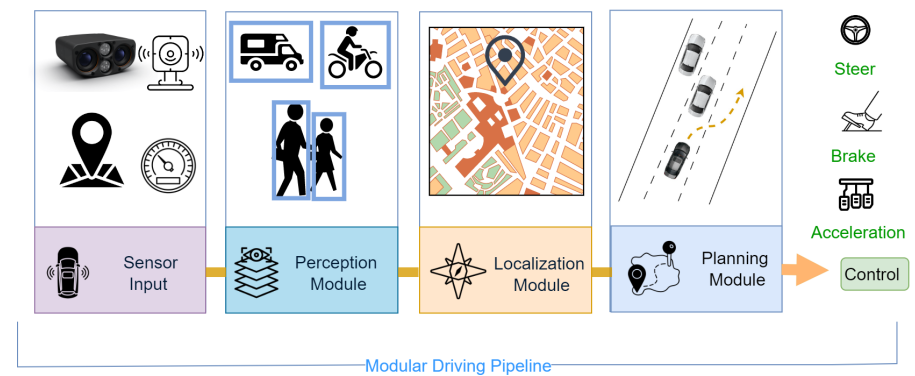
# End to End Autonomous Driving

The initial approaches to Autonomous Driving used modular architecture, which divides the driving pipeline into individual, but interconnected modules – sensor drivers, perception, localization, planning, and control.

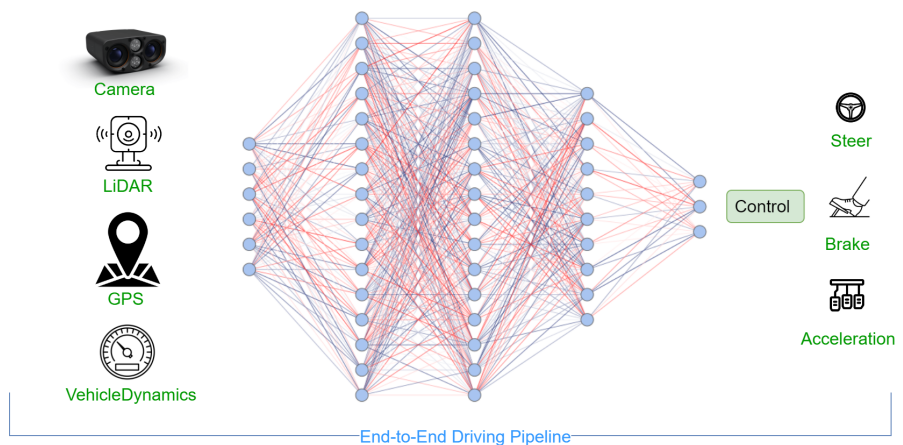


[Image source](https://arxiv.org/pdf/2307.04370.pdf)

While this is how most companies started with, due to its simplicity, interpretability, soon, we started seeing drawbacks

* **Failure in complex scenarios** – Components in Modular architecture are trained for objectives, different than the actual necessity. For example, the perception module aims to compute the 3D pose, orientation and possibly predict the future trajectory of all agents in environment. While this is indeed useful from perspective of object detection, that’s not the end goal. This is reflected in inability to handle complex driving scenarios, as not all cases can be modelled and described.
* **Error propagation** – In such architecture, the downstream modules rely heavily on the upstream modules output. Example, we don’t pass the entire point cloud data to the Planning module. We send the 3D Bounding boxes / Tracks of the static and dynamic objects in environment. So, the planning module doesn’t know about any other object in the environment, if the perception module fails to detect it.
* **Inefficiency** – Each module comprises of a separate Neural network, totalling up to 4 or 5 neural networks**,** each consume their own compute resources, even though lot of features are redundant

One of the solutions is End to End Autonomous Driving, wherein the sensory inputs are directly mapped to control outputs – Steer, Throttle, Brake. It consists of a single learning pipeline, that learns task-specific features, reducing chances of error propagation. It’s also efficient, as features are shared with single goal of generating control signal.



[Image source](https://arxiv.org/pdf/2307.04370.pdf)

## Inputs to End-to-End Driving system

Inputs to End-to-End Driving system, are similar to other systems, mostly comprise the sensor inputs, such as Lidar, Radar, Camera, GPS, vehicle inputs (wheel speed, IMU etc). In some cases, HD (High Definition) maps can also be provided as input. Apart from this, we can have Navigation inputs, such as goal point, intermediate stop points.  
  
Multi-modal inputs offer opportunities for all three types of sensor fusion – early, mid and late or combination of these. For Example, in Transfuser, we have separate feature extraction networks for RGB Camera input and Lidar BEV input, with Transformer module at each intermediate stage to fuse / align features. The concatenated feature outputs are then used to output controls

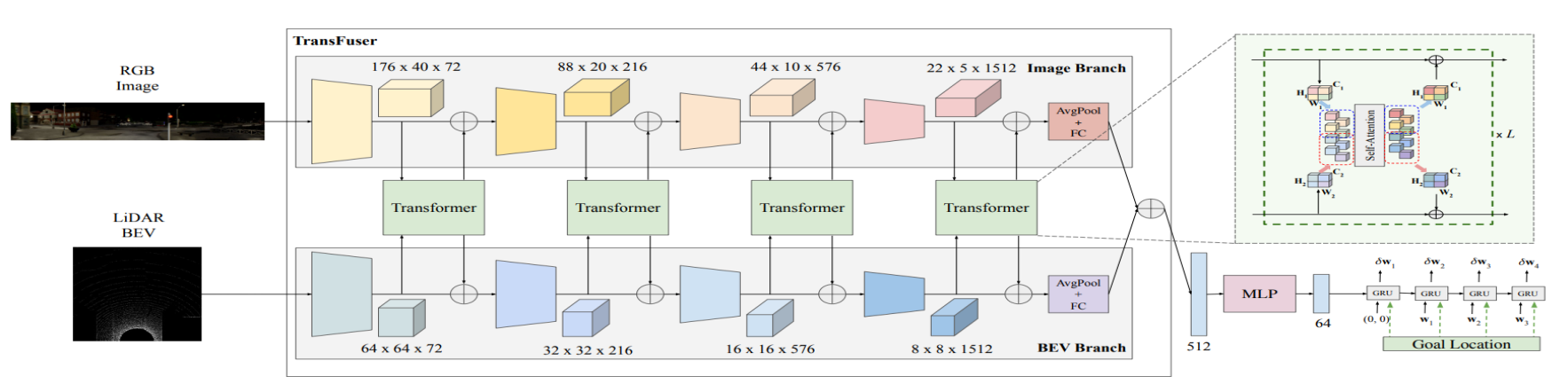


Image reference (Transfuser: Imitation with transformer-based sensor fusion for autonomous driving)

## Outputs from End-to-End Driving system

Most End-to-End models’ outputs are in two forms:

* **Waypoints** – The model predicts future waypoints to follow, in ego-vehicle coordinates. The waypoints are then converted to steering, throttle, brake outputs by traditional control systems such as PID or MPC (Model Predictive controller)
* **Direct Steer, acceleration –** NormalizedSteering angle, acceleration values are provided, which are calibrated according on per-vehicle basis.

Apart from these standard outputs, models can also output auxiliary output such as segmentation / attention maps, occupancy grid, BEV map etc, which help model learn better feature representations and are useful for visualisation as well

There are two main approaches for End-to-End driving – Reinforcement Learning and Imitation Learning. We’ll briefly look into both these categories now

## Imitation learning

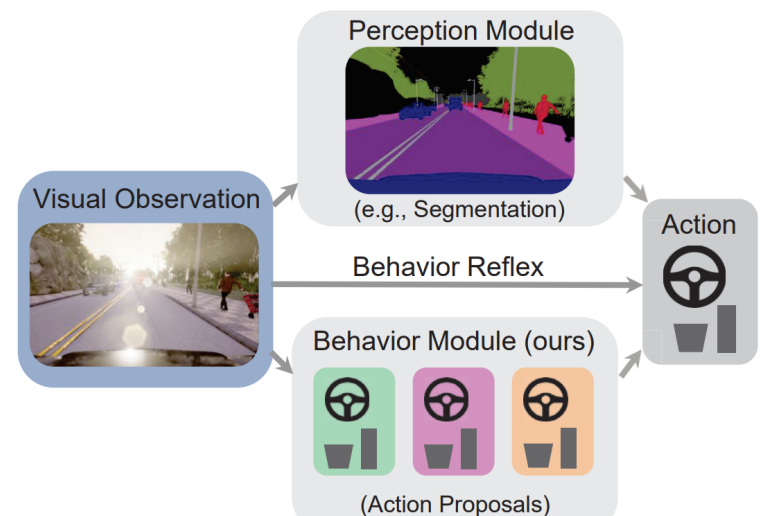
Imitation Learning (IL) is a supervised training method where the model is trained to resemble human driving behaviour. The model learns the driving style from expert demonstrations, which serve as training examples for the model.

As with any supervised training method, we have a dataset with inputs (state x) and outputs (action a). The main objective is to learn a policy 𝜋𝜃 (𝑠) that maps each given state to a corresponding action, and minimizes the difference b/w its own actions and expert actions in a situation. There are 3 variants of Imitation learning:

* Behavioural Cloning (BC)
* Direct Policy Learning (DPL)
* Inverse Reinforcement Learning (IRL)

### Behavioural Cloning

In Behavioural cloning, each dataset sample (e.g. RGB frame and corresponding action) is treated independently, and the aim is to minimize the difference between expert’s actions and the model’s actions. Simply put, all samples are considered independent from each other.



[Image reference] (Learning situational driving)

The main drawback of Behaviour cloning is overfitting. Actual scenarios, can be very different from training, as the dataset is limited.

### Direct Policy Learning

### Inverse Reinforcement Learning

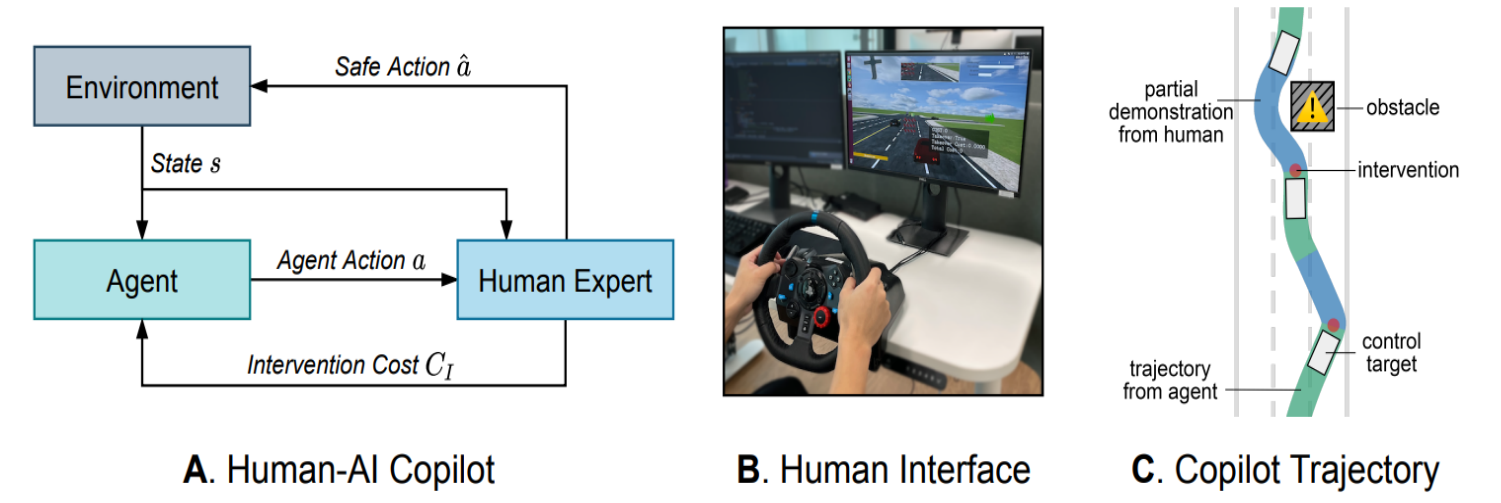
It is difficult to scale Imitation learning, as it is impossible to cover every instance during the learning phase. learning.

## Reinforcement learning

Reinforcement Learning (RL) works by maximizing cumulative rewards over time through interaction with the environment, and the network makes driving decisions to obtain rewards or penalties based on its actions. The training occurs online and allows exploration of the environment during training. But it is less effective in utilizing data compared to IL.

In line with Reinforcement learning terminology, we have an agent (in this case, our model), that can take set of actions (x% throttle, brake, steering) based on its state (position, velocity, other agents’ information) within an environment, which will eventually result in reward or penalty, based on whether goal is accomplished. The model learns to take the action, which will maximise its rewards.

In recent times, Human interaction has been found to improve performance of RL models, particularly in critical scenarios.



[Image reference] (Efficient learning of safe driving policy via human-ai copilot optimization)

In HACO (Human AI Copilot optimization), they allow the agent, to explore hazardous scenarios in a simulated environment. If the agent gets into harmful situations / irrelevant actions, a human expert will intervene, which involves a penalty.

Common approaches in Reinforcement learning include Deep Q-Networks (DQN), Actor-critic based models like Deep Deterministic Policy Gradient (DDPG) and policy-based optimization methods such as Proximal Policy Optimization (PPO).

## Evaluation

End-to-End systems are evaluated on two fronts – open loop and closed loop.

### Open loop Evaluation

Open loop testing means the model’s outputs are not fed to the system. We already have a reference driving behaviour (either by human / simulation), and model’s outputs are compared against this reference. Common metrics include

* Mean squared error (L2 norm) between the predicted and actual trajectories
* Percentage of time the system remains within a certain distance of the desired trajectory
* Ability to handle specific scenarios, such as intersections, lane changes, obstacles

Open loop testing is faster to compute and provide quick initial assessment of model, but they don’t generalize to wider geographies

### Closed loop Evaluation

Closed loop testing means the model’s actions are fed back to the system, Models are evaluated based on completion of specific routes, which involves risky scenarios such as unexpectedly crossing pedestrians or sudden lane changes. Benchmarks such as [CARLA Leaderboard], use following metrics for

## Simulations

Large-scale virtual scenarios can be constructed in virtual engines, enabling the collection of a significant quantity of data more readily. Such simulations are very crucial, particularly for End-to-End autonomous Driving, as there are limited public datasets, compared to traditional computer vision tasks. Simulators offer following advantages:

* Simulate weather conditions
* Simulate traffic flow
* Simulate road agents’ behaviour

Safety-critical scenarios can be generated using simulators, which helps the model detect and prevent such scenarios, in real world conditions. Common simulators

* CARLA
* LGSVL
* Gazebo + ROS
* CarSim + PreScan + MATLAB, Simulink

Apart from standard simulators, Generative models, based on GAN, NeRFs, have been gaining popularity too.

## References

### Research papers

* [TransFuser: Imitation with Transformer-Based Sensor Fusion for Autonomous Driving](https://www.cvlibs.net/publications/Chitta2022PAMI.pdf)
* [End-to-end Autonomous Driving: Challenges and Frontiers](https://arxiv.org/pdf/2306.16927.pdf)
* [PlanT: Explainable Planning Transformers via Object-Level Representations](https://arxiv.org/pdf/2210.14222)
* [NEAT: Neural Attention Fields for End-to-End Autonomous Driving](https://www.cvlibs.net/publications/Chitta2021ICCV.pdf)
* [A Survey of End-to-End Driving: Architectures and Training Methods](https://arxiv.org/pdf/2003.06404.pdf)
* [NeuroNCAP](https://arxiv.org/pdf/2404.07762)
* [DriveGPT4 – Interpretable End to End Autonomous Driving via Large Language model](https://arxiv.org/pdf/2310.01412v4)
* [Recent Advancements in End-to-End Autonomous Driving using Deep Learning: A Survey](https://arxiv.org/pdf/2307.04370.pdf)
* [Planning-oriented Autonomous Driving](https://arxiv.org/abs/2212.10156)
* [ST-P3 Spatio Temporal Feature Learning](https://arxiv.org/pdf/2207.07601.pdf)
* [ReasonNet - E2E with Spatio Temporal global reasoning](https://openaccess.thecvf.com/content/CVPR2023/papers/Shao_ReasonNet_End-to-End_Driving_With_Temporal_and_Global_Reasoning_CVPR_2023_paper.pdf)
* [MOTR: End-to-End Multiple-Object Tracking with Transformer](https://arxiv.org/pdf/2105.03247.pdf)

### Git repos

* [OpenDriveLab E2E driving](https://github.com/OpenDriveLab/End-to-end-Autonomous-Driving)
* [UniAD paper](https://github.com/OpenDriveLab/UniAD)
* [OpenDriveLab ST-P3](https://github.com/OpenDriveLab/ST-P3)
* [MOTR github](https://github.com/megvii-research/MOTR)
* [Transfuser repo](https://github.com/autonomousvision/transfuser)
* [PlanT](https://github.com/autonomousvision/plant)
* [NEAT repo](https://github.com/autonomousvision/neat)

Others

* [CARLA leaderboard](https://leaderboard.carla.org/)
* +[CARLA leaderboard papers with code](https://paperswithcode.com/sota/autonomous-driving-on-carla-leaderboard)