Assignment-3

This report documents the implementation, experiments, and results for Programming Assignment 3: comparing two Dueling-DQN variants (Type-1 mean-normalized advantage and Type-2 max-normalized advantage) and Monte Carlo REINFORCE (with and without a learned TD(0) baseline) on CartPole-v1 and Acrobot-v1.

Implementation Summary

Dueling-DQN: shared feature extractor with two heads — value V(s) and advantage A(s,a). Two aggregation rules implemented:

* Type-1 (mean-normalized): Q(s,a)=V(s) + (A(s,a) - mean_a A(s,a))
* Type-2 (max-normalized) : Q(s,a)=V(s) + (A(s,a) - max_a A(s,a))

REINFORCE: Monte-Carlo policy gradient with optional learned baseline V(s;φ). Baseline is trained using TD(0) updates while policy is updated using episode returns (advantage = G_t - V(s_t)).

Network architectures

All networks use small MLPs appropriate to each environment observation size.

Example:

shared: Linear(128) -> ReLU -> Linear(128) -> ReLU

value head: Linear(1)

advantage head: Linear(n_actions)

Policy networks use a small MLP ending with a softmax over discrete actions for CartPole/Acrobot.

Training details & hyperparameters

Discount factor γ = 0.99 (fixed for all experiments)

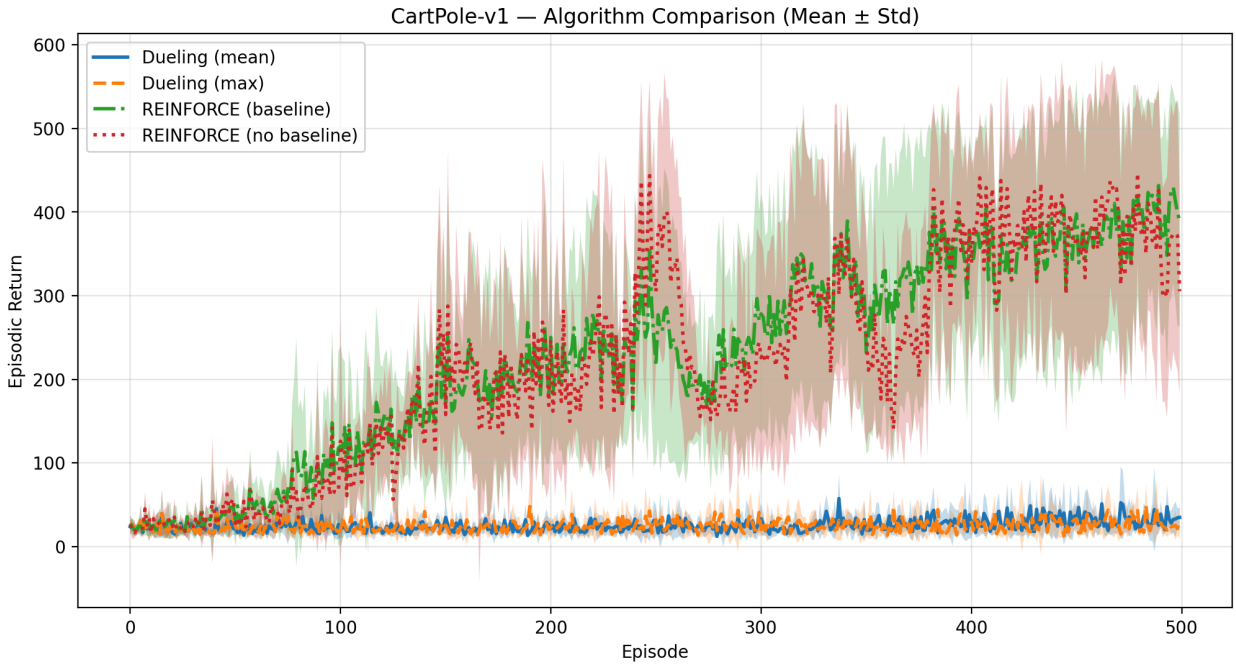
DQN (suggested): Adam lr=1e-3, buffer=50k, batch=64, target_update=500 steps, ε: 1.0→0.01

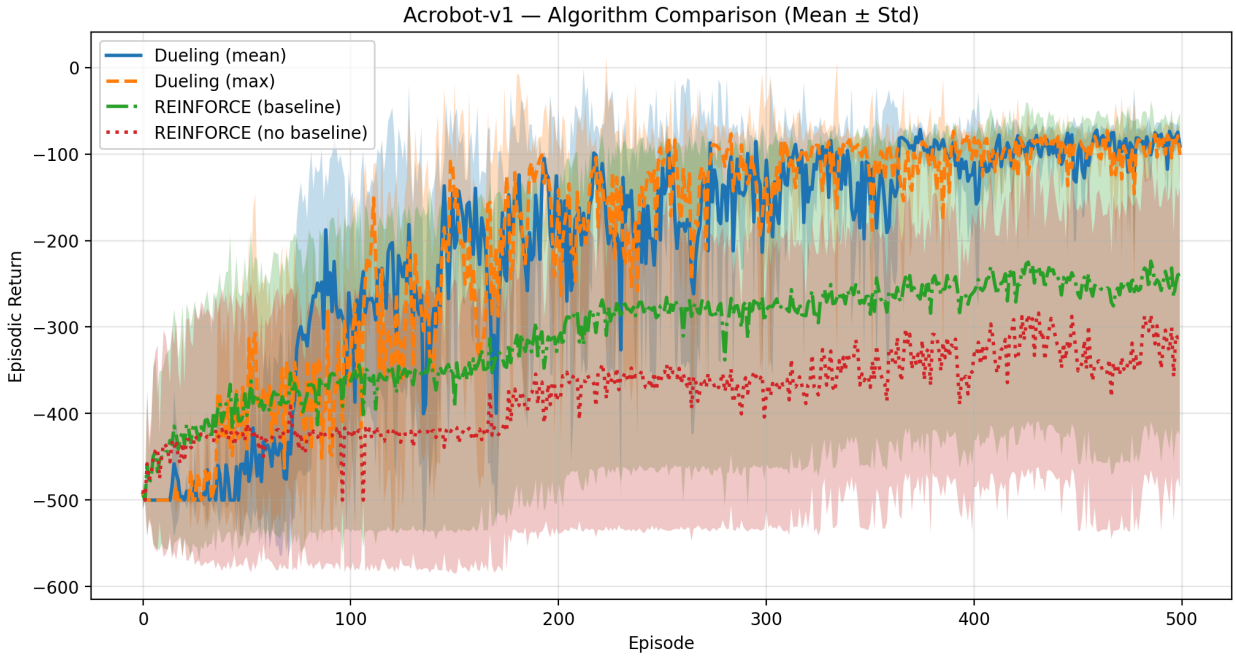
REINFORCE: policy lr=1e-3, baseline lr=1e-3, episodes: 1000–5000 depending on environment

Seeds: [0,1,2,3,4] and aggregate mean ± std across seeds

Save per-episode returns and checkpoints.

PLOTS(5 SEEDS)





Observations & insights

Acrobot-v1

1. Dueling-DQN (both variants) performs best. Both mean and max variants quickly climb from the initial plateau (~-500) to roughly -100 by ~300–500 episodes. They show substantial early variance but converge upward.

2. REINFORCE struggles here. With a baseline it does better than without, but both REINFORCE curves remain far worse than Dueling-DQN (baseline ≈ -250, no-baseline ≈ -320).

3. Advantage-normalization variant has little qualitative impact. mean vs max curves are similar in speed and final performance — normalization choice didn't radically change learning here.

4. High early variance, then stabilizing. All methods show large seed-wise variance early; DQN reduces variance faster than REINFORCE.

Why: Acrobot is a fairly sparse / harder control task where bootstrapped value-based methods (DQN with replay + target network) can accumulate low-variance Q estimates and exploit them better. On-policy Monte-Carlo REINFORCE suffers from high variance in returns and (with MC targets) slow, noisy updates — baseline helps but not enough to match DQN.

CartPole-v1

1. REINFORCE dominates for CartPole. Both REINFORCE variants (with baseline better than without) steadily increase to high returns (~300–500), consistent with CartPole being friendly to policy-gradient methods.

2. Dueling-DQN fails to learn here. Both `mean` and `max` variants remain near small returns (~10–40) across 500 episodes — not improving meaningfully.

3. High variability for policy gradient early, but converges upward. Baseline reduces variance and stabilizes learning (green smoother than red).

4. Advantage normalization again not decisive. Dueling mean vs max both show similar poor behavior.

Why: CartPole-v1 is a low-dimensional control task with dense, frequent rewards and smooth dynamics. In such environments, on-policy policy-gradient methods like REINFORCE tend to perform very well because they update the policy directly toward actions that produced higher returns in the most recent trajectories. This direct optimization allows REINFORCE to rapidly discover the sequence of balanced actions required to keep the pole upright.

In contrast, value-based methods like Dueling-DQN learn by estimating action values and improving the policy indirectly. This involves additional approximation steps such as bootstrapping from target networks and sampling from replay memory. While these mechanisms provide stability in more complex or sparse-reward environments, they can introduce extra delay in propagating the CartPole reward signal, making learning slower relative to REINFORCE. As a result, in this environment, the on-policy REINFORCE updates align more immediately with maximizing cumulative rewards, leading to faster and smoother improvement.