Designing controllers for the MegaVanderTest using reinforcement learning

---

Nathan Lichtlé (with Kathy Jang, Eugene Vinitsky and Adit Shah)

June 21, 2023

Traffic and Autonomy Conference, Maiori, Italy

The MegaVanderTest: problem statement

Given that we

- Deploy 100 cars on the highway
- Have a lot of highway trajectory data
- Need to adhere to physical sensing and actuation constraints

How to develop a traffic-smoothing energy-reducing controller?

Reinforcement learning (RL) to the rescue

Pros

- Can handle a lot of data and thrives in simulated environments
- Can be given many inputs
- Neural networks have a large expressive power
- Can optimize for a desired metric over time

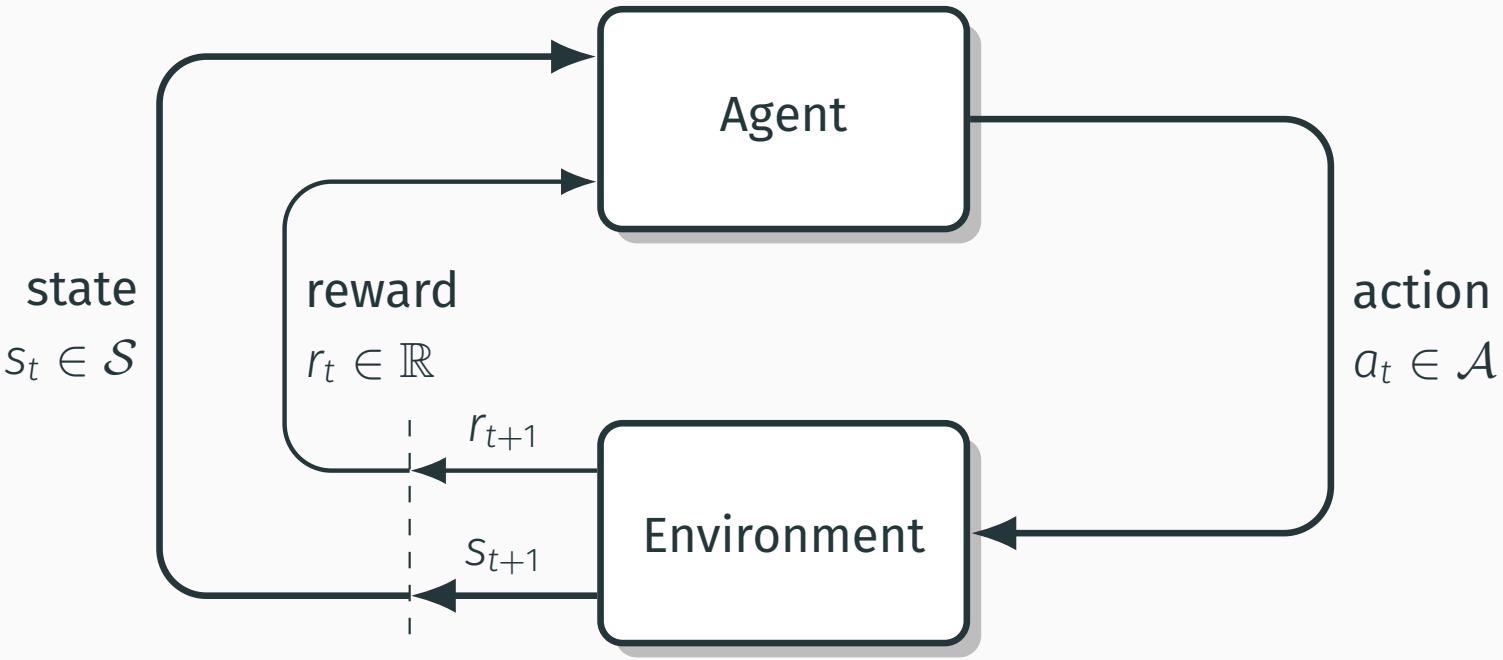
Cons

- Designing a good objective function is hard
- No convergence, generalization or safety guarantees
- Harder interpretability

MDP: Markov Decision Process

- An MDP is a model for sequential decision making under uncertainty.
- Components of an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R, \gamma, \mu)$:
  - State space $\mathcal{S}$: Set of all possible states.
  - Action space $\mathcal{A}$: Set of all possible actions.
  - Transition model $T(s'|s, a)$: Probability of transitioning to state s' given action a in state s.
  - Reward function $R(s, a) \in \mathbb{R}$: Immediate reward received after taking action a in state s.
  - Discount factor γ: Determines the importance of future rewards.
  - Initial state distribution $\mu(s_0)$: Probability distribution over initial states.

The RL loop


Agent
state
$s_t \in \mathcal{S}$
reward
$r_t \in \mathbb{R}$
action
$a_t \in \mathcal{A}$
r_{t+1}
s_{t+1}
Environment

Objective: Maximizing Sum of Discounted Rewards

- The policy $\pi(a \mid s)$ is a mapping from states to actions.
- The objective in reinforcement learning is to maximize the sum of discounted rewards.
- Sum of discounted rewards (return) over a trajectory τ:

$$G(\tau) = \sum_{t=0}^{T} \gamma^t r_t$$

  where r_t is the reward at time step t, $\gamma \in [0, 1)$ is the discount factor and T is the horizon (we could have $T = \infty$).

- We want to find a policy π that maximizes the expected return:

$$\pi^* = \arg\max_{\pi} \mathbb{E}_{\pi} \left[\sum_{t=0}^{T} \gamma^t r_t \right]$$

Basic Policy Gradient Algorithm

- The basic policy gradient algorithm uses gradient ascent to update the policy parameters.
- Update rule for policy parameters:

$$\theta \leftarrow \theta + \alpha \nabla J(\theta)$$

where α is the learning rate and $J(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{T} \gamma^t r_t\right]$.

- The policy gradient theorem:

$$\nabla J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) G(\tau)\right]$$

where $J(\theta)$ is the objective function, $\nabla J(\theta)$ is the gradient with respect to the policy parameters θ, and $G(\tau) = \sum_{t=0}^{T} \gamma^t r_t$ is the return over a horizon T.

Trust Region Methods

- Trust region methods aim to limit policy updates to a trust region for stability.
- The objective is to maximize the surrogate objective within the trust region:

$$L(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^\pi(s,a)$$

  where $A^\pi(s,a)$ is the advantage function.
- The trust region constraint:

$$\text{KL}\left[\pi_{\theta_{\text{old}}}(a|s) \,||\, \pi_\theta(a|s)\right] \leq \delta$$

  where $\delta > 0$ is a small threshold.
- The objective of trust region methods is to find the policy update that solves the following constrained optimization problem:

$$\max_\theta L(\theta) \quad \text{subject to} \quad \text{KL}\left[\pi_{\theta_{\text{old}}}(a|s) \,||\, \pi_\theta(a|s)\right] \leq \delta$$

PPO (Proximal Policy Optimization)

- PPO is an example of a trust region method.
- The PPO algorithm:
  1. Collect samples in the environment using the current policy.
  2. Compute advantages $A^{\pi_\theta}(s, a)$ using a value function estimate.
  3. Update the policy with gradient ascent using the clipped surrogate objective to ensure a conservative policy update within the trust region:

     $$L(\theta) = \mathbb{E}_{\pi_\theta}\left[\min\left(q_t(\theta)A^{\pi_\theta}(s,a), \text{clip}\left(q_t(\theta), 1-\epsilon, 1+\epsilon\right)A^{\pi_\theta}(s,a)\right)\right]$$

     where $q_t(\theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}$ is the importance sampling ratio and $\epsilon > 0$ controls the size of the trust region.
  4. Update the value function estimate with gradient descent on the mean square error between estimates and measured values.
  5. Repeat the process until convergence.

Dataset of highway driving trajectories

- Team collected 10 hours / 800km of trajectory data on I-24
- Wide range of driving conditions
- We cleaned up and smoothed the data

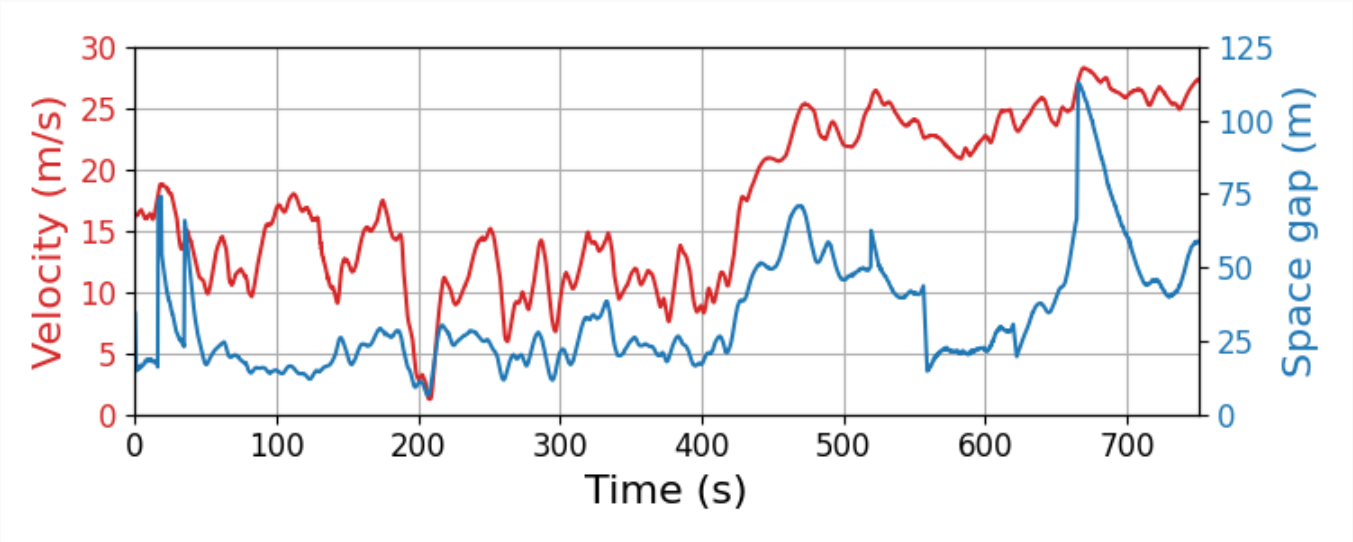


Figure 1: Example trajectory from the dataset

Building a simulation

- We replay trajectories in a one-lane simulation
- AVs are controlled using RL
- Humans are modeled using the Intelligent Driver Model
- We train single-agent and evaluate with many AVs

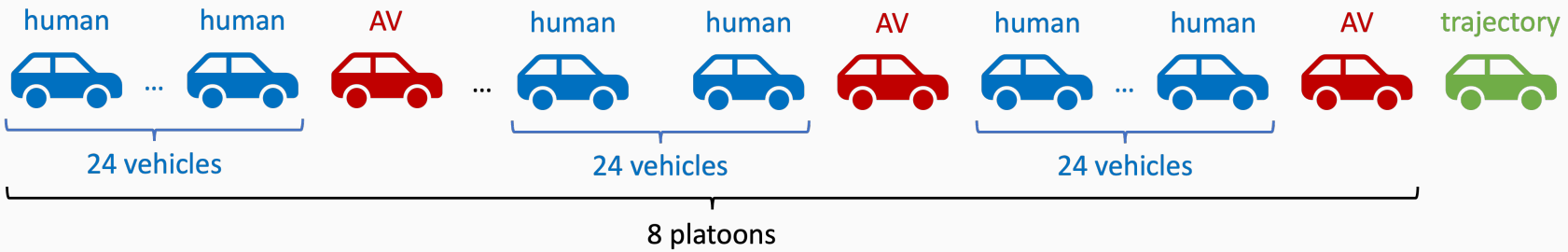


Figure 2: Example evaluation platoon in simulation

Controller design: observations

We consider the following observations:

- Ego speed v_t^{av}
- Leader speed v_t^{lead}
- Space gap $h_t = x_t^{\text{lead}} - x_t^{\text{av}} - \ell^{\text{lead}}$ where x is the vehicle front bumper position and ℓ is vehicle length
- Target speed v_t^{SP} from a speed planner
  - Downstream average speed information is collected over 500m to 1500m segments
  - A highway speed profile estimate is interpolated from the collected points
  - A speed planner guide is created by smoothing the speed profile

Controller design: actions

The AV instantaneous acceleration a_t is computed as follows:

- v1: acceleration-based controller

$$a_t = \pi_\theta(v_t^{\text{av}}, v_t^{\text{lead}}, h_t)$$

- v2: ACC-based controller (adaptive cruise control)

$$\begin{bmatrix} v_t^{\text{ACC}} & h_t^{\text{ACC}} \end{bmatrix} = \pi_\theta(v_t^{\text{av}}, v_t^{\text{SP}}, v_{t-1}^{\text{ACC}}, h_{t-1}^{\text{ACC}})$$

$$a_t = F_{\text{ACC}}(v_t^{\text{ACC}}, h_t^{\text{ACC}})$$

where the ACC model F_{ACC} is controlled by a speed setting v_t^{ACC} and a gap setting h_t^{ACC}, and internally depends on v_t^{av}, v_t^{lead}, and h_t.

Continuous Policy: Gaussian Distribution

Neural Network Architecture: 4x64 fully-connected neural network

$$
\begin{aligned}
h_1 &= \tanh(W_1 s + b_1) \\
h_2 &= \tanh(W_2 h_1 + b_2) \\
h_3 &= \tanh(W_3 h_2 + b_3) \\
h_4 &= \tanh(W_4 h_3 + b_4) \\
\mu(s) &= W_\mu h_4 + b_\mu \\
\sigma(s) &= \exp(W_\sigma h_4 + b_\sigma)
\end{aligned}
$$

where $\theta = (W_1, W_2, W_3, W_4, W_\mu, W_\sigma)$ are matrices of parameters of appropriate dimensions.

Policy Representation:

$$\pi_\theta(a|s) = \mathcal{N}(\mu(s), \sigma(s))$$

where $\mathcal{N}(\mu, \sigma)$ represents a Gaussian distribution with mean μ and standard deviation σ.

Failsafes and gap-closing

- We wrap the controller output a_t with a failsafe/gap-closing term, to stabilize training and encourage reasonable behavior:

$$a_t^{\text{final}} = \begin{cases} a_{\min} & \text{if } \Delta_t^{\text{TTC}} \leq 6 \\ a_{\max} & \text{if } \Delta_t^{\text{TTC}} > 6 \text{ and } h_t \geq \max(120, 6v_t^{\text{av}}) \\ a_t & \text{otherwise} \end{cases}$$

  where

$$\Delta_t^{\text{TTC}} = \begin{cases} \dfrac{h_t}{v_t^{\text{diff}}} & \text{if } v_t^{\text{diff}} > 0 \\ +\infty & \text{otherwise} \end{cases} \quad \text{and} \quad v_t^{\text{diff}} = \left[v_t^{\text{av}} \left(1 + \frac{4}{30} \right) + 1 \right] - v_t^{\text{lead}}$$

  and $a_{\min}, a_{\max}$ are the minimum and maximum accelerations we allow, respectively.
- We penalize stepping over the boundaries so the controller learns to stay within.

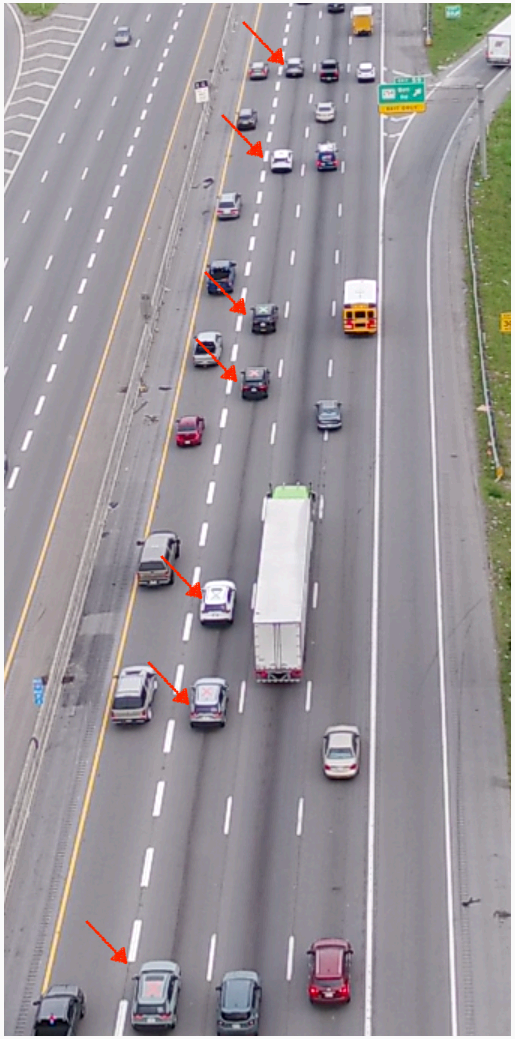
Reward function design

The reward r_t at time t is the weighted sum of several penalty terms:

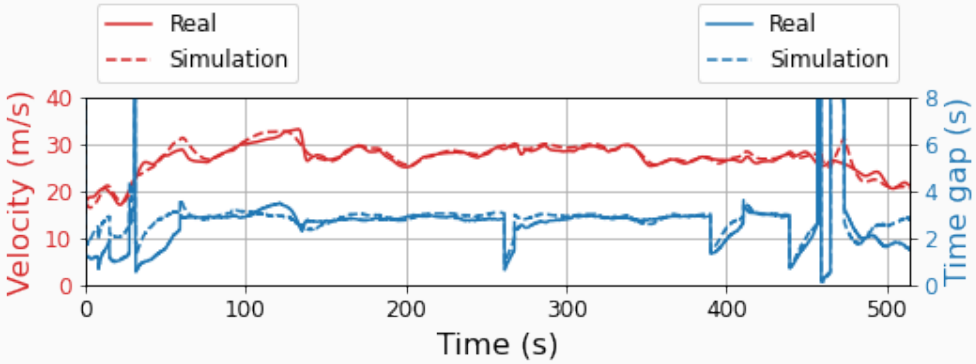
- Platoon energy consumption: $\frac{1}{n} \sum_{i=0}^{n} E_t^i$
  - E_t^i is the instantaneous energy consumption at time t of vehicle i (0 being the AV).
  - E_t^i depends on vehicle model, speed, acceleration and road grade.
  - n is the number of IDM-controlled vehicles following the AV.
- Acceleration amplitude: a_t^2
- Following the speed planner: $(v_t^{\text{av}} - v_t^{\text{SP}})^2$
- Staying within gap constraints: $\mathbb{1}\left[h_t \notin [h_t^{\min}, h_t^{\max}]\right]$
  - where $h_t^{\min}$ (resp. $h_t^{\max}$) is the minimum (resp. maximum) gap allowed by the failsafe (resp. gap-closing) function.
- Headway penalty: $\frac{h_t}{v_t^{\text{av}}} \mathbb{1}\left[h_t > 10 \wedge v_t^{\text{av}} > 1\right]$

where $\mathbb{1}[\mathcal{P}] = 1$ if $\mathcal{P}$ is true, 0 otherwise.

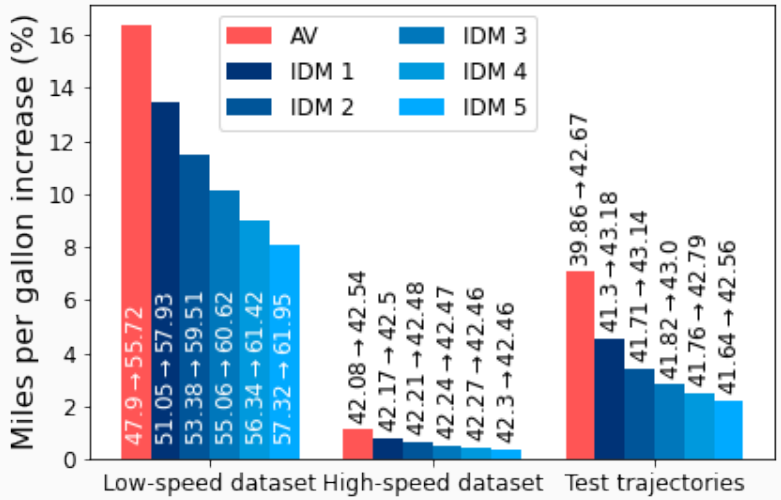
Last year: the VanderTest



(a) Deployment platoon



(b) Sim-to-real comparison



(c) Energy savings in simulation

AVs smoothing traffic waves

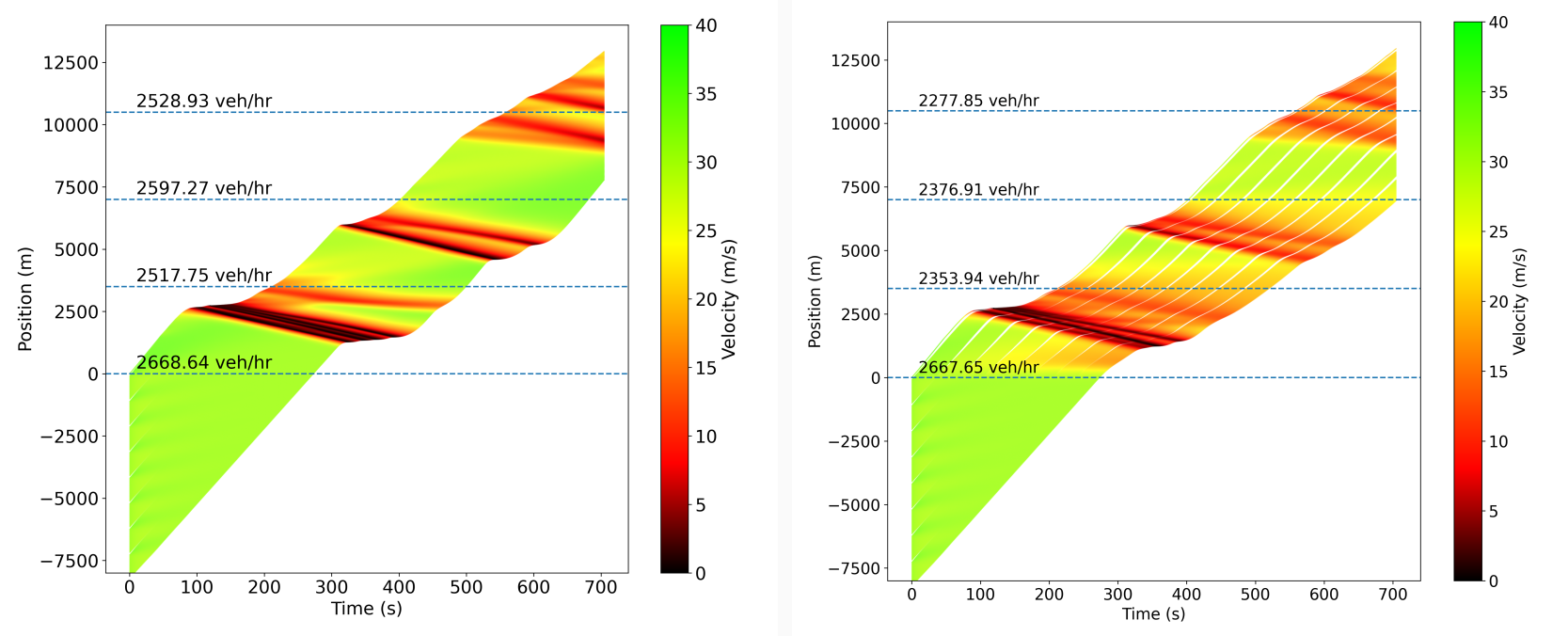


Figure 4: Time-space diagrams of baseline (left) and RL (right) at a 4% penetration rate on a simulation of 200 vehicles.

- Final control achieves around 23% energy savings in shockwaves
- Combination of low-speed and high-speed controllers

AV velocity profile

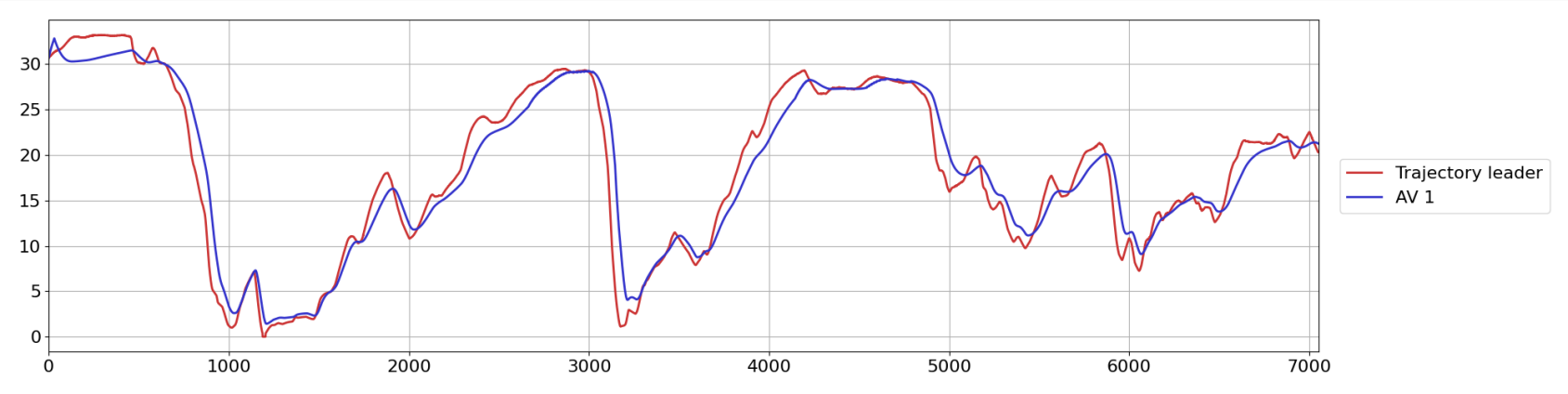


Figure 5: Velocity profile of an AV (blue) compared to its leader (red).

- We can observe at the scale of a single AV, how it attempts to cut through the steep accelerations and slowdowns

Deployment week

- A lot of last-minute debugging on the car directly, sitting in the parking lot or driving on the highway
- Making fixes and adjustments, reuploading controllers everyday
- In the end, RL controller safely and successfully deployed on 100 cars

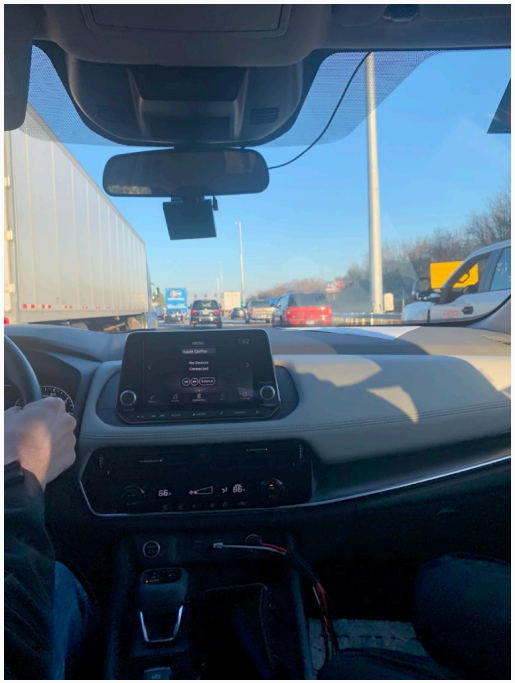


Figure 6: Driving with RL control