COMPSCI 687 - Final Project - Fall 2025

Autonomous Driving in Foggy Environments

By: Gomathy Dhanya Sankara Subramanian
Jeevesh Krishna Arigala

Introduction

Driving in dense fog is a major safety hazard because the limited visibility requires highly cautious navigation from both human drivers and autonomous vehicles. Modern autonomous vehicles must be equipped to handle subnormal weather conditions where visibility drops, road curvature becomes uncertain, and the positions and speeds of other cars become difficult to judge. Motivated by this challenge, this project introduces FoggyDriving, a fully custom reinforcement learning environment designed to study autonomous driving under restricted perception due to fog. To simplify a complex continuous environment, motion is discretized into unit speed changes and lane shifts along a simplified grid-like road that scrolls, while still supporting rich interactions through dynamic fog, noisy lidar sensing, and realistic traffic. The ego agent receives only partial observations consisting of lidar distances, its current lane, speed, and fog level, making the task a partially observable MDP. Surrounding vehicles follow behaviors derived from the IDM car-following model and the MOBIL lane-changing heuristic, creating stochastic, multi-agent traffic flow that the agent must predict and react to.

Using this environment, agents were trained with PPO, A2C, and DQN using Stable Baselines 3. The learned policies showed adaptive behavior such as maintaining safe distances, adjusting speed, safe overtaking and performing lane changes under fog-impaired visibility. PPO achieved the most stable and effective learning, A2C showed moderate performance, and DQN struggled with the noisy, partially observable dynamics.

To summarize, the project is a custom RL environment incorporating fog-dependent lidar visibility, IDM/MOBIL traffic behavior, and discrete driving actions. Using Stable Baselines 3, PPO, A2C, and DQN agents were trained to navigate safely by learning policies that adapt to visibility constraints and dynamic traffic. A custom renderer visualizes the foggy road, traffic, lidar beams, and agent behavior.

Environment Description

The FoggyDriving environment models a simplified, fog-affected highway as a discrete, grid-like world with two lanes and a finite visible horizon. Time is discretized into steps, and at each step the ego vehicle interacts with surrounding traffic under limited visibility imposed by fog and noisy lidar sensing.

Road Geometry

- The road is modeled as a 2 × 40 grid where grid_width = 2 represents the two lanes (Lane 0 – left lane, Lane 1 – right lane) and grid_height = 40 represents the visible stretch of road ahead.
- The ego vehicle is positioned at the bottom of the grid and remains fixed in place in the renderer while other vehicles move relative to it.
- The y-axis scrolls infinitely, simulating continuous forward motion, and cars move along the road in unit-distance increments.

Ego Car

- The ego car is the learning agent, fixed at the bottom of the 2×40 grid while the world scrolls around it.
- Its state consists of lane (0 or 1), speed (between min_speed and max_speed), and cumulative distance traveled.
- It chooses from five discrete actions: maintain speed, accelerate, decelerate, move left, or move right.
- The ego receives only partial observations (lane, speed, fog level, and lidar), never the raw positions of other cars.

Other Cars

- Other vehicles are defined by a lane, distance from the ego, current speed(actual velocity at the current step), and desired speed (cruising speed used by IDM)
- Longitudinal movement is governed by the IDM model, which adjusts speed smoothly based on traffic ahead.
- Lane changes follow the MOBIL rule, allowing cars to switch lanes only when it is safe and beneficial.
- Cars move relative to the ego; their distance increases or decreases depending on speed differences.
- New cars are spawned near the top of the grid when space permits, and cars leaving the visible region are despawned.

Intelligent Driver Model (IDM)

- IDM coined by Treiber, Hennecke, and Helbing applied to the Foggy Driving environment:
    1. Find the leader in the same lane (the nearest car ahead). If none exists, assume the gap is very large.
    2. Compute the actual gap s = leader.dist − car.dist − car_length
    3. $s' = s_0 + vT + (v\,\Delta v) / (2\sqrt{(a\,b)})$ where
        a. s' : desired dynamic gap (how much space the car wants to maintain)
        b. s_0 : minimum spacing at standstill
        c. v : the car's current speed
        d. T : preferred time distance to the leader
        e. Δv: relative speed:

4. $a_{IDM} = a(1 - (v / v_0)^{\delta} - (s^* / s)^2)$ where
    a. a_{IDM} : the IDM-computed acceleration applied to the car
    b. a : maximum comfortable acceleration
    c. v_0 : desired (free-flow) speed the car aims to reach
    d. δ : acceleration exponent controlling how sharply acceleration decreases near v_0

Minimizing Overall Braking Induced by Lane Changes (MOBIL)

- MOBIL coined by Treiber and Kesting applied to the foggy driving environment
    1. Check lane validity where target lane must be within bounds
    2. Calculate current acceleration $a_{current} = a_{IDM}(car, current\ leader)$
    3. Identify hypothetical leader in the target lane: $a_{target} = a_{IDM}(car, target\ leader)$
    4. Check incentive criterion: $a_{target} - a_{current}$ >threshold
    5. Identify follower in the target lane and compute the follower's hypothetical acceleration after the lane change $a_{follower} = a_{IDM}(follower, car)$ and check if $a_{follower} > - safe\_brake$
    6. If all conditions pass, update lane

Lidar Sensing

- The ego vehicle receives N discrete lidar beams (lidars), evenly spaced across a horizontal field of view spanning ±45°.
- Each lidar beam is cast forward from the ego car into the grid, stepping along the ray until either:
    a. it intersects another car's bounding box, or
    b. it reaches the maximum visible range allowed by the current fog level.
- The maximum sensing range is determined by fog intensity max_range_by_fog[fog_level] and distance readings are normalized by the fog-limited maximum range
- Small multiplicative Gaussian noise is added to each reading, with noise magnitude slightly increasing under heavier fog providing a noisy, range-limited, partial observation of the scene.

Fog Model

- The environment defines multiple fog levels ranging from 0 (clear) to a maximum configured value where each fog level corresponds to a visibility range, computed as:
  $max\_range\_by\_fog[fog] = grid\_height * (decay^{fog})$
- Fog directly affects lidar readings by limiting the maximum distance a beam can travel and the fog level changes stochastically during an episode.

MDP Formulation

$S\ (State\ Space)\ =\ [lane\_onehot,\ speed_{norm},\ fog_{norm},\ lidar_1, ..., lidar_N]$ where

- **lane_onehot**: one-hot encoding of lane ∈ {0,1} (2 dimensions)
- **normalized speed**: ego speed scaled to [0,1]
- **normalized fog level**: current fog ∈ [0,1,2…max_fog_levels]
- **normalized lidar distances**: N lidar beams scaled by fog-limited max range

$A\ (Action\ Space)\ =\ \{0, 1, 2, 3, 4\}$ where:

- **0**: maintain speed
- **1**: accelerate
- **2**: decelerate
- **3**: change lane left
- **4**: change lane right

$p\ (Transition\ Dynamics)$

1. **Fog transitions:**
   With probability 0.2, each step a new fog level is sampled uniformly from $\{0, ..., max\_fog\_levels\}$, otherwise the fog level remains unchanged.
2. **Traffic lane changes:**
   Each car independently considers a lane change with probability 0.2**,** but the MOBIL criteria deterministically decide whether the change occurs.
3. **Traffic spawning:**
   For each lane, if sufficient free space exists, a new car is spawned with probability 0.2, with its distance and desired speed sampled uniformly from predefined ranges.
4. **Lidar noise:**
   Each lidar beam reading is perturbed by multiplicative Gaussian noise such that $\epsilon \sim N(0, \sigma^2)\ with\ \sigma\ =\ 0.02(1\ +\ 0.03fog)$

$d_0\ (Initial\ State\ Distribution)$

1. Ego lane : $ego\_lane \sim Uniform\{0, 1\}$
2. Ego Speed : $ego\_speed\ =\ (min\_speed\ +\ max\_speed)/2$
3. Fog Level : $fog \sim Uniform\{0, 1, ..., max\_fog\_levels\}$
4. Initial Number of cars : $N \sim UniformInteger(5, 9)$ and for each of the cars:
   - $lane \sim Uniform\{0, 1\}$
   - $dist \sim Uniform(4, grid\_height)$
   - $v \sim Uniform(min\_speed,\ max\_speed\ -\ 1)$
   - $v0 \sim Uniform(max(v, min\_speed\ +\ 1),\ max\_speed)$
5. 3 cars such that Lane:
   - $lane\ =\ ego\_lane$
   - $Distance: dist \sim Uniform(2, 4)$
   - $Speed: v\ =\ min\_speed$
   - $Desired\ speed: v0 \sim Uniform(min\_speed\ +\ 0.5,\ max\_speed\ -\ 1)$

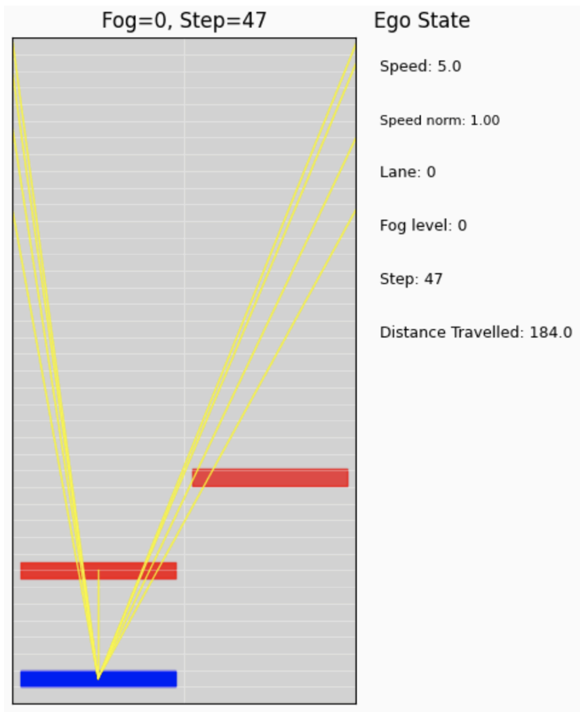
$R\ (Reward)$

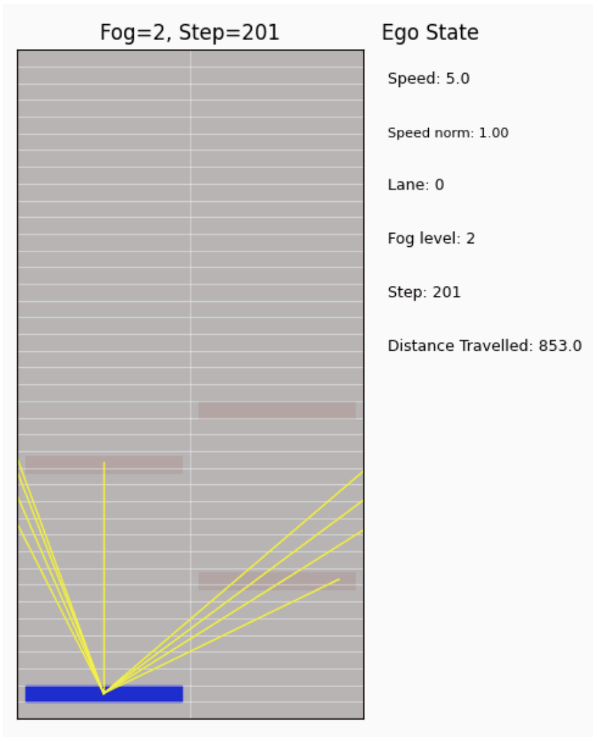
1. Base Reward : next ego speed
2. If collision occurs when taking action: -50
3. If maxsteps reached without collision : +100

γ (Discount factor) = 0.99

Renders

Blue : Reinforcement learning agent
Red : Other cars
Grey overlay : Fog
Yellow : Lidar ray casts





Training Setup

To evaluate the performance of reinforcement learning algorithms on the FoggyDriving environment, we trained three widely used model-free RL methods from Stable Baselines 3

1. **Proximal Policy Optimization (PPO):**

   PPO is an on-policy actor–critic method that stabilizes policy learning through clipped policy gradient updates that restrict the updates to stay within a region.

   The objective function :

   $$L^{PPO}(\theta) = E[min(r(\theta)\hat{A}, clip(r(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A})]$$

   Where $r(\theta) = \pi_{\theta}(a|s)/\pi_{\theta old}(a|s)$ is the importance sampling ratio that is clipped using the clipping parameter ϵ and $\hat{A}$ is the advantage estimate.

Hyperparameter	Value	Effect on Learning
Learning rate	3×10^{-4}	lower values increase stability but slow learning
Clip range (ε)	0.2	Limits policy shifts per update; prevents instability
Rollout length	2048	Longer rollouts reduce gradient variance and improve advantage estimation
Batch size	64	Larger batches yield smoother gradients and better convergence
Discount factor (γ)	0.99	Higher γ values encourage long-term returns
Entropy coefficient	Default SB3 (0)	Encourages exploration when > 0
GAE (λ)	Default SB3 (0.95)	Controls bias–variance tradeoff in advantage estimation
Policy type	MLP	Standard dense network for vector observations

2. **Advantage Actor-Critic (A2C)**

A2C is a synchronous actor–critic algorithm where parallel workers generate trajectories, and gradients are averaged to reduce variance. A2C uses the advantage function to update the policy without clipping.

Policy gradient update: $\nabla\theta J = E[\nabla_\theta log \pi_\theta(a|s)\, A(s, a)]$

Value function update: $V_\phi(s) \approx E[R_t]$

Where $\pi\theta$ is the policy, $A(s, a)$ is the advantage function,
$V_\phi(s)$ is the learned value baseline,
R_t is the discounted return.

Hyperparameter	Value	Effect on Learning
Learning rate	7×10^{-4}	Lower values increase stability but slow learning
n-step returns	128	Better credit assignment
Discount factor (γ)	0.99	Higher γ values encourage long-term returns
gae_lambda	0.95	Controls bias variance tradeoff
Entropy coefficient	0.001	Helps exploration
Number of parallel workers	SB3 default (1)	More workers reduce variance and speed up training
Policy type	MLP	Standard dense network for vector observations

3. Deep Q-Network (DQN)

DQN is an off-policy value-based reinforcement learning algorithm. It learns a Q-function using temporal-difference updates. DQN assumes the environment is fully observable.

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma max_{a'} Q_{target}(s', a') - Q(s,a))$$

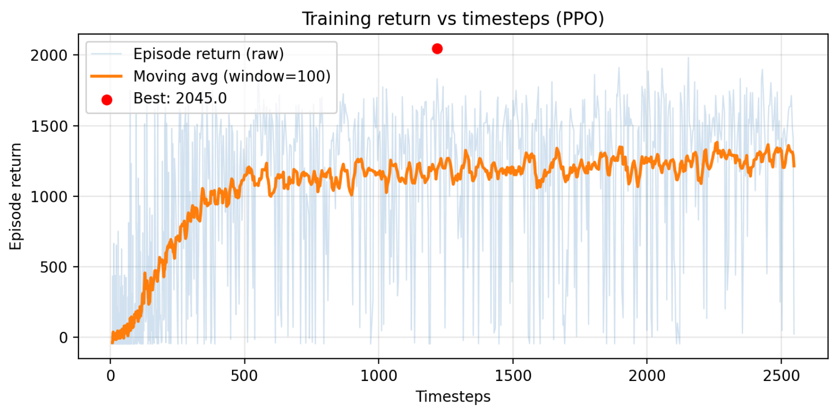
Where α is the learning rate,
r is the reward,
γ is the discount factor,
Q_{target} is the target network,

Hyperparameter	Value	Effect on Learning
Learning rate	1×10^{-3}	Lower values increase stability but slow learning
Replay buffer size	100,000	Larger buffers decorrelate samples; improve stability
Batch size	32	Smaller batches lead to noisier Q-value gradients
Target network update	Every 500 steps	Stabilizes Q-learning
Discount factor (γ)	0.99	Higher γ values encourage long-term returns
ε-greedy exploration	Enabled (1.0 to 0.1)	Random exploration prevents premature convergence
Policy type	MLP Q-network	Standard dense network for vector observations

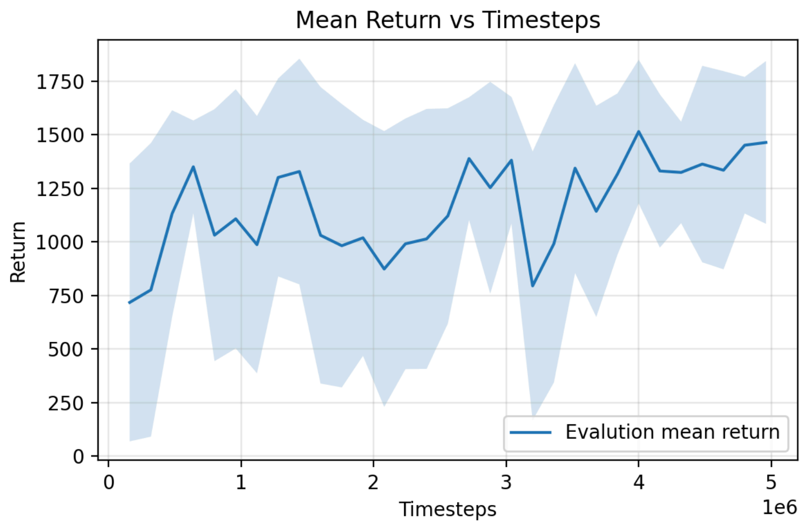
Results

1. Proximal Policy Optimization (PPO)

Training Curve
The PPO training curve shows a rapid increase in return during 0–500k steps, followed by a plateau around 1100k–1300k as the agent refines its driving policy.



Evaluation curve
The evaluation curve exhibits consistently high mean returns of 1000–1500) with a broad variance band caused possibly by random fog levels and traffic configurations.



Final Evaluation
The agent survives most of the episode horizon, rarely collides, and successfully learns behaviors such as lane changes, speed control, and safe following under fog-impaired visibility.

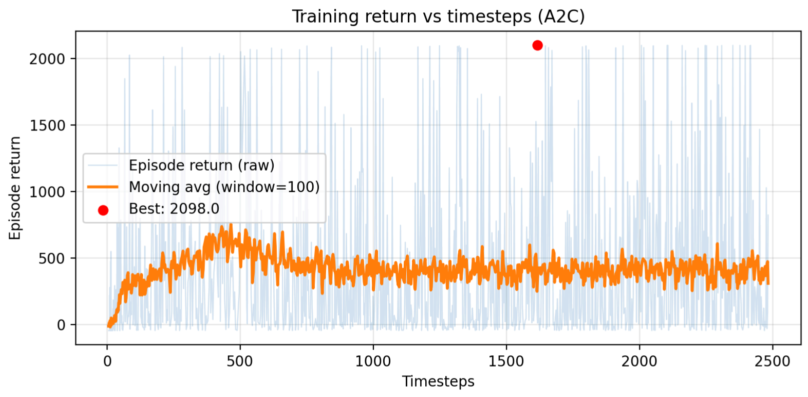
```
Evaluation : 50 episodes:
  Avg return     : 1218.88
  Avg ep length  : 328.0 steps
  Avg Collision  : 0.26
```

Some notable behaviours

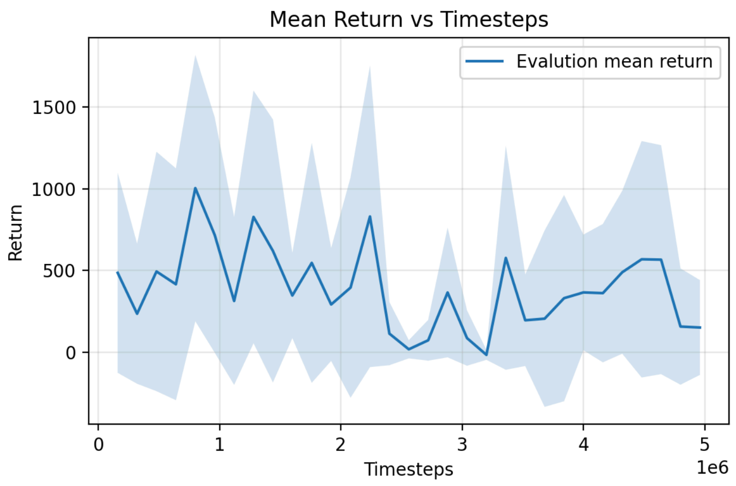
- Under low visibility, it slows down considerably, maintaining larger safety margins and avoiding unnecessary lane changes or overtakes.
- When visibility improves, the agent accelerates and confidently closes the gap to vehicles ahead.
- If both lanes are congested, the agent settles into a steady following behavior, matching the speed of the leading car.
- When a safe opportunity exists, it performs overtaking by switching lanes.
- Across conditions, the agent consistently maintains safe distances.

2. Advantage Actor-Critic (A2C)

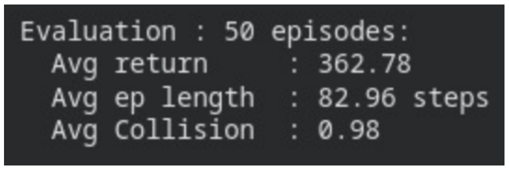
Training curve
A2C shows a strong initial improvement during the first ~300k timesteps), but quickly plateaus. Trraining returns remain highly noisy as A2C's updates are seem more sensitive to variance.



Evaluation curve
Evaluation performance remains inconsistent, fluctuating widely between low and moderately high returns. The mean evaluation return rarely exceeds 500 and shows no clear upward trend.



Final Evaluation
A2C agent fails to maintain safe distances, survives for only short durations, and crashes in nearly every episode. The policy lacks stability and does not meaningfully adapt to fog-induced partial observability.



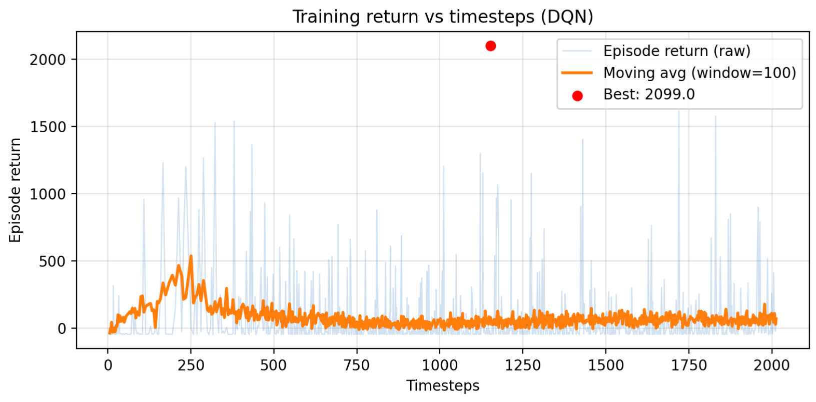
```
Evaluation : 50 episodes:
  Avg return     : 362.78
  Avg ep length  : 82.96 steps
  Avg Collision  : 0.98
```

Some takeaways

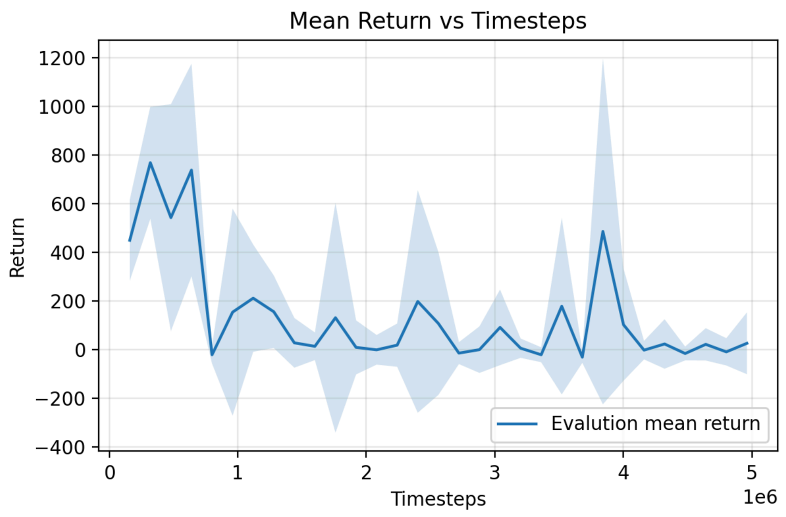
- A2C frequently alternates between speeding, braking, and lane switching without a stable pattern
- A2C fails to maintain safe headway; it approaches cars too quickly and has trouble matching the speed of the vehicle ahead.
- Lowering the learning rate, increasing the n-step horizon, and using more parallel environments would stabilize learning.
- Because fog limits visibility, incorporating memory through an LSTM policy might help A2C infer motion and react more safely.
- Better reward shapings, such as penalties for near-collisions or unnecessary lane changes, could further guide the agent toward safer policies.

3. Deep Q-Network (DQN)

Training curve
After a brief initial rise, the moving average quickly collapses and remains near zero for most of training. The Q-network struggles to approximate value targets in this noisy, partially observable environment.



Evaluation curve
Evaluation performance is inconsistent and generally poor. Returns vary dramatically between episodes, often dipping into negative values.



Final Evaluation
DQN crashes in almost every episode and fails to learn meaningful survival or safe-driving behavior.

```
Evaluation : 50 episodes:
  Avg return     : 261.64
  Avg ep length  : 62.32 steps
  Avg Collision  : 0.98
```

Some takeaways

- Behaviors are erratic and inconsistent across episodes.
- The agent accelerates blindly, often crashing almost immediately.
- Very little strategic lane-changing or following behavior emerges.
- DQN fails to incorporate fog cues or lidar distance effectively.
- Driving policy collapses early during training and never recovers.
- If the environment were fully observable, DQN would likely perform much better since Q-learning could correctly estimate future returns without needing memory or inference about hidden variables.

4. PPO vs A2C vs DQN

Among the 3 algorithms tested, PPO was the most reliable in the FoggyDriving environment. It learned stable, safe driving behaviors even under fog and noisy observations. A2C showed some improvement but remained inconsistent and prone to crashing, while DQN struggled almost entirely due to the limited visibility and high uncertainty of the task. Overall, PPO proved best suited for this type of partially observable, dynamic driving scenario.

Metric	PPO	A2C	DQN
Performance	High	Low	Low
Stability	Stable	Unstable	Very unstable
Collision Rate	Low (0.26)	Very High (0.98)	Very high (0.98)
Avg Return	1218	363	262
Learns Safe Driving?	Yes	Partially	No
Handles Fog / Noise?	Strong	Moderate	Poor
Strength	Robust policy learning	Fast updates	Simple architecture
Weakness	Slower per update	High variance	Fails with partial observability

Takeaways

The FoggyDriving environment proved to be a challenging task for studying autonomous driving under fog and limited perception. The combination of fog-dependent lidar, stochastic noise, and IDM/MOBIL-based traffic produced a partially observable setting where the agent needed to balance speed with safety.

Patterns and behaviours

Reward shaping played a decisive role in how the agent learned to behave. When the reward was based solely on speed (it was more of an indicator of distance travelled per timestep), the agent quickly discovered a loophole: it was more profitable to accelerate aggressively, maintain a high speed, and crash early than to drive cautiously and survive the full episode. A full 400-step episode at slow speeds (1–2 units) yields a cumulative reward of roughly 800, whereas driving at higher speeds (3–5 units) for only 250 steps before crashing produces a greater return. As a result, the agent consistently chose reckless acceleration over safe driving.

On the flip side, adding too many intermediate rewards for actions like safe overtaking, maintaining headway, or staying in a lane also created problems. Instead of learning genuine driving strategies, the agent began chasing reward "checkpoints" and fell into unnatural, instruction-following behavior. One example occurred when we rewarded successful overtakes where the agent learned to repeatedly go back and forth across a slower car, performing the same overtaking motion over and over again to accumulate extra rewards.

Traffic generation also influenced learning heavily. When we first began experimenting with the traffic model, it became clear that spawning cars only at the top of the grid made interactions extremely sparse. As a result, the ego vehicle did not learn meaningful behaviors such as following, braking, or overtaking. To compensate, we briefly tried an unrealistic setup where cars were spawned very close to the ego at high frequency. Surprisingly, the agent learned much faster in this configuration and achieved long survival times (often averaging ~394 steps), simply because it consistently had traffic to react to.

Poorly tuned spawning rules allowed agents to exploit periodic gaps or memorize traffic patterns, rather than learning genuine adaptive behavior. For example, the agent would sometimes slow down until traffic naturally drifted out of the environment bounds, then accelerate at full speed through an empty road. In other runs, the agent discovered that avoiding lane changes altogether reduced its chances of crashing, resulting in overly cautious, stationary-like behavior.

Limitations

- Two-lane grid simplification: Real roads include curvature, multiple lanes, merges, ramps, and off-ramps, which are not represented.
- No continuous dynamics: The environment uses discrete lane shifts and speed changes rather than realistic longitudinal and lateral control.
- Forward-only lidar: Full 360° sensing, blind-spot detection, and rear perception are not modeled.
- Simplified fog and physics: Fog affects only visibility, not braking distance, tire friction, or sensor dropout.

Conclusion

To conclude, this project demonstrated that reinforcement learning driving agents can learn meaningful driving behaviors even under fog, noise, and limited perception when supported by realistic traffic dynamics and balanced reward design. PPO proved most effective, while A2C and DQN revealed the challenges of high-variance learning and partial observability.