Benchmarking Policy-Gradient Methods for Deep Reinforcement Learning in High-Fidelity Autonomous Vehicles Simulators

Anonymous submission

Abstract

Deep learning has accelerated development in autonomous vehicles, leading to more advanced self-driving systems that can accomplish complex tasks, make real-time decisions, and enhance overall road safety. However, supervised deep learning approaches require ample expert-labeled data for training, which is both time-consuming and expensive to acquire. Deep reinforcement learning (DRL) provides an alternative approach to training an agent by having it learn from interactions from an environment. DRL benchmarks have shown success in low-dimensional spaces with simple goals. DRL can train high-achieving agents in well-defined environments such as Atari games, grid worlds, and 2D PyGames (Towers et al. 2023). Therefore, a high-fidelity benchmark suite for DRL algorithms is needed. This paper addresses this gap through a series of experiments using the VISTA and Racecar-Gym simulator. We measure state-of-the-art methods including policy gradients and integrate existing architectures with attention layers. Our target task is obstacle avoidance while following a car lane. Our work contributes a novel reference for researchers looking to asses and improve collision avoidance strategies in realistic and scalable environments.

Introduction

In recent years, DRL has gained increasing popularity as a robust (Kuutti et al. 2021) machine learning approach, especially for autonomous vehicles (AV). Autonomous driving is a high-dimension problem: there are myriad variables in AV that need to be taken into account, e.g. sensor data, road conditions, and vehicle control. This high-dimensional data and continuous action spaces, make DRL an attractive choice to train an agent.

However, deploying and training AVs in real-world scenarios can be costly and pose potential safety risks. Simulators mitigate these challenges by offering cost-effective and realistic platforms to train AVs via policy-gradients. Simulators like VISTA (Amini et al. 2019) process real-world driving data to synthesize high quality camera images, LiDAR point clouds, and GPS data. VISTA can also simulate various tasks like lane following, multi-agent maneuvering, or trajectory stabilization. Previous work has used VISTA as a tool to apply imitation-learning (Amini et al. 2022); this relies on a human reference. Through this study, we focus on collision avoidance between two cars using DRL methods.

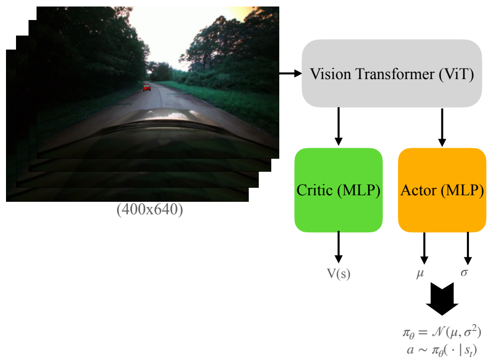
(400x640)
Vision Transformer (ViT)
Critic (MLP)
Actor (MLP)
V(s)
μ
σ
$\pi_\theta = \mathcal{N}(\mu, \sigma^2)$
$a \sim \pi_\theta(\cdot \mid s_t)$

Figure 1: Vision Transformer variant where in addition to an MLP returning classes there is an MLP returning a critic value.

In this paper, we present a benchmark of policy-gradient methods applied to car lane following tasks. We train an end-to-end RL algorithm from RGB[1] images into steering commands. Specifically, we asses three algorithms: REINFORCE (Williams 1992), A2C (Mnih et al. 2016), and PPO (Schulman et al. 2017). We also introduce a novel vision transformer architecture that is adapted for actor-critic functionalities (Dosovitskiy et al. 2017). We evaluate all of these approaches in a unified testing framework. In summary, our paper makes three novel contributions:

- A standardized experimental framework for evaluating DRL approaches in the autonomous vehicle domain.
- A benchmark suite to measure the performances among PPO, A2C, and REINFORCE.
- An adapted attention-based neural network architecture.

Our paper is organized in the following: Background provides an overview of relevant literature surrounding DRL, autonomous vehicles simulators, and neural network (NN) architectures. Experiment Framework describes our policy gradient framework. Actor-Critic Vision Transformer introduces the vision transformer model. Experiment Protocol

[1] Red-Green-Blue image format

explains technical detail about how the experiment is carried out. Results presents and analyzes our findings across the algorithms. Discussion deliberates the implications of our experiment's results. Finally, we discuss future work and limitations.

Background and Previous Works

Consider a Markov Decision Process (MDP) to model our reinforcement learning environment. Let $\mathcal{M}$ designate the MDP composed of $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. $\mathcal{S}$ is a measurable state space, indexed as $s \in \mathcal{S}$; $\mathcal{A}$ is a measurable action space, indexed as $a \in \mathcal{A}$; p is a transition function which maps s and a to a set of probability distributions; r is a reward function that maps a state and action to a real number; γ is the discount factor, represented as a real number between zero and one (Gummadi et al. 2022).

An agent's policy is defined as $\pi(a|s)$, a probability distribution. At each time step, t, the agent draws an actions $a_t \sim \pi(\cdot|s_t)$. The agent then obtains a reward $r_{t+1} = r(s_t, a_t)$. The formalization of $\pi(a|s)$ is given in (Lillicrap et al. 2015).

In the MDP, policy gradient (PG) methods find an optimal policy, π^* that maximizes the expected sum of discounted rewards. Note that PGs optimize without knowledge of the transition kernel, p. More formally, $\pi^* \in argmax_\pi J(\pi)$, where

$$J(\theta) = E[\sum_{t=0}^{\infty} \gamma^t r_{i+1}(s_{t+1}, a_{t+1})]. \tag{1}$$

One of the first PG methods, REINFORCE (Williams 1992), uses likelihood ratios to update the policy. REINFORCE applies gradient descent to optimize $J(\theta)$ by scaling log-likelihood probabilities from a policy distribution by discounted rewards. However, REINFORCE has high variance in performance (Schulman et al. 2017).

A2C (Mnih et al. 2016) is an actor-critic method that addresses variance by applying a critic function. This approach combines elements of both value and policy based RL methods. The advantage value is calculated through a value function. A2C can be multi-processed so that multiple agents train in their own environments simultaneously. All the parallelized agents share a global policy and critic. However, scaling log-probabilities by an advantage value is still prone to over-correction when updating the model's parameters (Espeholt et al. 2018).

We turn to Proximal Policy Optimization (PPO) for its stability. PPO is a policy gradient method where a clipped surrogate objective optimizes a policy (Schulman et al. 2017). Specifically,

$$L^{CLIP}(\theta) = E[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)] \tag{2}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}$. The ratio, r_t, comes from TRPO (Schulman et al. 2015). PPO addresses this through limiting the gradient descent step. PPO is taking the minimum between a PG objective and the clipped objective. PPO's update step is curbed, e.g., the objective function is flatten (Schulman et al. 2017), ensuring that the agent does not over-correct itself. As a result, PPO's objective function also allows multiple epochs for policy updating. There exist baseline PPO implementations for standard Gym environments (Raffin et al. 2021). However, many PPO implementations for autonomous vehicles are platform specific (Wang and Zou 2022; Muzahid, Kamarulzaman, and Rahman 2021), making it difficult for standardizing robust autonomous vehicle benchmarking.

AV benchmarks have been proposed for DRL algorithms in simulation rather than real-world data. VISTA is a photo-realistic AV simulator (Amini et al. 2019). VISTA is an autonomous driving perception and control simulator that simulates an environment from real-world images. Conventionally, 3D AV simulators like Duckietown (Chevalier-Boisvert et al. 2018), TORCS (Espié et al. 2005), and Racecar-Gym (Towers et al. 2023) are purely rendered, simulated environments. VISTA, on the other hand, takes in real-world driving data and synthesizes novel camera perspectives. External data passed into VISTA is defined as a trace. A trace features a collection of sensor-based time-series data that VISTA uses for generating novel data. VISTA's sensors include a front RGB camera, LiDAR, and event camera, mimicking a real-world autonomous vehicle. At each time step t, we can control the latitude and longitude dynamics of an agent by specifying steering commands and speed respectively. Typical autonomous control is performed by deep neural networks (DDNs) (Kuutti et al. 2021).

Convolutional neural networks (CNNs) can be modeled as end-to-end systems where camera images are mapped directly to steering commands (Pomerleau 1988). CNNs extract features from images and produce a vehicle control value. However, CNNs do not consider temporal features when given a sequence of images. This can lead to high-variance in output. Recurrent neural networks (RNNs) address this shortcoming by having an internal memory mechanism (Kuutti et al. 2021). Furthermore, RNNs like Long short-term memory (LSTM) are designed to reduce vanishing gradients while learning temporal differences (Eraqi, Moustafa, and Honer 2017).

Lastly, Transformer neural network architectures have shown promising success in natural language processing (Vaswani et al. 2017), computer vision (Dosovitskiy et al. 2020), and reinforcement learning (Parisotto et al. 2020). Transformers excel in sequence modeling and surpass LSTMs. A few key difference between transformers and LSTMS are that transformers are parallelizble, scalable, and feature an attention mechanism. These attributes allow transformers to capture longer temporal dependencies than LSTMs. AV tasks that involve taking in sequences of images make transformers an attractive option to measure contextual patterns (Li et al. 2022); Lampinen et al. (2021) has done research on applying attention mechanisms to hierarchical DRL tasks.

Experiment Framework

In this section, we introduce a reinforcement learning framework to standardize measuring policy gradient algorithms in VISTA. We focus on two tasks: collision avoidance and lane following. While there are two vehicles in the task, the ego agent is the one we are trying to optimize while the other vehicle serves as the obstacle. The policy for the obstacle is static.

States are defined as RGB images of size (70×310) in pixels. VISTA renders full images with size (400×640) pixels and are pre-processed before being passed into the model. Pre-processing entails cropping and resizing the image for the region of interest (Amini et al. 2020). Images are then normalized so that pixel values are between $[0.0, 1.0]$ instead of $[0.0, 255.0]$. This pre-processing technique is adapted from the a subsequent VISTA paper (Wang et al. 2022).

We define the reward function as the sum of two parts: lane following (r_{lane})and collision avoidance ($r_{\text{collision}}$).

$$r(s,a) = r_{\text{lane}}(s,a) + r_{\text{collision}}(s,a) \tag{3}$$

We use Wang et al. (2022)'s formal lane and collision reward definitions

$$r_{\text{lane}} = 1 - (\frac{q_{\text{lat}}}{Z_{\text{lat}}})^2 \tag{4}$$

$$r_{\text{collision}} = -\frac{|\text{Dilate}(P_{ego}) \cap P_{other}|}{|P_{ego}|} \tag{5}$$

Equation 4 penalizes an agent proportionate to its distance away from the center of the lane. q_{lat} is the agent's distance from the center of the lane and Z_{lat} designates how far the agent moves left or right (Wang et al. 2022). We set Z_{lat} as half of the road's width. Meanwhile, equation 5 computes the overlap of the dilated ego agent and other agent's polygons over the original ego agent polygon. We dilate the ego agent's polygon by 10 fold. Under this reward function, the maximum reward value is 1.0.

An agent terminates when it goes out of lane, exceeds maximum rotation (steering angle exceeds 130 degrees), crashes with another object, or finishes a track. Terminating automatically issues a reward value of zero.

In each algorithm, we use a 5-layer CNN with GroupNorm and ReLU (Wang et al. 2022). This CNN takes in the pre-processed camera images and returns two values: mean and standard deviation. The mean and standard deviation are then used as parameters for a Gaussian distribution. Steering command values are sampled from this Gaussian distribution. In PPO, we also store the log of the probability density evaluated at that steering commands value. This allows us to compute the ratio of the current policy log probabilities over previous policy log probabilities. This is for stability in PPO (Schulman et al. 2017).

The ego agent is controlled through steering commands. Steering commands, also known as curvature in VISTA (Amini et al. 2019), affect the latitudinal direction and yaw of the car. Speed, which influences longitudinal control, is given. Moreover, our goal is for an agent to follow the road and avoid obstacles which only requires latitudinal commands.

Actor-Critic Vision Transformer

Vision transformers (ViTs) have been increasingly showing success over CNN and RNN models (Dosovitskiy et al. 2020; Touvron et al. 2021; Dehghani et al. 2023). Parallelizable self-attention mechanisms in ViTs have mitigated the vanishing gradient problem RNNs face. A transformer's attention mechanism, multi-modal architectures, and scalability make them an attractive option for reinforcement learning (Agarwal et al. 2023). End-to-end autonomous driving, in a DRL setting, can also benefit from transformers. (Liu et al. 2022). An ego-agent should be trained to predict the next action, following a sequence of past actions (Kuutti et al. 2021). Hence, multi-modal transformers can be applied to learn sequential patterns in an AV setting.

We implement a ViT model based off Dosovitskiy et al. (2020)'s implementation. Figure 1 illustrates the ViT we adapted for our DRL task. We specify a ViT model to take in RGB images of size (144×144). Images are reshaped so that they are squares to extract image patches of size 16. Furthermore, we add an extra MLP to process critic values given a sequence of images in addition to the classifier MLP. This adjustment is motivated by A2C and PPO requiring a critic value for the advantage function (Liu et al. 2022). Both Actor and Critic MLPs have a hidden dimension size of 2048. We call this adapted ViT model, AC-VIT.

Similar to the CNN model, the actor predicts μ and σ values for a Gaussian distribution ($\mathcal{N}$) that we sample actions from. Sampled actions should probabilistically give the highest reward.

A noteworthy difference from the A2C-CNN implementation is that we induce gradient clipping in the transformer implementation; we use a gradient clip value of 2.0. This is because a deeper neural network like ViT and an unconstrained update step can lead to exploding gradients.

Experiment Protocol

In this section we describe how experiments were conducted to measure the PG algorithms and NN architectures.

Algorithm 1: Policy Gradient Training

```
Parameter: Env, Model, Advantage type
Output: csv file
 1: Initialize Env
 2: Initialize csv file
 3: while Episode$_i \leq N$ do
 4:     $s_t \longleftarrow$ Env reset
 5:     Let $t, r = 0$.
 6:     for $t \longrightarrow T$ do
 7:         $\mu, \sigma \longleftarrow$ Model($s_t$)
 8:         $\pi_t \longleftarrow$ Gaussian($\mu, \sigma^2$)
 9:         $a \longleftarrow \pi_t(s_t)$
10:         $s_{t+1}, r_{t+1}$, done $\longleftarrow$ Env.step(a)
11:         memory.add($s_{t+1}, r_{t+1}$, done)
12:         if done then
                break
13:         end if
14:     end for
15:     $\theta \longleftarrow \theta + \Delta J(\theta)$ {Update the model's parameters}
16:     Evaluate model
17:     Write evaluation to csv file
18: end while
19: return csv file
```

Table 1: A summary of the PG/AC algorithms, NN models, and learning rates we used in our benchmark.

Algorithm	NN	LR	PC	p-value
REINFORCE	CNN	5e-5	-0.111	0.0788
A2C	CNN	5e-5	0.298	1.599e-06
	AC-ViT	5e-6	-0.059	0.352
PPO	CNN	5e-5	0.175	0.006
	AC-ViT	5e-6	0.387	2.468e-10

Algorithm 1 features a generalized set of instructions used to train an agent. It is adapted from Vaswani et al. (2022)'s pseudo code, as well as Amini's lecture [2], to include different environments and model specifications. The main idea behind it is to update the NN (Model/Policy) after every T steps for N episodes. s_t denotes the image(s) passed into the NN model. It is assumed that s_t undergoes some preprocessing step, depending on the NN architecture. All NN models return a μ and σ value. The μ and σ values then specify a Gaussian distribution. We then sample actions, a, from the Gaussian distribution. The agent takes the next step in the environment with the newly sampled action. Stepping in the environment returns the next state, s_{t+1}, reward at $t+1$, and a boolean value (indicating if the agent terminated).

Note that for every timestep, t, we keep track of the current state, reward value, log probabilities, and terminal boolean in a external Memory class. This Memory class serves two purposes: batch learning and computing advantage functions.

The reason why our update step is ambiguous is because the Advantage value depends on the algorithm. PPO and A2C have a critic network that calculate the Advantage value. However, REINFORCE only uses the cumulative discounted rewards (Williams 1992). Moreover, PPO's objective function is curbed to prevent overstepping gradients, while A2C and REINFORCE do not have as much constraints in their objective functions.

Lastly, Algorithm 1 returns a CSV file. This CSV file records an agent's performance at each episode, i. In line 16, *Evaluate model*, starts a new instance of the environment and runs the agent with π_θ until termination. We track the cumulative rewards received, amount of steps taken, and percentage of the track completed. These metric will help us observe the rate at which an agent learns how to complete a task.

Table 1 lists the algorithms, NN architectures, learning rate, Pearon correlation coefficient (PC), and p-values. We found greater success in lowering the learning rate for A2C and PPO under the AC-ViT model due to their complexity over the task's scale. Our last protocol we specify is the number of agents to train concurrently. A2C and PPO offer multi-processing capabilities (Schulman et al. 2017; Mnih et al. 2016). This means we can run separate instances of an environment with an agent. After t steps, we aggregate the data from each environment and update the model. This increases data collecting efficiency. We decide to run three separate threads for A2C and a single thread for PPO. Our rationale is to highlight PPO's stable gradient-step. Moreover, PPO takes multiple update steps over a single batch, so it has a built-in data efficient mechanism.

Results

This section presents results from various experiments conducted across three algorithms and two NN architectures. All algorithms were trained on two GeForce RTX-3090 GPUs. We only consider front-camera RGB images in training and testing. All images are size (400×640) pixels but are subject to pre-processing.

The ego agent is trained to follow a lane and avoid an object down the road simultaneously. The road width is 4 meters. In every instance of an environment, we randomly spawn an object (obstacle vehicle) within [50.0,60.0] meters away from the starting coordinate of the ego agent. The obstacle vehicle is also initialized with lateral noise within [-3.0, 3.0] meters. Meanwhile, the ego agent is initialized in the middle of the lane, beginning with the highest reward it can receive. We deem an agent's learning has successfully converged if it consistently scores a cumulative reward of 100 or higher which is consistent with previous work [3].

VISTA

VISTA has a total of four traces (two are reversed traces), so two unique tracks for an agent to drive in. We train and test on one trace at a time. Among the four traces, we present results on the trace recorded at 5:46PM. This challenging trace is chosen due to a variety of lighting difficulties. For example, the sun goes directly into the camera frame.

Our VISTA experiment compares PPO, A2C, and REINFORCE using a CNN model. Figure 2, illustrates the average cumulative rewards garnered over 250 episodes. In this experiment, each algorithm is trained with a CNN model. Each algorithm was trained multiple separate times, resulting multiple trials. We present four trial results. Figure 2 shows the average reward at each episode across the four trials. Four trials helps give a better idea of the model's true performance since these DRL algorithms can yield high variance at any step.

PPO and A2C performed best, having the highest average reward. REINFORCE, on the other hand, had the lowest cumulative average reward. PPO starts out favorably in early episodes with rewards between 100 to 200 points, while A2C finishes with a higher score. PPO's rolling average reward value is consistently over 100 points.

The left plot in Figure 1 presents results for PPO and A2C trained with the AC-ViT model. Overall, the models trained with AC-ViT performed worse than with CNN. However, a noteworthy observations is that PPO outperformed A2C on average.

We present the PC coefficient and p-value for each model and algorithm. The PCcoefficient measures the stability of

[2]https://github.com/aamini/introtodeeplearning
Amini et. al. Introduction to Deep Learning

[3]https://github.com/aamini/introtodeeplearning
Amini et. al. Introduction to Deep Learning

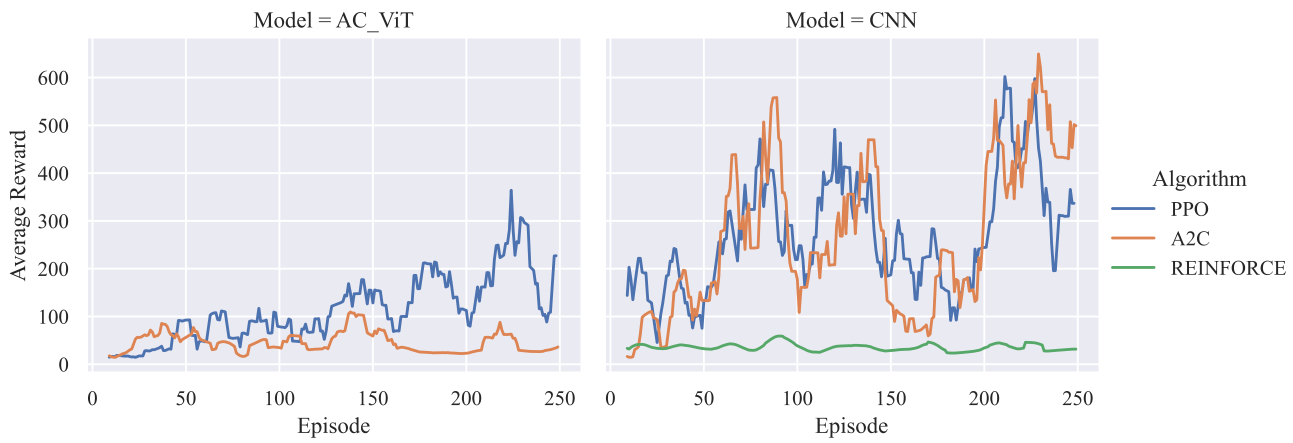

Figure 2: Learning speed comparison among PPO, A2C, and REINFORCE in VISTA

the agent's learning. The PC coefficients values and p-values are shown in Table 1.

Racecar-Gym

We measure A2C's performance in another environment, Racecar-Gym (Brunnbauer and Berducci 2020). Racecar-Gym is an AV environment based off PyBullet, a Python wrapper for the Kubric simulator (Greff et al. 2022). This is a purely 3D generated environment with more complicated driving environments than VISTA. We are interested in observing how A2C works in an environment like Racecar, where fine-dynamic control is required to successfully avoid another car and complete a race course. We also use a more intricate training architecture, coupled with this complex experimental environment. Our experiments show how A2C performs with a ResNet50 architecture (He et al. 2016). Our framework is modular by design, so we implement a ResNet50 for a task that might require finer control.

Since course completion is a sub-goal, in Racecar-Gym we define a "distance" reward for each step taken as 0.11376564. This value is calculated as the total length of the track divided by total number of steps. Crashing results in a reward of 0.0. Similar to the VISTA environment setup, we use a lane reward that penalizes the agent for deviating too far away from the lane. Given that the tracks are more challenging we only observe lane following without obstacle avoidance.

Figure 3 shows the rewards per episode over 500 episodes. We show 500 episodes, rather than the 250 episodes shown in the previous experiment to show the lack of convergence.

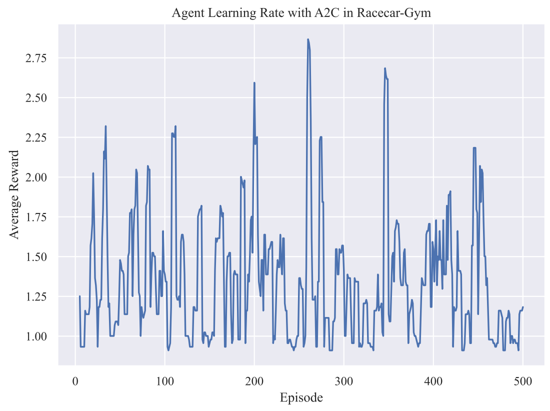

Figure 3: Learning speed of A2C in Racecar Gym with a ResNet50 model

Discussion

Through experimenting in our PG framework, we measure well-established DRL methods as well as attempt to improve them using a ViT.

In analyzing the results for VISTA, between PPO and A2C, we see periodic spikes in performance at similar episodes. We believe that these spikes are caused by the agent successfully avoiding the obstacle vehicle because it passed it and continued to follow the lane. Since the obstacle car is placed approximately 50 to 60 meters away, this behavior suggests that PPO successfully learned how to follow a lane with an obstacle car. This shows that lane following can be done in conjunction with other types of obstacles and tasks.

However, this performance is highly dependent on the underlying algorithm. For example, REINFORCE achieves poor performance on VISTA. This can be attributed to an unstable gradient step, as shown in previous work (Schulman et al. 2017). Currently, there are no measures taken to stop the car from over-correcting itself. This induces high variance, which is why the ego agent fails to achieve high rewards.

The underlying NN architecture also affects performance. For example, one reason why A2C performed worse than AC-ViT is due to model complexity. A2C is prone to overstepping in gradient descent. Whereas PPO's stable gradient descent could be the reason why it outscored A2C. It is also worth mentioning that PPO under the AC-ViT model showed consistent, gradual improvement with less variance than PPO under the CNN model. These architecture insights are defended by the p-value analysis.

In the CNN model, we receive a correlation coefficient value of 0.175 and 0.298 for PPO and A2C respectively; the p-values are 0.006 and 1.599e-6. This suggests that there is significant evidence to reject the null hypothesis that episode number of average reward are positively correlated.

Conversely, AC-ViT shows PPO outperforming A2C with a pearson correlation coefficient value of 0.387 and a p-value of 2.468e-10. The A2C correlation coefficient and p-value with AC-ViT are -0.059 and 0.352. Hence, PPO indicates stable, increasing learning when trained with the AC-ViT model.

We found that a single threaded agent trained with PPO performs adequately the same as a multi-threaded agent trained with A2C under a CNN model. While the agents trained with a ViT did not show results that align with the current literature in transformers, they warrant further investigation to properly incorporate transformers in a PG and AV setting.

In Racecar-gym, the agent trained with A2C-CNN presents lackluster results. There are multiple sources to point to this performance. Reward design is a crucial component to any DRL application. A reward function that emphasized lane and obstacle avoidance could have a different outcome over distance traveled. Figure 3 could also speak to A2C gradient step instability. Lastly, applying ResNet50 in a non-photorealistic environment like Racecar-Gym might be too complicated for the task and overfitted early on - this could explain the occasional spikes in performance.

Future Work and Limitations

A2C and PPO are just two of many policy optimization approaches that can be applied in AV. Deep Deterministic Policy Gradients (DDPG) (Chou, Maturana, and Scherer 2017), Twin delayed DDPG (Fujimoto, Hoof, and Meger 2018), and Soft actor-critic (Haarnoja et al. 2018) are other state-of-the-art DRL techniques.

Regarding neural networks models, a further investigation involving transformers in a DRL application is warranted. Approaches like GTrXL (Parisotto et al. 2020), hierarchical memory (Lampinen et al. 2021), actor-critic transformers in DDPG all show promising results in their respective DRL environments. Additionally, a video vision transformer (Arnab et al. 2021) could be an interesting application, where instead of sequential episodes, chunks of episodic steps are passed into the NN model like a video.

Lastly, expanding our framework would increase usability in other AV simulators. The first place for expansion is integrating more AV environments. E.g., Duckietown (Chevalier-Boisvert et al. 2018) is another PyBullet-based 3D AV simulator that focuses on tasks like lane following and obstacle avoidance. More environments also opens the possibility of measuring more complicated tasks. Duckietown notably has a full-city grid which avails tasks like city navigation with obstacles (Chevalier-Boisvert et al. 2018). The next step for a more complicated task in VISTA is for an agent to overtake a moving vehicle (Wang et al. 2022).

Conclusion

In this paper, we present a standardized experimental framework, a benchmark suite, and an adapted attention-based architecture for testing RL algorithms in the autonomous driving domain. We demonstrated results in two AV simulation environments: VISTA and Racecar-Gym. We analyzed the results and suggested new areas of exploration in attention-based methods. The contributions of our work enhances the reproducibility of results, while also streamlining progress of RL techniques in the real-world challenges of autonomous vehicles.

We showed that there are many potential opportunities for improvements in RL algorithm performance in autonomous driving. One key future area of research are attention architectures, which could help to enable safer autonomous driving in unknown environments. In future and ongoing work, we will examine these mechanisms further in both VISTA and Racecar-gym as well as other autonomous driving simulation platforms like Duckietown (Chevalier-Boisvert et al. 2018).

Overall, this work contributes a standardized experimental framework, benchmark suite, and a adapted attention-based architecture for refining RL algorithms in autonomous driving. This enables safer, more efficient, and more reliable autonomous vehicles in both testing and in development.

Acknowledgments

Removed to preserve anonymity.

References

Agarwal, P.; Rahman, A. A.; St-Charles, P.-L.; Prince, S. J. D.; and Kahou, S. E. 2023. Transformers in Reinforcement Learning: A Survey. .

Amini, A.; Gilitschenski, I.; Phillips, J.; Moseyko, J.; Banerjee, R.; Karaman, S.; and Rus, D. 2020. Learning Robust Control Policies for End-to-End Autonomous Driving From Data-Driven Simulation. 5(2): 1143–1150. Conference Name: IEEE Robotics and Automation Letters.

Amini, A.; Rosman, G.; Karaman, S.; and Rus, D. 2019. Variational End-to-End Navigation and Localization. In *2019 International Conference on Robotics and Automation (ICRA)*, 8958–8964. IEEE. ISBN 978-1-5386-6027-0.

Amini, A.; Wang, T.-H.; Gilitschenski, I.; Schwarting, W.; Liu, Z.; Han, S.; Karaman, S.; and Rus, D. 2022. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *2022 International Conference on Robotics and Automation (ICRA)*, 2419–2426. IEEE.

Arnab, A.; Dehghani, M.; Heigold, G.; Sun, C.; Lučić, M.; and Schmid, C. 2021. Vivit: A video vision transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, 6836–6846.

Brunnbauer, A.; and Berducci, L. 2020. racecar$_{gym}$.

Chevalier-Boisvert, M.; Golemo, F.; Cao, Y.; Mehta, B.; and Paull, L. 2018. Duckietown Environments for OpenAI Gym. https://github.com/duckietown/gym-duckietown.

Chou, P.-W.; Maturana, D.; and Scherer, S. 2017. Improving stochastic policy gradients in continuous control with deep reinforcement learning using the beta distribution. In *International conference on machine learning*, 834–843. PMLR.

Dehghani, M.; Mustafa, B.; Djolonga, J.; Heek, J.; Minderer, M.; Caron, M.; Steiner, A.; Puigcerver, J.; Geirhos, R.; Alabdulmohsin, I.; Oliver, A.; Padlewski, P.; Gritsenko, A.; Lučić, M.; and Houlsby, N. 2023. Patch n' Pack: NaViT, a Vision Transformer for any Aspect Ratio and Resolution. .

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, 1–16.

Eraqi, H.; Moustafa, M.; and Honer, J. 2017. End-to-End Deep Learning for Steering Autonomous Vehicles Considering Temporal Dependencies.

Espeholt, L.; Soyer, H.; Munos, R.; Simonyan, K.; Mnih, V.; Ward, T.; Doron, Y.; Firoiu, V.; Harley, T.; Dunning, I.; Legg, S.; and Kavukcuoglu, K. 2018. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. In *Proceedings of the 35th International Conference on Machine Learning*, 1407–1416. PMLR. ISSN: 2640-3498.

Espié, E.; Guionneau, C.; Wymann, B.; Dimitrakakis, C.; Coulom, R.; and Sumner, A. 2005. TORCS, The Open Racing Car Simulator.

Fujimoto, S.; Hoof, H.; and Meger, D. 2018. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, 1587–1596. PMLR.

Greff, K.; Belletti, F.; Beyer, L.; Doersch, C.; Du, Y.; Duckworth, D.; Fleet, D. J.; Gnanapragasam, D.; Golemo, F.; Herrmann, C.; et al. 2022. Kubric: A scalable dataset generator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3749–3761.

Gummadi, R.; Kumar, S.; Wen, J.; and Schuurmans, D. 2022. A Parametric Class of Approximate Gradient Updates for Policy Optimization. In *Proceedings of the 39th International Conference on Machine Learning*, 7998–8015. PMLR. ISSN: 2640-3498.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. PMLR.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Kuutti, S.; Bowden, R.; Jin, Y.; Barber, P.; and Fallah, S. 2021. A Survey of Deep Learning Applications to Autonomous Vehicle Control. 22(2): 712–733.

Lampinen, A.; Chan, S.; Banino, A.; and Hill, F. 2021. Towards mental time travel: a hierarchical memory for reinforcement learning agents. In *Advances in Neural Information Processing Systems*, volume 34, 28182–28195. Curran Associates, Inc.

Li, G.; Qiu, Y.; Yang, Y.; Li, Z.; Li, S.; Chu, W.; Green, P.; and Li, S. E. 2022. Lane change strategies for autonomous vehicles: a deep reinforcement learning approach based on transformer. *IEEE Transactions on Intelligent Vehicles*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Liu, H.; Huang, Z.; Mo, X.; and Lv, C. 2022. Augmenting Reinforcement Learning with Transformer-based Scene Representation Learning for Decision-making of Autonomous Driving.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. In Balcan, M. F.; and Weinberger, K. Q., eds., *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, 1928–1937. New York, New York, USA: PMLR.

Muzahid, A. J. M.; Kamarulzaman, S. F.; and Rahman, M. A. 2021. Comparison of PPO and SAC Algorithms Towards Decision Making Strategies for Collision Avoidance

Among Multiple Autonomous Vehicles. In *2021 International Conference on Software Engineering Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM)*, 200–205.

Parisotto, E.; Song, F.; Rae, J.; Pascanu, R.; Gulcehre, C.; Jayakumar, S.; Jaderberg, M.; Kaufman, R. L.; Clark, A.; Noury, S.; Botvinick, M.; Heess, N.; and Hadsell, R. 2020. Stabilizing Transformers for Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning*, 7487–7498. PMLR. ISSN: 2640-3498.

Pomerleau, D. A. 1988. ALVINN: An Autonomous Land Vehicle in a Neural Network. In *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann.

Raffin, A.; Hill, A.; Gleave, A.; Kanervisto, A.; Ernestus, M.; and Dormann, N. 2021. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268): 1–8.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, 1889–1897. PMLR. ISSN: 1938-7228.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jegou, H. 2021. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, 10347–10357. PMLR. ISSN: 2640-3498.

Towers, M.; Terry, J. K.; Kwiatkowski, A.; Balis, J. U.; Cola, G. d.; Deleu, T.; Goulão, M.; Kallinteris, A.; KG, A.; Krimmel, M.; Perez-Vicente, R.; Pierré, A.; Schulhoff, S.; Tai, J. J.; Shen, A. T. J.; and Younis, O. G. 2023. Gymnasium.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Vaswani, S.; Bachem, O.; Totaro, S.; Müller, R.; Garg, S.; Geist, M.; Machado, M. C.; Castro, P. S.; and Roux, N. L. 2022. A general class of surrogate functions for stable and efficient reinforcement learning. In *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, 8619–8649. PMLR. ISSN: 2640-3498.

Wang, T.-H.; Amini, A.; Schwarting, W.; Gilitschenski, I.; Karaman, S.; and Rus, D. 2022. Learning Interactive Driving Policies via Data-driven Simulation. In *2022 International Conference on Robotics and Automation (ICRA)*, 7745–7752.

Wang, Y.; and Zou, S. 2022. Policy Gradient Method For Robust Reinforcement Learning. In *Proceedings of the 39th International Conference on Machine Learning*, 23484–23526. PMLR. ISSN: 2640-3498.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. 8(3): 229–256.