**Function Approximation in Reinforcement Learning**

**Project Overview**

This project implements and compares three reinforcement learning algorithms—DQN, REINFORCE, and A2C—using neural networks as function approximators on the CartPole-v1 environment. The aim is to highlight the impact of algorithmic approach on learning performance and stability when using the same network architecture, and to analyze results statistically across multiple independent runs.

**Project Structure**

**1. Main Code (RL-HW2.ipynb)**

* Setup: Prepares the CartPole-v1 environment using Gymnasium, sets random seeds, and configures all agents to use a shared neural network architecture.
* Algorithm Implementations:
  + DQN: Q-value estimation using experience replay and target networks. Implements early stopping on convergence.
  + REINFORCE: Monte Carlo policy gradient using returns, with returns normalization for stability.
  + A2C: Actor-Critic learning with shared policy and value networks; advantage estimation is used to reduce variance.
* Hyperparameter Studies: Hyperparameters (learning rates, batch size, buffer size) are tuned and optimized for Colab CPU runtime.
* Experiment Loop: Each algorithm is evaluated over 50 independent runs (with different seeds) to provide statistically meaningful learning curves and convergence data.
* Statistical Analysis & Visualization: Computes the mean and standard deviation of rewards across runs. Generates a comparison plot using matplotlib.

**2. Results and Figures**

* results/learning\_curves\_comparison.png: Learning curve comparison of DQN, REINFORCE, and A2C (mean ± one std).
* results/best\_hyperparameters.json: Saved dictionary of tuned hyperparameters for reproducibility.
* results/statistical\_summary.json: JSON file with summary statistics (final mean reward, stdev, episodes to converge, etc).
* results/dqn\_run\_0.pkl to results/a2c\_run\_49.pkl: Raw results per run, per algorithm. Each is a pickled list of episode rewards.

**3. Codebase Files**

* RL-HW2.ipynb: The main Colab notebook, fully executable and includes all code, experiments, and plotting setup.
* README.md: This file, providing structure and usage notes.

**Installation & Setup**

1. Clone or upload the notebook to Google Colab.
2. Dependencies:
   * Gymnasium, torch, matplotlib, numpy
   * Install in Colab with:

text

!pip install gymnasium torch matplotlib

1. Adjust experiment count/batch size as needed for runtime constraints. The code is optimized for 50 runs per algorithm for smooth Colab CPU usage.

**Running the Experiments**

1. Open and run all cells in RL-HW2.ipynb.
2. Results are saved in the results/ folder.
3. Plot and stats:
   * The notebook will automatically display the comparative learning curve at the end.
   * Key statistics (means, stds, episodes to converge) are printed and also saved to statistical\_summary.json.

**Experiment Workflow**

* Each algorithm (DQN, REINFORCE, A2C) is trained using the shared network over 50 independent random seeds.
* Hyperparameters are set for best performance on CPU within runtime limits.
* For each algorithm, the episode rewards are saved per run and aggregated for statistical analysis.

**Output**

* Figures: learning\_curves\_comparison.png holds the mean learning curves with ±1 standard deviation shaded.
* Data: Individual run rewards as .pkl files, and aggregate stats in JSON files.
* Hyperparameter log: Best experimental settings in best\_hyperparameters.json.

**Summary**

By running the provided notebook, you systematically compare the function approximation capabilities of three core RL algorithms using neural networks. The produced plots, raw results, and numerical summaries support deeper assignment analyses and can be used in your report as figures and tables.

**Example Figures**

* Learning Curve Comparison: Composite plot with DQN/REINFORCE/A2C (mean ± std).
* Statistical Table: Mean, stdev, and convergence info for each algorithm.

All required outputs and analysis ready for assignment submission.