DeepLane: Parallel DQN for Autonomous Lane Keeping

Jose Tomas, Gabriel Gonzalez, Nawaljit Dhaliwal
Cal Poly Pomona
Pomona, California
jatomas@cpp.edu, gigonzalez@cpp.edu, nsdhaliwal@cpp.edu

Abstract

Lane keeping should be done with precise lateral control and must handle noise and changes in road conditions. Traditional rule-based controllers are typically insufficient when lane shapes change or sensor readings become unreliable, which motivates the use of reinforcement learning for making flexible decisions. In this paper, we present DeepLane, a lane-keeping framework that uses a parallel Deep Q-Network to accelerate learning and improve policy stability. Different from the standard DQN setup, which relies on a single simulation, DeepLane draws experience from eight parallel environments, where the agent is exposed to a wider range of lane deviations, heading errors, and curvature patterns. The system takes in a five-dimensional state vector that encodes lateral offset, heading error, vehicle position, and curvature previews, and outputs discrete steering actions that are smoothed by a simple safety layer to ensure realistic control. Training incorporates experience replay, a target network, and observation normalization over 400,000 environment steps. A well-designed reward function promotes centered driving, smooth steering, and consistent alignment with the lane centerline. A DeepLane agent trained in this fashion exhibits low centerline error, strong lane retention, and stable steering on new randomized lane geometries, even in the presence of noise and curvature changes, clearly outperforming the single-environment DQN training methods.

1. Introduction

Lane keeping is a principal requirement for autonomous vehicles that requires precise steering control in the presence of noise, changing curvature, and imperfect sensing. Traditional rule-based controllers can work well on predictable roads but often fail when lane markings fade, geometry varies, or disturbances push the vehicle off center. These limitations have motivated the use of reinforcement learning as a more general alternative that can adapt to a wide range of driving conditions.

However, many of these RL methods rely on a single simulation environment during the training phase, which narrows their experiences to a small set and converges slowly. In this regard, we propose DeepLane, a lane-keeping framework based on a Parallel Deep Q-Network.By collecting experience from eight simultaneously running simulated environments, the agent exposes itself to more lateral errors, heading deviations, and curvature patterns, which speeds up and stabilizes learning. DeepLane adopts a compact five-dimensional state representation with a safety-filtered discrete steering action space. Training includes experience replay, target network updates, and observation normalization, utilizing 400,000 environment steps. Experimental results on unseen, randomized lane geometries have demonstrated that DeepLane achieves low lateral error while keeping the vehicle in the lane boundaries and producing smooth steering; all these are considered as a benefit of using parallelized RL in an autonomous lane-keeping task.

1.1. Loss Function

DeepLane uses the standard Deep Q-Network (DQN) temporal-difference (TD) loss to measure how far the network's predicted Q-values deviate from a stable target. The loss is defined as

$$L(\theta) = \mathbb{E}\left[(y_{\text{target}} - Q_\theta(s_t, a_t))^2\right], \tag{1}$$

where the TD target is computed as

$$y_{\text{target}} = r_t + \gamma \max_{a'} Q_{\theta^-}(s_{t+1}, a'). \tag{2}$$

Here, Q_θ is the online network, Q_{θ^-} is the target network (updated every 8000 steps), and $\gamma = 0.99$ is the discount factor. The TD loss encourages the online network to match its predictions to a slowly changing target, which stabilizes learning.

A loss function measures how "wrong" the model's predictions are. In reinforcement learning, the TD loss signals the agent to increase the predicted value of good steering actions and decrease the predicted value of poor ones. In

DeepLane, this loss plays a central role in shaping steering behavior. Because the reward function penalizes lateral drift, heading error, and aggressive steering, poor lane-keeping results in large losses. Conversely, smooth and centered driving produces small TD errors.

DeepLane's use of a 300,000-transition replay buffer, eight parallel environments, and a target network significantly reduces the variance of the TD loss and speeds convergence. As training progresses and the TD loss steadily decreases, the policy learns to maintain low lateral error, avoid boundary violations, and produce smooth steering adjustments.

1.2. Reward Function

To encourage stable, centered, and smooth vehicle motion, the agent is trained with a shaped reward defined at each time step t as

$$r_t = 1 - \lambda_y y_t^2 - \lambda_\psi \psi_t^2 - \lambda_{\dot{y}} \dot{y}_t^2 - \lambda_\delta \delta_t^2 - \lambda_{\Delta\delta}(\delta_t - \delta_{t-1})^2 - \text{edge\_pen}(y_t) + \text{progress\_bonus}(x_t, \psi_t), \tag{3}$$

where y_t is the lateral displacement from the lane center, ψ_t is the heading error, $\dot{y}_t = v \sin(\psi_t)$ is the lateral velocity, and δ_t, δ_{t-1} denote the current and previous steering angles.

The weighting coefficients are

$$\lambda_y = 0.8, \quad \lambda_\psi = 0.25, \quad \lambda_{\dot{y}} = 0.08, \quad \lambda_\delta = 0.002, \quad \lambda_{\Delta\delta} = 0.05.$$

A soft boundary penalty edge_pen(y_t) is applied as the vehicle approaches the lane edges. This term is zero near the centerline and increases smoothly within the outer $0.3\,\text{m}$ of the lane, reaching a maximum penalty of 0.5 as $|y_t|$ approaches $1.8\,\text{m}$.

Additionally, a forward progress bonus progress_bonus(x_t, ψ_t) provides a small positive reward proportional to $v\cos(\psi_t)$, encouraging forward motion aligned with the lane heading.

This reward structure penalizes lateral and heading deviations, discourages abrupt steering inputs, and promotes smooth, centered lane keeping.

1.3. Optimization

To train the DeepLane controller, we rely on the Adam optimizer to reduce the standard DQN TD-error. In practice, the network updates are carried out using batches of 256 samples drawn from a replay buffer of 300,000 transitions, which are collected from all eight parallel environments. We use a discount factor of $\gamma = 0.99$ and apply a linearly decaying learning rate that starts at 3×10^{-4} and gradually approaches zero as training finishes. This schedule helps the network take larger steps early on and smaller, more stable steps later. A separate target network is updated every 8,000 steps using a soft-update rate of $\tau = 0.005$, which keeps learning stable by preventing the targets from changing too quickly. Altogether, these settings allow the policy to improve steadily while avoiding the instability that can occur in deep reinforcement learning.

2. Train setup

2.1. Training Setup and Hyperparameters

DeepLane is trained using a Parallel DQN configuration that runs eight independent environments concurrently. Each `SubprocVecEnv` instance executes its own copy of the custom `LaneKeepingEnv`. The vehicle moves at a constant speed of $15\,\text{m/s}$, with a simulation time step of $\Delta t = 0.05\,\text{s}$. A wheelbase of $2.7\,\text{m}$ is used, matching the dynamics defined in the environment model. To improve training stability, we apply `VecNormalize` to normalize observations while keeping the original rewards unchanged.

DQN Hyperparameters.

- **Network architecture:** MLP with hidden layers $[256, 256]$ and ReLU activations
- **Discount factor:** $\gamma = 0.99$
- **Replay buffer size:** 300,000 transitions (shared across all environments)
- **Batch size:** 256
- **Learning rate:** linearly decayed from 3×10^{-4} to 0 during training
- **Target network update interval:** every 8,000 environment steps
- **Exploration schedule:** ε decays from 1.0 to 0.02 over the first 35% of training
- **Total training steps:** 400,000 environment steps

Sampling trajectories from eight parallel environments significantly increases the diversity of experiences added to the replay buffer. This parallelized design, combined with a large replay memory, leads to faster learning and more stable convergence than a single-environment DQN.

3. Experiment

3.1. Evaluation and Generalization Performance

This work investigates whether a Parallel DQN architecture can learn a stable and reliable lane-keeping policy in a noisy and dynamically changing road environment. Our objective is to determine whether parallelized training improves learning speed, policy smoothness, and generalization when compared to a standard single-environment

DQN. Specifically, we analyze the agent's ability to maintain lane position, track the centerline, and produce smooth steering behavior. To assess robustness, we evaluate the trained policy on lane geometries that were not observed during training. Overall, the goal is to demonstrate that *DeepLane* can provide safe and efficient autonomous lane keeping suitable for real-world deployment.

For quantitative evaluation, we test the learned policy on a held-out set of randomly generated lane geometries and report two simple metrics:

- **Centerline RMSE**: the root-mean-square lateral deviation, $\sqrt{\mathbb{E}[y_t^2]}$.
- **Lane retention**: the percentage of time steps in which $|y_t| \leq 1.8\,\text{m}$.

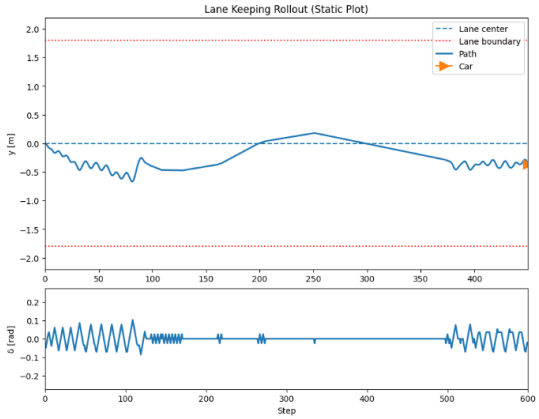

Figure 1. Lane Keeping Rollout (Static Plot)

Even with sensor noise and randomized curvature characteristics, the Parallel DQN often achieves low lateral RMSE and remains within lane boundaries for the full 30-second evaluation episodes. Figure **??** shows an example rollout from the evaluation set.

3.2. Trajectory and Steering Behavior

Trajectory Plot (Top). The top panel illustrates the vehicle's lateral position y_t over the course of the episode. The dashed horizontal line marks the lane center at $y = 0$, and the red dotted lines indicate the lane boundaries at $y = \pm 1.8\,\text{m}$. The blue curve shows the vehicle's lateral trajectory, while the orange marker represents the final position of the agent.

Interpretation. The vehicle begins with a small initial offset but rapidly returns toward the lane center, demonstrating that the learned controller effectively corrects heading and lateral deviations. Throughout the full 600-step rollout, the agent remains within the lane boundaries. Minor oscillations appear as the controller makes fine steering adjustments, yet the overall trajectory is stable. This behavior indicates that the Parallel DQN policy achieves reliable lane keeping even under sensor noise and variations in lane curvature.

Steering Command Plot (Bottom). The bottom panel displays the steering angle δ_t. Although the action space contains 15 discrete steering commands, a safety layer enforces a maximum rate of change of $0.5\,\text{rad/s}$, resulting in smoother, more realistic steering profiles.

Interpretation. During the initial correction phase, steering adjustments are larger as the agent compensates for its early lateral deviation. Once the vehicle stabilizes near the lane center, the policy maintains position using small, well-regulated oscillations. Near the end of the rollout, sensor noise and varying road curvature introduce slightly stronger oscillations, but the rate limiter ensures that steering remains smooth and physically plausible. These steering patterns confirm that the combination of the reward design and the safety layer encourages not only accurate lane keeping but also smooth control behavior.

3.3. Training Performance summary

The table highlights important training measurements gathered during the Parallel DQN learning process. These numbers show how the agent's actions, results, and learning changed during its training. The measurements are divided into three main groups: how well the rollout is doing, information about the timing, and details about training results.

Table 1. Training Performance Summary for DeepLane

Mean episode length	56.5
Mean episode reward	10.8
Exploration rate	0.636
Total timesteps	52,056
Episodes	1316
TD loss	0.00392
Learning rate	2.61e-4

Rollout Metrics: The rollout statistics show that the agent is beginning to learn stable lane-keeping behavior. The mean episode length and reward indicate improving performance, while the exploration rate shows the agent is still in an early-to-mid learning phase.

Time Metrics: The time metrics highlight efficient training throughput. The parallel environments allow rapid collection of episodes and timesteps, completing over 52k steps in only 41 seconds of training.

Training Diagnostics: The diagnostics indicate stable learning progress. The low TD loss and decayed learning

rate show that the Q-network is converging and producing more accurate value estimates as training continues.

4. Relation to Class

4.1. Deep Learning for Lane Understanding

Lane keeping relies on understanding lane boundaries, road edges, and changes in curvature directly from visual input. Deep Learning supports this through convolutional neural networks, which extract meaningful features from camera frames. The convolution operation applies a learned filter across the image to detect patterns such as lane edges and painted lines. This process is represented by

$$S(i,j) = \sum_m \sum_n X(i+m, j+n)\, K(m,n)$$

where X is the input image and K is a learned kernel. The output highlights features that the lane keeping system can interpret. After convolution, nonlinear activation functions such as ReLU are applied to remove insignificant values and enhance structural information important for steering. The ReLU operation

$$a = \max(0, z)$$

helps the model emphasize meaningful lane features even when lighting or contrast varies.

4.2. Nonlinear Mapping from Road Geometry to Steering Control

Lane keeping ultimately involves predicting an appropriate steering command based on the current road geometry. This relationship between state variables and the steering action is nonlinear and cannot be modeled with simple equations. Deep neural networks learn this mapping from data, expressed as

$$a = f_\theta(s)$$

where s represents the road state, including curvature, lateral offset, and heading error. Each neural network layer further transforms the representation using

$$h^{(l)} = \sigma\left(W^{(l)} h^{(l-1)} + b^{(l)}\right)$$

which allows the model to combine multiple road cues into a coherent steering output. Through training, the network learns how different road geometries correspond to corrective steering actions needed to keep the vehicle centered.

4.3. Generalization Across Real World Road Conditions

Lane keeping systems must operate reliably across a wide range of driving environments, including varying lighting, worn lane markings, shadows, and road surface inconsistencies. Deep Learning improves generalization by regularizing the model to avoid overfitting to conditions seen during training. This is represented by the loss function

$$\mathcal{L} = \mathcal{L}_{task} + \lambda \|W\|^2$$

which discourages overly complex models and makes the network more adaptable to new roads. In addition, sensor noise and imperfections are unavoidable in real-world driving, so robustness is improved by considering noise in the state estimation process. A noisy observation can be modeled as

$$\tilde{s} = s + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

which prepares the network to handle uncertainty in input measurements.

4.4. Continuous and Smooth Steering Behavior

A critical part of lane keeping is generating smooth steering commands rather than abrupt or jerky corrections. Since steering is a continuous value, the output of the neural network is treated as a real-valued action

$$a_t \in \mathbb{R}$$

which allows fine-grained control of the vehicle. To encourage stable driving behavior, smoothness can also be included in the training objective. A commonly used penalty term

$$\mathcal{L}_{smooth} = (a_t - a_{t-1})^2$$

discourages large jumps between consecutive steering outputs. This leads to smoother trajectories and more human-like lane following, especially on curved roads.

4.5. Training Stability and Learning from Experience

Training a lane keeping model requires repeated interaction with driving data, where the network gradually refines how it interprets lane geometry and selects steering actions. Parameter updates are performed using gradient descent

$$\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}$$

which decreases the model's error over time. To improve stability, many systems use experience replay, where past driving states and actions are stored and sampled during training:

$$(s_t, a_t, r_t, s_{t+1}) \sim \mathcal{D}.$$

This exposes the model to more diverse driving situations and prevents forgetting. The learning signal itself is strengthened through temporal difference learning, represented by

$$\delta = r_t + \gamma V(s_{t+1}) - V(s_t)$$

which measures how much better or worse the new state is compared to expectations. This allows the model to adjust its steering strategy with each training cycle.

5. Conclusion

DeepLane demonstrates that using a Parallel DQN setup with eight simultaneous environments can significantly improve training speed, policy stability, and generalization for autonomous lane keeping. The agent learns to maintain low lateral error, stay within lane boundaries, and produce smooth steering behavior even under noise and varying curvature. The combination of a shaped reward, a safety-filtered action space, and observation normalization further contributes to stable learning. Overall, the results show that parallelized reinforcement learning is a practical and effective approach for robust lane-keeping control systems.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] A. Raffin, et al., "Stable-Baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, pp. 1–10, 2021.

[3] Farama Foundation, "Gymnasium Documentation," 2023. Available: https://gymnasium.farama.org/

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.