Reinforcement Learning for CarRacing-v3: SAC+DrQ and PPO Implementation Study

EE569 Deep Learning

University of Tripoli, Libya

Authors: Ahmed Mohamed Bakory (2200207974)
Faisal Ali Elhouderi (2200208864)
Muhammed Ali Muhmoud (2200208982)

Instructor: Dr. Nuri Benbarka

Date: December 2025

Repository: https://github.com/G-0-4/EE569-CarRacing-Challenge-G-0-4

A study of pixel-based reinforcement learning for autonomous racing using Soft Actor-Critic with Data-regularized Q-learning and Proximal Policy Optimization.

Abstract

This report presents our implementation of reinforcement learning agents for CarRacing-v3. We implement **SAC+DrQ** for continuous control and **PPO** using stable-baselines3. Our SAC+DrQ achieves **861.47** ± **52.73** mean reward, exceeding both the 700-point success and 800-point excellence thresholds. We analyze learning dynamics, compare sample efficiency, and discuss challenges including memory constraints and gradient flow in pixel-based RL.

Keywords: reinforcement learning; soft actor-critic; PPO; CarRacing; continuous control; pixel observations

Contents

1 Introduction

1.1 Problem Statement

CarRacing-v3 from Gymnasium is a continuous control task where an agent learns to drive using 96×96 pixel observations. The action space is continuous: steering ($[-1, 1]$), gas ($[0, 1]$), and brake ($[0, 1]$).

The reward structure incentivizes fast track completion: $+1000/N$ for each of N tiles visited, -0.1 per timestep. Successful episodes yield 800–950 points.

1.2 Performance Targets

- **Success:** Average score > 700 over 3 episodes
- **Excellence:** Average score > 800 with minimal interactions
- **Bonus 1:** Highest evaluation score among groups
- **Bonus 2:** Lowest total steps while achieving > 700

1.3 Approach

We implement two approaches:

1. **SAC+DrQ:** Off-policy actor-critic with entropy regularization and data augmentation, using 2D continuous actions (steer + acceleration)
2. **PPO:** On-policy policy gradient using stable-baselines3's CNN policy

2 Theoretical Background

2.1 Soft Actor-Critic (SAC)

SAC [1] optimizes a maximum entropy objective:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}\left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right] \tag{1}$$

Key components: twin Q-networks to mitigate overestimation, soft Gaussian policy with tanh squashing, automatic temperature α tuning, and target networks for stable bootstrapping.

2.2 Data-regularized Q-learning (DrQ)

DrQ [2] applies random shift augmentation during training:

$$\text{aug}(o) = \text{RandomCrop}(\text{Pad}(o, 4), 84, 84) \tag{2}$$

This simple augmentation dramatically improves sample efficiency for pixel-based RL.

2.3 Proximal Policy Optimization (PPO)

PPO [3] uses a clipped surrogate objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right] \tag{3}$$

PPO is stable and parallelizable but less sample-efficient than off-policy methods.

3 Implementation Details

3.1 Environment Preprocessing

- Grayscale conversion (3 channels $\rightarrow$ 1)
- Crop to 84×84, remove HUD
- Stack 4 frames for temporal information
- 2D action space: steer + accel (positive=gas, negative=brake)

3.2 SAC+DrQ Architecture

Encoder: Atari-style CNN (Conv 8×8/4 $\rightarrow$ Conv 4×4/2 $\rightarrow$ Conv 3×3/1 $\rightarrow$ FC 50 + LayerNorm + Tanh)

Actor: MLP (50 $\rightarrow$ 1024 $\rightarrow$ 1024 $\rightarrow$ 2×2) outputting mean and log-std for squashed Gaussian

Critic: Twin Q-networks, each with encoder + MLP (52 $\rightarrow$ 1024 $\rightarrow$ 1024 $\rightarrow$ 1)

3.3 Hyperparameters

Table 1: Key hyperparameters for SAC+DrQ and PPO.

Parameter	**SAC+DrQ**	**PPO**
Replay/Buffer size	30,000	–
Batch size	256	64
Learning rate	1×10^{-4}	3×10^{-4}
Discount (γ)	0.99	0.99
Feature dim	50	CNN default
Hidden dim	1024	MLP default
DrQ padding	4	–
Parallel envs	1	6
Steps per update	1	2,048
Clip range	–	0.2
Entropy coef	auto (α)	0.01

3.4 Training Infrastructure

Hardware: NVIDIA RTX 3050 (4GB), 8GB RAM (WSL2: ~5.8GB available). The 30k replay buffer uses ~1.7GB RAM.

4 Experimental Design

Table 2: Summary of experimental runs.

Run	Algorithm	Configuration	Steps	Time
1	SAC+DrQ	Default, seed=1	1.91M	25h 15m
2	PPO	SB3 CnnPolicy, 6 envs	1.00M	3h 35m
3–5	SAC+DrQ	OOM tests (killed)	20–40k	–
6	SAC+DrQ	Modified config	∼1.7M	∼38h

Run 1 (SAC+DrQ Baseline): Primary experiment, trained in two phases (0–600k, then resumed to 1.9M).

Run 2 (PPO): Completed but achieved significantly lower performance.

Runs 3–5: Terminated by OOM, helped identify maximum buffer size.

Run 6: Configuration study without Aim tracking, limiting analysis capability.

5 Results

5.1 Final Performance

Table 3: Evaluation results (3 episodes each).

Run	Mean	Std	Max	Min
SAC+DrQ (Run 1)	**861.47**	52.73	936.00	822.36
PPO (Run 2)	306.92	42.89	352.90	249.67
SAC+DrQ Modified	209.51	23.00	239.48	183.58

SAC+DrQ exceeds both success (700) and excellence (800) thresholds with **861.47 mean reward**.

5.2 Learning Curves

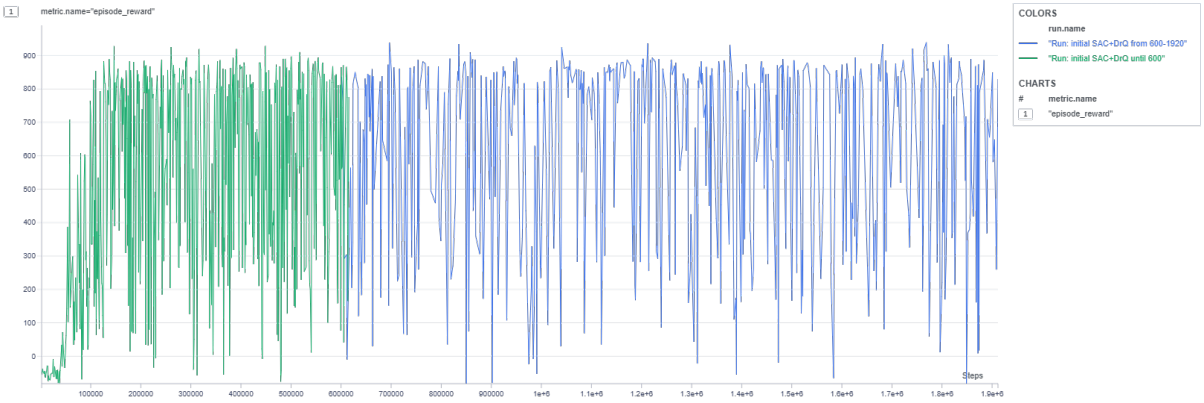


Figure 1: SAC+DrQ episode rewards. Gap at step 600k is the training resumption.

Figure 2: SAC+DrQ evaluation rewards (mean ± std). First exceeds 800 at step 126,000.

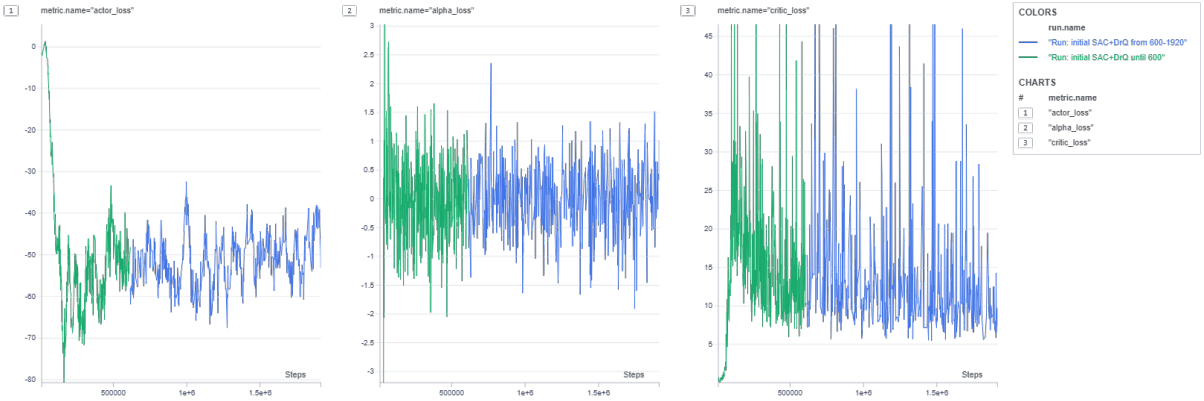
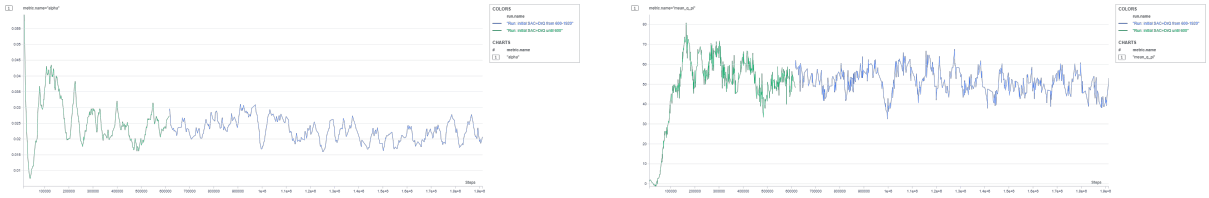


Figure 3: SAC+DrQ training losses: critic, actor, and alpha.



(a) Temperature α (b) Mean Q-values

Figure 4: SAC+DrQ entropy temperature and Q-value progression.

5.3 Sample Efficiency

Table 4: Sample efficiency comparison.

Algorithm	Steps to 800+	Final	Time
SAC+DrQ	126,000	861.47	25h 15m
PPO	Never	306.92	3h 35m

SAC+DrQ reaches 800+ in only 126k steps; PPO fails to exceed 400 after 1M steps.

5.4 Experiment Tracking

Run	Experiment	Run		
Name	Description	Hash	Date	Duration
• Run: PPO		3ecc764cc8124b5ea310053e	20:05:52 · 24 Jan, 26	3hrs 35min
• Run: initial SAC+DrQ from 600-1920		7612abfe36594b058ccf3e55	02:03:51 · 24 Jan, 26	17hrs 11min
• Run: initial SAC+DrQ until 600		645f75a64dab4388838caa5b	17:16:31 · 23 Jan, 26	8hrs 4min
• Run: initial killed at 40		87cb4cbc28334b47a3fcaaee	04:50:04 · 23 Jan, 26	2hrs 4min
• Run: initial killed at 20-2		1179c5cf2aae4af0bc0ab20b	03:47:11 · 23 Jan, 26	48min 44sec
• Run: initial killed at 20		406872990f64490197afc347	03:21:35 · 23 Jan, 26	1hrs 14min

Figure 5: Aim dashboard showing tracked experiments.

6 Discussion

6.1 Why SAC+DrQ Succeeds

- **Continuous actions:** Smooth, coordinated steering + acceleration
- **DrQ augmentation:** Critical regularization for pixel-based learning
- **Entropy regularization:** Prevents premature convergence
- **Off-policy learning:** Efficient experience reuse via replay buffer

6.2 Why PPO Underperforms

- **Suboptimal hyperparameters:** Small batch size (64), wide clip range (0.2)
- **Gaussian policy:** Unbounded; Beta distribution works better for bounded actions [5]
- **No augmentation:** More prone to overfitting
- **Missing metrics:** Aim only captured system metrics, preventing diagnosis

6.3 Ablation: Configuration Changes

Run 6 modified: buffer 30k→70k, updates/step 1→2, feature dim 50→64. Result: **209.51** (vs 861.47), showing that more is not always better. Possible causes: overfitting from more updates, larger features needing more data.

6.4 Memory Constraints

- 30k buffer (∼1.7GB): Stable
- 50k buffer (∼2.8GB): Marginal
- 70k+ buffer: Frequent OOM

6.5 Lessons Learned

1. Establish working baseline before improvements
2. Change one hyperparameter at a time
3. Always log experiments (Run 6 lacked tracking)
4. DrQ augmentation is essential for pixel-based RL

7 Conclusion

7.1 Summary

- SAC+DrQ achieved **861.47** ± **52.73**, exceeding both thresholds
- Reached 800+ in only **126,000 steps**
- PPO achieved 306.92, requiring hyperparameter optimization
- Naive hyperparameter scaling severely degraded performance

7.2 Recommendations

1. Use SAC+DrQ for pixel-based continuous control
2. Apply DrQ augmentation (random shifts)
3. Start with conservative hyperparameters
4. Ensure comprehensive experiment tracking

7.3 Future Work

- Beta-distribution PPO for bounded actions
- Systematic hyperparameter sweeps
- Action repeat for better sample efficiency
- RGB vs grayscale comparison

References

[1] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep RL with a Stochastic Actor. *ICML*, 2018.

[2] I. Kostrikov, D. Yarats, R. Fergus. Image Augmentation Is All You Need: Regularizing Deep RL from Pixels. *ICLR*, 2021.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347*, 2017.

[4] V. Mnih et al. Human-level Control Through Deep Reinforcement Learning. *Nature*, 518(7540):529–533, 2015.

[5] P.-W. Chou, D. Maturana, S. Scherer. Improving Stochastic Policy Gradients in Continuous Control with the Beta Distribution. *ICML*, 2017.

A Hyperparameter Details from Aim


```
1 item {
  "hparams": 27 items {
    "action_repeat": int 1
    "actor_lr": float 0.0001
    "alpha_lr": float 0.0001
    "batch_size": int 256
    "critic_lr": float 0.0001
    "device": string "cuda"
    "drq_pad": int 4
    "env_id": string "CarRacing-v3"
    "eval_every_steps": int 50000
    "feature_dim": int 50
    "frame_stack": int 4
    "gamma": float 0.99
    "grad_clip_norm": int 10
    "grayscale": bool True
    "hidden_dim": int 1024
    "init_alpha": float 0.1
    "num_eval_episodes": int 3
    "replay_size": int 30000
    "reward_scale": int 1
    "seed": int 1
    "start_steps": int 10000
    "target_entropy": None
    "tau": float 0.01
    "total_steps": int 2000000
    "update_after": int 10000
    "update_every": int 1
    "updates_per_step": int 1
  }
}
```

(a) SAC+DrQ

```
1 item {
  "hparams": 14 items {
    "action_repeat": int 1
    "batch_size": int 64
    "clip_range": float 0.2
    "ent_coef": int 0
    "frame_stack": int 4
    "gae_lambda": float 0.95
    "gamma": float 0.99
    "grayscale": bool True
    "learning_rate": float 0.0003
    "n_envs": int 1
    "n_epochs": int 10
    "n_steps": int 2048
    "seed": int 1
    "total_timesteps": int 1000000
  }
}
```

(b) PPO

Figure 6: Hyperparameters from Aim experiment tracking.

B Reproducibility

B.1 Training Commands

```
# SAC+DrQ (best result)
python train.py --total-steps 2000000 --seed 1 \
    --checkpoint-dir checkpoints/sac_run --run-name sac_drq_baseline

# PPO
```

```
python train_ppo.py --total-timesteps 1000000 --seed 1 \
    --checkpoint-dir checkpoints/ppo_run --run-name ppo_baseline
```

B.2 Evaluation

```
# SAC
python inference.py --checkpoint checkpoints/sac_run/best_sac_drq.pth \
    --algo sac_drq --episodes 3 --no-render --seed 1

# PPO
python inference.py --checkpoint checkpoints/ppo_run/best_model.zip \
    --algo ppo --episodes 3 --no-render --seed 1
```

B.3 Environment Setup

```
conda create -n car_racing python=3.10
conda activate car_racing
pip install -r requirements.txt
aim up  # View logs
```