University of Camerino

School of Science and Technology

Master Degree in Computer Science

Clustered Federated Deep Reinforcement Learning with Selective Aggregation

A Framework for Chess Playstyle Preservation

Supervisor

Prof. Massimo Callisto De Donato

Co-Supervisor

PhD. Student Martina Zannotti

Student

Francesco Finucci

Academic Year 2024-2025

Abstract

Federated learning enables collaborative model training across distributed nodes while preserving data privacy, but its application to reinforcement learning presents unique challenges, particularly the tension between collaborative learning and behavioral diversity. In chess, different playing styles (tactical vs. positional) represent valuable strategic diversity that traditional federated averaging would homogenize into a single global model.

This thesis presents a novel clustered federated deep reinforcement learning framework that maintains specialized chess engines while enabling knowledge transfer. We implement a three-tier hierarchical aggregation system: (1) local training on distributed nodes, (2) intra-cluster federated averaging within playstyle groups, and (3) selective inter-cluster aggregation that shares only low-level feature extraction layers while preserving cluster-specific policy and value heads. Our approach employs an AlphaZero-style neural network architecture with a 119-plane board representation and dual policy-value heads, combining Monte Carlo Tree Search (MCTS) for move exploration with self-play reinforcement learning. The system is bootstrapped using playstyle-filtered Lichess databases and tactical puzzle datasets before transitioning to self-play training.

The system comprises tactical and positional clusters, each containing four federated nodes that collaboratively learn cluster-specific strategies. By sharing only generic feature extractors (convolutional and early residual blocks) while maintaining separate decision-making layers (policy and value heads), our selective aggregation preserves strategic diversity while accelerating convergence through knowledge transfer. We evaluate trained models against Stockfish across multiple ELO levels and measure cluster diversity through specialized metrics.

Experimental results demonstrate that clustered federated learning successfully balances collaboration and specialization, producing distinct chess engines that maintain playstyle characteristics while benefiting from distributed training. This framework extends federated learning to reinforcement learning domains requiring behavioral diversity, with applications beyond chess to multi-agent systems, personalized AI assistants, and distributed robotics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rise of distributed computing and machine learning has created unprecedented opportunities for collaborative AI systems, yet also introduced fundamental challenges in how these systems learn and share knowledge. Federated learning has emerged as a paradigm that enables multiple agents to collaboratively train models while preserving data privacy and locality. However, when applied to reinforcement learning, particularly in domains requiring diverse behavioral strategies, traditional federated approaches face a critical tension: the trade-off between collaborative learning efficiency and the preservation of strategic diversity.

This thesis addresses this challenge in the context of chess, a domain where strategic diversity is not merely desirable but essential. Different playing styles, from aggressive tactical combinations to patient positional maneuvering, represent distinct approaches that have value in different game contexts. While traditional federated averaging would blend these approaches into a homogeneous strategy, we propose a clustered architecture that maintains this diversity while still enabling knowledge transfer across distributed agents.

1.1 Motivation

The success of deep reinforcement learning in complex domains like chess, Go, and Atari games has demonstrated the potential for AI systems to achieve superhuman performance through self-play and iterative improvement. AlphaZero, in particular, revolutionized computer chess by combining deep neural networks with Monte Carlo Tree Search, learning entirely from self-play without human knowledge. However, these achievements typically rely on massive centralized computational resources and homogeneous training data, limiting their applicability in distributed settings where data and computation are naturally partitioned.

Federated learning addresses some of these limitations by enabling collaborative model training across distributed nodes without centralizing data. This approach

offers several advantages: preservation of data privacy, reduced communication overhead, and the ability to leverage diverse computational resources. In the context of reinforcement learning for chess, federated approaches could allow multiple training agents to share knowledge while maintaining local control over their training processes and data.

Yet traditional federated learning, designed primarily for supervised learning tasks, faces a fundamental challenge when applied to reinforcement learning in strategic domains. The standard federated averaging algorithm converges toward a single global model, effectively homogenizing the strategies learned by different agents. In chess, this homogenization is problematic. Human chess has evolved numerous distinct playing styles, each with strengths in different positions and game phases. Tactical players excel at calculating concrete variations and exploiting immediate opportunities, while positional players specialize in long-term strategic maneuvering and structural advantages. These diverse approaches are not simply different paths to the same solution; they represent fundamentally different strategic philosophies that have coexisted and enriched the game for centuries.

The tension between collaboration and diversity becomes acute in federated reinforcement learning. While agents benefit from sharing knowledge about general chess principles and pattern recognition, forcing them to converge to identical strategies eliminates the very diversity that makes chess rich and complex. A purely tactical model may miss subtle positional nuances, while a purely positional model may overlook sharp tactical opportunities. An ideal system would preserve these distinct strategic identities while still enabling agents to learn from each other's experiences.

Furthermore, maintaining strategic diversity has practical benefits beyond chess. In multi-agent systems, personalized AI assistants, and distributed robotics, diverse behavioral strategies enable systems to adapt to different contexts, user preferences, and environmental conditions. A framework that can balance collaborative learning with behavioral preservation addresses a broader challenge in distributed artificial intelligence: how to build systems that are both cooperative and specialized.

1.2 Research Questions

This thesis investigates the following research questions:

1. **How can federated learning be adapted to preserve strategic diversity in reinforcement learning domains?** Traditional federated averaging produces a single global model, but many domains benefit from maintaining distinct behavioral strategies. We investigate whether a clustered federated architecture can balance knowledge sharing with playstyle preservation.

2. **What aggregation mechanisms enable knowledge transfer without homogenizing agent behaviors?** We explore selective aggregation strategies that share low-level feature representations while maintaining cluster-specific decision-making layers. The question is whether this approach can accelerate learning while preserving the distinct characteristics of different playing styles.

3. **Can playstyle-specific training data effectively bootstrap distinct strategic identities in a federated setting?** We investigate whether filtering training data by chess opening classifications (ECO codes) and puzzle types can establish and maintain tactical versus positional specializations throughout federated training rounds.

4. **How can we measure and quantify strategic diversity in federated chess engines?** Beyond standard performance metrics like ELO ratings, we need methods to assess whether cluster-specific models maintain distinct strategic characteristics or converge toward homogeneous play.

5. **Does clustered federated learning provide performance benefits compared to isolated training?** We examine whether the proposed three-tier aggregation system (local training, intra-cluster averaging, inter-cluster selective sharing) improves learning efficiency and final playing strength compared to agents training independently.

These questions guide our exploration of clustered federated deep reinforcement learning, with chess serving as a concrete testbed for principles applicable to broader distributed AI systems requiring both collaboration and specialization.

1.3 Contributions

This thesis makes the following contributions:

1. **A novel clustered federated learning architecture for reinforcement learning.** We introduce a three-tier hierarchical aggregation system that maintains multiple cluster-specific models rather than converging to a single global model. This architecture enables collaborative learning while preserving behavioral diversity.

2. **Selective inter-cluster aggregation mechanism.** We design and implement a selective weight-sharing strategy that aggregates only low-level feature extraction layers across clusters while maintaining separate policy and value

heads for each playstyle. This mechanism enables knowledge transfer without homogenizing strategic characteristics.

3. **Playstyle-aware data filtering methodology.** We develop a systematic approach for establishing distinct strategic identities using ECO opening code classification and puzzle type filtering, demonstrating how domain-specific data curation can initialize and maintain behavioral diversity in federated settings.

4. **Complete federated AlphaZero implementation for chess.** We provide an end-to-end system integrating AlphaZero-style deep reinforcement learning with federated infrastructure, including cluster management, asynchronous communication, distributed aggregation, and both supervised bootstrapping and self-play training phases.

5. **Evaluation framework for strategic diversity.** We develop metrics to quantify and track the preservation of cluster-specific playing styles throughout federated training, providing tools for assessing whether models maintain distinct characteristics or undergo homogenization.

6. **Empirical analysis of collaboration-diversity tradeoffs.** Through systematic experiments, we provide insights into the benefits and limitations of clustered federated learning compared to isolated training and traditional federated averaging.

1.4 Thesis Structure

The remainder of this thesis is organized as follows:

Chapter 2: Background provides the theoretical foundation for this work. We review deep reinforcement learning and the AlphaZero algorithm, including neural network architectures, Monte Carlo Tree Search, and self-play training. We then introduce federated learning principles, covering federated averaging and its applications. Finally, we discuss related work in distributed reinforcement learning and behavioral diversity preservation.

Chapter 3: Methodology presents our clustered federated learning framework. We describe the three-tier hierarchical aggregation system, detailing local training, intra-cluster federated averaging, and selective inter-cluster aggregation. We explain the selective weight-sharing mechanism that preserves strategic diversity while enabling knowledge transfer. We also present our playstyle-aware data filtering methodology using ECO codes and puzzle classifications.

Chapter 4: Implementation describes the technical realization of our system. We detail the AlphaZero-style neural network architecture with its 119-plane board representation and dual policy-value heads. We explain the server architecture for cluster management and distributed aggregation, the client-side training infrastructure, and the data processing pipeline for Lichess databases and puzzle datasets.

Chapter 5: Experiments outlines our experimental setup and evaluation methodology. We describe the cluster topology with tactical and positional specializations, training configurations, and hyperparameters. We present our evaluation framework, including Stockfish-based performance testing, diversity metrics for measuring playstyle preservation, and baseline comparisons against isolated training and traditional federated averaging.

Chapter 6: Results presents our empirical findings. We analyze training convergence across clusters, evaluate playing strength through ELO estimation, and measure strategic diversity preservation throughout federated rounds. We compare clustered federated learning against baseline approaches and provide insights into the collaboration-diversity tradeoff.

Chapter 7: Conclusion summarizes our contributions and findings. We discuss the implications of our work for federated reinforcement learning and distributed AI systems. We acknowledge limitations of the current approach and propose directions for future research, including extensions to other domains, alternative aggregation strategies, and scalability improvements.

Chapter 2

Background and Related Work

This chapter provides the theoretical foundation for our clustered federated deep reinforcement learning framework. We begin by describing our literature search methodology to ensure transparency and reproducibility. We then provide an overview of reinforcement learning fundamentals and the AlphaZero algorithm that forms the basis of our chess engine. We introduce federated learning principles and discuss how they can be adapted to reinforcement learning settings. Finally, we review related work in distributed reinforcement learning, behavioral diversity preservation, and federated learning applications.

2.1 Literature Review Methodology

To ensure a comprehensive and systematic review of relevant literature, we conducted a multi-stage search process across academic databases, preprint repositories, and technical documentation. This section details our search strategy, inclusion criteria, and the tools used to identify and synthesize relevant work.

2.1.1 Search Strategy

We performed systematic searches across multiple academic databases and repositories between September 2024 and January 2025. The primary sources included:

- **Google Scholar:** Broad coverage of computer science literature and citation tracking
- **arXiv.org:** Recent preprints in machine learning (cs.LG, cs.AI, cs.MA)
- **ACM Digital Library:** Conference proceedings (NeurIPS, ICML, ICLR, AAAI)
- **IEEE Xplore:** Systems and distributed computing literature

- **Semantic Scholar:** AI-powered search and paper recommendations

2.1.2 Search Terms and Queries

Our literature search employed combinations of core terms, connected with Boolean operators. Table 2.1 shows the primary and secondary search queries used across different databases.

Table 2.1: Literature Search Queries

Category	**Search Query**
Primary Queries	
Federated RL	"federated learning" AND "reinforcement learning"
Clustered FL	"clustered federated learning" OR "personalized federated learning"
Distributed Chess AI	"AlphaZero" AND ("federated" OR "distributed")
Behavioral Diversity	"behavioral diversity" AND "multi-agent"
Chess Playstyle	"chess AI" AND ("playing style" OR "playstyle")
Selective Aggregation	"selective aggregation" AND "federated learning"
Distributed MCTS	"Monte Carlo Tree Search" AND "distributed"
Secondary Queries	
Heterogeneous FL	"heterogeneous federated learning"
Non-IID Data	"non-IID federated learning"
Transfer Learning	"transfer learning" AND "deep reinforcement learning"
Distributed Self-Play	"self-play" AND ("distributed" OR "federated")
Quality Diversity	"quality diversity algorithms"
Model Divergence	"model divergence" AND "federated"

We also performed backward citation tracking (reviewing references of key papers) and forward citation tracking (identifying papers that cite foundational work) to ensure coverage of relevant literature.

2.1.3 Inclusion and Exclusion Criteria

Papers were included if they met the following criteria:

Inclusion criteria:

- Published between 2015 and 2025 (with exceptions for seminal earlier work)

- Directly relevant to federated learning, reinforcement learning, or chess AI
- Peer-reviewed or from reputable preprint repositories (arXiv)
- Available in English
- Sufficient technical detail to understand methodology

Exclusion criteria:

- Purely theoretical work without implementation insights
- Domain-specific applications unrelated to game-playing or multi-agent systems
- Duplicate publications or superseded versions
- Insufficient detail on methods or results

2.1.4 AI-Assisted Literature Discovery

In addition to traditional database searches, we leveraged AI tools to assist with literature discovery and synthesis. Table 2.2 shows the AI-assisted queries used for research assistance.

Table 2.2: AI-Assisted Literature Discovery Queries

Tool	**Query / Purpose**
Claude (Anthropic)	
Federated RL Overview	"Summarize recent advances in federated reinforcement learning, focusing on methods that handle heterogeneous agents"
Behavioral Diversity	"What are the key challenges in maintaining behavioral diversity in multi-agent systems?"
Selective Aggregation	"Compare different approaches to selective aggregation in federated learning"
Distributed AlphaZero	"Find papers that combine AlphaZero-style training with distributed or federated approaches"
Semantic Scholar	
Recommendations	AI-powered paper recommendations based on citation graphs and content similarity
Connected Papers	
Citation Networks	Visualizing citation networks and identifying research clusters

These AI-assisted searches were particularly useful for:

1. Quickly understanding the landscape of a new research area

2. Identifying terminology variations (e.g., "behavioral diversity" vs "policy diversity" vs "strategic heterogeneity")

3. Discovering connections between seemingly disparate research communities (e.g., federated learning and chess AI)

4. Generating additional search terms based on paper abstracts

2.1.5 Documentation and Synthesis

We maintained a structured database of reviewed papers using reference management software, tracking:

- Paper metadata (authors, venue, year)
- Key contributions and findings
- Methodological approaches
- Relevance to our research questions
- Gaps or limitations identified

This systematic approach ensured comprehensive coverage of relevant literature while maintaining focus on our core research questions about clustered federated learning for reinforcement learning with behavioral diversity preservation.

2.2 Reinforcement Learning

Reinforcement learning (RL) is a machine learning paradigm where an agent learns to make decisions by interacting with an environment to maximize cumulative rewards. Unlike supervised learning, where correct answers are provided, RL agents must discover effective strategies through trial and error, receiving only sparse feedback about the quality of their actions.

2.2.1 Markov Decision Processes

Reinforcement learning problems are typically formalized as Markov Decision Processes (MDPs). An MDP is defined by a tuple (S, A, P, R, γ) where:

- S is the set of possible states the environment can be in
- A is the set of actions the agent can take

- $P(s'|s,a)$ is the transition probability of reaching state s' after taking action a in state s
- $R(s,a,s')$ is the reward received when transitioning from state s to s' via action a
- $\gamma \in [0,1]$ is the discount factor that determines how much future rewards are valued relative to immediate rewards

The agent's behavior is determined by a policy $\pi(a|s)$ that specifies the probability of taking action a in state s. The goal of reinforcement learning is to find an optimal policy π^* that maximizes the expected cumulative discounted reward, known as the return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.1}$$

The value function $V^\pi(s)$ represents the expected return when starting in state s and following policy π:

$$V^\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{2.2}$$

Similarly, the action-value function $Q^\pi(s,a)$ represents the expected return when taking action a in state s and then following policy π:

$$Q^\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{2.3}$$

2.2.2 Deep Reinforcement Learning

Traditional RL algorithms use tabular representations to store value functions, which becomes impractical for large state spaces. Deep reinforcement learning addresses this limitation by using neural networks as function approximators to estimate value functions and policies. This enables RL to scale to complex domains like video games, robotics, and board games.

Deep Q-Networks (DQN) pioneered this approach by using convolutional neural networks to approximate the action-value function $Q(s,a)$ for Atari games. The key innovations included experience replay, where transitions are stored in a buffer and sampled randomly for training, and a separate target network that stabilizes learning.

Policy gradient methods provide an alternative approach by directly parameterizing the policy $\pi_\theta(a|s)$ with neural network parameters θ. The policy gradient theorem allows us to compute gradients of the expected return with respect to these parameters:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)] \tag{2.4}$$

Actor-critic methods combine these approaches by maintaining both a policy network (actor) and a value network (critic). The critic evaluates the quality of the actor's actions, providing lower-variance gradient estimates.

2.2.3 AlphaZero and Monte Carlo Tree Search

AlphaZero represents a breakthrough in deep reinforcement learning for board games, achieving superhuman performance in chess, Go, and shogi through pure self-play learning without human knowledge. The algorithm combines three key components: a deep neural network for position evaluation, Monte Carlo Tree Search for move planning, and reinforcement learning for continuous improvement.

Neural Network Architecture

The AlphaZero neural network takes the current board position as input and produces two outputs:

- A **policy head** $p = f_\theta^p(s)$ that outputs a probability distribution over legal moves
- A **value head** $v = f_\theta^v(s)$ that outputs a scalar value estimating the probability of winning from the current position

The network uses a deep residual architecture with convolutional layers to process spatial patterns on the board. This dual-headed design allows the network to both suggest promising moves and evaluate position quality, which are used together during search.

Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a best-first search algorithm that builds a search tree incrementally through random sampling. Unlike traditional minimax search used in classical chess engines, MCTS focuses computational effort on the most promising variations.

Each node in the search tree represents a board position and stores statistics about visits and values. The search proceeds through four phases:

1. **Selection:** Starting from the root, choose child nodes that balance exploration (trying less-visited moves) and exploitation (following moves with high

estimated value) using the PUCT formula:

$$UCT(s,a) = Q(s,a) + c_{puct} P(s,a) \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)} \tag{2.5}$$

where $Q(s,a)$ is the mean action value, $P(s,a)$ is the prior probability from the neural network, and $N(s,a)$ is the visit count.

2. **Expansion:** When a leaf node is reached, evaluate the position using the neural network to get policy priors and value estimate.

3. **Simulation:** In AlphaZero, this phase is replaced by direct neural network evaluation rather than random rollouts.

4. **Backpropagation:** Update statistics along the path from leaf to root, incrementing visit counts and updating action values.

After running many MCTS simulations (typically 800 for AlphaZero), the final move is selected based on visit counts, which represent a refined estimate of move quality informed by deep search.

Self-Play Training

AlphaZero improves through iterative self-play. The current neural network generates training games by playing against itself using MCTS-guided move selection. Each position in these games provides training data:

- The **policy target** is the distribution of MCTS visit counts π, which represents an improved policy compared to the raw network output
- The **value target** is the final game outcome $z \in \{-1, 0, 1\}$ (loss, draw, win)

The network is trained to minimize a combined loss function:

$$L = (z - v)^2 - \pi^T \log p + c||\theta||^2 \tag{2.6}$$

This loss function encourages the network to predict game outcomes accurately (value loss) and match the improved MCTS policy (policy loss), with L2 regularization to prevent overfitting.

The key insight of AlphaZero is that MCTS can be viewed as a policy improvement operator. By repeatedly training the network on self-play games where moves are selected by MCTS, the network gradually improves, which in turn makes future MCTS searches more effective. This creates a positive feedback loop that leads to continuous improvement without requiring any domain knowledge beyond the game rules.

2.3 Federated Learning

Federated learning is a distributed machine learning paradigm that enables multiple participants to collaboratively train a shared model while keeping their data decentralized. Unlike traditional centralized training where all data is aggregated in one location, federated learning brings the model to the data rather than the data to the model. This approach addresses privacy concerns, reduces communication costs, and enables learning from data that cannot be easily centralized due to legal, technical, or practical constraints.

2.3.1 Federated Averaging Algorithm

The foundational algorithm for federated learning is Federated Averaging (FedAvg), proposed by McMahan et al. FedAvg coordinates distributed training across multiple clients through a central server that aggregates local model updates.

The basic FedAvg procedure consists of several rounds of communication between the server and clients:

1. **Model Distribution:** The server sends the current global model parameters w_t to a subset of clients

2. **Local Training:** Each selected client k trains the model on their local dataset D_k for E epochs, producing updated parameters w_t^k

3. **Aggregation:** The server collects the updated models and computes a weighted average:

$$w_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_t^k \tag{2.7}$$

   where $n_k = |D_k|$ is the size of client k's dataset and $n = \sum_k n_k$ is the total data size

4. **Iteration:** This process repeats for multiple rounds until convergence

The key advantage of FedAvg is that it performs multiple local optimization steps before communication, significantly reducing the number of communication rounds needed compared to sending gradients after each batch. This is crucial since communication is often the bottleneck in distributed settings.

2.3.2 Challenges in Federated Learning

While federated learning offers many benefits, it also introduces several challenges compared to centralized training:

Non-IID Data: In typical federated settings, data across clients is not independently and identically distributed. Different clients may have data from different distributions, different label distributions, or different amounts of data. This heterogeneity can slow convergence and lead to suboptimal models.

Communication Efficiency: Since clients may have limited bandwidth or intermittent connectivity, minimizing communication rounds is essential. Techniques like gradient compression, quantization, and local optimization help reduce communication costs.

System Heterogeneity: Clients may have varying computational capabilities, storage capacity, and availability. Some clients may be able to train quickly on powerful hardware while others are limited by mobile device constraints.

Privacy and Security: While federated learning keeps raw data decentralized, model updates can still leak information about training data. Differential privacy and secure aggregation techniques can provide stronger privacy guarantees but add computational overhead.

2.3.3 Personalization in Federated Learning

A key limitation of vanilla FedAvg is its assumption that all clients should converge to a single global model. However, in many real-world scenarios, different clients or groups of clients may benefit from specialized models tailored to their specific data distributions or preferences.

Personalized federated learning addresses this by allowing some degree of model customization per client or client group while still leveraging collaborative learning. Several approaches have been proposed:

Fine-tuning: Clients start with a global model but continue training locally after federation completes, adapting the model to their specific data.

Multi-task Learning: The model is split into shared and personalized layers. Shared layers learn common representations across all clients, while personalized layers adapt to individual client characteristics.

Clustered Federated Learning: Clients are grouped into clusters based on data similarity or other criteria. Each cluster maintains its own model through federated averaging within the cluster, while potentially sharing knowledge across clusters.

Meta-Learning: The global model is trained to be easily adaptable to new clients with minimal fine-tuning, using techniques like Model-Agnostic Meta-Learning (MAML).

These personalization techniques recognize that a one-size-fits-all global model is not always optimal, especially when client data distributions are significantly

different. Our work builds on the clustered federated learning approach, extending it to reinforcement learning settings where behavioral diversity is not just a result of data heterogeneity but a desired outcome.

2.4 Chess Engines and AI

Computer chess has been a central domain for artificial intelligence research since the field's inception. The evolution of chess engines reflects broader trends in AI, from symbolic rule-based systems to search algorithms to modern deep learning approaches.

2.4.1 Classical Chess Engines

Traditional chess engines rely on three core components: board representation, move generation, and position evaluation combined with tree search.

Minimax and Alpha-Beta Pruning: Classical engines use minimax search to explore the game tree, assuming both players play optimally. The algorithm recursively evaluates positions by assuming the maximizing player wants the highest score while the minimizing player wants the lowest. Alpha-beta pruning dramatically reduces the search space by eliminating branches that cannot affect the final decision.

Hand-Crafted Evaluation Functions: Classical engines evaluate positions using carefully designed functions that consider material balance, piece activity, pawn structure, king safety, and other strategic factors. These evaluation functions encode centuries of human chess knowledge into numerical scores.

Stockfish, currently the strongest classical chess engine, represents the pinnacle of this approach. It combines sophisticated search algorithms, aggressive pruning techniques, and finely tuned evaluation heuristics to search billions of positions per second. Despite being based on traditional methods, Stockfish remains competitive with neural network engines in many positions.

2.4.2 Neural Network Chess Engines

The introduction of AlphaZero in 2017 demonstrated that neural networks trained through self-play could achieve superhuman chess performance without domain knowledge. Unlike classical engines that use hand-crafted evaluation functions, AlphaZero learned position evaluation and move selection entirely from self-play.

The success of AlphaZero inspired several open-source projects, most notably Leela Chess Zero (LC0), which reimplemented the AlphaZero approach using distributed training across thousands of volunteer computers. LC0 has evolved to

match and sometimes exceed Stockfish's playing strength, particularly in positions requiring long-term strategic planning.

Neural network engines exhibit different playing characteristics compared to classical engines. They tend to favor positional understanding and long-term planning over tactical calculation depth. This has enriched computer chess by introducing more varied and sometimes more human-like playing styles.

2.4.3 Playing Style in Chess

Chess players, both human and computer, exhibit distinct playing styles that reflect different strategic philosophies. Two broad categories often used to characterize playing style are:

Tactical Play: Emphasizes concrete calculation, immediate threats, and combinative play. Tactical players excel at spotting forcing sequences, sacrifices, and sharp variations. They prefer dynamic positions with many pieces on the board where calculation depth determines the outcome.

Positional Play: Focuses on long-term strategic advantages like pawn structure, piece coordination, and space control. Positional players excel at gradual maneuvering, prophylaxis, and converting small advantages into wins. They prefer positions where understanding trumps calculation.

In human chess, players typically develop preferences for certain opening systems and strategic themes that align with their style. Mikhail Tal exemplified tactical brilliance with his sacrificial attacks, while Anatoly Karpov demonstrated the power of refined positional technique. Most strong players can play both styles but show preferences and strengths in certain types of positions.

For chess engines, playing style has traditionally been less pronounced. Classical engines tend toward tactical play due to their search depth, while neural network engines often display more positional understanding. Our work explores whether distinct playing styles can be deliberately cultivated and maintained in federated learning settings, creating specialized engines rather than homogeneous ones.

2.5 Related Work

Our work draws on several areas of research: distributed reinforcement learning, federated learning applications to RL, behavioral diversity in multi-agent systems, and clustered federated learning.

2.5.1 Distributed Reinforcement Learning

Distributed training has become essential for reinforcement learning in complex domains due to the computational demands of both environment interaction and neural network training.

Parallel Experience Collection: Many RL systems use multiple actors to collect experience in parallel, dramatically increasing sample efficiency. A3C (Asynchronous Advantage Actor-Critic) introduced asynchronous updates from multiple workers to a shared model. IMPALA (Importance Weighted Actor-Learner Architecture) separates experience collection from learning, using importance sampling to handle the resulting off-policy data.

Distributed AlphaZero: The original AlphaZero training used distributed self-play, with many workers generating games in parallel while a central learner updates the neural network. This architecture enables the massive scale of training required for superhuman performance—AlphaZero played nearly 5 million games during training.

However, these distributed RL approaches still rely on centralized aggregation and aim for a single global model. They distribute computation for efficiency but do not address the challenge of maintaining behavioral diversity or training multiple specialized models collaboratively.

2.5.2 Federated Reinforcement Learning

Applying federated learning to reinforcement learning is an emerging research area. While traditional supervised federated learning deals with fixed datasets, federated RL must handle the added complexity of exploration, temporal dependencies, and non-stationary data distributions as policies improve.

Policy-Based FedRL: Some approaches extend FedAvg directly to policy gradient methods, aggregating policy network parameters across agents. However, this faces challenges when agents experience different environments or have different reward functions, as policies optimized for different MDPs may not meaningfully average.

Value-Based FedRL: Other work focuses on sharing value function estimates or Q-functions across agents. This can be effective when agents share the same environment but experience different parts of the state space.

Exploration vs. Exploitation Trade-offs: Federated RL introduces unique challenges for exploration. If all agents follow similar exploration strategies, they may collectively fail to explore the state space adequately. Some work addresses this through coordinated exploration strategies or by encouraging diversity in local training.

Most federated RL research has focused on settings where agents face different but related tasks, aiming to share knowledge across task distributions. Our work differs by considering agents working on the same task (chess) but seeking to maintain distinct behavioral strategies rather than converging to a single solution.

2.5.3 Behavioral Diversity in Multi-Agent Systems

Maintaining diversity in multi-agent systems has been studied in several contexts, motivated by applications in team behavior, robust learning, and ensemble methods.

Quality Diversity Algorithms: MAP-Elites and related algorithms explicitly optimize for both performance and behavioral diversity. They maintain archives of solutions that exhibit different behaviors, even if some are suboptimal, creating a diverse collection of strategies.

Diversity-Driven Exploration: In multi-agent RL, some work uses diversity objectives to encourage agents to explore different parts of the state space or learn different policies. This can improve collective exploration efficiency and robustness.

Emergent Communication and Specialization: Research in multi-agent communication has shown that agents can spontaneously develop specialized roles when working toward common goals, with different agents handling different subtasks.

Our work differs from these approaches in that we seek to maintain diversity not just during training but in the final models, and we do so in a federated setting where agents cannot directly observe each other but must coordinate through aggregation.

2.5.4 Clustered Federated Learning

Clustered federated learning recognizes that in heterogeneous settings, forcing all clients to converge to a single global model may be suboptimal. Instead, clients are grouped into clusters, with each cluster maintaining its own model.

Automatic Clustering: Several methods propose to automatically discover clusters during training. Clients are initially assigned to clusters randomly or based on data characteristics, then cluster membership is refined based on model similarity or gradient alignment. This allows the system to discover natural groupings in the data distribution.

Multi-Center Federated Learning: Some approaches maintain multiple global models and allow clients to contribute to the model that best matches their data. This creates a form of competitive federated learning where models specialize to different data distributions.

Hierarchical Aggregation: Similar to our approach, some work uses hierarchical aggregation where updates are first aggregated within clusters, then partial

aggregation occurs across clusters. This balances the benefits of local specialization with global knowledge sharing.

However, existing clustered federated learning work focuses on supervised learning tasks with heterogeneous data distributions. The clustering emerges from data heterogeneity rather than being designed to preserve behavioral characteristics. Our work extends these ideas to reinforcement learning where we explicitly initialize and maintain clusters based on strategic playing style, and we introduce selective layer-wise aggregation to balance knowledge transfer with behavioral preservation.

2.5.5 Transfer Learning in Deep RL

Transfer learning in reinforcement learning aims to leverage knowledge from one task to accelerate learning on related tasks. This is relevant to our selective aggregation mechanism.

Progressive Neural Networks: Freeze previously learned networks and add new capacity for new tasks, allowing lateral connections. This prevents catastrophic forgetting but increases model size.

Fine-Tuning and Layer Freezing: Standard practice is to fine-tune pretrained networks on new tasks, often freezing early layers that learn general features while adapting later layers to task specifics.

Multi-Task Learning: Training a single network on multiple related tasks can improve performance on all tasks by learning shared representations. However, this typically assumes tasks are sufficiently similar that a shared representation helps rather than hinders.

Our selective inter-cluster aggregation draws on these insights. We share early feature extraction layers across playing style clusters, analogous to sharing general features in transfer learning, while keeping decision-making layers cluster-specific to preserve strategic differences.

2.5.6 Gaps in Existing Work

While the related work provides valuable insights and techniques, several gaps remain that our research addresses:

1. **Federated RL with Intentional Diversity:** Existing federated RL work focuses on handling unavoidable data heterogeneity, not deliberately maintaining behavioral diversity as a goal.

2. **Selective Layer Aggregation:** While some clustered FL work uses hierarchical aggregation, selective layer-wise aggregation based on functional role (feature extraction vs. decision making) is underexplored.

3. **Playing Style Preservation:** Chess AI research has not addressed how to maintain distinct playing styles in collaborative training settings where the default would be homogenization.

4. **Comprehensive Framework:** No existing work provides an end-to-end system combining clustered federated learning, selective aggregation, and self-play RL for behavioral diversity preservation.

Our work addresses these gaps by developing a framework specifically designed to balance collaborative learning with behavioral preservation in reinforcement learning domains, using chess as a concrete and measurable testbed.

Chapter 3

Methodology

3.1 Problem Formulation

3.1.1 Markov Decision Process Formulation

3.1.2 Strategic Diversity Objective

3.1.3 Federated Learning Constraints

3.1.4 Performance Metrics

3.2 Clustered Federated Learning Framework

3.2.1 Framework Overview

3.2.2 Cluster Design

3.2.3 Client-Server Architecture

3.2.4 Communication Protocol

3.3 Neural Network Architecture

3.3.1 Input Representation

3.3.2 Residual Network Structure

3.3.3 Policy and Value Heads

3.3.4 Layer Grouping for Selective Aggregation

3.4 Three-Tier Aggregation System

3.4.1 Local Training Phase

3.4.2 Intra-Cluster Aggregation

Chapter 4

Implementation

4.1 System Design

4.2 Technologies Used

4.3 Challenges

Chapter 5

Experimental Setup

5.1 Experimental Design

5.2 Datasets and Benchmarks

5.3 Hyperparameters

Chapter 6

Results and Discussion

6.1 Results

6.2 Analysis

6.3 Discussion

Chapter 7

Conclusion

7.1 Summary

7.2 Contributions

7.3 Future Work

Bibliography

[1] A. Author and B. Author. Example paper title. *Journal Name*, 1:1–10, 2023.

Appendix A

Additional Material