**Improved Project Proposal: Mancala AI Agent Development Using Advanced Deep Reinforcement Learning**

**Problem Definition**

Mancala, a family of ancient strategy games with global variations, presents a compelling challenge for AI development due to its combinatorial complexity and turn-chaining mechanics. The Kalah variant, popular in North America, involves strategic stone distribution and captures, offering a manageable yet non-trivial search space. This project aims to develop a high-performance AI agent for Kalah using **advanced deep reinforcement learning (DRL)** techniques, prioritizing both winning efficiency and computational resource optimization. While prior work by Hunter (2021) evaluated traditional algorithms (Minimax, MCTS) and basic DRL (A3C), this project will explore **hybrid architectures** (e.g., neural MCTS) and **modern DRL frameworks** (e.g., PPO, Rainbow DQN) to surpass existing benchmarks in win rate and computational efficiency.

**Brief Literature Review**

1. **Traditional AI Approaches**:  
   Gifford et al. (2008) explored **heuristic-driven Minimax** with **Alpha-Beta pruning** to play Mancala. They identified dominant strategies, such as **maximizing the stone differential (H1)** and **prioritizing "go-again" moves**, which contributed to effective decision-making. Their work emphasized the use of **heuristic combinations** to enhance gameplay but lacked **adaptive learning**. This approach, while efficient, could not dynamically improve through **self-play** or experience. Hunter (2021) expanded on Gifford's work by comparing various algorithms, specifically highlighting the advantages of the **A3C (Asynchronous Advantage Actor-Critic)** method over traditional heuristic-based approaches. However, Hunter's A3C agent achieved a **67.1% win rate** against the **Advanced Heuristic Minimax (AHM)** and proved **computationally intensive**. The A3C method also required **manual tuning of heuristics**, which limited its scalability and generalization across different game variants.
2. **Reinforcement Learning (RL)**:  
   Hunter’s **A3C implementation**, while effective in its ability to **learn through self-play**, used a **shallow neural network** that limited its potential for deeper exploration. The lack of a more robust **neural network architecture** constrained the agent’s ability to learn more complex strategies and optimize its performance. Additionally, the limited **self-play regime** restricted the agent’s exposure to **diverse game scenarios**, preventing it from fully exploring the strategic space of Mancala. There is significant room for improvement by incorporating **deeper architectures** and optimizing **exploration techniques**, which could lead to higher performance and generalization across various game scenarios.
3. **Gaps**:  
   There are key **gaps** in the existing literature on Mancala AI. Notably, no prior research has combined **neural networks** with **Monte Carlo Tree Search (MCTS)** to enhance decision-making for Mancala, despite the proven success of this hybrid approach in **AlphaGo**. This combination has the potential to significantly improve gameplay by efficiently exploring the **game tree** and evaluating moves. Furthermore, there has been limited exploration of **transfer learning** in the context of Mancala, which could help generalize learned strategies across different **Mancala variants**. Another important gap is the need for **efficient model compression** techniques, which would enable deployment of AI agents on **low-resource devices** without sacrificing performance.

**Proposed Methodology**

1. **Algorithm Design:**
   * Minimax, Alpha-Beta Pruning, Minimax with Advanced Heuristic (4 heuristics)
   * Hybrid Exploration: **Neural MCTS Hybrid**: Integrate a deep neural network (DNN) with Monte Carlo Tree Search, where the network predicts move probabilities (policy) and state value, guiding MCTS simulations
   * **Efficient DRL: Double DQN with Prioritized Experience Replay:** Mitigate overestimation bias and prioritize impactful transitions during training. Or Dueling DQN Architecture: Decouple value and advantage estimation to better evaluate board states and moves.
2. **Training Framework:**
   * Curriculum Learning: Train the agent on progressively complex game states (e.g., starting with 2 stones/bin, scaling to 4).
   * Reward Engineering: Define rewards for intermediate goals (e.g., +1 for a capture, +4 for increasing store,+5 for consecutive turns, +10 for a win).
3. **Evaluation Metrics**:
   * **Win Rate**: Against Hunter’s A3C  (baseline: 67.1%) and AHM over 100 simulated games.
   * **Computational Efficiency**: Measure training time, inference latency (<100 ms/move), and memory footprint (<50 MB).
   * **Generalization**: Test performance on unseen Mancala variants (e.g., Oware, Sungka).

**Tools**

* **Deep Learning**: PyTorch, TensorFlow.
* **Optimization**: TensorRT for model pruning.
* **Game Frontend + Backend**: JavaScript + Flask.

**Expected Outcomes**

1. A state-of-the-art AI agent achieving **>60% win rate** against Hunter’s A3C agent.
2. Open-source release of trained models, codebase, and benchmarking toolkit for reproducibility.
3. Comparative analysis of DQN variants vs. traditional methods (Minimax, AHM) and A3C.
4. A game interface for human interactions.

**References**

Gifford, C., Lee, J., & Peters, J. (2008). *Searching and game playing: An artificial intelligence approach to Mancala*. University of Kansas, Tech. Rep. ITTC-FY2009-TR-03050-03.

Hunter, T. J. (2021). *The exploration and analysis of Mancala from an AI perspective* (Honors Theses). <https://doi.org/10.32597/honors/257>.