

# **COMPARING VALUE BASED AND ACTOR CRITIC REINFORCEMENT LEARNING METHODS**

**DQN VS PPO ON CARTPOLE AND LUNARLANDER**

By Kingsley Chukwuma

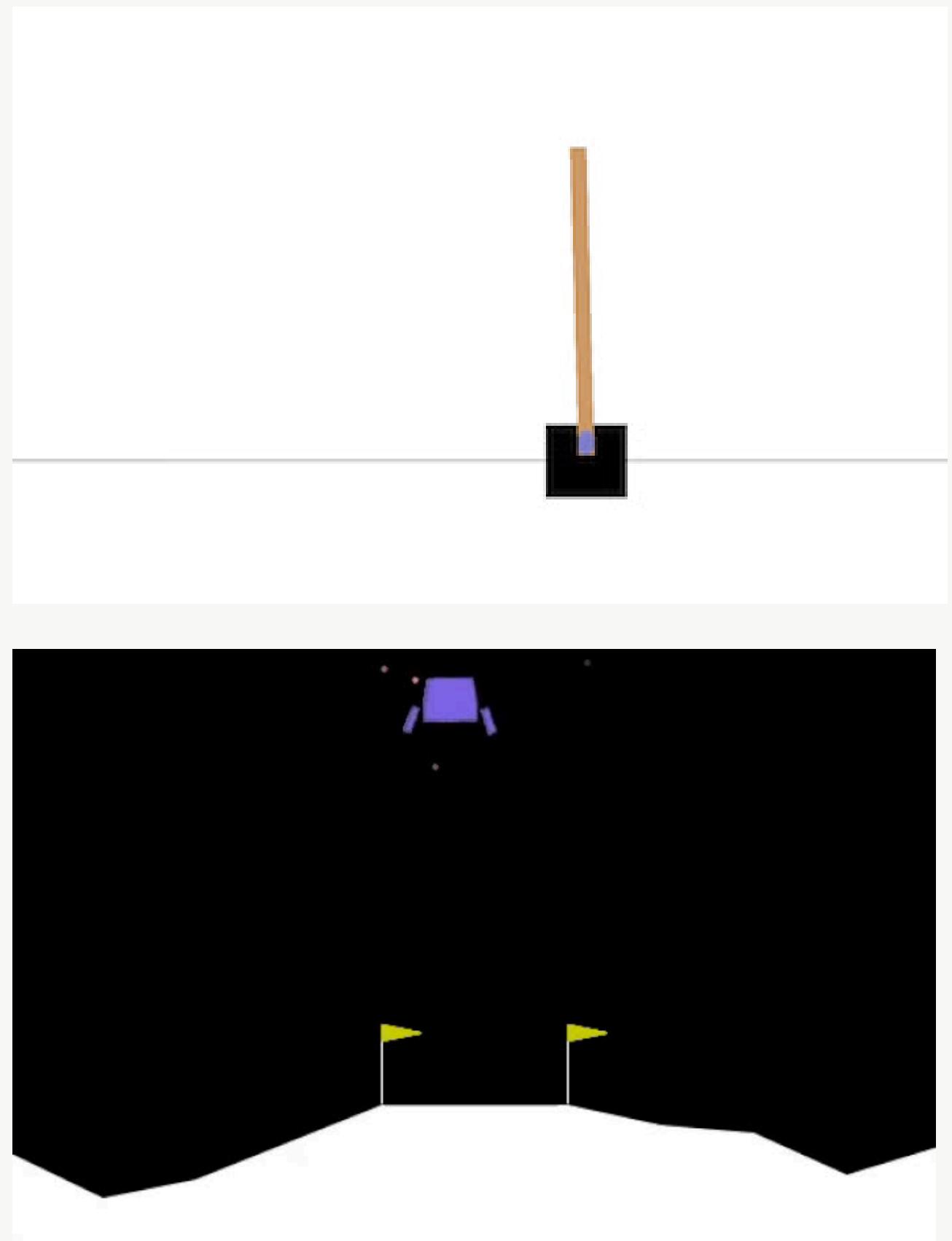
[github.com/KingsleyEbukaChukwuma/Comparing-Value-Based-and-Actor-Critic-Reinforcement-Learning-Methods](https://github.com/KingsleyEbukaChukwuma/Comparing-Value-Based-and-Actor-Critic-Reinforcement-Learning-Methods)

# INTRODUCTION

In this project, we compare a value based method (DQN) and an actor critic method (PPO) on two environments of increasing complexity. We evaluate the impact of reward shaping, hyperparameter tuning, and algorithmic stability using multi-seed evaluation.

## Goals:

- Implement and compare two RL algorithms
- Evaluate performance across environments
- Study reward shaping effects
- Assess robustness using multi-seed evaluation



# ENVIRONMENTS, ALGORITHMS & PARADIGMS

## ■ ENVIRONMENTS

CartPole-v1

- Low-dimensional
- Dense reward
- Stability benchmark

LunarLander-v3

- Higher-dimensional
- Sparse + shaped reward
- Sensitive to reward design

## ■ ALGORITHMS & PARADIGMS

DQN (Value-Based)

- Learns Q-values
- Experience replay + target network
- Sensitive to hyperparameters

PPO (Actor-Critic)

- Direct policy optimization
- Clipped updates for stability
- Robust across environments

# REWARD FUNCTION DESIGN

## ■ REWARD VARIANTS TESTED

- v0: environment default
- v1: mild state-based shaping
- v2: refined shaping in critical states

## ■ METHODOLOGY

- Reward treated as an ablation
- Hyperparameters fixed during reward comparison

```
def step(self, action: Any):
    obs, reward, terminated, truncated, info = self.env.step(action)
    shaped = float(reward)

    env_id = getattr(self.env.unwrapped.spec, "id", "")
    if self.cfg.variant == "v0":
        return obs, shaped, terminated, truncated, info

    if env_id.startswith("CartPole"):
        shaped = self._shape_cartpole(obs, shaped, variant=self.cfg.variant)
    elif env_id.startswith("LunarLander"):
        shaped = self._shape_lunarlander(obs, shaped, variant=self.cfg.variant)

    info = dict(info)
    info["reward_raw"] = float(reward)
    info["reward_shaped"] = float(shaped)
    info["reward_variant"] = self.cfg.variant
    return obs, shaped, terminated, truncated, info
```

```
def _shape_cartpole(obs: np.ndarray, reward: float, variant: str) -> float:
    """
    v1: tiny penalty for large pole angle
    v2: v1 + tiny penalty for cart position away from center
    """
    x, x_dot, theta, theta_dot = obs
    if variant in ("v1", "v2"):
        reward -= 0.01 * abs(theta)
    if variant == "v2":
        reward -= 0.005 * abs(x)
    return float(reward)
```

# HYPERPARAMETER TUNING & SETUP

## ■ TUNING APPROACH

- Random search
- Focused tuning parameters:
  - learning rate
  - discount factor
  - exploration schedule
  - network architecture

```
cfg = load_yaml(args.base_config)
base_kwargs: Dict[str, Any] = dict(cfg.get("model_kwargs") or {})
policy_space: Dict[str, Any] = cfg.get("policy_kwargs_space") or {}
dqn_space: Dict[str, List[Any]] = cfg.get("dqn_space") or {}
ppo_space: Dict[str, List[Any]] = cfg.get("ppo_space") or {}
```

```
for t in range(1, args.n_trials + 1):
    sampled = dict(base_kwargs)
    for k, vals in space.items():
        sampled[k] = sample_from(vals)

    pk = build_policy_kwargs_sample(policy_space)
    if pk:
        sampled["policy_kwargs"] = pk
```

## ■ EVALUATION

- 20–30 evaluation episodes
- Final results: 5 seeds

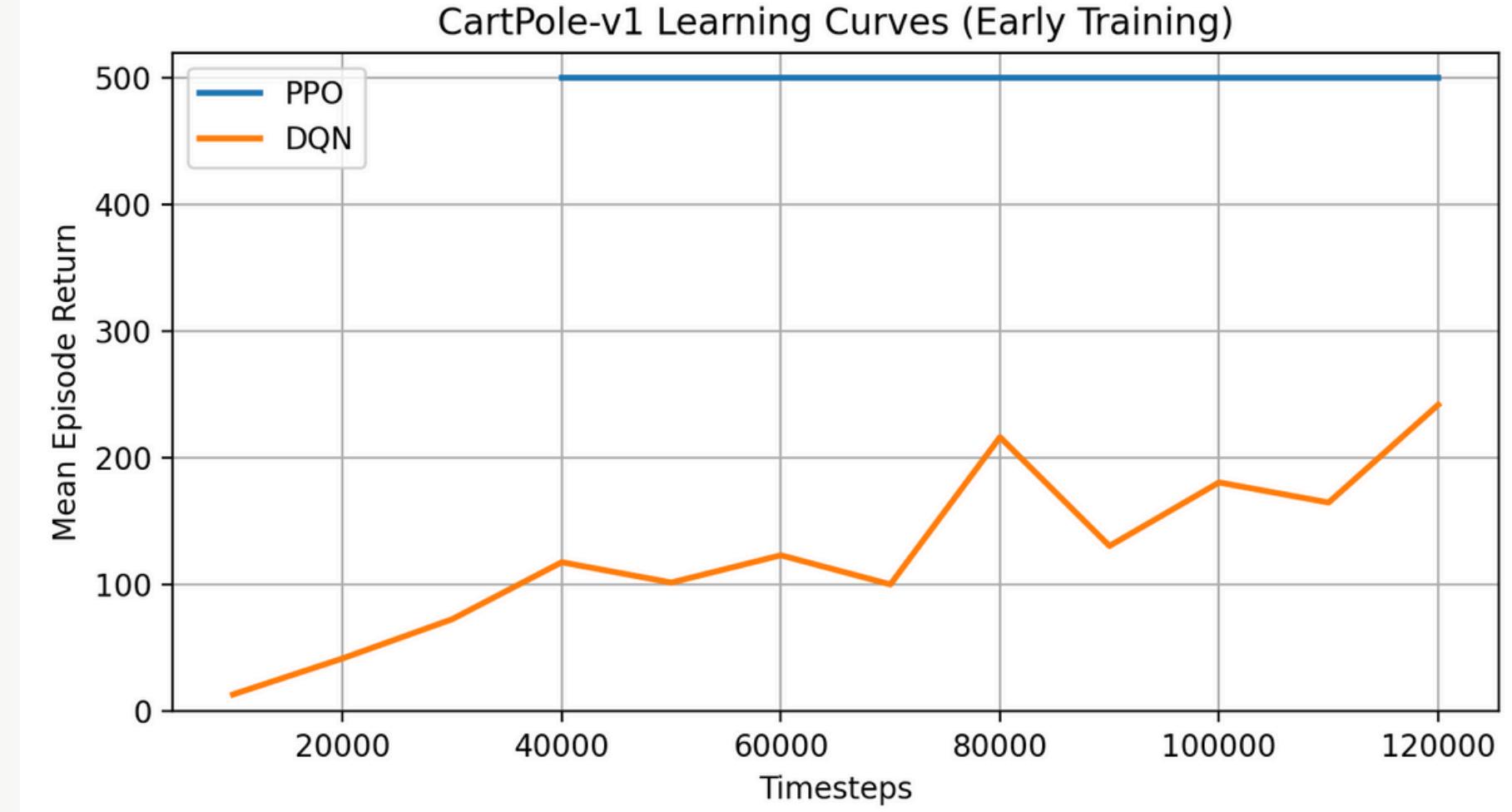
```
mean_r, std_r = evaluate_policy(model, eval_env, n_eval_episodes=args.eval_episodes, deterministic=True)

with open(results_csv, "a", newline="", encoding="utf-8") as f:
    csv.writer(f).writerow([t, float(mean_r), float(std_r), float(train_time), json.dumps(sampled, sort_keys=True, default=str)])

if mean_r > best_mean:
    best_mean = float(mean_r)
    best_kwargs = sampled
    with open(
        os.path.join(
            out_dir,
            f"best_{args.env_id}_{args.algo}_{args.reward}.yaml".replace("/", "-"),
        ),
        "w",
        encoding="utf-8",
    ) as f:
        safe_sampled = sanitize_for_yaml(sampled)
        yaml.safe_dump({"model_kwargs": safe_sampled}, f, sort_keys=False)
```

# RESULTS (CARTPOLE)

Algorithm	Mean ± Std
PPO	500.0 ± 0.0
DQN	343.7 ± 191.4



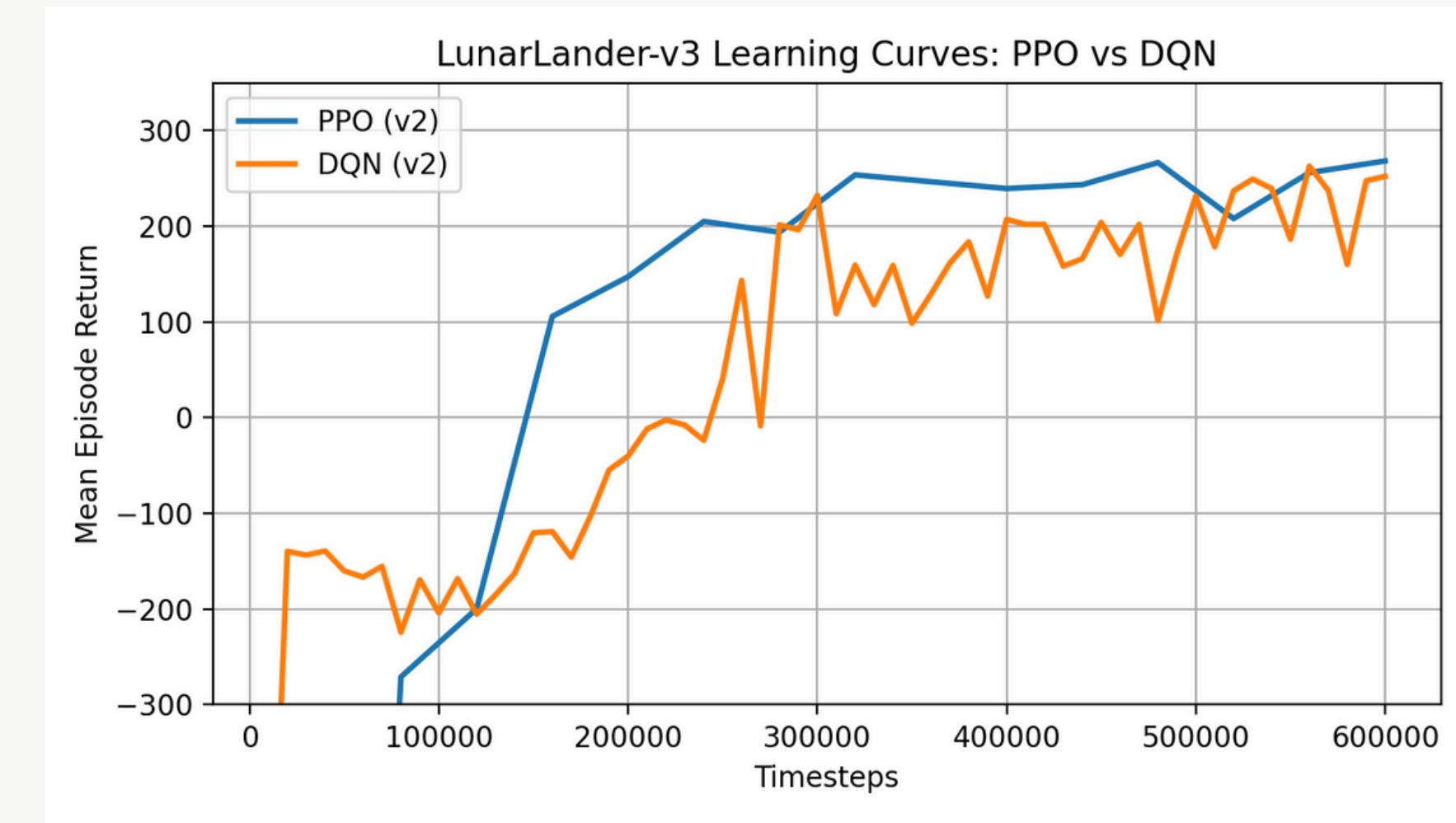
## ■ KEY TAKEAWAY

- PPO solved CartPole consistently
- DQN showed high variance despite tuning

PPO converges early and maintains optimal performance, while DQN exhibits continued variance even with extended training

# RESULTS (LUNARLANDER)

Algorithm	Mean ± Std	Time
PPO (v2)	$264.2 \pm 9.8$	~760 s
DQN (v2)	$245.0 \pm 20.6$	~1450 s



## ■ KEY TAKEAWAY

- PPO more sample-efficient
- Reward shaping significantly improved stability

On LunarLander, PPO (v2) achieves higher return with lower variance and converges faster than DQN (v2).

# CONCLUSIONS

## ■ CHALLENGES

- DQN highly seed sensitive
- Single seed tuning did not generalize
- Longer training improved performance but not robustness

## ■ INSIGHTS

- Actor critic methods are more stable
- Reward shaping matters more in complex environments
- Multi seed evaluation is essential

## ■ FINAL CONCLUSIONS

- PPO consistently outperformed DQN in stability and efficiency
- Reward shaping helped in LunarLander but not in CartPole
- Value based methods require careful tuning and remain sensitive
- Algorithm choice and evaluation methodology matter as much as raw performance.

**THANK YOU FOR YOUR TIME  
AND ATTENTION**