CSC 496 Mid-Project Report

Learning to Play Snake: A Deep Q-Learning Study in a Custom Gym Environment

Gaurvendra Pratap Singh Pundhir

Abstract

I am training agents to play a custom 2-D Snake game using reinforcement learning. The environment is built from scratch with the Gymnasium API and supports both feature and pixel observations. The project proceeds from tabular baselines to Deep Q-Networks (DQN) with replay and target networks. By mid-term, the training/evaluation pipeline, logging, and checkpoints are complete. On the small-board configuration, DQN learns a stable policy; over the last 50 episodes of a representative run (seed 123) the **average return ≈ 11.66**, **median survival ≈ 136.0 steps**, and **return standard deviation ≈ 6.28**. Next, I will add Double- and Dueling-DQN ablations, a modest curriculum, and stronger evaluation across multiple seeds.

---

1. Problem & Contributions

Goal. Learn policies that survive and collect apples efficiently in Snake, avoiding self-collision.

How this project is different.

- **Custom Gym environment** (feature and pixel modes) enabling both tabular and deep methods.
- **Reproducible harness:** YAML configs, seeds, TensorBoard + CSV logging, and checkpoints.
- **Systematic methodology:** tabular baselines → DQN → upcoming ablations (Double/Dueling/PER).
- **Open, re-runnable code** suitable for peer replication and extension.

---

2. Environment Formulation

State space.

- Feature state (current): compact features including head direction, relative food location, and local hazards; optionally frame-stacked.

- Pixel stack (optional): stack of the last k binary planes (snake/food/walls).

Action space. {Up, Down, Left, Right} with a guard against 180° reversals.

Rewards. +1 for apple, −1 for death, small per-step penalty (default −0.01, tuned to −0.003 for the best run). Optional potential-based shaping (progress toward food) is available without altering optimality.

Termination. Collision with boundary or self.

API. reset()->(obs, info), step(a)->(obs', r, terminated, truncated, info) per Gymnasium.

This design satisfies MDP assumptions and scales from tabular methods to function approximation.

---

3. Methods

3.1 Baselines (v1)

Algorithms: Q-Learning / SARSA / Expected-SARSA (tabular).
Policy: ε-greedy with linear decay.
Outcome: Learn simple survival on very small boards; quickly saturate due to state explosion and sparse reward.

3.2 Deep Q-Network (v2)

Network (feature mode): Flatten → 256 ReLU → 256 ReLU → |A| (MLP). (Pixel mode swaps MLP for a small CNN; not used in mid-term runs.)
Stability: Experience replay (uniform), target-network periodic sync, Huber loss, grad-norm clipping.
Exploration: ε decays from 1.0 → 0.05 over a chosen schedule.
Typical hyperparameters: γ = 0.99, batch = 128, buffer = 50–100k, warm-up = 8–10k, target sync = 1–2k, lr = 1e-3 → 5e-4 (best stability), total steps = 150–300k.
Planned ablations (next phase): Double DQN (reduce overestimation), Dueling heads (value/advantage), Prioritized Replay (sample efficiency), and board-size curriculum (8×8 → 10×10 → 12×12).

---

4. Implementation & Reproducibility

Repository: envs/ (custom Gym), agents/ (DQN + Replay), configs/ (YAMLs), train_dqn.py, make_report.py, runs/ (TB + CSV), checkpoints/.

Logging

- TensorBoard scalars: charts/episode_return, charts/episode_length, loss/td_loss.
- Per-episode CSV: runs/episodes.csv with config, seed, episode, return, length, epsilon, loss.
- Summary table: make_report.py aggregates the last K (default 50) episodes per (config, seed) into runs/summary.csv.

Checkpoints: checkpoints/dqn_latest.pt (periodic) and dqn_final.pt (end of run).
Seeding: NumPy and PyTorch seeds set from CLI (--seed).

Quickstart (PowerShell)

- conda create -n RL python=3.11 -y
- conda activate RL
- python -m pip install -r requirements.txt
- python train_dqn.py --config configs\dqn_small.yaml --seed 123
- tensorboard --logdir runs
- python make_report.py

---

5. Experiments

5.1 Configurations

Small board (feature state) — best-performing mid-term setup.
Key values that improved stability: step_penalty: -0.003, lr: 0.0005, target_sync: 2000, eps_decay_steps: 150000.

5.2 Evaluation Protocol

Train with seed(s) per config; log every episode to CSV and TensorBoard.
Report metrics from the last 50 episodes per (config, seed): **average return (↑)**, **median survival steps (↑)**, and **return std (↓)**.
Qualitatively inspect learning curves (EMA-smoothed plots) for upward trends and stability.

---

6. Results (Mid-Term)

6.1 Quantitative (seed 123, last-50 episodes)

Config	S	Avg. Return	Median Survival	Return Std
configs/dqn_small.yaml	1	**11.6628**	**136.0**	**6.2820**

(Derived from runs/summary.csv after running make_report.py on the current runs/episodes.csv.)

6.2 Qualitative

Figure 1 – Episode Return (last ~400 episodes, EMA 0.9).
Shows a clear upward trend; the smoothed curve rises steadily across training.

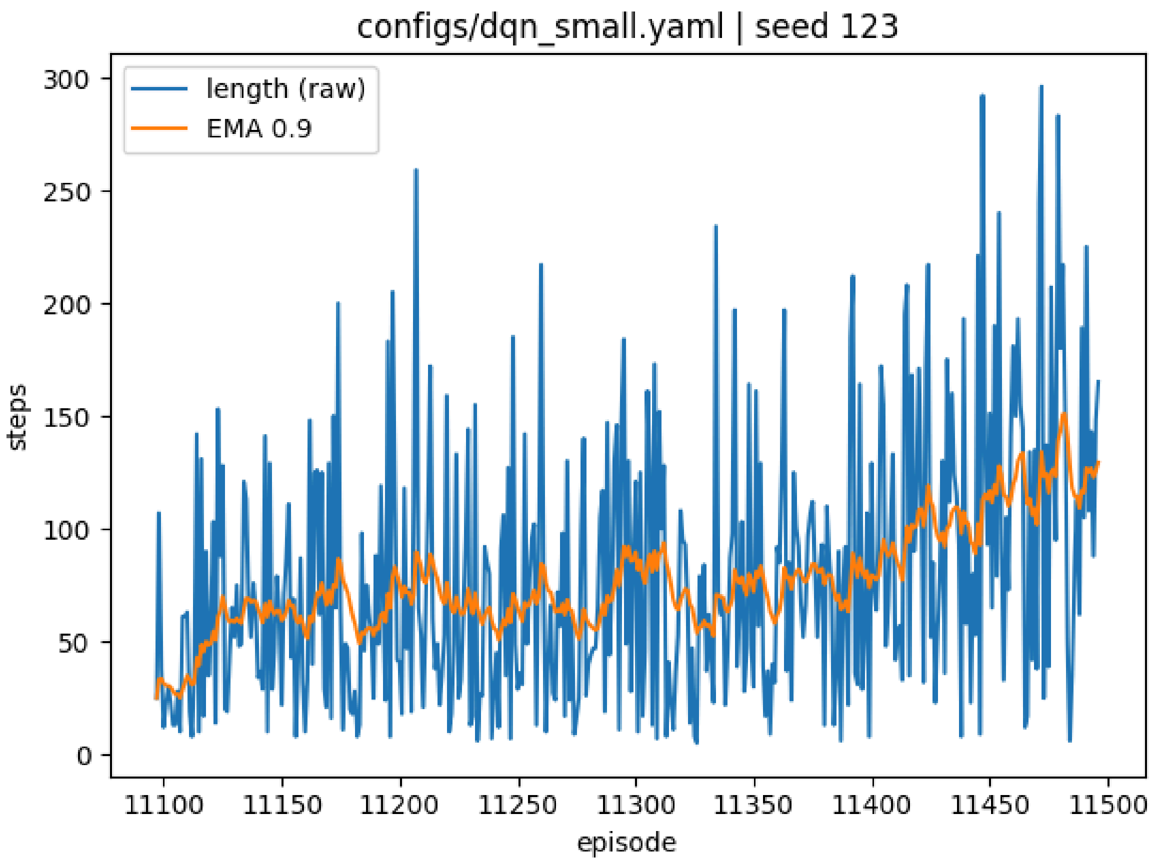
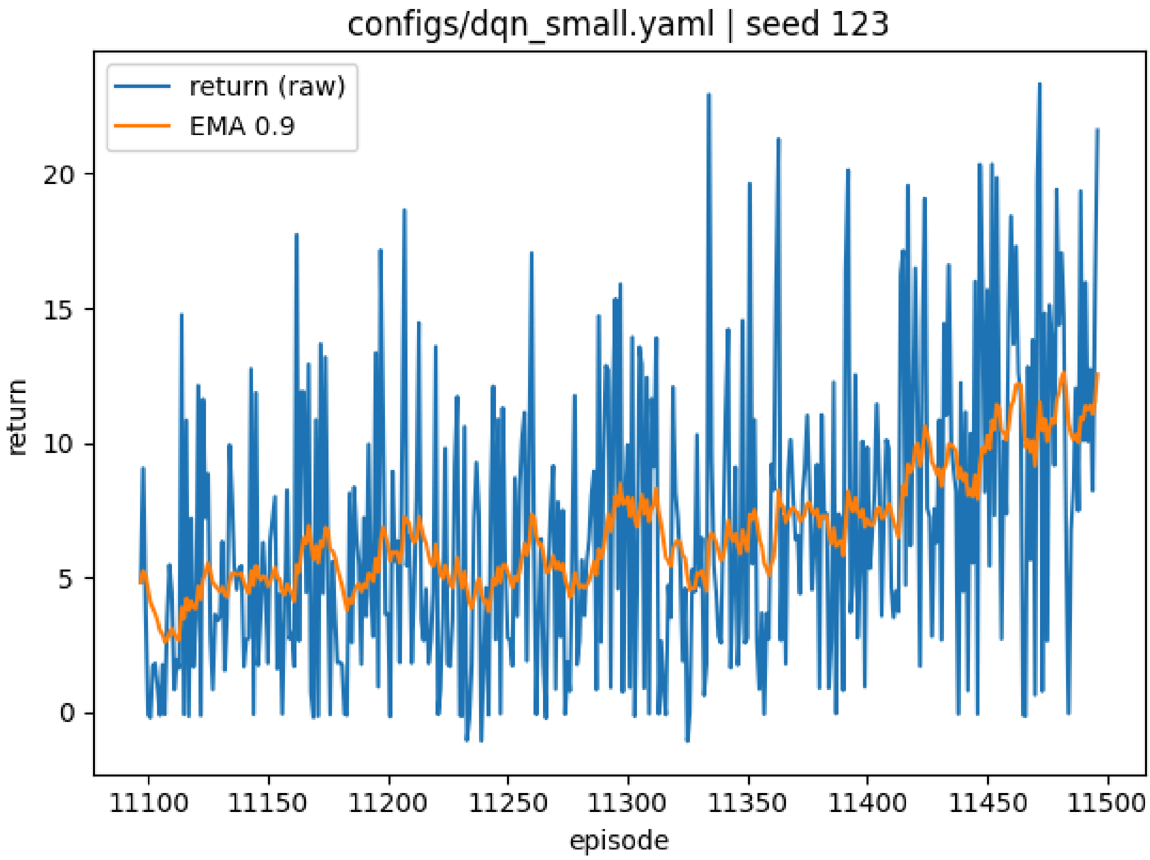


Figure 2 – Episode Length (last ~400 episodes, EMA 0.9).
Median survival increases with frequent long episodes (>100–200 steps) by the end.



---

7. Discussion

Learning achieved. DQN transitions from survival-only behavior to purposeful apple collection on the small board.
What helped. Gentler step penalty, slower ε-decay, slightly lower lr, and longer target-sync interval reduced instability and prevented "wander with cost" failure modes.
Variance. Return variance remains non-trivial (std ≈ 6.28) due to stochastic starts and sparse reward; averaging across seeds and/or Double DQN should tighten dispersion.
Limitations. Current results use feature state; pixel stacks and generalization to larger boards are pending. Baselines were used for calibration but not extensively profiled in mid-term plots.

---

8. Risks & Mitigations

Risk	Impact	Mitigation

Overestimation bias	Noisy Q-values, unstable returns	**Double DQN** target selection
Sparse rewards	Long plateaus, looping behavior	Potential-based shaping; curriculum; tuned step penalty
Seed variance	Unreliable single-run conclusions	Report across ≥3 seeds; fixed eval seeds
Scaling to larger boards	Slower learning	Curriculum (8×8→10×10→12×12); PER; dueling heads

---

9. Next-Phase Plan (2–3 weeks)

Algorithms. Implement **Double DQN**; add **Dueling** head; consider **Prioritized Replay** with β-annealing.
Training. Curriculum 8×8 → 10×10 → 12×12; grid search (step penalty, ε schedule, lr, target_sync).
Reporting. Standardized evaluation (≥3 seeds), learning curves per config, ablation tables, demo GIF.

---

10. Timeline (updated)

Date	Milestone	Deliverables
Oct 17	Proposal & Literature Review	Proposal doc + references
Nov 10	**Mid-Project**	Working env + DQN results + this report + plots

Nov 24	Ablations Complete	Double/Dueling/PER + curriculum; updated tables/plots
Dec 5	Final Report	6–8 pp paper + code + demo GIF/video

---

11. References

Mnih, V. et al. Human-level control through deep reinforcement learning. **Nature**, 2015.
Gymnasium documentation — https://gymnasium.farama.org (API reference).
Hessel, M. et al. Rainbow: Combining Improvements in Deep RL. **AAAI**, 2018.

---

12. Additional Figures (TensorBoard, seed 123)

Figure 3 — Episode Return (TB export). Return rises from near 0 to **>4–5** by ~11.5k episodes; EMA shows steady improvement.

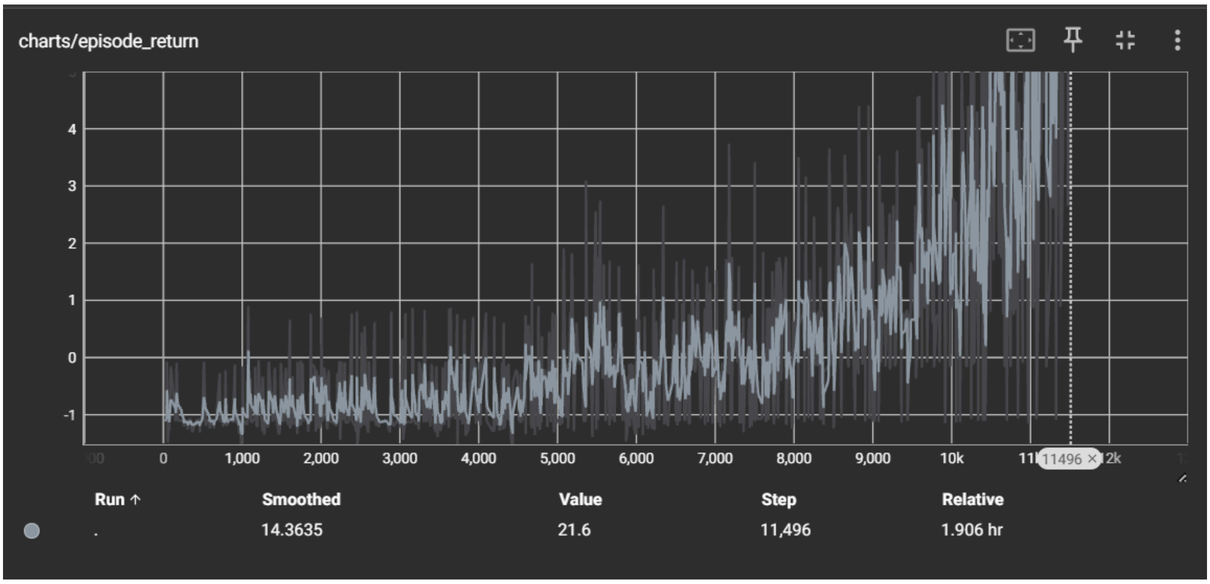
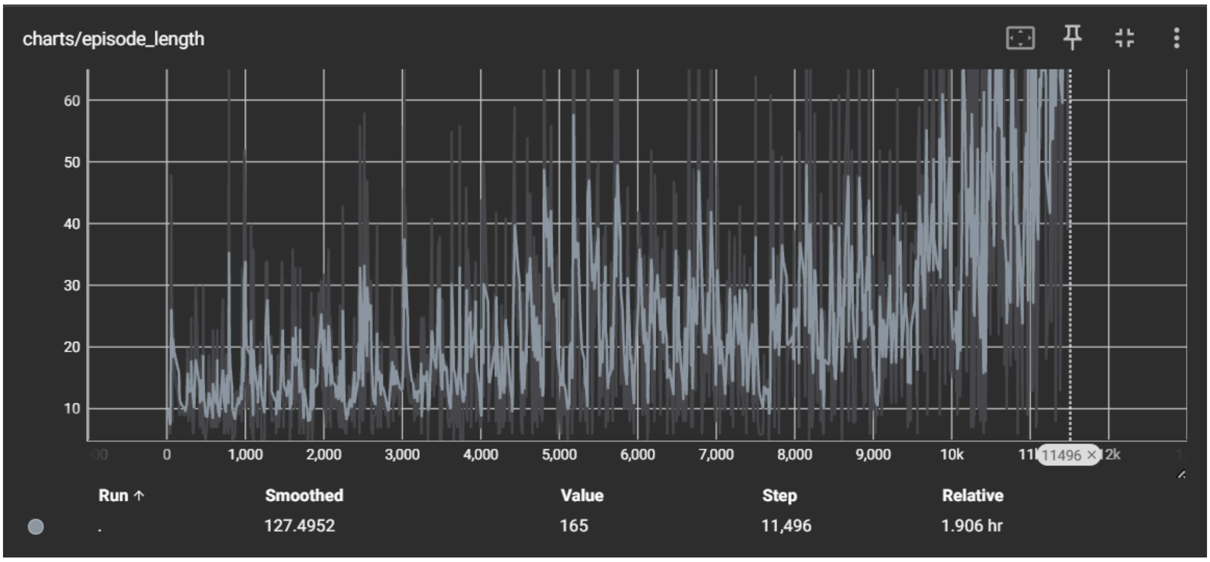


Figure 4 — Episode Length (TB export). Survival increases with frequent long episodes (50–150+ steps), consistent with improved navigation.



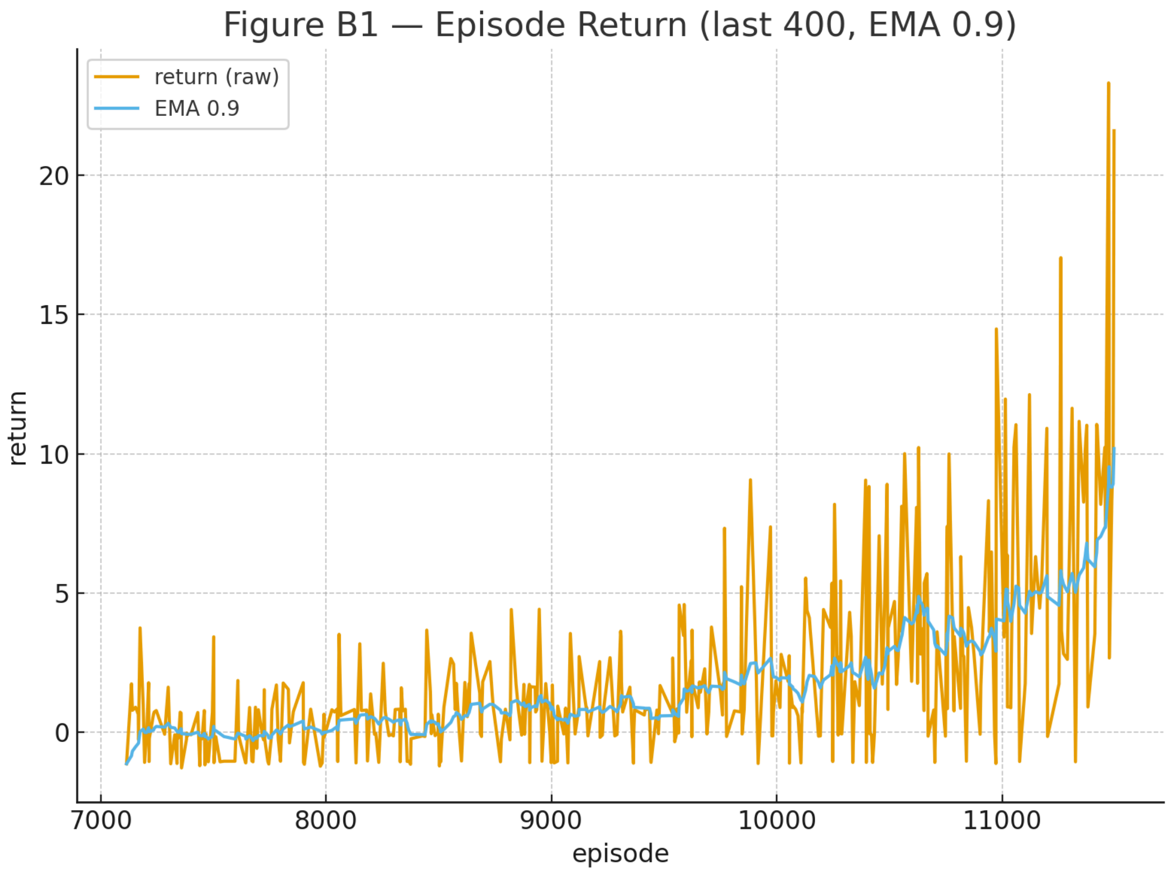
Appendix A — Commands Used

- # Train (small board)
- python train_dqn.py --config configs\dqn_small.yaml --seed 123
-
- # Live metrics
- tensorboard --logdir runs
-
- # Build table & plots
- python make_report.py
- python make_plots.py --csv runs\episodes.csv --outdir submission --last 400 --smooth 0.9
-
- # Collect artifacts
- Copy-Item runs\summary.csv submission\
- Copy-Item runs\episodes.csv submission\episodes_full.csv
- Copy-Item configs\dqn_small.yaml submission\config_used.yaml
- Copy-Item checkpoints\dqn_final.pt submission\ -ErrorAction SilentlyContinue
- Copy-Item checkpoints\dqn_latest.pt submission\ -ErrorAction SilentlyContinue
- Compress-Archive -Path submission* -DestinationPath submission_midproject.zip -Force

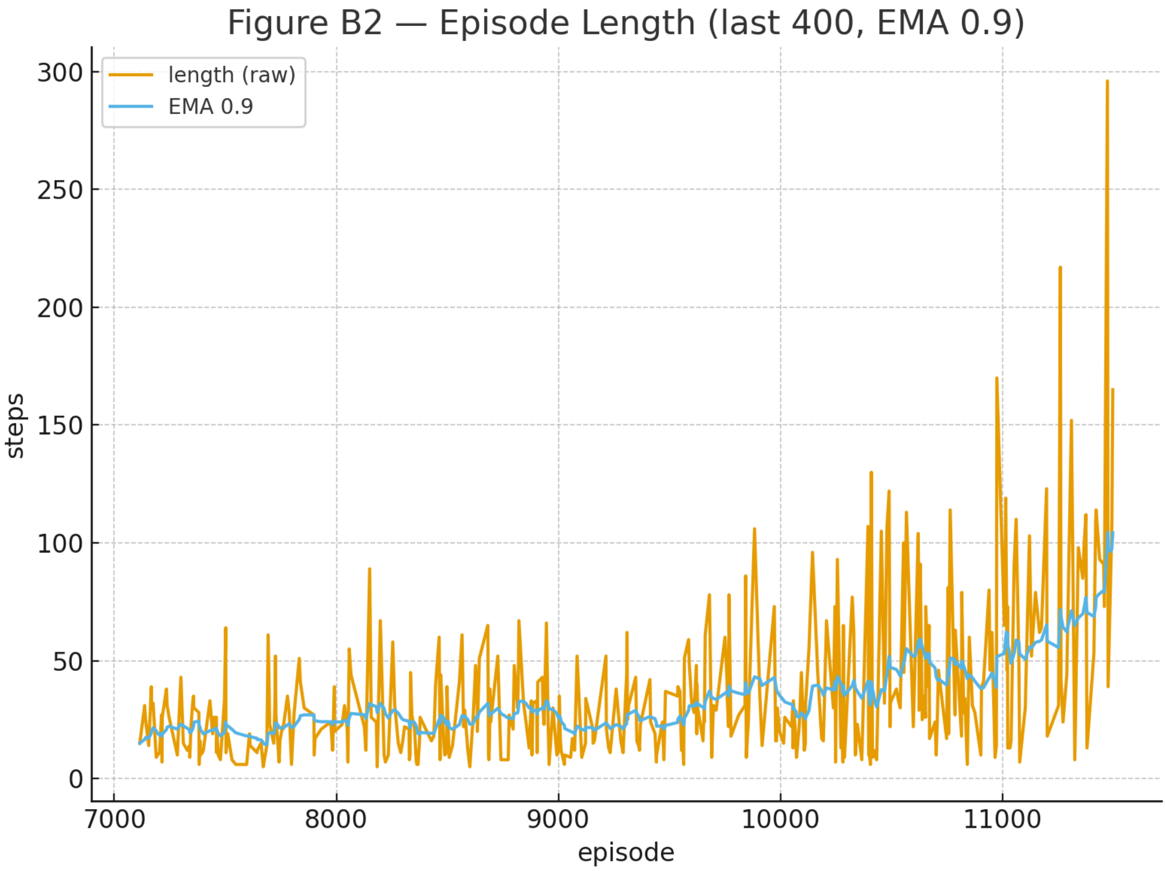
---

Appendix B — Raw Plots

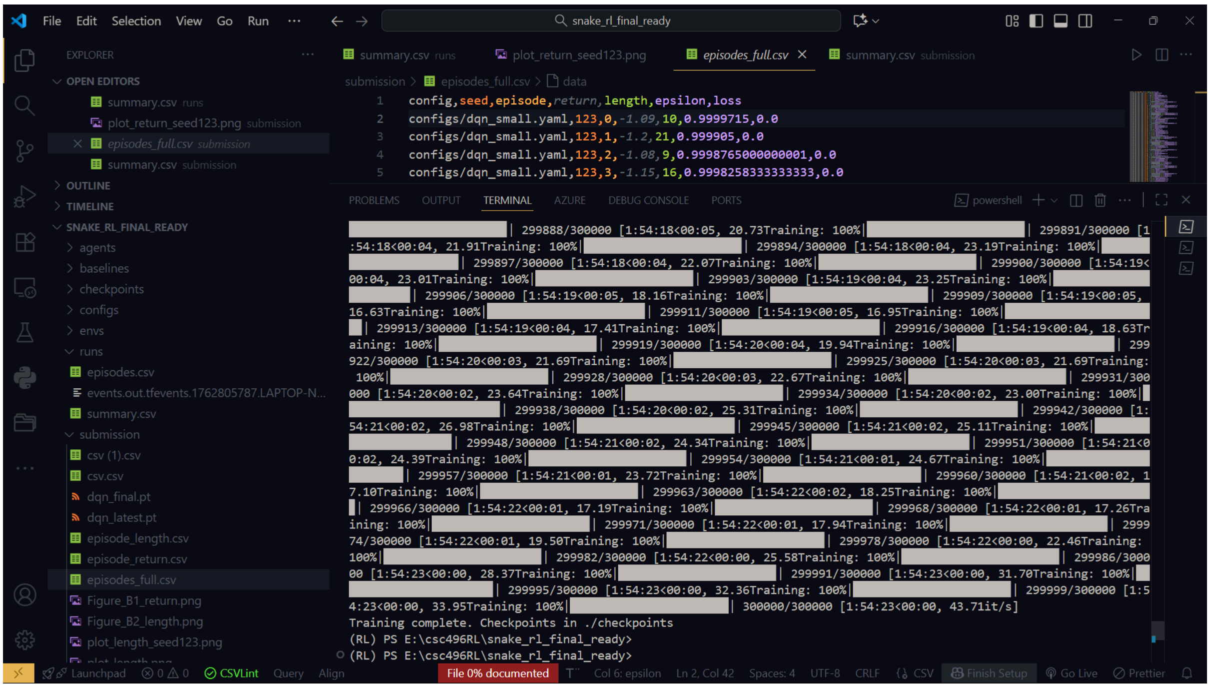
- **Figure B1 — Episode Return (last 400, EMA 0.9).**



- **Figure B2 — Episode Length (last 400, EMA 0.9).**



Project Snapshot



- There isn’t a git commit history - I haven’t put it on github yet as I’m the sole collaborator. So, I didn’t had any need to branch the project out - for the final submission - I’ll put the files on github, and share the link