标准答案(Ground-Truth)计算回归损失，以更新模型

权重。

值得注意的是，为了避免mode collapse 问题，OpenPilot采取了一个特别的策略。mode collapse 是指模型倾向于输出非常相似或重复的解决方案，从而忽视了可能的多样性。为了解决这一问题，OpenPilot不仅仅预测一条轨迹，而是生成了M条（在此文章中为5条）轨迹作为候选方案。然后，使用一个分类器来从这些候选轨迹中选择最终的轨迹，分类器的训练方法为从这M条轨迹中选择一条与Ground-turth 最相近的候选者，这条候选者的标记为1，而其他则标记为0，使用Cross-entropy loss来做分类任务的损失计算。

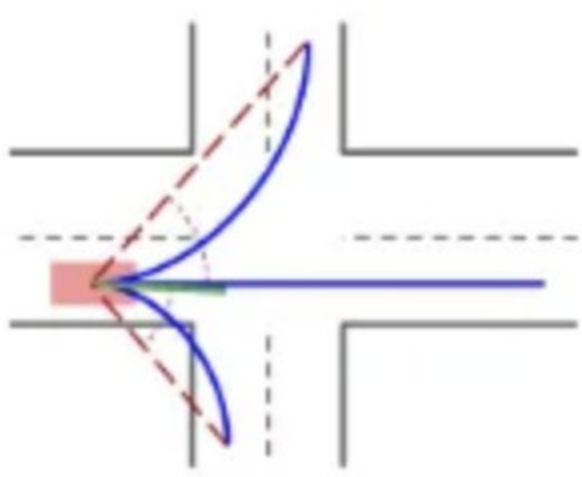


Figure 8: How MTP loss works [17]. In this bird's-eye view (BEV), the pink rectangle represents the ego vehicle. The ground truth trajectory is shown in green. The predicted trajectories are shown in blue. It's obvious that we should choose the trajectory that goes straight as P .

$$\mathcal{L} = \mathcal{L}_{reg} + \alpha \mathcal{L}_{cls},$$

(2)

Experiment

该篇论文使用的公开资料集分别为nuScenes以及Comma2k19，资料集的统计数据如下表所示

Table 1: Comparison between the two datasets used in this report.

Dataset	Raw FPS (Hz)	Aligned FPS (Hz)	Length Per Sequence (Frames/Second)	Altogether Length (Minutes)	Scenario	Locations
nuScenes [18]	12	2	40 / 20	330	Street	America Singapore
Comma2k19 [19]	20	20	1000 / 60	2000	Highway	America

评比的指标有两种: Euclidean distance error 以及Hit rate

- **Average Euclidean Distance Error.** As the name suggests, we calculate the Euclidean distances in 3D space between the corresponding predicted and ground truth points. Then, we average them. Besides, we also calculate the Euclidean distances when the points are projected onto x and y axis. The lower, the better. The unit is *meter*.
- **Average Precision.** If the distance between two corresponding points is smaller than a threshold, we mark it "Hit". Otherwise, we mark it "Miss". Then, we can calculate the hit rate, and average them. For example, AP@0.5 means the average hit rate under threshold 0.5. The higher the better.

从Table 2.可以看出在nuScenes资料集上的比较，OP-Deepdive模型与未经过微调（fine-tuning）的Supercombo模型的表现相当。这表明OP-Deepdive模型在某种程度上能够匹敌原始的Supercombo模型。然而，值得注意的是，当Supercombo模型在nuScenes数据集上进行了微调之后，其性能会有显著提升，成为三者中表现最佳的模型。作者在分析中指出，模型在nuScenes资料集上的Euclidean error 相对较高。这可能是由于nuScenes资料集的特点所导致的。nuScenes数据集的对齐数据是以2FPS（每秒两帧）的频率进行的，这可能使得任务变得较为困难，因为低帧率可能导致重要的时序信息丢失，从而影响模型对车辆动态和环境变化的预测能力。

Table 2: Results on nuScenes dataset.

(a) Imitation metrics for our reimplemented model, OP-Deepdive. **D.E.:** Euclidean Distance Error. **D.E. \vec{x} :** Euclidean distance error projected onto x axis. **AP@0.5:** Average precision under threshold 0.5 meter. The rest can be inferred.

Range (Meter)	D.E. (Meter)	D.E. \vec{x} (Meter)	D.E. \vec{y} (Meter)	AP@0.5	AP@1	AP@2
0-10	2.02	1.92	0.25	0.28	0.51	0.73
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) Imitation metrics for the original Supercombo model.

Range (Meter)	D.E. (Meter)	D.E. \vec{x} (Meter)	D.E. \vec{y} (Meter)	AP@0.5	AP@1	AP@2
0-10	1.96	1.85	0.32	0.237	0.408	0.692
10-20	5.80	5.42	1.04	0.025	0.064	0.170
20-30	8.54	8.11	1.26	0.013	0.038	0.090
30-50	11.36	11.05	1.23	0.008	0.023	0.053

(c) Imitation metrics for the finetuned Supercombo model.

Range (Meter)	D.E. (Meter)	D.E. \vec{x} (Meter)	D.E. \vec{y} (Meter)	AP@0.5	AP@1	AP@2
0-10	1.39	1.31	0.22	0.305	0.568	0.809
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

从Table 3. 可以看出在Comma2k19 dataset中，OP-deepdive的表现甚至比Supercombo model还要好

Table 3: Results on Comma2k19 dataset.

(a) Imitation metrics for our reimplemented model, **OP-Deepdive**.

Range (Meter)	D.E. (Meter)	D.E. \vec{x} (Meter)	D.E. \vec{y} (Meter)	AP@0.5	AP@1	AP@2
0-10	0.451	0.426	0.067	0.909	0.951	0.971
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) Imitation metrics for the original model, **Supercombo**.

Range (Meter)	D.E. (Meter)	D.E. \vec{x} (Meter)	D.E. \vec{y} (Meter)	AP@0.5	AP@1	AP@2
0-10	0.4396	0.2679	0.1998	0.7966	0.9510	0.9829
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

最后，作者也用横向加速度(Lateral acceleration)以及加加速度(Jerk)来衡量预测的轨迹是否舒适，若lateral acceleration以及jerk的值很大，则代表轨迹很不平滑，反之亦然。从下表可以看到目前的模型在Jerk以及Lateral acceleration的指标都远远不及于Human demo，但实际在控制车子时，Openpilot还会做一些后处理来让轨迹变得更平滑。

(c) Comfort metrics.

Method	Average Jerk	Max Jerk	Average Lateral Acc.	Max Lateral Acc.
Supercombo	2.2243	10.8209	0.3750	1.0505
OP-Deepdive (ours)	4.7959	24.007	0.4342	1.7931
Human Demo	0.3232	2.2764	0.3723	0.7592

结语

作为刚进入自驾车领域的我来说，我觉得这篇论文显得很浅写易懂，并且带我初入未来轨迹预测这个任务，而作者提供的source code也很简单，推荐给刚进入自驾车领域的大家，若有任何问题也欢迎跟我讨论! 若要看demo 影片，可至project webpage观看，上面有许多作者们的实车录影。