# IRL-DAL: Safe and Adaptive Trajectory Planning for Autonomous Driving via Energy-Guided Diffusion Models

*Seyed Ahmad Hosseini Miangoleh* *Amin Jalal Aghdasian* *Farzaneh Abdollahi*

S.A.H.Miangoleh, A.J.Aghdasian, F.Abdollahi
Department of Electrical Engineering, Amirkabir University of Technology (Tehran Polytechnic)
Tehran, Iran
Corresponding author: Farzaneh Abdollahi, f_abdollahi@aut.ac.ir

This paper proposes a novel inverse reinforcement learning framework using a diffusion-based adaptive lookahead planner (IRL-DAL) for autonomous vehicles. Training begins with imitation from an expert finite state machine (FSM) controller to provide a stable initialization. Environment terms are combined with an IRL discriminator signal to align with expert goals. Reinforcement learning (RL) is then performed with a hybrid reward that combines diffuse environmental feedback and targeted IRL rewards. A conditional diffusion model, which acts as a safety supervisor, plans safe paths. It stays in its lane, avoids obstacles, and moves smoothly. Then, a learnable adaptive mask (LAM) improves perception. It shifts visual attention based on vehicle speed and nearby hazards. After FSM-based imitation, the policy is fine-tuned with Proximal Policy Optimization (PPO). Training is run in the Webots simulator with a two-stage curriculum. A 96% success rate is reached, and collisions are reduced to 0.05 per 1k steps, marking a new benchmark for safe navigation. By applying the proposed approach, the agent not only drives in lane but also handles unsafe conditions at an expert level, increasing robustness.

Keywords: *Autonomous Vehicles, Diffusion Models, Trajectory Planning, Inverse Reinforcement Learning (IRL), Adaptive Attention, Proximal Policy Optimization (PPO), Hybrid Reward*

## 1 Introduction

The main challenge in autonomous vehicles is building systems that can operate at human levels of safety and reliability in highly dynamic environments [1, 2]. This challenge mainly comes from the need to avoid obstacles in a strong and reliable way. Obstacle avoidance is the key safety task that shows whether a system can work in the real world. Even small mistakes in rare situations can cause very serious failures. This means the agent must remain safe even when encountering situations it was not trained on [3]. To address these challenges, recent research has investigated a range of advanced methods. These approaches aim to enhance the safety, reliability, and adaptability of autonomous systems operating in dynamic environments.

### 1.1 Related Work

This work falls within the overlap of four main areas: hybrid learning, reward inference, generative planning, and adaptive perception. The following section reviews some recent studies in each of these areas.

#### 1.1.1 Hybrid Imitation and Reinforcement Learning

Imitation Learning (IL), especially Behavioral Cloning (BC), is widely used in autonomous driving [4]. It provides a data-efficient way to learn a mapping from expert demonstrations to control actions using supervised learning [5, 6]. The primary advantage of BC is its computational efficiency and its requirement for no explicit knowledge of the underlying environment dynamics. Within BC, methods such as Conditional Imitation Learning improve performance by conditioning the policy on high-level commands. This helps address the challenge of having many possible correct actions for a given situation [7]. Even though it is efficient, IL has a problem called covariate shift. In this case, small mistakes grow over time when the agent reaches states that were not in the training data. Unlike traditional methods, RL can develop strong and resilient behaviors by learning from trial-and-error experience. However, it usually needs a lot of data and depends on hand-designed reward signals that can be unreliable [8].

The hybrid method combines IL and RL. This lets the self-driving car first learn basic behavior from examples and then improve it through trial, feedback, and optimization [9]. In [10], a hybrid method

is used that combines BC with the PPO algorithm in the Unity Agents environment. This setup trains racecar agents to drive and avoid obstacles. These combinations capitalize on expert supervision for fast convergence while allowing the agent to explore recovery and adaptation strategies [11]. In [12], the BC-SAC model fuses BC and Soft Actor-Critic to enhance policy robustness and safety in autonomous driving. By employing supervised imitation and reinforcement optimization, one achieves higher generalization and a 38% reduction in failure rate across complex real-world scenarios.

#### 1.1.2 Inverse Reinforcement Learning for Reward Shaping

The problem of designing a good reward for complex tasks can be handled with Inverse Reinforcement Learning (IRL). IRL tries to find the hidden reward rules directly from expert demonstrations [13, 14]. The Conditional Predictive Behavior Planning model combines Conditional Motion Prediction and Maximum Entropy IRL to make driving more like a human. It predicts how nearby cars will react to each possible move and scores these moves using expert driving data [15]. Adversarial inverse reinforcement learning (AIRL) uses a GAN to learn the reward and the driving policy at the same time, which helps the agent adapt to new environments. Safety-aware AIRL then adds safety rules to block risky actions and lower the chance of crashes [16].

#### 1.1.3 Adaptive Perception and Attention in Driving

With the growing use of attention mechanisms in deep learning [17, 18], these methods have been added to end-to-end driving models. They help the system focus more on key visual elements such as vehicles, pedestrians, and road signs [19]. Effective driving requires perception systems that can adapt attention dynamically to changing contexts. In [20], it uses spatiotemporal, uncertainty-aware attention over multi-modal sensors with crossmodal alignment and multiscale fusion to prioritize important actors and regions for downstream planning. In [21], it adds a temporal residual block, multiscale feature fusion, and global plus double attention to use time and image cues better. In [22], it combines adaptive channel attention and grouped spatial attention with channel shuffle to highlight important features. It plugs into standard CNNs, can replace a 3×3 convolution layer, and improves accuracy with little extra compute.

#### 1.1.4 Diffusion Models for Safe Motion Planning

Diffusion models [23] are now leading tools for generating data. They are also being used more and more for planning tasks. When generating trajectories, they can produce diverse future motions that still obey real-world physics [24]. Their key advantage is their flexibility. By guiding the backward diffusion steps with extra rules or helper models, they can add safety rules such as avoiding crashes and keeping motions that the vehicle can really follow [25, 26]. Current methods use diffusion models as direct policy models, mainly in offline reinforcement learning settings [27]. Moreover, some methods use stand-alone planners that first produce open-loop trajectories. A separate controller then follows these trajectories [28].

### 1.2 Research Gap

Despite progress in each area, a key gap appears at their intersection. Current stacks rarely combine generative planning, reward inference, and online policy learning into one system. This causes three main limits:

1. Lack of an end-to-end unified loop: Most diffusion-based planners generate trajectories open loop and are executed separately from the RL policy, creating a distribution mismatch between planned motions and closed-loop control. This separation weakens robustness under disturbances and hinders consistent transfer from trajectory proposals to joint steering–speed commands.

2. Non-adaptive safety trade-offs: Fixed cost weights for lane keeping, collision avoidance, and stability cannot re-balance as scene risk and sensor uncertainty change. Without real-time, perception-

conditioned guidance, systems oscillate between over-conservative and overly aggressive behavior and fail to maintain safety while preserving efficiency.

3. Inefficient learning signals: Reliance on sparse, hand-crafted rewards leads to sample-hungry training, unstable convergence, and sub-expert driving. The absence of dense rewards inferred from demonstrations and a staged curriculum limits generalization to out-of-distribution states and degrades control smoothness.

### 1.3 Motivation

Reliable autonomous driving must jointly address three coupled challenges: learning efficiency, decision safety, and perceptual adaptability.

**Learning** Pure RL needs a lot of data and becomes unstable when reward signals are rare. IL uses data well but still drifts under covariate shift, and detailed rewards made by people often fail to match expert goals in complex traffic.

**Safety and Planning** Reactive policies are fast to respond, but cannot plan very far ahead. Generative planners can guess future events, but they often cannot run in real time or give strong safety guarantees, so there is still a gap between long-term planning and moment-to-moment control.

**Perception** Standard vision encoders treat almost all regions of an image uniformly and struggle to leverage contextual information effectively. In driving, however, attention mechanisms should be dynamic: prioritizing near-field lane coherence to ensure lateral stability at high speeds, while intensifying focus on immediate proximity when potential hazards are detected.

These limitations highlight the need for a unified framework that connects stable imitation with exploratory reinforcement learning.

### 1.4 Contributions

We introduce a IRL–DAL, a cohesive framework that confronts the above challenges through three complementary components:

**(1) Hybrid IL–IRL–RL Training** A structured pipeline that integrates BC for initialization with PPO fine-tuning under a hybrid reward combining sparse environment feedback and dense GAIL-based intent rewards. This ensures stability, sample efficiency, and policy alignment with expert intent.

**(2) Diffusion Planner for On-Demand Safety** A conditional diffusion model serves as a short-horizon, risk-aware planner, activated only in uncertain or high-risk states. It generates candidate trajectories optimized by an energy-based objective penalizing collisions and abrupt control variations, allowing the main policy to internalize safer behaviors via planner feedback.

**(3) Learnable Adaptive Mask (LAM)** A lightweight perception module that dynamically modulates spatial attention based on vehicle kinematics and LiDAR proximity. The mask directs the visual encoder toward context-critical regions—amplifying lower-field road features at high speed for precise lane keeping and highlighting proximate surroundings near hazards—achieving interpretable and efficient attention allocation without heavy self-attention overhead.

## 2 Problem formulation

The paper first formalizes the autonomous driving task as a partially observable Markov decision process POMDP defined by the tuple:

$$(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, \mathcal{Z}, \gamma)$$

which captures the agent partial observability and reliance on noisy sensor data.

State Space ($\mathcal{S}$) The latent true state includes the ego car position, orientation, and speed. It also includes nearby moving objects, the road layout. The policy cannot see this state directly.

Action Space ($\mathcal{A}$) Each action is a continuous two-dimensional vector:

$$a_t = [\,\text{steering},\ \text{speed}\,]$$

which is later mapped by a low-level controller into executable control commands. This design provides smooth, normalized continuous control.

Transition Function ($\mathcal{T}$) The stochastic dynamics:

$$\mathcal{T}(s_{t+1} \mid s_t, a_t)$$

defines how the true state evolves given the action.

Observation Space ($\mathcal{O}$) The multimodal observation at each time step is given by:

$$o_t = \{I_t, L_t, K_t\}$$

Where:

- **Camera Image** ($I_t \in \mathbb{R}^{H \times W \times 3}$): front-facing RGB frame.
- **LiDAR Scan** ($L_t \in \mathbb{R}^{180}$): range readings from a forward LiDAR.
- **Vehicle Kinematics** ($K_t$): normalized ego-vehicle speed.

The LAM is generated internally based on $K_t$ and $L_t$. Thus, they belong to the perception module rather than the raw observation space.

Observation Function ($\mathcal{Z}$) The conditional distribution:

$$\mathcal{Z}(o_t \mid s_{t+1}, a_t)$$

Models the sensing process that generates observations.

Reward Function ($\mathcal{R}$) Since the agent perceives the environment only through observations, the reward depends on both $o_t$ and $a_t$. IRL-DAL uses a hybrid reward. It combines an environment-defined term with an intrinsic term learned by the IRL discriminator.

$$r_t(o_t, a_t) = (1 - w_{\text{IRL}})\, r_{\text{env}}(o_t, a_t) + w_{\text{IRL}}\, r_{\text{IRL}}(o_t, a_t)$$

where $w_{\text{IRL}}$ is a phase-dependent weight used during the mixed training phase, the influence of the learned reward throughout training. Both terms depend solely on observable quantities, ensuring consistency under partial observability.

Discount Factor ($\gamma$) A scalar $\gamma$ that balances short-term rewards with long-term performance.

Objective The agent aims to learn a stochastic policy $\pi_\theta(a_t \mid o_t)$, parameterized by $\theta$, that maximizes the expected discounted return:

$$\mathbb{E}_{\pi_\theta, \mathcal{T}, \mathcal{Z}} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]$$

## 3 Methodology

Figure 1 provides an overview of the entire architecture. Our proposed framework integrates safety, stability, and expert-like decision-making within a single autonomous driving system. As illustrated in Figure 1, the architecture combines four interacting components that collectively enable robust and adaptive behavior: (1) context-aware perception via the LAM, (2) FSM-aware structured replay buffers, (3) PPO fine-tuning with a hybrid reward, and (4) diffusion-based safety supervision with experience correction.

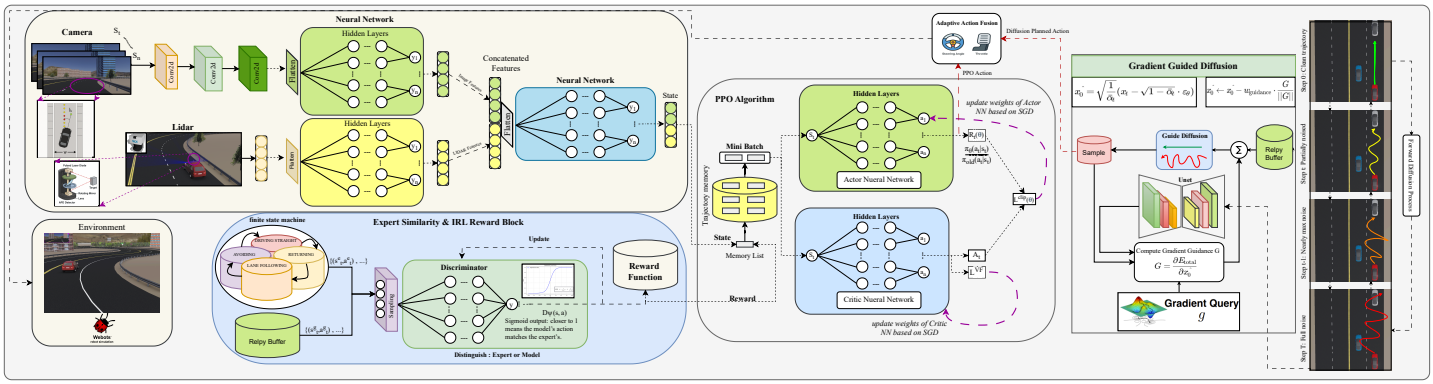


Figure 1: Overview of the IRL-DAL architecture. The training process unfolds in two phases: the policy is first initialized by Behavioral Cloning (BC) and then fine-tuned with Proximal Policy Optimization (PPO) using a hybrid reward $r_{\text{total}}$. A LAM enhances state-aware perception. During RL rollouts, the Diffusion-based Adaptive Lookahead (DAL) planner serves as a safety supervisor, correcting unsafe PPO actions through energy-guided sampling so that only safe experiences are stored in the replay buffer $\mathcal{D}_{\text{PPO}}$.

## 3.1 Perception Module

The perception module uses a learnable attention mechanism that dynamically focuses on the driving situation. This adaptive mask helps the agent build a compact but informative state representation $s_t$ from high-dimensional, multi-sensor inputs. The quality of the driving policy strongly depends on how well it sees and encodes the environment. Standard end-to-end methods often treat all image pixels equally, wasting model capacity. To fix this, we introduce the LAM, a small differentiable module that adds safety-related prior knowledge to the visual input using top-down attention.

Figure 2 shows the LAM architecture and how it connects to the PPO policy. The module takes two inputs: the current vehicle speed $v_t$ and the minimum LiDAR distance $d_{\min,t} = \min(\min(l_t), d_{\max})$, where $l_t \in \mathbb{R}^{180}$ is the raw LiDAR range vector, clipped at the maximum range $d_{\max}$. These signals are then scaled to the interval $[0, 1]$.

$$v_t^{\text{norm}} = \text{clamp}\left(\frac{v_t}{v_{\max}}, 0, 1\right) \tag{1}$$

$$h_t = \text{clamp}\left(\frac{d_{\text{safe}} - d_{\min,t}}{d_{\text{safe}}}, 0, 1\right) \tag{2}$$

where $v_{\max}$ and $d_{\text{safe}}$ are predefined maximum speed and safety distance thresholds, respectively. LAM computes context-dependent modulation factors using two learnable scalar parameters $\alpha_{\text{speed}}, \alpha_{\text{lidar}} \in \mathbb{R}$. They are initialized to 0.5:

$$f_{\text{speed}} = 1 + \alpha_{\text{speed}} \cdot v_t^{\text{norm}}, \quad f_{\text{hazard}} = 1 + \alpha_{\text{lidar}} \cdot h_t \tag{3}$$

These factors scale a base lower-bound intensity weight $w_{\text{base, lower}} = 1.0$, while the upper-bound weight is fixed at $w_{\text{base, upper}} = 0.0$. The resulting dynamic lower intensity is:

$$w_{\text{lower}} = w_{\text{base, lower}} \cdot f_{\text{speed}} \cdot f_{\text{hazard}} \tag{4}$$

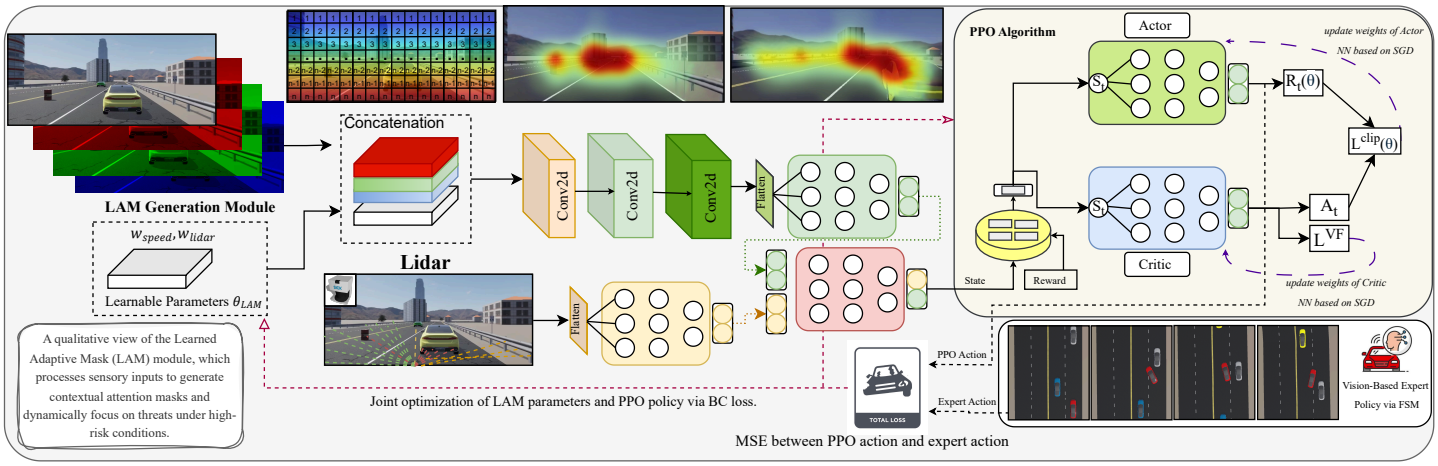


Figure 2: Architecture of the LAM and its integration with the PPO policy. Normalized speed $v_t^{\text{norm}}$ and hazard level $h_t$ modulate a vertical gradient mask via learnable parameters $\alpha_{\text{speed}}$ and $\alpha_{\text{lidar}}$. The resulting mask is concatenated with the RGB image to form a 4-channel input, enabling context-aware visual processing. LAM is trained end-to-end via BC gradients, allowing the agent to discover adaptive attention patterns that prioritize safety-critical regions.

A smooth vertical gradient mask is then constructed for each row $y \in [0, H-1]$ of the input image:

$$M_t(y) = w_{\text{base, upper}} + (w_{\text{lower}} - w_{\text{base, upper}}) \cdot \frac{y}{H-1} \tag{5}$$

To ensure numerical stability and bounded output, the mask is normalized:

$$\hat{M}_t = \frac{M_t}{\max(M_t) + \varepsilon}, \quad \varepsilon = 10^{-6} \tag{6}$$

yielding $\hat{M}_t \in \mathbb{R}^{1\times H\times W\times 1}$.

The normalized RGB image $I_t \in \mathbb{R}^{H\times W\times 3}$ (scaled to $[0,1]$) is concatenated channel-wise with $\hat{M}_t$ to form a 4-channel input tensor:

$$I'_t = \text{concat}(I_t/255.0, \hat{M}_t) \in \mathbb{R}^{H\times W\times 4} \tag{7}$$

This augmented input is processed by a shared convolutional backbone, which also fuses embedded LiDAR features before feeding into the actor and critic heads.

The LAM parameters $\theta_{\text{LAM}} = \{\alpha_{\text{speed}}, \alpha_{\text{lidar}}\}$ are optimized end-to-end during BC alongside the policy using the Adam optimizer with learning rate $\eta_{\text{BC}}$ and L2 regularization applied selectively to policy weights:

$$\mathcal{L}_{\text{BC}} = \frac{1}{B}\sum_{i=1}^{B} \left\| \pi_\theta(s_t^{(i)}) - a_{\text{expert}}^{(i)} \right\|^2 + \lambda_{\text{L2}} \sum_{p\in\theta_{\text{policy}}} \|p\|^2 \tag{8}$$

where gradients flow through the 4-channel observation to update $\theta_{\text{LAM}}$. Training includes gradient clipping (max norm $G_{\max}$) and learning rate scheduling via plateau detection.

As shown in Figure 2, the learned masks adapt dynamically to the driving context. Rather than shifting the geometric center of attention, the mechanism modulates the intensity of the spatial gradient. At high speeds, the mask amplitude significantly increases (via $f_{\text{speed}}$), which strengthens the feature extraction across the entire driveable area, effectively expanding the usable visual range while maintaining a strong prior on the immediate lane path. Similarly, when proximity to obstacles is detected ($d_{\min,t}$), the hazard factor ($f_{\text{hazard}}$) further amplifies the mask intensity. This ensures that the network receives a sharper, high-contrast signal of the immediate surroundings for precise collision avoidance, boosting safety without relying on hand-crafted heuristic rules.

### 3.2 Multi-Phase Learning Curriculum

Policy learning happens in two main phases. First, BC gives the agent a stable and safe starting point by training the policy network on expert demonstrations. This warm start reduces risky, high-variance exploration and ensures the agent begins with reasonable behavior. Then, the policy is improved with PPO [29], which allows controlled on-policy exploration and lets the agent go beyond simple imitation.

To keep PPO updates aligned with expert behavior, an adversarial IRL module based on GAIL [30] is added. This module provides dense, behavior-focused reward signals. The hybrid reward is defined as:

$$r_t = w_{\text{env}} r_{\text{env}}(s_t, a_t) + w_{\text{irl}} r_{\text{irl}}(s_t, a_t) \tag{9}$$

where $r_{\text{env}}(s_t, a_t)$ denotes the environment reward and $w_{\text{env}} + w_{\text{irl}} = 1$. The IRL reward is:

$$r_{\text{irl}}(s_t, a_t) = -\log(1 - D_\psi(s_t, a_t) + \varepsilon) \tag{10}$$

it is bounded within a predefined range.

The curriculum begins with an FSM-partitioned expert dataset $D_{\text{expert}}$, enabling balanced sampling across driving modes. During the imitation phase, both policy and diffusion planner are trained via BC with interval $T_{\text{BC}}^{\text{init}}$. In the mixed phase, PPO updates use hybrid rewards, with discriminator trained every $T_{\text{disc}}$ steps and BC continued adaptively every $T_{\text{BC}}^{\text{mixed}}$ steps. Safety interventions, which are triggered when $d_{\text{min},t} < d_{\text{safe}}$ or $|d_{\text{lane},t}| > e_{\text{lane}}$, are corrected and stored with a metadata flag indicating diffusion-based intervention. This progressive design, which moves from imitation to hybrid RL and then to safety distillation, leads to robust, safe, and efficient learning.

### 3.3 Expert Data Generation via FSM-Aware Experience Replay

A key part of the framework is a reliable and high-quality source of expert demonstrations. Instead of dealing with the noise and inconsistency of real human driving logs, an expert policy is built from scratch using a deterministic FSM policy $\pi^*$. As shown in Figure 3, this FSM controller moves smoothly between four behavior modes: Lane Following, Obstacle Avoidance, Driving Straight, and Returning, with transitions defined by simple sensor-based rules that describe the current driving situation. What sets this approach apart is the FSM-aware experience replay strategy. Each collected transition $(o_t, a_t^*, s_{\text{FSM},t})$ is stored in a separate buffer $D_s$ that corresponds to its active FSM state $s$. The full expert dataset is then formed as follows:

$$D_{\text{expert}} = \bigcup_s D_s$$

This clear, state-based structure helps solve a common problem in autonomous driving datasets, where rare but important events are often underrepresented, such as passing through a narrow gap or recovering from a strong lane drift. By sampling mini-batches evenly across FSM states, the training process sees a balanced mix of normal cruising and challenging edge cases, instead of being dominated by simple highway driving.

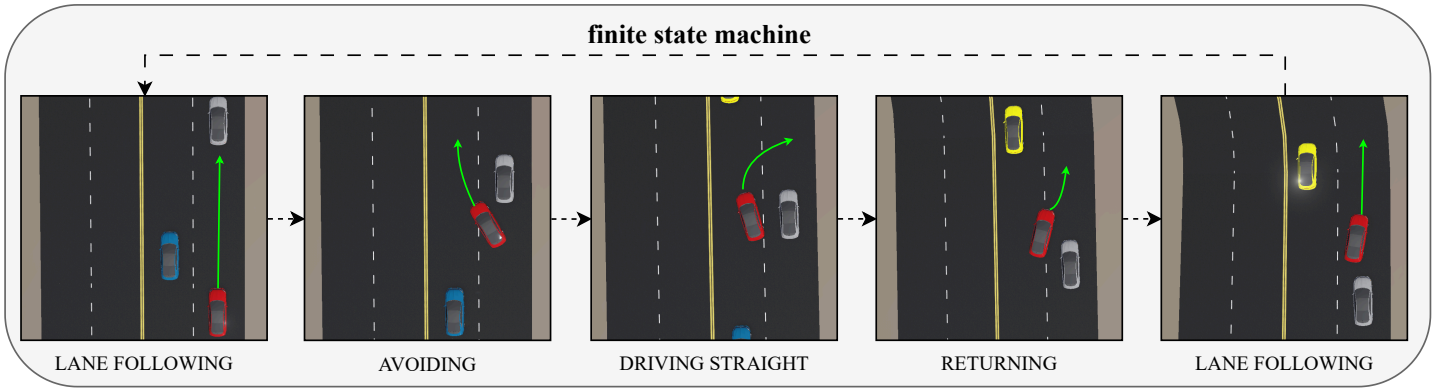


Figure 3: The FSM expert policy in action. It switches smoothly between the modes (Lane Following, Obstacle Avoidance, Driving Straight, Returning) using sensor-based transition rules. The FSM-aware experience replay stores each sample in its matching state buffer, which gives balanced exposure to both normal and risky driving situations.

#### 3.3.1 Phase 1: Foundational Pre-training

The training process starts with a supervised warm up over $N_{\text{imitation}}$ timesteps, using only $D_{\text{expert}}$ to build a safe and reliable behavioral base. This phase protects the agent from unstable and unsafe exploration that often appears at the start of RL. Under BC, the policy $\pi_\theta$ is trained on compact embeddings $s_t = \phi(o_t)$ from the perception module. It learns to match the expert actions using a mean-squared error loss. Balanced sampling from the FSM-partitioned buffers ensures that all driving modes are well represented during training:

$$\mathcal{L}_{\text{BC}}(\theta) = \mathbb{E}_{(o_t,a_t^*)\sim D_{\text{expert}}^{\text{balanced}}}\left[\|\pi_\theta(s_t) - a_t^*\|^2\right] + \lambda_{\text{L2}} \sum_{p\in\theta_{\text{policy}}} \|p\|^2 \tag{11}$$

This loss keeps the policy close to expert behavior long before any reward signal is used. Diffusion planner training runs concurrently with BC. A conditional 1D U-Net diffusion planner is trained to generate smooth and feasible motion sequences. It is trained on consecutive expert action chunks $\{a_t^*, \ldots, a_{t+H-1}^*\}$ taken from $D_{\text{expert}}$ and optimized with the standard DDPM denoising objective [31]. In this way, the planner learns the smooth and physically consistent control patterns of the FSM expert and is prepared to act later as a reliable safety net.

#### 3.3.2 Phase 2: Online Fine-tuning with Adversarial Reinforcement Learning

After a strong base is built, the system runs for $N_{\text{mixed}}$ timesteps of online refinement with PPO. During this phase, the policy is improved through interaction with the environment, using a hybrid reward that combines clear task feedback with imitation-based signals. Under GAIL, a discriminator $D_\psi$ is trained to go beyond simple action matching. It learns to tell the difference between policy rollouts $(s_t, a_t)$ and true expert pairs $(s_t^*, a_t^*)$ using a binary cross-entropy loss:

$$\begin{aligned}\mathcal{L}_{\text{Disc}}(\psi) = &-\mathbb{E}_{(s_t,a_t)\sim\pi_\theta}[\log(1 - D_\psi(s_t, a_t))] \\ &-\mathbb{E}_{(s_t^*,a_t^*)\sim D_{\text{expert}}}[\log(D_\psi(s_t^*, a_t^*))]\end{aligned} \tag{12}$$

Once training is complete, the discriminator provides a dense and continuous imitation reward:

$$r_{\text{irl}}(s_t, a_t) = -\log(1 - D_\psi(s_t, a_t) + \varepsilon) \tag{13}$$

Balanced sampling across FSM states ensures that the discriminator sees the full range of expert behaviors.

In the hybrid reward formulation, the final PPO reward is designed to balance task completion with expert-level behavior.

$$r_t(s_t, a_t) = w_{\text{env}} r_{\text{env}}(s_t, a_t) + w_{\text{irl}} r_{\text{irl}}(s_t, a_t) \tag{14}$$

where $w_{\text{env}} + w_{\text{irl}} = 1$. This combination helps the agent achieve high-level goals, such as steady progress and zero collisions, while also matching the smooth, anticipatory style of the FSM expert.

## 3.4 Safety Assurance via Guided Diffusion and Experience Correction

Finally, a diffusion-based planner is added as a safety-focused backup system. The DAL module is activated only in high-risk situations and quickly generates short, feasible paths that keep the vehicle safe. At the same time, it improves learning by storing safe, corrected experiences in the policy memory, so the agent becomes better over time without reinforcing unsafe behaviors. Together, these parts (perception, curriculum, expert replay, and safety) form an integrated pipeline that combines fast reactions with careful planning and leads to behavior that is effective, safe, and close to human driving.

Even when the PPO policy learns complex driving patterns, it is still hard to stay safe in rare or unexpected situations. To reduce these risks, the framework uses two safety components:

1. An on-demand, energy-guided diffusion planner that is activated to generate safe actions when the situation becomes dangerous.
2. An experience correction mechanism that sends these safe corrections back to the main policy so that temporary fixes become lasting improvements.

### 3.4.1 On-Demand, Energy-Guided Trajectory Generation

The diffusion planner trained in the first phase now acts as a safety-critical motion generator. It stays inactive most of the time to save computation and is only turned on in high-risk states. A state is treated as high-risk when the closest LiDAR reading becomes too small or the vehicle moves far away from the lane center:

$$d_{\text{min},t} < d_{\text{trigger}} \quad \vee \quad |d_{\text{lane},t}| > e_{\text{lane}} \tag{15}$$

letting the PPO policy drive freely under normal conditions.

When DAL is called, it generates a short safe trajectory $\mathcal{A} = a_t, \ldots, a_{t+H-1}$ based on the current context embedding. This is done with a guided reverse diffusion process. At each denoising step $k$, the U-Net predicts a cleaner trajectory $\hat{\mathcal{A}}_k^0$, which is then moved toward safer behavior by following the gradient of a composite energy function $E(\mathcal{A}, o_t)$:

$$\tilde{\mathcal{A}}_k^0 = \hat{\mathcal{A}}_k^0 - w_g \cdot \frac{\nabla E(\hat{\mathcal{A}}_k^0, o_t)}{\|\nabla E\| + \varepsilon} \tag{16}$$

where $w_g$ determines how strongly the planner pushes trajectories toward safe regions during sampling, and $\tilde{\mathcal{A}}_k^0$ denotes the resulting safety-guided trajectory obtained after applying the energy gradient.

The energy $E$ is a weighted sum of five simple terms that together enforce lane keeping, obstacle clearance, smooth control, stability, and an optional expert-alignment term.

$$E = E_{\text{lane}} + E_{\text{lidar}} + E_{\text{jerk}} + E_{\text{stability}} + E_{\text{expert}} \tag{17}$$

Lane Adherence — $E_{\text{lane}}$: This term penalizes deviation from the lane center. Instead of computationally expensive forward modeling, the energy term utilizes the current lateral error state distance ($d_{\text{lat}}$) to strictly guide the diffusion sampling process toward immediate correction.

$$E_{\text{lane}} = w_{\text{lane}} \left( \frac{d_{\text{lane},t}}{s_{\text{lane}}} \right)^2 \tag{18}$$

with a risk-adaptive weight $w_{\text{lane}}$ that increases under higher hazard levels:

$$w_{\text{lane}} = w_{\text{base}}^{\text{lane}}(1 + \alpha_{\text{hazard}} h_t)$$

where $w_{\text{base}}^{\text{lane}}$ is the nominal weighting coefficient for lane keeping, $\alpha_{\text{hazard}}$ controls how aggressively the weight increases under hazardous conditions, and $h_t$ denotes the instantaneous hazard indicator.

Obstacle Avoidance — $E_{\text{lidar}}$: To keep a safe distance from obstacles, trajectories generated while the vehicle is in close proximity to obstacles are penalized based on the current sensor state ($d_{\min,t}$).

$$E_{\text{lidar}} = w_{\text{lidar}} \cdot \max\left(0, \frac{d_{\text{safe}}^{\text{plan}} - d_{\min,t}}{s_{\text{lidar}}}\right)^2 \tag{19}$$

where the weight $w_{\text{lidar}}$ increases with the hazard level, according to $w_{\text{lidar}} = w_{\text{base}}^{\text{lidar}}(1 + h_t)$.

Control Smoothness — $E_{\text{jerk}}$: To avoid jerky or hard-to-drive behavior, large changes between consecutive actions are penalized:

$$E_{\text{jerk}} = w_{\text{jerk}} \cdot \frac{1}{H-1} \sum_{i=1}^{H-1} \|a_i - a_{i-1}\|^2 \tag{20}$$

where $w_{\text{jerk}}$ is the weighting coefficient for smoothness and $H$ is the trajectory horizon.

Stability — $E_{\text{stability}}$: Steady, centered control is encouraged by penalizing large steering or speed deviations from a neutral reference:

$$E_{\text{stability}} = w_{\text{stab}} \cdot \frac{1}{H} \sum_{i=0}^{H-1} \left(\text{steer}_i^2 + (\text{speed}_i - v_{\text{ref}})^2\right) \tag{21}$$

where $w_{\text{stab}}$ is the stability weight and $v_{\text{ref}}$ denotes the target reference speed.

Expert Alignment — $E_{\text{expert}}$ (optional): When an expert reference is available, an additional term keeps the trajectory close to the expert demonstration:

$$E_{\text{expert}} = w_{\text{exp}} \cdot \frac{1}{H} \sum_{i=0}^{H-1} \|a_i - a_i^{\text{expert}}\|^2 \tag{22}$$

where $w_{\text{exp}}$ controls the strength of expert-matching and $a_i^{\text{expert}}$ is the corresponding expert action.

The hazard level $h_t$ adapts all risk-sensitive weights:

$$h_t = \text{clamp}\left(1 - \tanh\left(\frac{d_{\min,t}}{s_h}\right), 0, 1\right) \tag{23}$$

where $d_{\min,t}$ is the minimum LiDAR distance at time $t$, $s_h$ is a scaling factor determining sensitivity to obstacles, and the clamp limits the value to the range $[0, 1]$.

Conditioning is performed using a compact context vector $c_t \in \mathbb{R}^{64}$, which encodes current hazard level, lateral deviation, speed, steering, and LiDAR statistics. Each element is normalized, and unused entries are padded for dimensional consistency.

### 3.4.2 Action Blending and Execution

Only the first action $a_t^{\text{DAL}}$ from the refined trajectory is used. During the mixed phase, this action is smoothly blended with the PPO action using a dynamic blend weight:

$$w_b = \begin{cases} 1.0, & d_{\min,t} < d_{\text{critical}} \\ w_{\text{base}}^{\text{blend}} + w_{\text{scale}}^{\text{blend}} h_{\text{blend}} & \text{otherwise} \end{cases} \tag{24}$$

where:

$$h_{\text{blend}} = e^{-d_{\min,t}/s_1} + k_{\tanh} \tanh(|d_{\text{lane},t}|/s_2) \tag{25}$$

and $w_b$ is temporally smoothed via EMA. The final executed action is:

$$a_t^{\text{final}} = w_b a_t^{\text{DAL}} + (1 - w_b) a_t^{\text{PPO}} \tag{26}$$

### 3.4.3 Experience Correction

The executed transition is stored in the PPO replay buffer with a correction marker, and any unsafe PPO action is replaced by the safe blended output to prevent accidents (Runtime Shielding). However, the experience stored in the replay buffer remains anchored to the FSM expert action. This ensures the policy learns stable rule-based behaviors while the diffusion layer prevents premature episode termination.

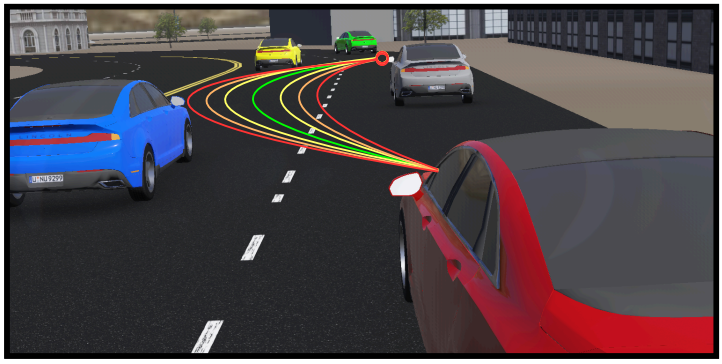

Figure 4: The safety pipeline in action. DAL is activated by high-risk signals, generates energy-guided safe trajectories, and blends the first action with the PPO output. The corrected experiences are stored in the replay buffer with markers, The diffusion planner acts as an active shield, allowing the agent to safely collect more high-quality expert data even in challenging scenarios.

## 3.5 Safety-Aware Experience Correction (SAEC)

To ensure robust training without premature episode termination, the SAEC mechanism employs the diffusion planner as a runtime safety shield. As described in Algorithm 1, the planner is activated when the agent enters a high-risk state, defined by:

$$d_{\text{min},t} < d_{\text{trigger}} \vee |d_{\text{lane},t}| > e_{\text{lane}} \tag{27}$$

where $d_{\text{min},t}$ is the closest LiDAR reading and $d_{\text{lane},t}$ measures the lateral deviation. Once triggered, DAL generates a safe action to prevent immediate collision. However, to ensure the policy converges to a stable kinematic behavior, we do not train on the corrective diffusion action itself. Instead, we utilize the concurrent Rule-Based Expert (FSM) to calculate the ideal ground-truth action ($a_{\text{Expert}}$) for that specific state.

The transition stored in the PPO replay buffer $D_{\text{PPO}}$ is therefore defined as:

$$D_{\text{PPO}} \leftarrow D_{\text{PPO}} \cup (o_t, a_{\text{Expert}}, r_t, o_{t+1}) \tag{28}$$

This mechanism provides two critical benefits:

- **Runtime Survival:** The diffusion action is executed to physically prevent collisions, allowing the episode to continue beyond states that would normally cause termination.
- **Safe Expert Labeling:** By keeping the agent active in near-accident scenarios, the system allows the FSM expert to label these "edge cases" with correct recovery actions. The policy $\pi_\theta$ thus learns obstacle avoidance based on the FSM's consistent logic, while the diffusion layer acts solely as a guardrail during this learning process.

## 3.6 Policy Optimization with Safety-Aware Curriculum

The core of the system, the driving policy, is trained with a safety-first curriculum that combines the stable reliability of IL with the flexibility of RL. This combination leads to behavior that is not only skilled but also naturally cautious and robust. The full training procedure is summarized in Algorithm 1.

### 3.6.1 RL Backbone: PPO

To improve the policy through interaction with the environment, PPO is used, an on-policy actor-critic method well suited to smooth, high-dimensional control. Its clipped objective keeps updates stable and prevents large policy changes that could disrupt training. The full optimization objective is:

$$L(\theta) = \hat{\mathbb{E}}_t \left[ L_t^{\text{CLIP}}(\theta) + c_1 L_t^{\text{VF}}(\phi) - c_2 S[\pi_\theta](s_t) \right] \tag{27}$$

where $L_t^{\text{CLIP}}$ is the clipped policy loss, $L_t^{\text{VF}}$ is the value function error, and $S[\pi_\theta]$ is an entropy term that encourages the policy to keep exploring. Raw observations are first converted into compact embeddings $s_t = \phi(o_t)$ by a shared perception backbone (Sec. 3.1). After this step, the actor and critic each use their own lightweight MLP head.

### 3.6.2 Multi-Phase Training Curriculum

Training is carried out in two phases. It starts with an imitation warm up and then moves to a reward driven refinement phase.

**Phase 1 — Imitation Pre-training:** Over the first $N_{\text{imitation}}$ timesteps, both the policy $\pi_\theta$ and diffusion planner are bootstrapped via BC on the FSM-aware expert dataset $D_{\text{expert}}$ (Sec. 3.3). Balanced sampling across driving modes guarantees exposure to rare but critical edge cases:

$$\mathcal{L}_{\text{BC}} = \mathbb{E}_{(o_t, a_t^*) \sim D_{\text{expert}}^{\text{balanced}}} \left[ \|\pi_\theta(s_t) - a_t^*\|^2 \right] + \lambda_{\text{L2}} \|\theta_{\text{policy}}\|^2. \tag{28}$$

This phase plants a **safe, expert-grade behavioral prior**, slashing unsafe exploration right from the start.

**Phase 2 — IRL-PPO Fine-tuning with Hybrid Reward:** Once a solid foundation is set, the agent goes for $N_{\text{mixed}}$ timesteps, refining itself with PPO under a **hybrid reward** that fuses crisp environmental feedback with rich, learned imitation cues:

$$r_t = w_{\text{env}} r_{\text{env}}(s_t, a_t) + w_{\text{irl}} r_{\text{irl}}(s_t, a_t) \tag{29}$$

where $w_{\text{env}} + w_{\text{irl}} = 1$.

IRL Reward from GAIL discriminator:

$$r_{\text{irl}}(s_t, a_t) = -\log(1 - D_\psi(s_t, a_t) + \varepsilon), \quad r_{\text{irl}} \in [r_{\min}, r_{\max}]$$

The environment reward $r_{\text{env}}$ is dense and FSM aware. It is adapted to the current driving mode and combines terms for precise lane centring, obstacle clearance, and sparse goal completion. A collision gives a large negative reward, a near miss gives a smaller penalty, and a minimum baseline keeps the reward positive in safe states.

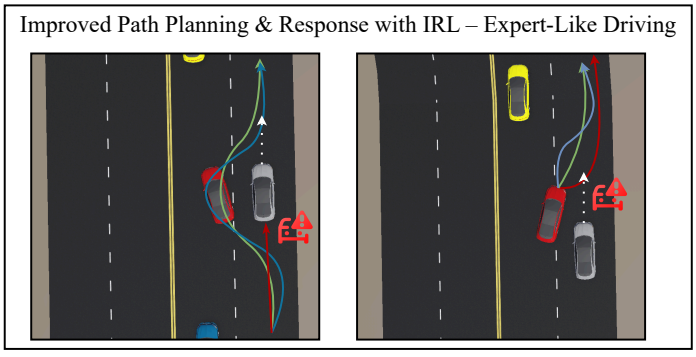


Figure 5: Impact of Hybrid Reward Shaping on Trajectory Smoothness. The blue trajectory illustrates the policy trained solely on the rule-based environment reward ($r_{\text{env}}$), resulting in oscillatory behavior and excessive lateral deviation. In contrast, the green trajectory incorporates the dense IRL signal ($r_{\text{IRL}}$), effectively regularizing the policy towards the expert's kinematic profile. This results in smoother lane-change maneuvers (left) and tighter collision avoidance (right), correcting the overshooting tendencies observed in the baseline.

### 3.7 Integrated Training Pipeline

Algorithm 1 ties everything together: FSM-aware replay, BC, GAIL, PPO, DAL, and SAEC.

**Algorithm 1:** Hybrid IRL–DAL Training Framework

**Initialize:** $\pi_\theta$, $V_\phi$, $P_{\text{DAL}}$, $D_\psi$, $\mathcal{D}_{\text{buffer}} \leftarrow \emptyset$
**Phase 1: Imitation (Warm-up & Data Collection)**
**for** $t \leftarrow 0$ **to** $N_{imitation} - 1$ **do**
  Observe $o_t$, compute expert action $a_t^* \leftarrow \text{FSM}(o_t)$
  Execute $a_t^*$, observe $o_{t+1}$
  Store $(o_t, a_t^*)$ in $\mathcal{D}_{\text{buffer}}$
  **if** $t \bmod T_{BC} = 0$ **then**
    Update $\pi_\theta$ via BC on $\mathcal{D}_{\text{buffer}}$ (Eq. 28)
  **end**
  **if** $t \bmod T_{diffusion} = 0$ **then**
    TrainDiffusion($\mathcal{D}_{\text{buffer}}$)
  **end**
**end**
**Phase 2: Mixed (RL + DAL Safety + Regularization)**
**for** $t \leftarrow N_{imitation}$ **to** $N_{total} - 1$ **do**
  Sample agent action $a_t^{\text{PPO}} \sim \pi_\theta(o_t)$
  Compute expert action $a_t^* \leftarrow \text{FSM}(o_t)$
  **if** $d_{\min,t} < d_{trigger}$ ***or*** $|d_{lane,t}| > e_{lane}$ **then**
    Sample $\mathcal{A} \sim P_{\text{DAL}}(o_t)$ with energy guidance
    $a_t^{\text{final}} \leftarrow \text{blend}(a_t^{\text{DAL}}, a_t^{\text{PPO}})$
  **else**
    $a_t^{\text{final}} \leftarrow a_t^{\text{PPO}}$
  **end**
  Execute $a_t^{\text{final}}$, observe $r_{\text{env}}, o_{t+1}$
  $r_{\text{irl}} \leftarrow -\log(1 - D_\psi(o_t, a_t^{\text{final}}) + \varepsilon)$
  $r_t \leftarrow w_{\text{env}} r_{\text{env}} + w_{\text{irl}} r_{\text{irl}}$
  Store $(o_t, a_t^{\text{final}}, a_t^*, r_t)$ in $\mathcal{D}_{\text{buffer}}$
  **if** $t \bmod T_{disc} = 0$ **then**
    TrainDiscriminator($\mathcal{D}_{\text{buffer}}$)
  **end**
  **if** $t \bmod T_{sync} = 0$ **then**
    SyncFeatures()
  **end**
  **if** *SHOULD_RUN_BC_TRAINING()* **then**
    TrainBC($\mathcal{D}_{\text{buffer}}$)
  **end**
  Update $\pi_\theta, V_\phi$ via PPO using collected rollouts
**end**

## 4 Simulations and Results

### 4.1 Experimental Setup

All experiments are done in the Webots simulator. Webots provides realistic vehicle motion and flexible scene settings. The virtual city has multi-lane curved roads, moving obstacles, and different lighting conditions from bright day to dark evening. This setup makes the tests close to real urban driving. For reproducibility, all details of the perception backbone are given. It takes a dictionary-style observation

that includes camera images and LiDAR scans.

1. **Input Data:** The model takes two input streams. The visual stream is a 4-channel tensor $I'_t \in \mathbb{R}^{H\times W\times 4}$, formed from a downscaled RGB image and the LAM output. The auxiliary stream is a 1D LiDAR scan $L_t \in \mathbb{R}^{N_{\text{beams}}}$ that provides distance measurements in all directions.

2. **Visual Encoder:** A three-layer convolutional network with kernel sizes 5, 3, and 3 processes the visual input. It uses ReLU activations and stride 2 for downsampling, and a final flatten layer produces a compact visual feature vector $z_{\text{vision}}$.

3. **LiDAR Encoder:** The LiDAR vector is first normalized and then passed through a three-layer MLP with ReLU activations. This network produces a 32-dimensional embedding $z_{\text{lidar}}$ that captures the scene geometry.

4. **Feature Fusion:** The two latents are concatenated and polished by a small fusion MLP to produce the final state $s_t \in \mathbb{R}^{512}$:
$$\boldsymbol{s}_t = \text{MLP}_{\text{fusion}}(\text{concat}[\boldsymbol{z}_{\text{vision}}, \boldsymbol{z}_{\text{lidar}}]). \tag{30}$$
This fused vector gives the policy a complete view of the scene, with fine local structure from LiDAR and global context from vision, and it is used in both the imitation warm-up and the later online RL training.

5. **Environment and Episode Termination:** A custom Gym wrapper updates observations at 10 Hz. At each step, it provides an $H \times W \times 4$ visual tensor (RGB plus mask) and an $N_{\text{beams}}$-beam LiDAR scan. An episode ends either when a collision occurs ($d_{\min} < d_{\text{collision}}$) or when the goal is reached. A successful episode is one that reaches the goal without any violations.

6. **Training Procedure:** Training followed the two-phase safety-aware curriculum shown in Algorithm 1, for a total of $N_{\text{total}}$ timesteps. All transitions were stored in FSM-partitioned replay buffer, so that rare safety-related events are well represented. The main hyperparameters are given in Table 1.

Table 1: Training hyperparameters

| Parameter | Value (from code) |
|---|---|
| PPO Steps / Batch / Epochs | 2048 / 64 / 10 |
| Discount Factor ($\gamma$) / GAE-$\lambda$ | 0.99 / 0.95 |
| BC / PPO Learning Rate | $3 \times 10^{-4}$ |
| Diffusion Horizon ($H$) / Diffusion Steps ($T$) | 8 / 100 |
| Total Buffer Capacity | 50,000 transitions |
| IRL Reward Weight ($w_{\text{irl}}$) | 0.3 |
| Environment Reward Weight ($w_{\text{env}}$) | 0.7 |
| Imitation Phase Duration ($N_{\text{imitation}}$) | 20,000 steps |
| Mixed Phase Duration ($N_{\text{mixed}}$) | 30,000 steps |
| Total Training Steps ($N_{\text{total}}$) | 50,000 steps |
| BC Update Interval (Imitation) | every 1,000 steps |
| BC Update Interval (Mixed) | every 500 steps |
| Diffusion Training Interval | every 500 steps |
| Discriminator Update Interval ($T_{\text{disc}}$) | every 2,000 steps |
| Feature Sync Interval ($T_{\text{sync}}$) | every 1,000 steps |
| L2 Regularization ($\lambda_{\text{L2}}$) | $10^{-6}$ |
| PPO Clip Range | 0.2 |
| Entropy Coefficient ($c_2$) | 0.01 |
| Value Loss Coefficient ($c_1$) | 0.5 |
| Gradient Clip Norm ($G_{\max}$) | 0.5 |
| Learning Rate Scheduler | Reduce on Plateau (patience=5) |
| LiDAR Beams ($N_{\text{beams}}$) | 180 |
| Image Size ($H \times W$) | $64 \times 64$ |
| Collision Threshold ($d_{\text{collision}}$) | 1.0 m |
| DAL Trigger: $d_{\text{trigger}}$ / $e_{\text{lane}}$ | 3.0 m / 120 px |
| Critical Blend Threshold ($d_{\text{critical}}$) | 1.5 m |
| Hazard Scale ($s_h$) | 3.0 |
| Blend Hazard Scales ($s_1$, $s_2$, $k_{\tanh}$) | 2.0 / 20 / 0.3 |
| DAL Guidance Weight ($w_g$) | 0.1 |
| Energy Weights ($w_{\text{jerk}}$, $w_{\text{stab}}$, $w_{\text{exp}}$) | 0.1 / 0.5 / 2.0 |
| Lane Scale ($s_{\text{lane}}$) / LiDAR Scale ($s_{\text{lidar}}$) | 0.5 / 2.0 |
| LAM Alphas Initial Value | 0.5 |
| Optimizer | Adam ($\beta_1 = 0.9$, $\beta_2 = 0.999$) |

## 4.2 Component-wise Evaluation of the IRL–DAL Framework

To measure the effect of each component, a clear ablation study was carried out. Four model variants were trained under the same conditions, adding components one by one to separate their individual contributions. The evaluated variants are as follows: The Baseline (PPO + Uniform Sampling) is the standard PPO algorithm with a uniform replay buffer. The second variant, + Structured Replay, builds upon the baseline model by incorporating FSM-aware replay, which aims to provide more coverage to safety-critical states. The third variant, + Generative Planner, includes the structured replay along with the diffusion-based safety planner, but without adaptive perception. Finally, the + LAM + SAEC (Full IRL-DAL) represents the full model, incorporating the adaptive mask, safety-aware experience correction, and all components enabled. These results are summarized in Table 2 and visualized in Fig. 6, which display reward curves, BC alignment, and intervention decay.

Table 2: Quantitative performance across architectural variants (10 seeds, mean ± std). Mean reward normalized to [0, 200]. Trajectory prediction metrics (ADE/FDE) from rollout evaluation. Arrows indicate improvement direction; **bold** denotes best.

| Model | Mean Reward ↑ | Coll./1k Steps ↓ | Success (%) ↑ | BC Loss ($\times 10^{-2}$) ↓ | Action Sim. (%) ↑ | ADE (m) ↓ | FDE (m) ↓ |
|---|---|---|---|---|---|---|---|
| PPO + Uniform Sampling | 85.2 ± 4.1 | 0.63 ± 0.12 | 78.1 ± 3.2 | 17.1 ± 1.4 | 65.3 ± 4.1 | 5.25 ± 0.31 | 11.8 ± 0.65 |
| + FSM Replay | 120.4 ± 3.8 (+41%) | 0.30 ± 0.08 | 88.4 ± 2.1 | 12.3 ± 1.1 | 75.1 ± 3.5 | 4.10 ± 0.27 | 9.5 ± 0.58 |
| + Diffusion Planner | 155.1 ± 3.2 (+29%) | 0.15 ± 0.05 | 92.0 ± 1.8 | 13.0 ± 1.0 | 80.2 ± 3.0 | 3.15 ± 0.22 | 7.2 ± 0.49 |
| **+ LAM + SAEC (Ours)** | **180.7 ± 2.9** (+16%) | **0.05 ± 0.03** | **96.3 ± 1.2** | **7.4 ± 0.8** | **85.7 ± 2.4** | **2.45 ± 0.18** | **5.1 ± 0.41** |

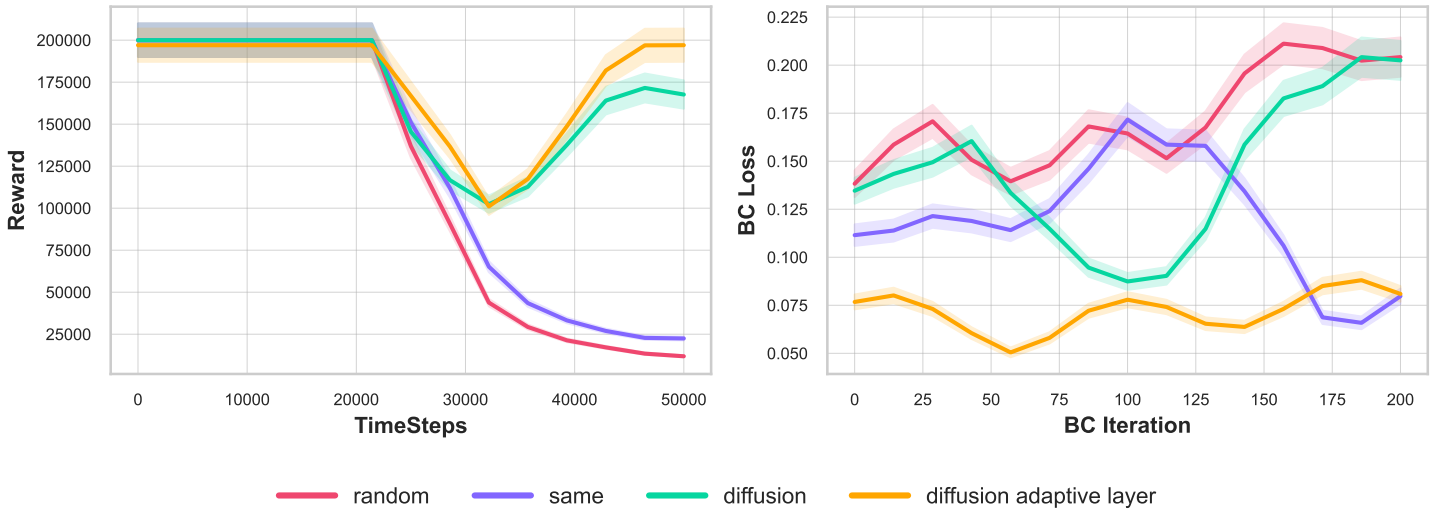


Figure 6: Training dynamics over 50k steps. After switching to mixed-mode at 20k steps, the full IRL–DAL model (*Diff+Adapt+SAEC*) rockets ahead in reward and stability. BC loss stays pinned low, and DAL interventions fade to near-zero—proof of **adaptive distillation**. See Table 2 for final numbers.

Structured replay and safety awareness have a key role in the results. Using only the FSM buffer increases the reward by 41% and reduces collisions by 52% (from 0.63 to 0.30 per 1k steps), which shows that rare events need special attention. Adding DAL increases the reward by another 29% and again cuts collisions by half (from 0.30 to 0.15), so it not only reacts to danger but also helps prevent it (Fig. 7). With the full model, including adaptive perception (LAM) and SAEC, collisions drop to 0.05, which is a 67% reduction compared to the previous step and 12.6 times better than the baseline. At the same time, BC loss stays low and action similarity to the expert increases, so expert-like behavior is reached without sacrificing stability.

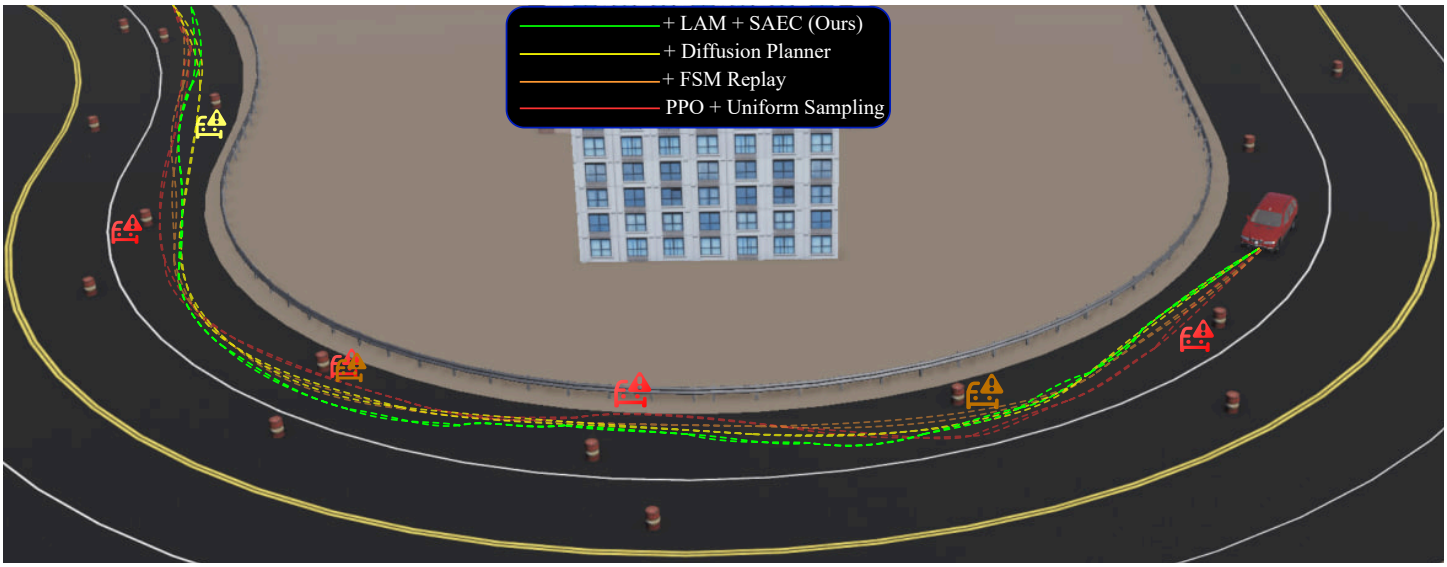


Figure 7: Qualitative ablation study on a high-curvature turn scenario in Webots. The **Baseline PPO (Red)** fails to negotiate the curve, resulting in a collision. Adding **FSM Replay (Orange)** improves lane adherence but exhibits oscillatory behavior. The **Diffusion Planner (Yellow)** successfully ensures safety but deviates from the center. The full **IRL–DAL framework (Green)** demonstrates superior trajectory smoothness and precise lane centering, attributing to the LAM's focused attention on road boundaries.

## 5 Conclusion

This work introduced IRL-DAL, a unified framework for safe and adaptive autonomous driving that combines inverse reinforcement learning, diffusion planning, and on-policy control. The method links three key ideas in a single loop: hybrid IL-IRL-RL training, a diffusion-based safety planner, and a LAM for perception. Together, these parts let the agent learn an expert-like driving policy that remains stable under online interaction and reacts safely in high-risk situations.

The framework was evaluated in the Webots simulator using a two-phase curriculum. In the first phase, an FSM expert and behavioral cloning provided a safe and reliable starting point for both the policy and the diffusion planner. In the second phase, PPO with a hybrid reward and GAIL-based IRL signals refined the behavior while DAL and SAEC enforced safety and turned interventions into useful training data. The final agent reached a 96% success rate and reduced collisions to 0.05 per 1k steps. Ablation studies showed clear gains from each component: FSM-aware replay improved coverage of rare events, the diffusion planner reduced further failures, and LAM plus SAEC delivered the larger safety and performance gains.

### Data Availability Statement

All data used in the experiments — including scenario configurations, expert FSM trajectories, and logged rollouts — were generated in the Webots environment and are made available via the project website.

### Code Availability Statement

The full implementation of the IRL-DAL framework is published and accessible at https://seyed07.github.io/Autonomous-Driving-via-Hybrid-Learning-and-Diffusion-Planning/. Researchers and practitioners can reproduce the experiments and use the code under the terms described there.

### Conflict of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References


[1] L. Crosato, K. Tian, H. P. Shum, E. S. Ho, Y. Wang, C. Wei, *Advanced Intelligent Systems* **2024**, *6*, 3 2300575.

[2] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. E. Sallab, S. Yogamani, P. Pérez, *IEEE Transactions on Intelligent Transportation Systems* **2021**, *23*, 6 4909.

[3] S. Ross, G. Gordon, J. A. Bagnell, In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. **2011** 627–635.

[4] M. Cui, Y. Hu, S. Xu, J. Wang, Z. Bing, B. Li, A. Knoll, *Advanced Intelligent Systems* **2023**, *5*, 10 2300269.

[5] G. Li, Z. Ji, S. Li, X. Luo, X. Qu, *IEEE Transactions on Intelligent Vehicles* **2022**, *8*, 2 1025.

[6] X. Wang, C. Wei, H. Tian, W. Wang, J. Hu, *IEEE Transactions on Intelligent Vehicles* **2024**.

[7] H. M. Eraqi, M. N. Moustafa, J. Honer, *IEEE Transactions on Intelligent Transportation Systems* **2022**, *23*, 12 22988.

[8] C. Yuan, C. Shuai, Z. Zhang, J. Ma, Y. Fang, Y. Sun, *Advanced Intelligent Systems* **2025**, 2400991.

[9] M. O. Radwan, A. A. H. Sedky, K. M. Mahar, In *2021 31st International Conference on Computer Theory and Applications (ICCTA)*. IEEE, **2021** 215–222.

[10] R. Mahmoudi, A. Ostreika, In *IVUS*. **2023** 214–221.

[11] F. Pinto, S. Al-Stouhi, In *SAE World Congress Experience*. SAE International, **2021** .

[12] Y. Lu, J. Fu, G. Tucker, X. Pan, E. Bronstein, R. Roelofs, B. Sapp, B. White, A. Faust, S. Whiteson, et al., In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, **2023** 7553–7560.

[13] R. Zhao, Y. Li, Y. Fan, F. Gao, M. Tsukada, Z. Gao, *IEEE Transactions on Intelligent Transportation Systems* **2024**.

[14] G. Lanzaro, T. Sayed, *Expert Systems with Applications* **2025**, *260* 125405.

[15] Z. Huang, H. Liu, J. Wu, C. Lv, *IEEE Transactions on Intelligent Transportation Systems* **2023**, *24*, 7 7244.

[16] P. Wang, D. Liu, J. Chen, H. Li, C.-Y. Chan, In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, **2021** 1036–1042.

[17] C. Lu, S. Ali, T. Yue, *IEEE Transactions on Software Engineering* **2024**.

[18] X. Xi, X. Jin, Q. Jiang, Y. Lin, W. Zhou, L. Guo, *Advanced Intelligent Systems* **2023**, *5*, 11 2300310.

[19] Y. Wang, W. Wan, H. Zhang, C. Chen, G. Jia, In *2024 4th International Conference on Electronic Information Engineering and Computer Communication (EIECC)*. IEEE, **2024** 1519–1522.

[20] M. Schwonberg, J. Niemeijer, J.-A. Termöhlen, N. M. Schmidt, H. Gottschalk, T. Fingscheidt, et al., *IEEE Access* **2023**, *11* 54296.

[21] X. Chi, Z. Guo, F. Cheng, *Alexandria Engineering Journal* **2024**, *105* 538.

[22] X. Ma, K. Hu, X. Sun, S. Chen, *International Journal of Intelligent Systems* **2024**, *2024*, 1 3934270.

[23] M. Peng, K. Chen, X. Guo, Q. Zhang, H. Zhong, M. Zhu, H. Yang, *IEEE Transactions on Intelligent Transportation Systems* **2025**.

[24] J. Wang, J. Guo, M. Feng, C. Li, X. Xue, J. Pu, *IEEE Transactions on Intelligent Vehicles* **2024**.

[25] T. Pearce, A. Canonaco, H. Zhu, I. Havoutis, N. Pugeault, *IEEE Robotics and Automation Letters* **2023**, *8*, 6 3607.

[26] C. Chi, V. Little, S. Zhang, Z. Wang, S. Wang, A. Chang, M. Sung, In *Conference on Robot Learning*. **2023** 2209–2220.

[27] Z. Wang, M. Janner, Y. Du, C. Finn, S. Levine, In *International Conference on Learning Representations*. **2023** .

[28] M. Janner, Y. Du, J. B. Tenenbaum, S. Levine, In *International Conference on Machine Learning*. **2022** 9489–9502.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, *arXiv preprint arXiv:1707.06347* **2017**.

[30] J. Ho, S. Ermon, *Advances in neural information processing systems* **2016**, *29*.

[31] J. Ho, A. Jain, P. Abbeel, In *Advances in Neural Information Processing Systems*, volume 33. **2020** 6840–6851.

## Biographies

### Biography

Seyed Ahmad Hosseini Miangoleh is currently pursuing the B.S. degree in electrical engineering (control) at Amirkabir University of Technology, Tehran, Iran. His research interests include control systems, robotics, machine learning, deep learning, reinforcement learning, and computer vision.

**Biography**

Amin Jalal Aghdasian received the B.S. degree in electrical engineering from Tabriz University, Tabriz, Iran, in 2018, and the M.S. degree in mechatronics engineering from Amirkabir University of Technology, Tehran, Iran, in 2023. His research interests include intelligent automotive systems, robust control, machine learning, neural networks, robotics, and reinforcement learning.

**Biography**

Farzaneh Abdollahi (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from the Isfahan University of Technology, Isfahan, Iran, in 1999, the M.Sc. degree in electrical engineering from the Amirkabir University of Technology, Tehran, Iran, in 2003, and the Ph.D. degree in electrical engineering from Concordia University, Montreal, QC, Canada, in 2008. She is currently an Associate Professor with the Amirkabir University of Technology and an adjunct Professor with Carleton University. Her research interests include intelligent control, robotics, control of nonlinear systems, control of multiagent networks, and robust and switching control.