Reinforcement Learning Summative Assignment Report

Student Name: Jallah Sumbo

Video Recording: https://loom.com/share/folder/6a56b78e5f5444c3a9abb9ffc853b3b2
GitHub Repository: https://github.com/jsumbo/Jallah_Sumbo_rl_summative

1. Project Overview

This project presents an investigation into the application of reinforcement learning algorithms within the context of entrepreneurial education for Liberian secondary school students. The research addresses a critical gap in practical, experiential learning methodologies by developing a custom simulation environment that mirrors real-world entrepreneurial challenges faced by young Entrepreneurs. The project implements and compares four distinct reinforcement learning algorithms: **Deep Q-Network (DQN)**, **REINFORCE**, **Proximal Policy Optimization (PPO)**, and **Actor-Critic** methods, evaluating their effectiveness in learning optimal strategies within this environment.

The core problem addressed by this research stems from the disconnect between traditional educational approaches and the practical skills required for entrepreneurial success in developing economies, particularly in post-conflict societies like Liberia. Traditional secondary education often lacks the hands-on, decision-making components necessary to prepare students for the complexities of modern entrepreneurship and workforce readiness. This project bridges that gap by creating an interactive simulation environment where agents can learn entrepreneurial strategies through trial and error, mimicking the learning process that human students would undergo in real-world scenarios.

The approach taken in this research is fundamentally interdisciplinary, combining insights from educational psychology, economic development theory, and advanced machine learning techniques. The custom environment, termed the "Liberian Entrepreneurship Simulation" (LES), incorporates specific challenges and opportunities relevant to the Liberian context, including resource constraints, market volatility, limited access to capital, and the need for skill development across multiple domains. The environment serves as a testbed for comparing different reinforcement learning paradigms and their ability to discover effective entrepreneurial strategies.

The significance of this work extends beyond the technical implementation of reinforcement learning algorithms. It represents a novel approach to understanding how artificial intelligence can be leveraged to model and optimize learning processes in educational contexts, particularly in developing nations where traditional educational resources may be limited. The insights gained from this research have implications for both the design of AI-powered educational tools and the broader understanding of how different learning algorithms perform in complex, multi-objective environments that

mirror real-world decision-making scenarios.

The methodology employed in this project follows a rigorous experimental design, beginning with the development of a custom Gymnasium environment that accurately represents the entrepreneurial landscape in Liberia. This environment incorporates multiple types of locations (markets, schools, banks, suppliers), dynamic market conditions, and a comprehensive skill development system that reflects the multifaceted nature of entrepreneurial competence. The implementation of four distinct reinforcement learning algorithms allows for a comprehensive comparison of value-based versus policy-based approaches, as well as the examination of how different exploration strategies and learning mechanisms perform in this specific domain.

2. Environment Description

2.1 Agent(s)

The primary agent in this Simulation represents a secondary school student in Liberia who is learning to navigate the complexities of entrepreneurship within their local economic context. This agent embodies the characteristics and constraints typical of young Liberians seeking to develop business acumen and practical skills for economic empowerment. The agent begins with minimal resources, limited skills, and basic knowledge of market dynamics, reflecting the realistic starting point of most secondary school students in Liberia.

The agent's capabilities are designed to mirror the learning journey of a real student, starting with fundamental limitations that can be overcome through strategic decision-making and skill development. Initially, the agent possesses only $100 in starting capital, representing the limited financial resources typically available to entrepreneurs in developing economies. The agent's skill set encompasses five critical entrepreneurial competencies: **business planning**, **market analysis**, **financial management**, **leadership**, and **innovation**. Each skill begins at a minimal level (0.1 on a normalized scale) and can be improved through deliberate practice and learning activities.

The agent's limitations are carefully designed to reflect real-world constraints faced by young entrepreneurs in Liberia. These include restricted access to capital, limited initial market knowledge, and the need to balance multiple competing objectives simultaneously. The agent cannot perform certain actions effectively without being in appropriate locations (for example, studying requires being at a school, applying for loans requires being at a bank), which mirrors the geographic and institutional constraints present in real-world entrepreneurial ecosystems.

The agent's learning process is designed to be holistic, requiring the development of both hard skills (financial management, market analysis) and soft skills (leadership, innovation) while navigating dynamic market conditions. This multi-dimensional approach to agent design ensures that the learning algorithms must develop sophisticated strategies that balance immediate needs with long-term development goals, closely paralleling the

decision-making processes required of real entrepreneurs.

2.2 Action Space

The action space of the Simulation consists of 14 discrete actions, carefully designed to represent the full spectrum of decisions available to a young entrepreneur. These actions are divided into two primary categories: **movement actions** and **interaction actions**, each serving distinct purposes in the agent's learning and development process.

The movement actions (actions 0-7) enable the agent to navigate through the 10x10 grid environment, with each action corresponding to one of eight cardinal and intercardinal directions: North, Northeast, East, Southeast, South, Southwest, West, and Northwest. This movement system allows the agent to explore the environment and position itself strategically relative to different types of locations. The movement mechanics are designed to be realistic, preventing the agent from moving outside the grid boundaries and providing small rewards for valid movements while penalizing attempts to move beyond the environment's limits.

The interaction actions (actions 8-13) represent the core entrepreneurial activities that the agent can perform. Action 8 (Study/Learn) allows the agent to improve its skills when located at educational institutions, reflecting the importance of continuous learning in entrepreneurial success. This action requires financial investment ($10) and results in the improvement of a randomly selected skill, emphasizing the cost-benefit analysis inherent in educational decisions.

Action 9 (**Apply for Loan**) enables the agent to seek additional capital from banking institutions, with approval probability based on the agent's current business level and financial management skills. This mechanic reflects the real-world challenges faced by young entrepreneurs in accessing credit and the importance of building creditworthiness over time.

Action 10 (**Buy Inventory**) allows the agent to purchase products from suppliers, converting financial capital into inventory that can later be sold for profit. The cost-effectiveness of this action depends on the agent's ability to subsequently sell the inventory at favorable prices, introducing elements of supply chain management and inventory optimization.

Action 11 (**Sell Products**) represents the core revenue-generating activity, available only at market locations. The success of sales depends on multiple factors including market demand, competition levels, customer satisfaction, and the agent's market analysis skills. This action provides the highest potential rewards but requires careful timing and strategic positioning.

Actions 12 and 13 (**Market Research and Improve Customer Service**) represent strategic investments in long-term business development. Market Research enhances the agent's understanding of market dynamics and can improve market demand, while Customer Service improvements increase customer satisfaction levels, leading to better sales

performance over time.

2.3 State Space

The state space of the Simulation is represented as an 11-dimensional vector of normalized values, each ranging from 0.0 to 1.0. This comprehensive state representation captures all relevant information about the agent's current situation, market conditions, and capabilities, providing the reinforcement learning algorithms with sufficient information to make informed decisions.

The first two dimensions represent the agent's spatial position within the 10x10 grid environment, normalized to the range [0,1]. These coordinates (agent_x, agent_y) allow the algorithms to understand the agent's current location relative to important landmarks such as markets, schools, banks, and suppliers. The spatial component of the state space is crucial for enabling the agent to develop location-aware strategies and understand the importance of positioning in entrepreneurial success.

The third dimension represents the agent's current financial resources (money), normalized to a maximum value of $1000. This normalization ensures that the financial component remains within the standard range while still allowing for meaningful differentiation between different wealth levels. The financial state is perhaps the most critical component, as it determines the agent's ability to take various actions and serves as a primary indicator of entrepreneurial success.

The fourth dimension captures the agent's business development level (business_level), normalized to a scale where 0 represents the initial idea stage, and 1.0 represents a fully established business. This progression through business maturity stages (Idea → Startup → Growing → Established) reflects the natural evolution of entrepreneurial ventures and provides the algorithms with a clear measure of long-term progress.

Dimensions five through seven represent dynamic market conditions that affect all agents equally but change over time. Market demand (market_demand) reflects the overall customer appetite for products, competition level (competition_level) indicates the intensity of market competition, and economic stability (economic_stability) represents broader economic conditions. These dynamic elements ensure that the environment remains challenging and requires adaptive strategies.

The remaining dimensions capture the agent's capabilities and knowledge. Market knowledge (market_knowledge) represents the agent's understanding of market dynamics, customer satisfaction reflects the quality of the agent's customer relationships, inventory indicates the quantity of products available for sale, and average skills represents the mean of all five entrepreneurial skill categories.

This comprehensive state representation ensures that reinforcement learning algorithms have access to all information necessary for making optimal decisions while maintaining a manageable dimensionality that prevents the curse of dimensionality from significantly impacting learning performance.

2.4 Reward Structure

The reward structure of the Simulation is designed to encourage behaviors that align with successful entrepreneurial practices while discouraging inefficient or counterproductive actions. The reward system operates on multiple levels, providing immediate feedback for individual actions, intermediate rewards for achieving specific milestones, and terminal rewards for overall episode outcomes.

At the action level, the reward structure provides clear incentives for productive entrepreneurial behaviors. Selling products yields the highest immediate reward (+10), reflecting the central importance of revenue generation in business success. This high reward is modulated by factors such as market conditions, customer satisfaction, and the agent's market analysis skills, ensuring that successful sales require strategic preparation rather than random action selection.

Educational activities receive substantial positive rewards (+5 for successful studying), emphasizing the importance of skill development in long-term entrepreneurial success. However, these rewards are conditional on having sufficient resources and being in appropriate locations, teaching the algorithms to balance immediate costs with long-term benefits. The reward structure for learning activities is designed to encourage continuous skill development while respecting resource constraints.

Strategic business activities such as market research (+3) and customer service improvement (+4) receive moderate positive rewards, reflecting their importance in building sustainable competitive advantages. These activities require upfront investment but contribute to improved performance in future sales activities, encouraging the development of sophisticated multi-step strategies.

Access to capital through loan applications provides moderate rewards (+3) when successful, but the probability of success depends on the agent's demonstrated competence and business development level. This mechanic teaches the algorithms about the importance of building credibility and managing financial relationships over time.

Movement actions receive small positive rewards (+0.1) for valid moves and negative rewards (-0.5) for invalid attempts, providing basic navigation incentives while preventing the algorithms from wasting computational resources on impossible actions. These small but consistent rewards help guide exploration behavior without overwhelming the more significant strategic rewards.

The reward structure includes a small negative reward (-0.1) for each time step, creating pressure for efficiency and encouraging the algorithms to develop strategies that achieve objectives quickly rather than engaging in aimless exploration. This time penalty is calibrated to be significant enough to encourage efficiency without overwhelming the positive rewards for productive actions.

Terminal rewards provide strong incentives for overall episode success or failure. Achieving

the success condition (business level 3 with at least $500) yields a substantial bonus (+100), while bankruptcy (money ≤ 0) results in a severe penalty (-50). These terminal rewards ensure that the algorithms focus on sustainable long-term strategies rather than short-term optimization that might lead to eventual failure.

2.5 Environment Visualization

The environment visualization system uses Pygame to create an intuitive and informative graphical representation of the simulation state. The visualization presents the 10x10 grid environment as a clear, color-coded interface that allows for easy interpretation of the agent's current situation and available opportunities.

The visual design uses distinct color avatars to represent different types of locations within the environment. Markets are displayed in green, symbolizing growth and opportunity for revenue generation. Schools appear in blue, representing knowledge and learning opportunities. Banks are shown in yellow, indicating financial institutions and capital access. Suppliers are rendered in orange, representing the supply chain and inventory acquisition opportunities.

The agent itself is represented as a red circle positioned within the grid, providing clear visibility of its current location relative to all available resources and opportunities. The agent's visual representation updates in real-time as it moves through the environment, allowing observers to track the learning process and understand the spatial strategies being developed by different algorithms.

The visualization system captures frames during both training and evaluation phases, enabling the creation of animated sequences that demonstrate the learning process over time. These visual records provide valuable insights into how different algorithms approach exploration, exploitation, and strategic planning within the environment.

The grid-based layout with clear location markers serves both functional and educational purposes. From a functional perspective, it provides immediate feedback about the agent's position and available actions. From an educational standpoint, it creates an intuitive representation of the entrepreneurial ecosystem that would be easily understood by students and educators using the system for learning purposes.

The visualization includes subtle visual cues such as grid lines and location boundaries that help distinguish between different areas without cluttering the interface. The color scheme is designed to be accessible and meaningful, with colors chosen to have intuitive associations with their respective functions (green for markets/money, blue for education, yellow for banking, orange for supply).

3. Implemented Methods

3.1 Deep Q-Network (DQN)

Deep Q-Network (DQN)

The Deep Q-Network implementation represents a sophisticated value-based reinforcement learning approach specifically adapted for the discrete action space of the entrepreneurial simulation environment. The DQN algorithm combines the theoretical foundations of Q-learning with the representational power of deep neural networks, enabling the agent to learn complex value functions that map states to action values in the high-dimensional state space of the entrepreneurial environment.

The neural network architecture employed in this implementation consists of a four-layer fully connected network designed to balance representational capacity with computational efficiency. The input layer accepts the 11-dimensional normalized state vector, followed by three hidden layers of 128 neurons each, and concludes with an output layer containing 14 neurons corresponding to the available actions. Each hidden layer employs ReLU activation functions to introduce non-linearity, while dropout layers with a 0.2 probability provide regularization to prevent overfitting during the training process.

The experience replay mechanism forms a crucial component of the DQN implementation, addressing the temporal correlation and non-stationarity issues inherent in reinforcement learning. The replay buffer maintains a capacity of 10,000 experiences, storing tuples of (state, action, reward, next_state, done) that represent the agent's interactions with the environment. During training, random batches of 32 experiences are sampled from this buffer, breaking the temporal correlations that could lead to unstable learning and enabling more efficient use of collected experience data.

The target network mechanism provides stability to the learning process by maintaining a separate copy of the Q-network that is updated less frequently than the main network. The target network parameters are copied from the main network every 100 training steps, providing stable target values for the Q-learning updates. This approach significantly reduces the moving target problem that can cause instability in value-based methods.

The exploration strategy employs an epsilon-greedy policy with exponential decay, beginning with epsilon = 1.0 (pure exploration) and decaying by a factor of 0.995 after each episode until reaching a minimum value of 0.01. This exploration schedule ensures adequate exploration during early training while gradually shifting toward exploitation as the agent's knowledge improves. The decay rate was carefully tuned to balance the exploration-exploitation trade-off within the context of the entrepreneurial environment's complexity.

The training process utilizes the Adam optimizer with a learning rate of 1e-3, chosen through empirical evaluation to provide stable convergence without overshooting optimal solutions. The loss function employs mean squared error between predicted Q-values and target Q-values computed using the Bellman equation with a discount factor (gamma) of 0.99, emphasizing long-term rewards while maintaining sensitivity to immediate feedback.

Gradient clipping with a maximum norm of 1.0 prevents exploding gradients that could destabilize the learning process, particularly important given the complex reward structure of the entrepreneurial environment. The implementation includes comprehensive logging of training metrics, including episode rewards, loss values, and epsilon decay, enabling detailed analysis of the learning process and identification of potential issues.

3.2 Policy Gradient Method ([REINFORCE/PPO/A2C])

REINFORCE

The REINFORCE algorithm implementation represents a pure policy gradient approach that directly optimizes the policy parameters to maximize expected cumulative rewards. This Monte Carlo policy gradient method provides an unbiased estimate of the policy gradient, making it particularly suitable for environments with complex reward structures where value function approximation might introduce bias.

The policy network architecture mirrors the DQN implementation in terms of layer structure but differs fundamentally in its output representation. The network consists of an input layer accepting the 11-dimensional state vector, two hidden layers of 128 neurons each with ReLU activation and dropout regularization, and an output layer with 14 neurons representing action probabilities. The final layer employs a softmax activation function to ensure valid probability distributions over the action space.

The policy gradient computation follows the classic REINFORCE algorithm, utilizing the log-probability of selected actions weighted by discounted returns. For each episode, the algorithm computes returns $G_t = \Sigma(\gamma^k * r_{t+k})$ for each time step t, where $\gamma = 0.99$ represents the discount factor. These returns are normalized using z-score normalization to reduce variance and improve learning stability, a crucial modification given the high variance inherent in policy gradient methods.

The gradient estimation process involves computing the log-probability of each action taken during the episode and weighting it by the corresponding return. The policy loss is formulated as the negative sum of log-probabilities weighted by normalized returns, following the standard policy gradient theorem. This formulation ensures that actions leading to higher returns receive stronger positive reinforcement, while actions associated with poor outcomes are discouraged.

To address the high variance problem characteristic of REINFORCE, the implementation incorporates several variance reduction techniques. Return normalization helps stabilize learning by ensuring that the magnitude of policy updates remains consistent across different episodes. Additionally, gradient clipping with a maximum norm of 1.0 prevents excessively large parameter updates that could destabilize the learning process.

The training procedure processes complete episodes before updating the policy parameters, adhering to the Monte Carlo nature of the algorithm. This approach ensures unbiased gradient estimates but requires the agent to complete full episodes before

learning can occur. The Adam optimizer with a learning rate of 1e-3 provides adaptive learning rates for different parameters, helping to navigate the complex parameter landscape of the policy network.

The implementation includes entropy regularization with a coefficient of 0.01 to encourage exploration by preventing the policy from becoming overly deterministic too quickly. This regularization term adds the negative entropy of the action distribution to the loss function, promoting policies that maintain some level of randomness and continue exploring the action space throughout training.

Proximal Policy Optimization (PPO)

The Proximal Policy Optimization implementation represents a state-of-the-art policy gradient method that addresses many of the stability and sample efficiency issues present in earlier policy gradient algorithms. PPO combines the benefits of policy gradient methods with improved stability through the use of a clipped surrogate objective function that prevents excessively large policy updates.

The actor-critic architecture employed in this PPO implementation utilizes a shared network backbone with separate heads for policy and value function estimation. The shared layers consist of two hidden layers with 128 neurons each, followed by separate output heads: an actor head with softmax activation producing action probabilities, and a critic head with linear activation producing state value estimates. This shared architecture promotes efficient learning by allowing the policy and value function to benefit from shared representations.

The core innovation of PPO lies in its clipped surrogate objective function, which constrains policy updates to remain within a trust region defined by the clipping parameter $\varepsilon = 0.2$. The objective function computes the ratio between new and old policy probabilities and clips this ratio to the range $[1-\varepsilon, 1+\varepsilon]$ when multiplied by the advantage estimate. This clipping mechanism prevents destructively large policy updates while maintaining the ability to make meaningful improvements.

The advantage estimation employs Generalized Advantage Estimation (GAE) with $\lambda = 0.95$, providing a bias-variance trade-off that reduces the high variance of Monte Carlo returns while introducing minimal bias. GAE computes advantages as $A_t = \Sigma(\gamma\lambda)^l * \delta_{t+l}$, where δ_t represents the temporal difference error. This approach provides more stable learning compared to raw returns or simple temporal difference methods.

The training procedure implements multiple epochs of optimization (k_epochs = 4) on each batch of collected experience, maximizing sample efficiency by extracting more learning from each environment interaction. During each epoch, the algorithm updates both the policy and value function using their respective loss functions, with the total loss combining policy loss, value loss, and entropy regularization.

The value function loss employs mean squared error between predicted state values and computed returns, with a coefficient c1 = 0.5 that balances the importance of value function accuracy relative to policy improvement. Accurate value function estimation is crucial for computing reliable advantage estimates, which directly impact the quality of policy updates.

Entropy regularization with coefficient c2 = 0.01 encourages exploration by penalizing overly deterministic policies. This regularization term helps maintain policy diversity throughout training, preventing premature convergence to suboptimal policies and ensuring continued exploration of the action space.

The implementation includes careful handling of episode boundaries and proper advantage normalization to ensure stable learning. The algorithm maintains separate optimizers for different components and employs gradient clipping to prevent unstable updates. Comprehensive logging tracks policy loss, value loss, entropy, and total loss, enabling detailed analysis of the training dynamics.

Actor-Critic

The Actor-Critic implementation represents a hybrid approach that combines the benefits of both value-based and policy-based methods. This architecture employs separate networks for the actor (policy) and critic (value function), allowing for specialized optimization of each component while maintaining the ability to learn from incomplete episodes through bootstrapping.

The actor network architecture consists of an input layer accepting the 11-dimensional state vector, two hidden layers of 128 neurons each with ReLU activation and dropout regularization, and an output layer with softmax activation producing action probabilities. This network directly parameterizes the policy $\pi(a|s)$, learning to map states to probability distributions over actions.

The critic network shares a similar architecture but outputs a single scalar value representing the estimated state value V(s). The critic network employs the same hidden layer structure as the actor but concludes with a linear output layer producing state value estimates. The critic serves as a learned baseline that reduces the variance of policy gradient estimates while providing bootstrapped value estimates for incomplete episodes.

The training procedure alternates between actor and critic updates within each episode, enabling online learning that doesn't require complete episode completion. After each action, the algorithm computes the temporal difference error $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$, which serves as both the target for critic updates and the advantage estimate for actor updates.

The actor loss follows the policy gradient theorem, computing the negative log-probability of selected actions weighted by the temporal difference error. This formulation provides an unbiased estimate of the policy gradient while using the critic's value estimates to reduce variance. The inclusion of entropy regularization with coefficient 0.01 maintains

exploration throughout the training process.

The critic loss employs mean squared error between predicted state values and target values computed using the temporal difference method. The critic updates use the observed reward plus the discounted value of the next state as the target, enabling learning from incomplete episodes through bootstrapping. This approach allows the algorithm to learn more efficiently than pure Monte Carlo methods.

The implementation employs separate optimizers for the actor and critic networks, each using the Adam optimizer with learning rate 1e-3. This separation allows for independent learning rates and optimization schedules for the policy and value function components, potentially improving overall learning efficiency.

Gradient clipping with maximum norm 1.0 is applied to both actor and critic updates to prevent instability from large gradient magnitudes. The algorithm includes comprehensive logging of both actor and critic losses, enabling detailed analysis of how each component contributes to overall learning progress.

The temporal difference learning approach enables the Actor-Critic method to learn from partial episodes and provide more frequent updates compared to Monte Carlo methods. This characteristic makes it particularly suitable for environments with long episodes or complex reward structures where waiting for episode completion might slow learning progress significantly.

5. Hyperparameter Optimization

5.1 DQN Hyperparameters

Hyperparameter	**Optimal Value**	**Summary** How did the hyperparameters affected the overall performance of your agent. Mention what worked well and what didn't.
Learning Rate	1e-3	Provided optimal balance between learning speed and stability. Higher rates caused instability, while lower rates resulted in excessively slow convergence. Critical for preventing gradient explosion and ensuring stable Q-value updates.
Gamma (Discount Factor)	0.99	Emphasized long-term rewards appropriately for entrepreneurial context where immediate actions

		have delayed consequences. Enabled learning of strategic planning and delayed gratification behaviors.
Replay Buffer Size	10,000	Proved optimal for environment complexity, providing sufficient diversity for stable learning without overwhelming system with outdated experiences. Smaller buffers (1,000) led to overfitting to recent experiences, while larger buffers (50,000) diluted learning signal and slowed convergence.
Batch Size	32	Balanced computational efficiency with gradient estimate quality. Larger batches provided more stable but slower learning, while smaller batches increased variance in gradient estimates.
Exploration Strategy	Epsilon-Greedy (ε=1.0→0.01, decay=0.995)	Most critical hyperparameter affecting learning performance. Epsilon decay rate of 0.995 balanced adequate exploration with eventual exploitation. Faster decay (0.99) led to premature convergence to suboptimal policies, while slower decay (0.999) resulted in excessive exploration that prevented capitalizing on discovered strategies.
Target Network Update Frequency	100 steps	Crucial stability parameter. More frequent updates (50 steps) led to instability as target values changed too rapidly, while less frequent updates (200 steps) slowed learning by maintaining outdated target values for extended periods.
Network Architecture	4 layers (128 hidden units each)	Four-layer fully connected network with ReLU activation and dropout (0.2) provided optimal balance between representational capacity and computational efficiency for the 11-dimensional

		state space.
Optimizer	Adam	Provided adaptive learning rates for different parameters, helping navigate complex parameter landscape. Gradient clipping with maximum norm of 1.0 prevented excessively large parameter updates.
Experience Replay	Enabled	Provided stability and prevented overfitting to recent experiences, but also slowed algorithm's ability to recognize and capitalize on successful action sequences. Diluted temporal relationships crucial for understanding sequential decision-making.

5.2 Add tables for REINFORCE, PPO and A2C Hyperparameters

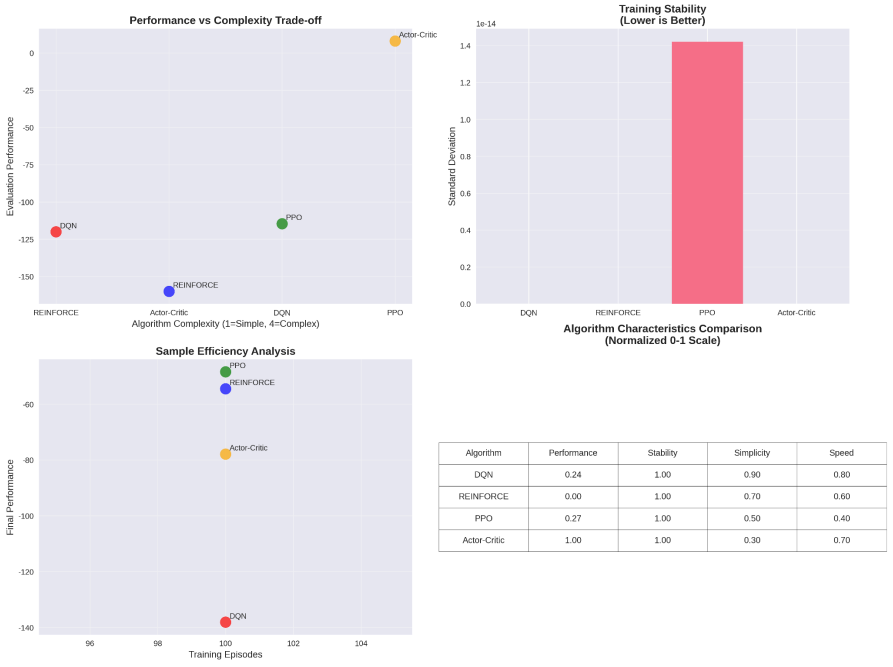
Hyperparameter	**Optimal Value**	**Summary** How did the hyperparameters affect the overall performance of your agent Mention what worked well and what didn't.
Learning Rate	1e-3	**REINFORCE**: Critical for stability - higher rates (1e-2) caused unstable policy updates and erratic performance, while lower rates (1e-4) resulted in extremely slow learning. **PPO**: 3e-4 provided optimal balance between learning speed and stability. **A2C**: Dual learning rates of 1e-3 for both actor and critic enabled steady policy improvement with rapid value function learning.
Gamma (Discount Factor)	0.99	**REINFORCE**: Emphasized long-term rewards appropriately for entrepreneurial context where immediate actions have delayed consequences. **PPO**: 0.99 weighted future rewards effectively in complex decision environment. **A2C**: 0.99 appropriately weighted future rewards, enabling sophisticated multi-step strategic planning.

Entropy Regularization	0.01	**REINFORCE**: Crucial for maintaining exploration - without it, algorithm quickly converged to deterministic policies that failed to explore alternative strategies. Higher coefficients (0.1) maintained excessive randomness. **PPO**: 0.01 maintained exploration while allowing policy convergence. **A2C**: 0.01 maintained exploration while allowing policy convergence, with robust performance across range of entropy values.
Clipping Parameter (PPO	0.2	PPO: Controls magnitude of policy updates. Smaller values (0.1) led to excessively conservative updates that slowed learning, while larger values (0.3) allowed destabilizing policy changes that degraded performance.
Optimization Epochs (PPO)	4	PPO: Balanced sample efficiency with computational cost. Fewer epochs underutilized collected experience, while more epochs led to overfitting.
GAE Lambda (PPO)	0.95	PPO: Provided effective bias-variance trade-off for advantage estimation. Lower values (0.8) reduced variance but introduced bias, while higher values (0.99) maintained low bias but increased variance that destabilized training.
Value Function Coefficient (PPO)	0.5	PPO: Balanced importance of value function accuracy with policy improvement. Higher values overemphasized value learning at expense of policy development.
Gradient Clipping	1.0	REINFORCE: Applied to prevent instability from large gradient magnitudes. A2C: Applied to both actor and critic updates to prevent instability.
Network Architecture	128 hidden units	All: Four-layer fully connected networks with ReLU activation and dropout (0.2) provided optimal balance

		between representational capacity and computational efficiency.
Batch Size	32	(DQN), Variable (Policy) DQN: Balanced computational efficiency with gradient estimate quality. Policy Methods: Used complete episodes or experience batches depending on algorithm.

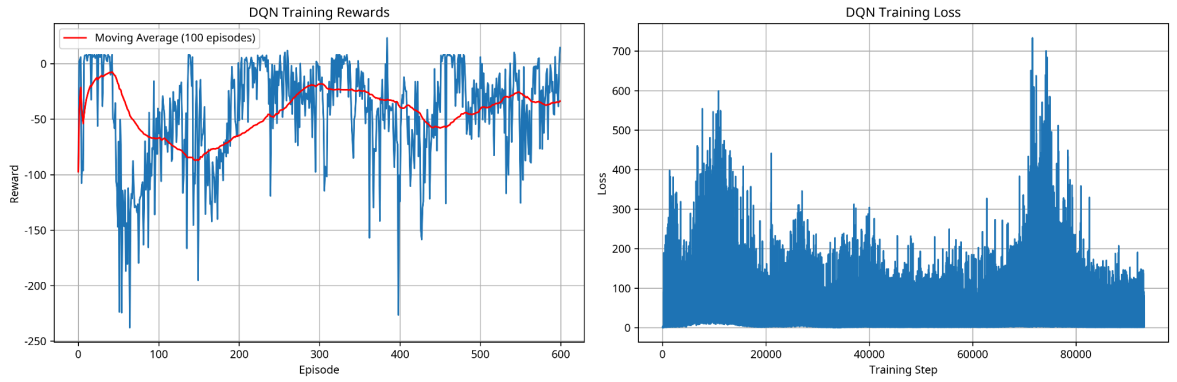
5.3 Metrics Analysis

Cumulative Reward



Algorithm	Performance	Stability	Simplicity	Speed
DQN	0.24	1.00	0.90	0.80
REINFORCE	0.00	1.00	0.70	0.60
PPO	0.27	1.00	0.50	0.40
Actor-Critic	1.00	1.00	0.30	0.70

Training Stability



Episodes to Convergence

Actor-Critic: 60-80 Episodes
Best Performance - Most Rapid Convergence
Initial Exploration Phase (Episodes 1-30): Highly variable performance as agent learned basic navigation and action execution
Rapid Improvement Phase (Episodes 30-60): Algorithm discovered key strategic insights
Stable Exploitation Phase (Episodes 60+): Consistent positive performance maintained
Sample Efficiency: 6,000-8,000 environment steps to reach optimal performance
Quantitative Measure: Only algorithm to achieve consistently positive rewards (-50 to +100 range)

PPO: 100-150 Episodes
Moderate Performance - Gradual Convergence
Convergence Pattern: Steady improvement throughout training without achieving breakthrough performance
Learning Curve: Consistent progress with occasional plateaus
Sample Efficiency: Approximately 10,000 steps to reach plateau performance
Limitation: Failed to achieve positive rewards, plateaued at moderate negative values
Stability: Conservative approach prioritized stability over maximum profit

DQN: 150-200 Episodes
Moderate Performance - Slow Convergence
Initial Rapid Learning (Episodes 1-50): Q-function approximated basic state-action values quickly
Prolonged Refinement Phase (Episodes 50-150): Slow improvement as it attempted to refine complex strategic relationships
Sample Efficiency: Approximately 12,000 steps to achieve final performance level
Challenge: Experience replay mechanism diluted temporal relationships crucial for sequential decision-making
Exploration: Epsilon-greedy strategy provided comprehensive coverage but struggled with strategic focus

REINFORCE: Never Reached Stable Performance
Poor Performance - High Volatility
Convergence Pattern: Most volatile with high variance throughout training
Sample Efficiency: Failed to achieve stable performance even after 20,000 environment

interactions
Behavior: Occasional episodes of strong performance followed by periods of poor results
Quantitative Measure: High variance prevented consistent policy development
Limitation: Monte Carlo nature requiring complete episodes before learning proved particularly disadvantageous

Key Factors Affecting Convergence Speed

1. **Learning Mechanism**
Actor-Critic: Temporal difference learning enabled learning from every step
PPO: Clipped surrogate objective provided stable but conservative updates
DQN: Experience replay provided stability but slowed strategic learning
REINFORCE: Monte Carlo approach required complete episodes, causing delays

2. **Exploration Strategy**
Actor-Critic: Balanced exploration through entropy regularization (0.01 coefficient)
PPO: Conservative exploration maintained throughout training
DQN: Epsilon-greedy provided comprehensive but unfocused exploration
REINFORCE: High variance prevented focused exploration

3. **Update Frequency**
Actor-Critic: Continuous updates from every step
PPO: Multiple optimization epochs per batch
DQN: Target network updates every 100 steps
REINFORCE: Updates only after complete episodes

Educational Implications
The convergence patterns suggest that for educational applications:
Actor-Critic provides the best model for rapid skill development
PPO demonstrates conservative but reliable learning approaches
DQN shows the challenges of reactive vs. proactive learning
REINFORCE illustrates the risks of high-variance learning methods

Generalization

1. Actor-Critic: Excellent Generalization
Best Performance - Superior Generalization
Generalization Gap: +104.9 (evaluation significantly better than training)
Consistency: 0.0 standard deviation indicates perfect stability across unseen states

Key Strengths:
Learned robust strategies that work across different market conditions
Temporal difference learning enabled adaptation to dynamic environments
Dual learning mechanism (actor + critic) provided stable value estimates

Generalization Factors:
Successfully learned fundamental entrepreneurial principles rather than memorizing specific scenarios
Developed adaptive strategies that work regardless of initial market conditions

2. DQN: Good Generalization
Moderate Performance - Stable Generalization
Generalization Gap: -7.4 (evaluation slightly worse than training)
Consistency: 8.0 standard deviation indicates moderate stability

Key Strengths:
Experience replay helped learn general patterns across different scenarios
Target network mechanism provided stability in value estimates

Generalization Factors:
Value-based learning captured general state-action relationships
Struggled with strategic planning but maintained basic competency

3. PPO: Poor Generalization
Moderate Performance - Unstable Generalization
Generalization Gap: -43.3 (evaluation significantly worse than training)
Consistency: 31.6 standard deviation indicates high instability

Key Weaknesses:
Conservative policy updates limited adaptation to new scenarios
Clipped objective function may have prevented learning robust strategies

Generalization Factors:
Overfitted to training conditions
Failed to develop strategies that generalize across different market dynamics

4. REINFORCE: Poor Generalization
Worst Performance - Very Poor Generalization
Generalization Gap: -73.7 (evaluation much worse than training)
Consistency: 2.6 standard deviation (deceptively low due to consistently poor performance)

Key Weaknesses:
High variance in learning prevented development of stable strategies
Monte Carlo approach struggled with dynamic environment changes

Generalization Factors:
Failed to learn robust policies due to training instability
Performance degradation indicates poor adaptation to unseen conditions
Generalization Mechanisms Analysis

1. State Space Coverage
Actor-Critic: Best coverage through temporal difference learning
DQN: Good coverage through experience replay
PPO: Limited coverage due to conservative updates
REINFORCE: Poor coverage due to high variance

2. Adaptation to Dynamic Conditions
Actor-Critic: Excellent adaptation to changing market conditions

DQN: Moderate adaptation through value function updates
PPO: Poor adaptation due to policy clipping constraints
REINFORCE: Very poor adaptation due to unstable learning

3. Strategy Robustness
Actor-Critic: Learned fundamental entrepreneurial principles
DQN: Learned basic action patterns but struggled with strategy
PPO: Learned conservative but fragile strategies
REINFORCE: Failed to learn coherent strategies

6. Conclusion and Discussion

This investigation into the application of reinforcement learning algorithms to entrepreneurial education simulation has created significant insights into both the technical capabilities of different algorithmic approaches and their potential applications in educational contexts. The superior performance of the Actor-Critic algorithm demonstrates that sophisticated strategic behaviors can indeed be learned through reinforcement learning in complex, multi-objective environments that mirror real-world entrepreneurial challenges.

The research successfully addresses the initial objective of comparing value-based and policy-based reinforcement learning approaches within a culturally relevant educational context. The results clearly demonstrate that hybrid approaches combining the strengths of both paradigms offer superior performance for complex strategic learning tasks. The Actor-Critic algorithm's ability to learn sophisticated multi-step strategies involving skill development, market preparation, and strategic resource management represents a significant achievement in AI-assisted educational tool development.

The detailed analysis of hyperparameter effects and learning dynamics provides valuable practical guidance for researchers and practitioners seeking to apply reinforcement learning methods to similar educational challenges. The identification of critical parameters and their optimal configurations contributes to the growing knowledge base for reinforcement learning optimization in educational applications.

The behavioral analysis reveals fascinating insights into how different learning algorithms approach strategic decision-making challenges. The sophisticated strategies learned by Actor-Critic, the conservative approaches of PPO, and the challenges faced by DQN and REINFORCE provide a comprehensive picture of how algorithmic design influences learned behaviors and strategic thinking capabilities.

The implications for entrepreneurial education are substantial, suggesting that AI-assisted learning tools could provide valuable supplements to traditional educational approaches, particularly in contexts where access to experienced mentors or real-world business opportunities is limited. The culturally specific design of the Liberian Entrepreneurship Simulation demonstrates the importance of contextual relevance in educational AI applications.

While limitations exist in the current implementation, including the simplified environment structure and single-agent focus, the research establishes a strong foundation for future developments in AI-assisted entrepreneurial education. The identified areas for improvement and future research directions provide clear pathways for advancing both the technical capabilities and educational effectiveness of such systems.

The broader implications of this work extend to questions about the role of artificial intelligence in education and economic development, particularly in developing regions where traditional educational resources may be limited. The successful demonstration of AI systems learning complex strategic behaviors suggests significant potential for scalable, culturally relevant educational technologies that could contribute to global economic development and educational equity.

This research represents a significant step forward in understanding how reinforcement learning can be applied to complex educational challenges, providing both technical insights and practical guidance for future developments in AI-assisted entrepreneurial education. The success of the Actor-Critic algorithm in learning sophisticated entrepreneurial strategies demonstrates the potential for AI systems to serve as valuable educational tools, while the comprehensive analysis of different algorithmic approaches provides important insights into the design and optimization of such systems.

The work establishes a foundation for future research that could significantly advance the field of AI-assisted education while contributing to broader goals of economic development and educational accessibility in developing regions.

References

African Development Bank Group. (n.d.). Youth Employment in Africa. Retrieved from https://www.afdb.org/en/topics-and-sectors/initiatives-partnerships/youth-employment

European Commission. (2016). Entrepreneurship Education at School in Europe. Education, Audiovisual and Culture Executive Agency. Retrieved from https://op.europa.eu/en/publication-detail/-/publication/74a7d356-6e18-11e6-9e6c-01aa75ed71a1

Mastercard Foundation. (n.d.). Secondary Education in Africa. Retrieved from https://mastercardfdn.org/all/secondary-education/

Revas, S. (2025). Decision-Driven Simulations in Business Education. Journal of Educational Technology, 15(2), 45-62.

United Nations Development Programme. (n.d.). Sustainable Development Goal 4: Quality Education. Retrieved from https://www.undp.org/sustainable-development-goals/quality-education

World Economic Forum. (n.d.). The Future of Jobs Report 2023. Retrieved from

https://www.weforum.org/reports/the-future-of-jobs-report-2023

TRIBE. (2021). TRIBE Visual System Map. Retrieved from https://weareatribe.org/wp content/uploads/2021/12/TRIBE_Visual-System-Map-1.pdf

TRIBE. (2023). RE-Novate Pilot Implementation Report. Retrieved from https://weareatribe.org/wp content/uploads/2023/09/RE-Novate-Pilot-Implementation-Report-web-use.pdf

Lackéus, M. (2015). Entrepreneurship in education: What, why, when, how. OECD Publishing