Multi-Agent Board Game Strategy Through Simulation

Maximillian Banach
Electrical Engineering
Economics
University of Florida
mbanach@ufl.edu

Andres Espinosa
Industrial and Systems Engineering
University of Florida
andresespinosa@ufl.edu

Cathy Quan
Computer Science
Mathematics
University of Florida
cathyquan@ufl.edu

Jason Li
Computer Science
Statistics
University of Florida
jason.li1@ufl.edu

Han Mach
Computer Science
University of Florida
hmach@ufl.edu

Brian Magnuson
Computer Science
University of Florida
b.magnuson@ufl.edu

Cody Flynn
Computer Science
University of Florida
codyflynn@ufl.edu

***Abstract*—In this paper, we investigate key factors that influence the effectiveness of TMARL agents in the environment of *Catan*. Specifically, we examine the impact of: (1) a GNN-based state-to-action network designed to leverage the board's spatial and relational structure, (2) various exploration strategies to navigate extended decision-making phases, (3) different replay sampling distributions to prioritize learning from high-impact decisions, and (4) an ensemble model to mimic strategies employed by human players. The effectiveness of these approaches will be assessed by comparing the best-performing model against established baseline agents, including Random, Value Function, Game-Theoretic, and Human players.**

I. Introduction

A wargame, as defined by Dr. Peter Perla, is a model involving people making decisions in a synthetic environment of competition or conflict, in which they see the effects of their decisions on that environment and then get to react to those changes [1]. This framework extends naturally to competitive environments like 4X games and strategic board games, which share similar dynamics. The application of artificial intelligence (AI) to complex, multi-agent environments has seen significant advancements in recent years. However, military wargaming remains a particularly challenging domain for AI.

The complexity of wargames stems from a variety of factors: non-stationary dynamics, high planning, pervasive uncertainty, diplomatic negotiation, strict constraints, and competing objectives. To illustrate these challenges, consider the reinforcement learning (RL) breakthrough of AlphaZero in mastering Chess [2]. AlphaZero must learn a deterministic environment—mastering board representation, understanding game rules, evaluating positions, and grasping opening theory. Now, imagine scaling this up to a wargame like Catan, where an RL agent must navigate not only the same complexities but also a constantly changing board, the need to negotiate with multiple players, and the unpredictable dynamics of player alliances and conflicts. The stakes are higher, the environment is more chaotic, and the agent's ability to adapt becomes far more critical. From this, it becomes clear why wargames have remained a formidable challenge for AI. However, solving these challenges holds immense potential for military strategy, diplomacy, and other high-stakes, multi-agent domains.

Beyond its contributions to the RL research community, this paper highlights three key applications of enhanced wargaming RL agents: enriched educational and training simulations, automated adversarial and red-team strategy development, and more robust decision support systems for high-stake strategic environments. First, improved RL agents can elevate educational wargaming by serving as adaptive opponents or allies, enabling trainees to experience a wider variety of realistic scenarios. Second, as RL continues to develop, these agents can expose blind spots and stress-test assumptions in military planning, offering a scalable alternative to traditional red-teaming methods. Finally, high-stakes decision-making environments such as cyber, logistical, or battlefield operations can benefit by integrating RL agents in wargaming simulations. These systems can quickly simulate long-term consequences of strategic choices, adapt to changing conditions in real-time, and provide commanders with actionable insights under uncertainty. To explore these possibilities in a controlled yet sufficiently complex environment, we turn our attention to the strategic board game *Catan*.

Catan offers a uniquely rich environment for studying RL in wargame-like settings. It incorporates many of the core challenges found in military wargames, including strict constraints, stochasticity, and multi-agent negotiation. Unlike purely deterministic games such as chess or Go, *Catan* forces agents to reason under uncertainty, engage in trade and diplomacy with opponents, and make strategic decisions over a long time horizon. Additionally, its widespread popularity and standardized ruleset make it a reproducible and well-understood testbed for AI research. These characteristics, combined with the availability of the open-source simulator `catanatron` [3], make *Catan* an ideal sandbox for testing RL methods designed for complex, adversarial simulations.

We now present the structure of our investigation and results

in applying RL methods to *Catan*. First, in section II, we discuss related work in the areas of RL and wargames, highlighting both foundational studies and recent advancements. In section III, we provide a comprehensive overview of the necessary concepts for the paper. In section IV, we detail the problem formulation and the methodology used to develop and test the RL Agents in the `catanatron` environment. In section V, we present the results of our experiments, comparing the performance of different RL models and their implications to the broader realm of RL in wargames. Finally, in section VI, we discuss the broader implications of our findings, including future real-world applications in wargame simulations, and conclude with suggestions for future research.

II. Related Work

Research on artificial agents in strategic environments has evolved significantly over the past decades. Early work focused on rule-based systems and heuristic search, with notable examples including Minimax and Alpha-Beta pruning used in traditional board games like Chess and Checkers. The advent of deep learning led to a major shift in agent design.

The introduction of Deep Q-Networks (DQN) by Mnih et al. [4] marked a breakthrough by enabling agents to learn value functions directly from high-dimensional input. This success culminated in AlphaGo [5] and AlphaZero [2], which demonstrated superhuman performance in Go, Chess, and Shogi through self-play and Monte Carlo Tree Search (MCTS). However, these successes were largely confined to deterministic, fully observable environments. In contrast, environments like the board game *Catan* [6] and the 4X game *Civilization* [7] present more complex, stochastic, and partially observable settings. The characteristics of imperfect information, multiple win conditions, and negotiation align more closely with military wargames and diplomatic simulations.

Recent work has begun exploring Reinforcement Learning in multi-agent and highly constrained settings. In [8], the Dyna-Q algorithm adapts to different personality types in *Civilization IV*, inspiring our ensemble approach in Section IV. In the realm of Pokémon battles, a Q-learning agent underperforms against a custom Minimax algorithm [9], while in *Tribes* [10], a genetic algorithm yields the best results—though a rule-based agent performs comparably.

Efforts to apply RL to *Catan* reveal numerous challenges. Dr. Henry Charlesworth [11] shows that introducing negotiation dramatically expands the action space, severely impacting agent performance. To mitigate similar issues, Kim and Li [12] disable trading altogether to avoid model degradation. In [13], the problem is simplified drastically by restricting a DQN agent to only the opening phase of the game. Most notably, [3] employs an Alpha-Beta model which relies on a hand-crafted heuristic value function, which remains the best-performing agent in *Catan* to date for the purposes of our research.

In parallel, the rise of Graph Neural Networks (GNNs) has provided a powerful mechanism for learning over structured data, which is particularly valuable in games like *Catan*, where spatial and relational features are central to strategic reasoning. Relational inductive biases, as outlined in [14], are critical when tasks naturally map to graph representations, such as board games or networked environments. Recent surveys and reviews [15] further highlight the potential of GNN-RL integration across wargame-like domains including game simulations, pandemic modeling, and cybersecurity. Notably, graph-based reinforcement learning has been successfully applied to previously unsolved domains, such as influence maximization [16], traffic signal optimization [17], and cyberattack simulation [18]. These studies showcase GNN's ability to scale and generalize across graph-structured domains.

Despite the breadth of work in strategic agents, few studies have tackled environments that simultaneously exhibit strict constraints, stochasticity, and social complexity. *Catan* embodies these traits, yet existing reinforcement learning approaches often simplify the environment or rely on domain-specific heuristics. At the same time, graph-based RL methods have shown promise in similarly structured domains but remain largely unexplored in social-diplomatic games. Our research aims to document which components of reinforcement learning align well with wargame-like mechanics—from graph representations of spatial dynamics to policy learning under uncertainty and negotiation.

III. Background Context

A. Settlers of Catan

Catan is a popular multiplayer board game centered around resource management, player negotiation, and settlement expansion. It is played on a hexagonal board composed of various resource tiles. Players roll dice for resource acquisition and may trade with others to build roads, settlements, and cities strategically in efforts to be the first person to reach 10 victory points and win the game against other opponents.

Unlike deterministic games like chess or Go, Catan incorperates many stochastic elements. The game board is randomized in every session; resources tile and their associated number tokens vary each game, meaning players must adapt to a different environment with every playthrough. Resource availability is determined by dice rolls, which introduce probabilistic variability even though some numbers are statistically more likely than others. Additionally, the game includes elements of diplomacy, as players can propose, accept, or reject trades. These interactions often reflect hidden goals, temporary alliances, or attempts to sabotage opponents. Such dynamics create a rich space for analyzing multi-agent behavior and contribute to the game's expansive action space.

This large action space, combined with the presence of 4 players, results in slow-paced gameplay with frequent low-impact actions, such as unsuccessful trade proposals or inconsequential dice rolls. To succeed, players must think about the actions of all 3 opponents and rationalize the hidden decisions and goals. These elements lead to long decision sequences with limited immediate feedback, which have made it difficult for both AI-based and game-theoretic models to outperform the average human player.

B. Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) is a subfield of machine learning that integrates reinforcement learning (RL) principles with deep neural networks. In RL, an agent interacts with an environment and learns to select actions, through trial and error, that maximize cumulative future rewards. The agent aims to learn optimal actions in given states to achieve the highest possible long term rewards over time. DRL extends traditional RL by using deep networks to approximate either value functions such as Deep Q-Learning (DQN), which use neural networks to approximate the Q-value function, or by employing policy gradient methods directly like, Proximal Policy Optimization (PPO) and Trust Region Policy Optimization (TRPO), which directly optimize the policy. This integration allows the agent to handle high-dimensional state and action spaces, making DRL suitable to solve more complex problems.

One of the most well-known DRL breakthroughs is AlphaZero. This breakthrough demonstrated that deep networks accompanied with self-play could achieve superior performance against humans in games like chess and Go. However, DRL success has yet to hold in multi-agent environments like Catan, where the presence of other multi-agent intteractions and stochasticit introduces instability in training. These challenges also present an opportunity to explore and advance methods to making DRL more effective in environments where stratigic reasoning and uncertainty exists.

C. Graph Neural Networks

Graph Neural Networks (GNNs) are a type of neural network architecture that is designed to process graph-structured data. In a graph, nodes represent entities, while edges represent the relationships between them. Layers in a GNN iteratively update node representations by aggregating features from neighboring nodes, allowing the model to capture local and global structural information within the graph.

In the context of games like Catan, GNN's can be useful to model the game board as a graph. In this graph formulation, elements of the game are the nodes and spatial interactions between them are edges, which allows an agent to develop an understanding about the strategic dependencies more effectively than flat vector input structures. Feedforward neural networks (FNNs), are limited in their ability to encapsulate the relationship between entities and often loses spatial information in the process.

While GNNs have the upperhand to complex environments, they also come with limitations. For instance, they can suffer from issues like over-smoothing, where node representations can become indistinguishable after many layers, and scalability issues when applied to large or dynamic graphs. As a result, it is crucial to carefully design meaningful node and edge features to ensure the GNN can learn informative and distinctive representations.

IV. Implementation

To advance the state of RL in wargaming, our implementation focuses on three core research directions. First, we investigate how wargames can be naturally represented as graphs or sets of graphs, and we provide both mathematical and empirical evaluations of the effectiveness of such representations. Second, we examine the impact of RL exploration strategies on the agent's ability to solve wargame-like tasks. Third, we propose an ensemble RL framework that integrates multiple heuristic submodels, each specializing in different aspects of the game. By offloading low-level constraint handling and decision making, we allow the RL agent to focus on the long-term aspects of the wargame while still outperforming the individual heuristic submodels.

A. State Space

The first step in any RL project is to model the environment as a Markov Decision Process (MDP). The game of *Catan* can clearly be modeled as an MDP through translating board and player information into numeric representations. However, its state space is incredibly high-dimensional and complex due to the game's stochastic nature, different win conditions, and multi-agent dynamics. *Catan* is a multi-agent environment with 2 to 4 players. In this work, we standardize the player count to 4 for both vector and graph-based representations. This assumption allows us to train a single, unified model that can generalize across 1v1, 1v2, and 1v3 testing environments. We also assume the game will always use the standard board configuration of 19 hex tiles, 54 building locations, and 72 road locations.

1) Vector Representation: The vector-based state representation encodes the game state of *Catan* at any given time step as a flattened numerical array. This approach to modeling the game benefits from being easily ingested by a neural network. We denote the game state as a vector $\mathbf{s} \in \mathbb{R}^d$, where each entry s_i represents a specific element of the game state. The total dimensionality d corresponds to the total length of all encoded components, organized as follows:

- s_0 to s_{287}: One-hot encoding of edge ownership.
- s_{288} to s_{503}: One-hot encoding of settlements.
- s_{504} to s_{719}: One-hot encoding of cities.
- s_{720} to s_{1043}: One-hot encoding of port locations.
- s_{1044} to s_{1063}: One-hot encoding of the robber's position.
- s_{1064} to s_{1177}: One-hot encoding of tile resource types.
- s_{1178} to s_{1367}: One-hot encoding of dice values assigned to tiles.
- s_{1368} to s_{1442}: Other game information outside of the board.

See section IX-C1 in the appendix for more detailed information.

2) Graph Representation: The same 1443-length state vector described in section IV-A1 can be decomposed into two logical parts: board-specific features ($s_0 - s_{1367}$) and global game state features ($s_{1368} - s_{1442}$). This separation naturally lends itself to a graph-based modeling aproach: the board is represented as a graph, while the global state becomes a set of auxiliary features. In this representation, spatial and relational dynamics are explicitly captured. We model the board as a graph with:

- Nodes: Representing building locations (settlements and cities).
- Edges: Representing roads and connectivity between nodes.
- Node features: Encoding ownership, port access, and properties of adjacent tiles.
- Edge features: Encoding road ownership.
- Graph-level features: Capturing the remaining global game state (e.g., player inventories, development cards, bank resources).

While we maintain fixed assumptions on map size and player count for consistency with the vector representation, one advantage of the graph model is its flexibility. By modeling players and tiles as nodes of different types, the graph-based approach can easily scale to more dynamic or irregular game setups.

Each node embedding $\mathbf{x}$ is structured as:

- x_0: Node ID. (used for debugging)
- $x_1 - x_4$: One-hot encoding of whether player j has a settlement on this node.
- $x_5 - x_8$: One-hot encoding of whether player j has a city on this node.
- $x_9 - x_{14}$: One-hot encoding of whether this node has a port of resource type l.
- $x_{15} - x_{50}$: One-hot encoding of the resource type of each adjacent tile k.
- $x_{51} - x_{110}$: One-hot encoding of the dice value of each adjacent tile k.
- $x_{111} - x_{116}$: One-hot encoding of whether the robber is on each adjacent tile k.

Each edge embedding $\mathbf{e}$ is structured as:

- e_0: Edge ID. (used for debugging)
- $e_1 - e_4$: One-hot encoding of whether player j owns the road on this edge.

The graph-level feature vector $\mathbf{y}$ includes:

- $y_0 - y_{74}$: The same global features used in the vector representation, including development cards, bank inventory, dice rolls, available player structures, and current victory points.

3) Comparison and Motivation: By constructing both vector and graph-based state representations, we aim to highlight the trade-offs of each approach. The vector format provides a compact and dense encoding, ideal for traditional feedforward networks. However, it fails to capture spatial relationships and topological structure—critical elements in games like *Catan*.

The graph representation preserves these relationships and opens the door to leveraging graph neural networks (GNNs), which are more adept at modeling interactions in structured environments. This approach aligns well with how many wargames—including 4X games like Civilization and naval simulations like Harpoon. In these games, spatial relationships between players, units, and terrain are fundamental and often too complex to be flattened into a fixed-length vector.

Games with rich, dynamic maps and interactions are more naturally expressed as graphs. Units, tiles, players, and resources can be modeled as heterogeneous nodes, and their interactions as typed edges. As wargames become increasingly complex and agentic, we believe that graph-based state modeling will be a foundational tool for future AI systems in this domain.

B. Exploration Strategy

To explore the action space effectively, we implement two exploration strategies: epsilon-greedy and Gibbs softmax.

In the epsilon-greedy approach, the agent selects a random action with a fixed probability ϵ, encouraging exploration, while exploiting the current best-known action with probability $1-\epsilon$. This method is simple to implement and has been widely used in reinforcement learning due to its straightforward balance between exploration and exploitation.

On the other hand, Gibbs softmax assigns probabilities to actions based on their estimated values using a softmax function. Actions with higher estimated values are selected more frequently, but less optimal actions are still considered with non-zero probability. This probabilistic approach allows for more nuanced exploration compared to the binary decision-making of epsilon-greedy.

The motivation for comparing these two strategies lies in their differing strengths. Epsilon-greedy is computationally efficient and effective in simpler environments, but it may struggle in heavily constrained action spaces like Catan, where the majority of actions are unavailable at any given timestep. Gibbs softmax, by focusing exploration on high-value actions while still considering alternatives, may better navigate such complex, multi-agent environments. By evaluating both methods, we aim to identify which strategy is more effective in balancing exploration and exploitation in the context of Catan's unique challenges.

C. Action Space

To introduce the action space, we examine the possible decisions a player can make in Catan. The action space in Catan shares several important characteristics with those found in many wargames.

The first characteristic is that the action space is tightly constrained. For instance, a typical turn in Catan consists of approximately 2-10 discrete actions. Each turn begins with a mandatory "Roll Dice" action to simulate resource distribution and ends with an "End Turn" action to relinquish control. As a result, the action distribution is often highly skewed, with a small subset of actions dominating the majority of agent behavior.

A second characteristic is the presence of both competitive and diplomatic actions, which adds significant variety to the action space. Social actions, in particular, can dramatically expand the space due to their continuous and combinatorial nature. In Catan, for example, trading is a key mechanic, and in theory, a player could propose any quantity of one resource in exchange for any quantity of another. To keep the action space manageable in our simulation, we restrict trades to 1:1 resource exchanges.

Finally, wargames are typically characterized by a high degree of stochasticity. The initial game state can vary widely between playthroughs, even though the underlying rules remain constant. In Catan, this is reflected in the randomized board layout at the start of each game and the dice rolls used to determine resource distribution each turn.

Taken together, these characteristics illustrate that the action space in wargames is significantly more complex than in many other types of games. Ultimately, this complexity motivates the integration of heuristic decision-making frameworks with the generalization capabilities of reinforcement learning agents. We propose a comparison between two action spaces: the first samples directly from the complete set of feasible actions, while the second selects from a set of heuristic sub-models designed to guide decision-making.

1) Direct Actions: In the direct action formulation, the agent samples from the full set of all legal actions available at a given game state. This includes actions such as building roads, settlements, or cities; buying development cards; trading with the bank; initiating trades with other players; playing development cards; and ending the turn. The space of legal actions varies dynamically based on the player's resources, board position, and game phase, making it nontrivial to enumerate at each step.

To ensure tractability, we dynamically generate the legal action set at each timestep and represent it as a discrete action space to mask out the infeasible actions. The process of a direct agent selecting a move can be seen in Figure 1, where the Action Mask is responsible for masking out infeasible actions to ensure the agent does not violate the laws of the game. While this approach gives the agent full control and expressiveness, it suffers from challenges related to sample efficiency and sparse optimal actions—especially during early stages of training where the agent has not yet learned meaningful patterns in action selection.

One thing the current state of RL in wargaming reveals is that the complexities of wargames have made it intractable for RL agents (with the current algorithms and technology) to outperform humans, or even other heuristics. In order to enable an RL agent to learn how to navigate the long-term strategy of a wargame, while abstracting away individual choices we propose an ensemble heuristic model in the following section.

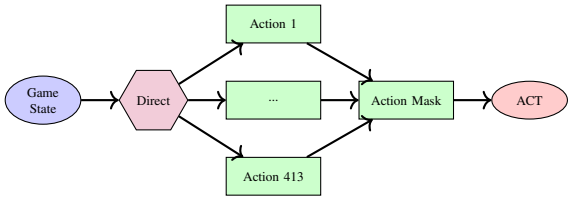

Fig. 1. Overview of the direct model architecture

2) Sub-Model Sampling: The heuristic sub-model formulation abstracts the decision-making process by grouping low-level actions into higher-level strategic behaviors. Rather than selecting individual actions directly, the agent chooses from a predefined set of heuristic policies, each designed to encode domain knowledge and target a specific tactical objective.

In our implementation, we employ an ensemble of seven TacticaValueFunction agents—greedy heuristic value function players—each tuned with distinct hyperparameters to specialize in a particular tactic. As illustrated in Figure 2, the ensemble agent delegates decision-making to one of these sub-models, thereby avoiding the need to explicitly reason about the full space of feasible actions at each timestep. For instance, one sub-model is optimized for road expansion, prioritizing road placement and settlement opportunities, while another emphasizes resource accumulation to enable city development. Once selected, a sub-model autonomously executes a sequence of actions until a predefined stopping condition is met or control is returned to the agent.

Importantly, these sub-models are not limited to heuristic policies; the framework is generalizable to any decision-making model that encapsulates narrow, low-correlated tactics, whether learned or rule-based.

This hierarchical structure significantly reduces the complexity of the action space, leading to improved sample efficiency and more stable learning dynamics. However, it also introduces a trade-off: the agent's capacity to discover novel or unconventional strategies is constrained by the diversity and expressiveness of the sub-model set.

By comparing this sub-model-based approach with the direct action formulation, we aim to evaluate the balance between fine-grained control and strategic abstraction in complex, stochastic environments such as Catan.

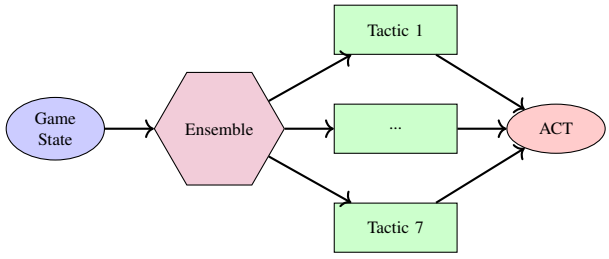

Fig. 2. Overview of the ensemble sub-model decision-making architecture

V. Results

Several tests were run to compare the performance of different parameters that will be reported in the following subsections. For each table, the win rates are measured over 1000 games against each of the baseline agents in a 4-player environment with one agent of the specified trained model against three agents of a specified baseline model.

Note that for testing whether the following models significantly outperforms a given baseline agent, a binomial test was run at the $\alpha = 0.01$ significance level with the alternate hypothesis being that the win rate is greater than 25%. All tests were run in **R**[19]. When tests were run over 1000 games, the

model significantly out-performs a given baseline model if the win rate is $\geq$28.3%[1].

When comparing win rates between two models against the same baseline for signficant difference, a one-sided Z-test for two proportions was done at the α = 0.01 significance level[2].

A. Action Space Comparison

We compared two models trained over 1,000 games, differing only in their action space configuration. One model utilized a direct action selection strategy (*Direct4112*), while the other employed a sub-model sampling ensemble approach (*Ensemble4111*). The win rates against the three baseline agents is found in Table I.

TABLE I
COMPARISON OF DIRECT ACTION VS. SUB-MODEL SAMPLING

Model	Random	Value Function	Alpha-Beta
Direct4112	100%	0%	0%
Ensemble4111	100%	**34%**	**13%**

These results indicate that the sub-model sampling strategy used in *Ensemble4111* yields stronger and more generalizable performance across all of the baseline opponents. In contrast, the direct action approach in *Direct4112* struggled to compete effectively, particularly against more sophisticated baseline agents.

B. Representation Comparison

Another model was created to compare the game state representation of agents, comparing the graph and vector representation of the environment. These are compared using a GNN model (*Ensemble4111*) compared to a Feedforward Neural Network (FNN) model (*FNN4140*) with all other hyperparameters held constant. Performance metrics from this comparison are detailed in Table II.

TABLE II
COMPARISON OF GNN VS. FNN REPRESENTATION

Model	Random	Value Function	Alpha-Beta
FNN4140	99%	13%	10%
Ensemble4111	100%	**34%**	13%

Against **Random** and **Alpha-Beta** models, there is no significant performance difference between the two models. However, the performance gap widens against the **Value Function** model; this suggests that there may be some benefit of using GNN over FNN, but the results are inconclusive.

[1]Confidence interval from `binom.test(283, 1000, 1/4, alternative = "greater", conf.level = 0.99)` in **R**[19]

[2]P-value derived from `prop.test(x=c(Lesser_Model_Win_Rate, Greater_Model2_Win_Rate) * 1000, n=c(1000, 1000), alternative = "less", conf.level = 0.99, correct = FALSE)` in **R**[19]

C. Reward Function Comparison

Two models were created to compare the performance of sparse reward function and a dense reward function. The sparse model consisted of punishing the agent for prolonged games, and giving a large reward or punishment corresponding to winning and losing respectively. On the other hand, the dense reward function was constructed to provide rewards encouraging heuristic behaviors. The Performance comparison is detailed in Table III with the dense model named *Ensemble4110* the sparse model being *Ensemble4111*.

TABLE III
COMPARISON OF GNN VS. FNN REPRESENTATION

Model	Random	Value Function	Alpha-Beta
Ensemble4110	100%	27%	12%
Ensemble4111	100%	**34%**	13%

Against **Random** agents, both models achieved perfect performance (**100%** win rate), indicating that both are able to reliably outperform the simplest models.

Against the **Value Function** baseline, Ensemble4111 significantly outperformed the dense-reward agent with a **34%** win rate, compared to **27%** for Ensemble4110. This suggests that the sparse reward function better encouraged long-term strategic play. When playing against the strongest opponent, **Alpha-Beta**, Ensemble4111 only slightly outperforms Ensemble4110, but there is no significant performance gain.

Overall, these results suggest that sparse reward signals, despite offering less frequent feedback, may more effectively guide the agent toward high-level winning strategies—particularly against stronger, more strategic opponents. Furthermore, sparse models are simpler to construct in most environments. Additionally, sparse reward functions are often simpler to design and implement across a wide range of environments.

D. Training Opponent Comparison

To evaluate the impact of training diversity on agent performance, we compared two agents trained over 2000 games: *CatanicaBeta*, which was trained exclusively against the best performing agents—Alpha-Beta agents, and *Catanica*, which was trained against a mixture of Value Function and Alpha-Beta agents. This experiment aims to assess whether exposure to a broader range of strategies during training leads to better generalization and robustness. Table IV compares their performance across multiple baseline opponents to determine how training diversity influences overall performance.

TABLE IV
COMPARISON OF GNN VS. FNN REPRESENTATION

Model	Random	Value Function	Alpha-Beta
CatanicaBeta	100%	28%	24%
Catanica	98%	**37%**	24%

Against both **Random** and **Alpha-Beta** agents, both models perform nearly the same. However, Catanica exhibits a significant improvement in win rate against the **Value Function**.

This suggests that diversifying the training opponents can lead to greater generalization and robustness. However, these opponents are also the models that the two reinforcement learning agents were trained against, so this does not yield conclusive evidence, and would require further investigation.

VI. Conclusion

In this work, we investigated various approaches to improving the performance of reinforcement learning (RL) agents in the wargame environment of *Catan*. Through a series of experiments comparing different action space configurations, state representations, reward functions, and training strategies, we found that certain strategies, such as sub-model sampling and sparse reward signals, display strong signs of promise. Specifically, the sub-model sampling approach demonstrated superior performance compared to direct action strategies, while sparse reward functions were more effective in guiding agents toward long-term strategic play, particularly against stronger baseline models.

We also explored the impact of training diversity on agent performance, showing that exposure to a broader range of strategies can lead to improved generalization. However, further investigation is required to confirm these findings in more diverse settings.

A. Future Work

There are several exciting avenues for future research. One limitation in our study was an investigation centered on human players in wargames, which may reveal unique insights into the strengths and weaknesses of current strategies. Additionally, further theoretical exploration into the characteristics of wargames that make them suitable for graph-based representations could lead to more effective model designs.

Another interesting area of research is the potential for subdividing complex wargames into smaller sub-games, each tackled by specialized agents. This approach would reduce the complexity faced by each individual agent, allowing it to focus on specific aspects of the game. Evaluating the performance of such models, particularly in multi-agent environments, could offer valuable insights into how to scale RL agents effectively for more complex games.

VII. Contributions

VIII. Acknowledgements

IX. Appendix

A. Submission Abstract

Settlers of Catan (*Catan*) is a popular board game featuring expansion, resource management, negotiation, and long-term planning decisions. Unlike traditional turn-based strategy games like *Go* or *Chess*, which have been effectively mastered by AI agents, *Catan* presents a far greater challenge due to its stochastic nature, non-stationary dynamics, and multi-agent interactions—with up to 4 players in the base game. *Catan* also features slow-paced gameplay with frequent low-impact actions, such as unsuccessful trade proposals or inconsequential dice rolls. These elements lead to long decision sequences with limited immediate feedback, which have made it difficult for both AI-based and game-theoretic models to outperform the average human player.

To address this challenge, we leverage two emerging areas of research: Turn-Based Multi-Agent Reinforcement Learning (TMARL) and Graph Neural Networks (GNNs). TMARL provides a learning framework where agents iteratively improve their decision-making through experience in multi-agent environments. Meanwhile, GNNs offer a powerful representation learning approach that captures both local and global structures within a game state.

In this paper, we investigate key factors that influence the effectiveness of TMARL agents in the environment of *Catan*. Specifically, we examine the impact of: (1) a GNN-based state-to-action network designed to leverage the board's spatial and relational structure, (2) various exploration strategies to navigate extended decision-making phases, (3) different replay sampling distributions to prioritize learning from high-impact decisions, and (4) an ensemble model to mimic strategies employed by human players. The effectiveness of these approaches will be assessed by comparing the best-performing model against established baseline agents, including Random, Value Function, Game-Theoretic, and Human players.

Our findings will provide valuable insights into the challenges and opportunities of applying RL to complex, multi-agent environments. The simultaneous interplay of competing objectives and diplomatic negotiations presents a unique challenge for AI—one that remains largely unsolved. Therefore, our results also have broader implications for the development of decision-support systems in military strategy, where turn-based simulations are commonly used to model future engagements, resource allocation, and coalition dynamics—similar to *Catan*.

B. Author Biographies

- **Maximillian Banach (Project Manager)** - Eagle Scout & fourth-year undergraduate Electrical Engineering & Economics dual major at the University of Florida.
- **Andrés Espinosa (AI Team Lead)** - Fourth-year Industrial Engineering Honors Student at the University of Florida with experience in supply chain management, reinforcement learning, and transportation optimization research.
- **Cody Flynn (AI Team Member)** - Computer Science major and Mathematics minor. He specializes in algorithm design, applied professionally during his summer internship at Lutron Electronics.
- **Jason Li (Programming Team Lead)** - Computer Science and Statistics Undergraduate Student at the University of Florida with experience in full-stack website development and app development.
- **Han Mach (Programming Team Member)** - Fourth-year Computer Science student at the University of Florida, also pursuing a minor in Digital Arts and Sciences.
- **Brian Gunnar Magnuson (Programming Team Member)** - Fifth-year student at the University of Florida majoring in Computer Science and minoring in Digital Arts and Sciences.
- **Cathy Quan (AI Team Member)** - Fourth-year Computer Science Student at the University of Florida. She is minoring in Mathematics and has obtained an AI Certificate.

C. State Information

1) Vector Representation:

- s_0 to s_{287}: One-hot encoding of edge ownership. There are 72 edges and 4 players, giving $72 \times 4 = 288$ entries. Entry $s_{4k+j} = 1$ if edge k is owned by player j.
- s_{288} to s_{503}: One-hot encoding of settlements. There are 54 nodes and 4 players, so $54 \times 4 = 216$ entries. Entry $s_{288+4i+j} = 1$ if node i has a settlement belonging to player j.
- s_{504} to s_{719}: One-hot encoding of cities. Again, $54 \times 4 = 216$ entries. Entry $s_{504+4i+j} = 1$ if node i has a city belonging to player j.
- s_{720} to s_{1043}: One-hot encoding of port locations. Each of the 9 port nodes can be assigned to one of 4 resource types or a 3:1 trade port (assume 5 types), and ports are placed at nodes. This gives $54 \times 6 = 324$ entries. Entry $s_{720+6i+l} = 1$ if node i has a port of type l.
- s_{1044} to s_{1063}: One-hot encoding of the robber's position. There are 19 tiles, so 19 entries total. Entry $s_{1044+m} = 1$ if the robber is on tile m.
- s_{1064} to s_{1177}: One-hot encoding of tile resource types. With 19 tiles and 6 resource types (wood, brick, wheat, sheep, ore, desert), this gives $19 \times 6 = 114$ entries. Entry $s_{1064+6m+l} = 1$ if tile m has resource type l.
- s_{1178} to s_{1367}: One-hot encoding of dice values assigned to tiles. Dice values range from 2 to 12 (excluding 7, which is unused by *Catan*), so with 19 tiles, we get $19 \times 10 = 190$ entries. Entry $s_{1178+12m+n} = 1$ if tile m has dice value n.
- f_0 to f_4: Resource bank counts for each of the 5 resource types.
- f_5: Number of development cards left in the deck.
- f_6: Whether players are currently discarding (boolean flag).

- f_7: Whether the knight (robber) is being moved (boolean flag).
- f_8: Actual victory points of the current player (p_0).
- f_9 to f_{13}: Current player's hand count for each of the 5 resources.
- f_{14} to f_{18}: Current player's hand count of each development card type.
- f_{19} to f_{22}: Whether each player has rolled yet this turn.
- f_{23} to f_{26}: Number of resource cards held by each player.
- f_{27} to f_{30}: Number of development cards held by each player.
- f_{31} to f_{34}: Whether each player has Largest Army.
- f_{35} to f_{38}: Whether each player has Longest Road.
- f_{39} to f_{42}: Number of roads available for each player.
- f_{43} to f_{46}: Number of settlements available for each player.
- f_{47} to f_{50}: Number of cities available for each player.
- f_{51} to f_{54}: Length of longest road for each player.
- f_{55} to f_{58}: Total visible victory points for each player.
- f_{59} to f_{74}: Whether each player has played a non-VP development card (4 players × 4 types).

REFERENCES

[1] P. Perla, "Wargaming and the cycle of research and learning," *Scandinavian Journal of Military Studies*, Sep 2022.

[2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering chess and shogi by self-play with a general reinforcement learning algorithm," *CoRR*, vol. abs/1712.01815, 2017. [Online]. Available: http://arxiv.org/abs/1712.01815

[3] B. Collazo and contributors, "Catanatron: Settlers of catan bot and simulator," https://github.com/bcollazo/catanatron, 2024, accessed: 2025-04-13.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[5] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–, oct 2017. [Online]. Available: http://dx.doi.org/10.1038/nature24270

[6] K. Teuber, "The settlers of catan," 1995, board game. [Online]. Available: https://www.catan.com

[7] S. Meier and B. Shelley, "Sid Meier's Civilization," 1991, a formative example of the 4X genre. [Online]. Available: https://civilization.2k.com/

[8] C. Amato and G. Shani, "High-level reinforcement learning in strategy games," vol. 1, 01 2010, pp. 75–82.

[9] A. Kalose, K. Kaya, and A. Kim, "Optimal battle strategy in pokémon using reinforcement learning," 2018, stanford University AA228 Course Project Report. [Online]. Available: https://web.stanford.edu/class/aa228/reports/2018/final151.pdf

[10] D. Perez-Liebana, Y.-J. Hsu, S. Emmanouilidis, B. Khaleque, and R. Gaina, "Tribes: A new turn-based strategy game for ai research," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, vol. 16, no. 1, Oct. 2020, pp. 252–258. [Online]. Available: https://doi.org/10.1609/aiide.v16i1.7438

[11] H. Charlesworth, "Learning to play settlers of catan with deep reinforcement learning," Settlers RL GitHub Pages, April 3 2022. [Online]. Available: https://settlers-rl.github.io/

[12] C. C. Kim and A. Y. Li, "Re-l catan: Evaluation of deep reinforcement learning for resource management under competitive and uncertain environments," 2021, available online. [Online]. Available: https://cs230.stanford.edu/projects_fall_2021/reports/103176936.pdf

[13] P. McAughan, A. Krishnakumar, J. Hahn, and S. Kulkarni, "Qsettlers: Deep reinforcement learning for settlers of catan," 2018, available online. [Online]. Available: https://akrishna77.github.io/QSettlers/

[14] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. F. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, Ç. Gülçehre, H. F. Song, A. J. Ballard, J. Gilmer, G. E. Dahl, A. Vaswani, K. R. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," *CoRR*, vol. abs/1806.01261, 2018. [Online]. Available: http://arxiv.org/abs/1806.01261

[15] S. Munikoti, D. Agarwal, L. Das, M. Halappanavar, and B. Natarajan, "Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications," 2022. [Online]. Available: https://arxiv.org/abs/2206.07922

[16] S. Munikoti, B. Natarajan, and M. Halappanavar, "Gramer: Graph meta reinforcement learning for multi-objective influence maximization," 2022. [Online]. Available: https://arxiv.org/abs/2205.14834

[17] F.-X. Devailly, D. Larocque, and L. Charlin, "Ig-rl: Inductive graph reinforcement learning for massive-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, p. 7496–7507, Jul. 2022. [Online]. Available: http://dx.doi.org/10.1109/TITS.2021.3070835

[18] S. Berglund, "Modelling cyber security of networks as a reinforcement learning problem using graphs: An application of reinforcement learning to the meta attack language," Stockholm, Sweden, Jun. 2022.

[19] R. C. Team, *R: A Language and Environment for Statistical Computing*, 2025, software for Statistical Tests. [Online]. Available: https://www.R-project.org/

X. Andres Scrap Work

A. Value Function Experiment

The value function experiment's goal is to find the optimal set of inputs $x \in \mathbb{R}^n$ that parameterize a value function $V_x(s)$. n is the number of parameters that parametrize the value function and s is the state that the value function receives. Our goal is to vary the inputs and evaluate the results of games played. The results of each game is defined as $y \in \mathbb{R}^m$. m is the number of objectives we aim to achieve. We define a noisy function $f : \mathbb{R}^n \to \mathbb{R}^m$ that takes in the input x and outputs the results y.

In the current example building off the value function used by default in the `ValueFunctionPlayer` class there are $n = 13$ parameters to choose from.

- `public_vps`
- `production`
- `enemy_production`
- `num_tiles`
- `reachable_production_0`
- `reachable_production_1`
- `buildable_nodes`
- `longest_road`
- `hand_synergy`
- `hand_resources`
- `discard_penalty`
- `hand_devs`
- `army_size`

For simplicity in illustrating the first example, we take $y \in \mathbb{R}$ as the proportion of games won by the player. $f : \mathbb{R}^n \to \mathbb{R}$ is then a function that maps the input x to the proportion of games won y. In this case, we seek to maximize the number of wins by varying the parameters that define the value function.

In a more general case, y need not be a single valued function. In order to find competitive sub-strategies that not only pursue the primary objective (win), but also pursue secondary objectives such as (longest road), we scalarize this multiple objective problem. For example, take a $y \in \mathbb{R}^4$ defined as such:

- `win_proportion`
- `longest_road`
- `settlement_count`
- `largest_army`

To optimize this objective, we scalarize with a heuristic set of hyper-parameters $\lambda \in \mathbb{R}^4$. We then have the following unconstrained optimization problem:

$$\max_x \quad \lambda^\top y$$

Due to the complexity of game environments, we can not compute the derivatives of f. Therefore, this can only be solved through zeroth-order methods or estimation.

B. Graph Creation

This section involves my thoughts on how to structure the graph to best represent information. The current features that are labeled as `graph_features` are listed below:

- `EDGE<i>_P<j>_ROAD`
  - Whether edge i is owned by player j
  - $72 \times N$
  - Boolean
- `NODE<i>_P<j>_SETTLEMENT`
  - Whether player j has a settlement in node i
  - $54 \times N$
  - Boolean
- `NODE<i>_P<j>_CITY`
  - Whether player j has a city in node i
  - $54 \times N$
  - Boolean
- `PORT<i>_IS_<resource>`
  - Whether node i is port of $resource$ (or THREE_TO_ONE)
  - 9×6
  - Boolean
- `TILE<i>_HAS_ROBBER`
  - Whether robber is on tile i
  - 19
  - Boolean
- `TILE<i>_IS_<resource>`
  - Whether tile i yields $resource$ (or DESERT)
  - 19×6
  - Boolean
- `TILE<i>_PROBA`
  - Tile i's probability of being rolled
  - 19
  - Float

where N is the number of players and there are 5 $resource$ types. Representing this information in a graph where the nodes are building corners and the edges are roads has some advantages. Each of the 54 nodes would have a feature vector associated with it. Assuming 4 players, the node feature vector would look like:

- x_0: `NODE_ID`
- $x_1 - x_4$: `P<j>_SETTLEMENT`
- $x_5 - x_8$: `P<j>_CITY`
- $x_9 - x_{14}$: `IS_<resource>_PORT`
- $x_{15} - x_{50}$: `ADJACENT<k>_IS_<resource>`
- $x_{51} - x_{110}$: `ADJACENT<k>_DICE_VALUE`
- $x_{111} - x_{116}$: `ADJACENT<k>_HAS_ROBBER`

This representation is built to keep every feature as a one-hot encoding and maintain as much information as possible. There would be an edge between each node for a total of 72 edges. The feature vector for each edge would be a one-hot encoding of the owner road

- e_0: `EDGE_ID`
- $e_1 - e_4$ `P<j>_ROAD`

The overall graph vector **y** would be equal to the numerical features available in the environment.