Unravelling Complex Sequential Social Dilemmas: In a Risky World with A2C Decision Transformers

Ashish Panchal[1 2], **Dr. Vandith Pamuru**[1]

[1]Indian School of Business, Hyderabad, India
[2]Georgia Institute of Technology, Atlanta, USA
ashish_panchal@isb.edu, vandith_pamuru@isb.edu

Abstract

Real-world decision-making involves complex sequential social dilemmas (SSDs) where current reinforcement learning (RL) algorithms struggle due to high non-stationarity caused by dynamic interactions and conflicting goals, particularly in games like Risk, Civilization, and Diplomacy. To address this, we introduce a simplified "Risk"-inspired environment to study explainable AI in complex SSDs, retaining key features like stochastic outcomes and temporary alliances. Experiments show traditional RL methods (DDPG, A2C, PPO) underperform against basic bots in this environment, suggesting limitations in capturing opponent intentions from isolated states. We also explored A2C Decision Transformer (A2C-DT) that leverages temporal context, showing performance gains over traditional methods.

Introduction

Most systems in this universe can be modeled as isolated environments with interacting elements governed by rules. By design, these elements might have goal/s that may or may not align with others. Dilemmas emerge when elements' goals conflict.

Such dilemmas are prevalent across scales, from international diplomacy (economic, environmental, military) to firm-level interactions, sports teams, and even individual healthcare plans. Multi-agent RL, a promising approach, mirrors human social interactions and has seen success in addressing aspects of these dilemmas, particularly in well-defined environments with bounded state and action spaces (Li et al. 2019; Bao and Liu 2019; Lee and Lee 2021; Palanisamy 2020; Ning and Xie 2024). While research has shed light on the emergence of cooperation in competitive environments like tragedy of commons, traditional approaches have shown limited success in achieving human-level cooperation (Perolat et al. 2017). Initial MARL successes in promoting cooperation include PPO-like algorithms demonstrating collaborative herding in predator-prey situation (Ritz et al. 2021). (Leibo et al. 2017) further investigated cooperation and defection strategies in games like Harvest and Wolfpack, highlighting environmental influences on their emergence. (Hughes et al. 2018) showed that agent-specific properties like inequity aversion can foster cooperation, even in intertemporal social dilemmas, by incorporating human-like moral values into agents' decision-making, leading to temporary team formations for resolutions. Subsequently, (Jaques et al. 2019), explored social influence, using influence-based rewards and opponent modeling (LSTM policies) with A3C, demonstrating improved returns. However, they do not particularly quantify cooperation and temporal change in the agent's attitude.

Challenges of Real-world complexity and limitations of existing approaches

While these studies advanced RL/MARL for social dilemmas, they often rely on simplified, synchronous environments with limited real-world applicability. Real-life systems are often asynchronous, with agents acting at different frequencies due to constraints or strategies. Furthermore, real-world scenarios are dynamic and non-stationary; opponent strategies evolve over time and may require long temporal horizons to discern. This leads to state-action spaces and transition/reward functions that are challenging to fully estimate.

For humans, these events are quite common, and as a species, we have shown resilience and identified solutions for the toughest of such problems. Board games like "Diplomacy" and "Civilization" known for their combinatorial state-action spaces and relevance to real-world social systems. Initial research explored Diplomacy as a testbed. (Paquette et al. 2019) introduced Dipnet (GCN-based, human imitation learning, A2C self-play fine-tuning), demonstrating dynamic coalition formation against bots. However, Dipnet showed limited skill improvement and incentive optimization. (Anthony et al. 2020) hen proposed BRPI (GNN encoder, LSTM decoder, imitation learning), achieving better performance against Dipnet, winning 27.3% times against dipnet A2C and losing only 1.3% times. However, their work did not elucidate cooperative strategies.

In their work (Gray et al. 2020) introduced Searchbot, a one-step lookahead agent with regret minimization trained on human data, achieving strong performance in Diplomacy. (Bakhtin et al. 2021) developed DORA, a double oracle RL agent, achieving superhuman performance in 2-player Diplomacy without human data. However, they also showed self-play models diverge from human strategies in larger settings, suggesting limitations of pure self-play. They further

noted, Transformer-based models outperformed GCNs studied earlier (Paquette et al. 2019; Anthony et al. 2020; Gray et al. 2020). (Jacob et al. 2022), addressed human alignment by regularizing regret minimization with imitation learning (IL), improving upon Searchbot. (Bakhtin et al. 2022) generalized this approach, achieving top performance in human-AI Diplomacy competitions. (FAIR) further explored human alignment using LLMs for negotiation and coordination, reinforcing the need for human data to achieve human-aligned strategies in complex social games.

Similar approaches are being explored in Civilization, another complex strategy game involving long-term empire-building and player interaction.In a recent effort, (Qi et al. 2024) proposed Mastaba, a hierarchical LLM-agent architecture for Civilization. The model showed moderate performance compared to humans and failed to learn defensive strategies. While these results may cast doubts on the utility of the LLMs while using long-term in-context information effectively in the gameplay, since the models were not specifically trained for the environment, the actual capability of these models remains unclear.

Motivating Social Learning and Evolutionary Insights for AI Agents

While human value alignment is important for AI, it's not immediately obvious why AI policies must resemble human policies for effective cooperation. The primary AI goal is to optimize utility in environments with diverse agents, cooperative or not, similar to human deliberations. However, it is evident from the mentioned research that current AI models seem to lack a "human element", which might be crucial to solving the demanding societal challenges. Furthermore, the complexity of the explored social environments might be too high to learn from, evaluate, and understand the reasoning of actions in a social interaction.

Although early research focused on environmental and agent characteristics to explain cooperation and deception, subsequent work shifted to algorithm and architecture optimization, yet often without full success. Scaling models and data, performance factors in learning tasks, alone may be insufficient, potentially missing a key "social learning" component – learning with society, not just in society.

Evolutionary biology offers insights into social intelligence. Social Exchange Theory (Cosmides, Barrett, and Tooby 2010) suggests, the presence of human cognitive architecture incorporating specialized reasoning, shaped by natural selection to solve adaptive problems like social interaction and avoiding dangers. The Cultural Brain Hypothesis (Muthukrishna et al. 2018)) posits that the brain has evolved to acquire, store, and manage adaptive knowledge gained through social and individual learning. In particular, larger groups and inherited knowledge favor social learning.

Hypotheses, Tiny-Risk Environment, and A2C-DT Model

Based on the above provided arguments, we hypothesize:. (1) Current environments for studying cooperation are either too complex (Diplomacy, Civilization) or too simplistic for real-world social dilemmas (SSDs). (2)Evolutionary insights and initial MARL research suggest exploring algorithms that leverage social learning, modeling social exchanges and adaptive policies, potentially without relying solely on human data. (3) Inspired by (Cosmides, Barrett, and Tooby 2010; Muthukrishna et al. 2018; Ashton, Thornton, and Ridley 2018; Rosati 2017), we hypothesize that learning in large, dense social environments could result in improved performance when transferred to smaller-scale SSDs. We aim to validate if AI agents can develop and transfer such socially learned skills..

To address hypothesis 1, we first consider Risk (hasbrp.com 1993). While simpler than Diplomacy, Risk still presents complexity with unbounded actions, dynamic alliances, delayed rewards, and non-stationarity (Gordon 2013), making it suitable for negotiation studies (Marks 1998; Asal et al. 2014). Despite the success of AlphaZero-like models in games like Chess and Go (EXPTIME complete) (Tromp and Farnebäck 2006; Robson 1983; Zhang and Yu 2020; Fraenkel and Lichtenstein 1981; Adachi, Kamekawa, and Iwata 1987), initial Risk explorations with MCTS and self-play have been limited (Blomqvist 2020; Carr 2020; Olsson 2005; Tan 1997; Osborne 2003). Risk and Diplomacy's unbounded actions and negotiation possibilities lead to NP-hard Nash equilibrium and value estimation problems (Fatima, Wooldridge, and Jennings 2007; De Jonge and Sierra 2015; De Jonge and Zhang 2017) .The exponential branching factor creates a vast state-action space, hindering effective learning of average human-level complex strategies. Furthermore, player intentions are opaque over many turns, complicating strategy analysis. To address these challenges and enable strategy analysis, we introduce Tiny-Risk, a simplified Risk variant retaining core complexities but with constrained dynamics, detailed in the next section.

Initial experiments in Tiny-Risk show that standard RL methods like DDPG, A2C and PPO struggle to learn effective strategies, exhibiting low win rates even against random legal action bots.

To address hypothesis 2, we propose A2C-DT, an Advantage Actor-Critic Decision Transformer variant, designed for efficient and stable policy gradient optimization with a target network.

Expected Contributions and Future Directions: A Decision Transformer for Social Strategy Learning

Our main contribution is Tiny-Risk for studying adaptability in dynamic SSDs. Additionally, we also provide an initial analysis of using only RL training in DT.

A2C-DT leverages temporal trajectories of agent and opponent actions (without explicit opponent modeling) to intrinsically learn opponent strategies. We postulate that this enables dynamic alliance formation and emergent cooperation/defection, even without human data. Preliminary results show A2C-DT outperforms baselines in Tiny-Risk. We anticipate A2C-DT will excel against "personality" agents with predictable temporal patterns, leveraging Transformers' success in sequence modeling. The attention mechanism in DTs may allow for estimating opponent strategy horizons. Future work will analyze social interaction mech-

anisms, meta-strategies (cooperation/defection), and value function design, alongside validating hypothesis 3, and exploring models like RWKV and Mamba for enhanced long-range context modeling.

Tiny-Risk: Environment Description and workflow

Risk is a turn-based world conquest game where players eliminate opponents by controlling territories and deploying armies. In the process, players can make temporary alliances by directly communicating with other players on dedicated channels or by supporting actions in the game. To reduce the complexity, the game setting is restricted to no-press.

Setup and Game Dynamics

The game board comprises 4 continents build of 10 territories. with continental connections via 2 specific territories per continent. fig. 1 Each game cycle consists of sequential player turns with 3 phases:

A. Troop Reinforcement: Players start with 3 troops and receive a fixed number each cycle (unlike Risk's territory-based reinforcement), encouraging collaboration and strategic positioning.

Start of the game: Players place initial troops in unoccupied territories. Later players have positional defense advantages. Card-to-troop exchange is removed to emphasize cooperation. Attack outcomes primarily introduce stochasticity fig. 1b.

After setting the 1st territory: fig.(1c, 1d, 1e), Actions are bounded only by the number of spare troops and player strategy. Therefore, players can reinforce occupied territories by placing spare troops. If they have troops left after one action they can take multiple actions and spread out through territories or concentrate strategically for their next move. This flexibility increases game complexity due to an almost unbounded number of actions. With each action in a phase, reachable states grow exponentially, encouraging more thoughtful gameplay.

End of the game for the player: fig. 1f, Once all the occupied territories of a player are lost, they lose the game and cannot place new troops, even in the empty territories, even if they have spare troops. The game progresses with the remaining players.

B. Attacking: Following 1st phase, players can attack adjacent territories owned by opponents or those that are still unoccupied.

Attack dynamics: Attacks on adjacent territories require at least one troop. During the attack, the player selects territory, proportion of troops, and attack position. Combat outcome is probabilistic, determined by normal distributions based on attacker and defender troop counts (wider for attacker advantage, narrower for defender advantage). Losing attacks results in troop losses for both sides, equal to the minimum of attacking and defending troops. During winning attacks, with an attacker advantage, remaining troops are sent to the newly captured location, unless only 1 troop remains, then the defender territories is freed. Defender advantage, enables the attacker to incorporate risks in their strategies, if it wins the defender loses all its troops, and the attacker is left with 1 troop.

Similar to the 1st phase, the actions in the attack phase are only bounded by troops on the board; allowing multiple sequential attacks actions within a turn, and are the sole method for territory acquisition.

Old and new sources of Stochasticity: Stochasticity is introduced through normal distributions for combat outcomes, with better error and function estimation gaurenties, simplifying Risk's dice rolls uneven distributions. Further, Troop deployment to conquered territories is also simplified to leave one behind strategy.

C. Fortifying: Players can transfer armies between connected controlled territories for defense or attack preparation. Unlike Risk transfers are limited to directly connected territories, with only one action per turn, forcing them to find the path in a dynamically changing environment.

Additional Constraints

While dice control, Risk cards, continent bonuses, and acquisition of opponent resources from eliminated opponents from the game are removed as they only added to the theoretical stochasticity of the environment. To enhance gameplay by increasing exposure to starting and ending scenarios, additional restrictions have been implemented

Maximum number of troops in a territory: Although this feature is editable, the experiment limits maximum capacity to 50 per territory. This forces agents to use different techniques after they achieve a certain level of defense. This is specifically to avoid strategies that involve creating a single indefeasible territory that can result in a deadlock when multiple agents try the same, resulting in a draw, which is not the worst but neither the best outcome, but is certain, therefore can result in high confidence, followed by a restricted exploration of social strategies.

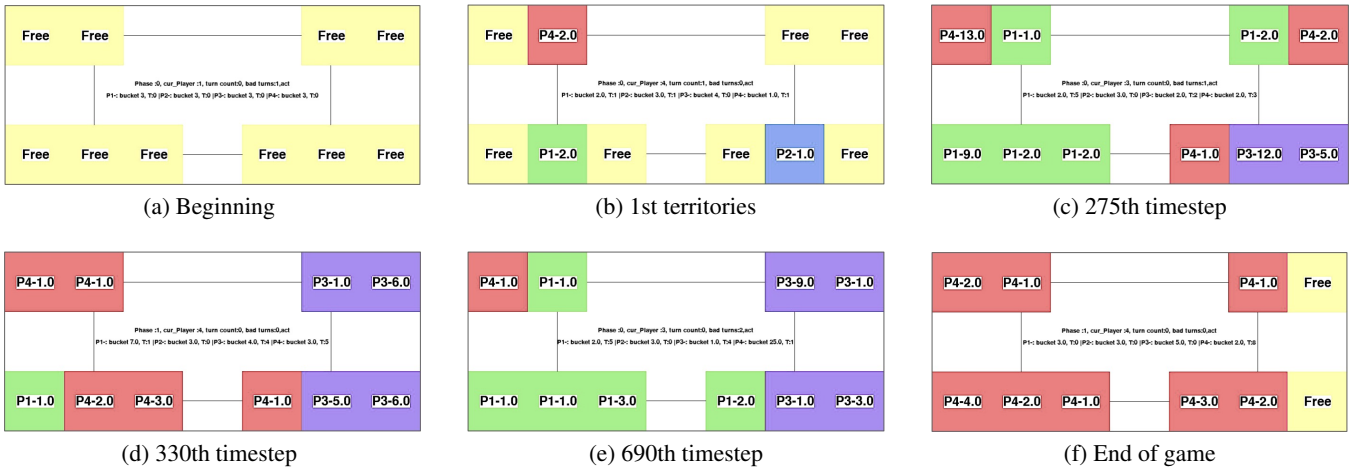
Time limit of the game: While the game can run for days, we created an editable feature, restricting the total number of steps per game episode to 3000 steps in our experiments. On average, a cycle could take 200 steps. Therefore we are forcing agents to learn strategies that can be achieved in 15 game cycles, restricting the search space.

Reward Function and Illegal Moves

In-game rewards: A reward of +1 is given for conquering every new territory. Conquering is defined by the presence of the player's troops at the new location. Additionally, -1 is given for losing a previously conquered territory.

End-game rewards: If a player is eliminated in the middle of a game, a -100 is rewarded, and at the end of the game, a winner is rewarded with a +100, and all losers again get a -100, which means the 2^{nd} best player only gets -100 and previously eliminated players get -200 in total apart from in-game rewards. In case of a draw by reaching the maximum time, all the players are given -100, resulting in all alive players sharing the 2nd place.

Penalties for illegal actions: A changeable variable is defined for taking any illegal move in the game. At every timestep, given the current territory acquisitions, allocated troops, spare troops, and phase, the possible actions that can



(a) Beginning (b) 1st territories (c) 275th timestep

(d) 330th timestep (e) 690th timestep (f) End of game

Figure 1: Tiny Risk: Gameplay and map, showcasing different stages of the game, with players' occupied territory and troops

be taken in the game are restricted. As these configurations change, so do the legal actions. Humans can make visual inferences, however, this is not the case for Agents, therefore the game board identifies and shares these actions as an action mask. However, given the model does not know how to use the mask, it can still make illegal moves. A predefined constant penalty is given for each such action. In our environment, we used -0.01, given a single player plays all the possible moves, for example, if the step limit per game is 3000, then a cumulative reward of -30 is accumulated in addition to the game draw reward.

Phase-Turn skip: Maximum possible illegal actions, as a changeable variable, are defined to ensure the game's progress for every phase of the turn in a given cycle. Once it is crossed, the game automatically moves to the next phase. In our experiments, we allowed the player to make 4 illegal actions per phase.

Environment Design

This subsection defines the state and action space, along with other observables.

State space: A 2D matrix of shape (10,2), each row corresponding to a location on the map, and the 1st column defines ownership of the territory, given by a number dedicated to the player, and the 2nd column represents the number of troops deployed.

Other observations: Specific information like the number of player's spare troops, along with the action mask depicting its legal actions for the timestep, are always visible to the player in their and other's turn. Common information such as the current phase, current agent, and current timestep is visible to all agents.

Hidden information: Information specific to other players is not visible, however is not difficult to estimate. Edges of the game are not directly visible and should be learned, given it is a static information it is not provided separately.

Action space: Each action is a pair of 2 variables, having different meanings during different actions. There are 32 possible action values for the 1st variable per timestep in any given phase; this is done to maintain uniformity of action space throughout the game. The 2nd variable is a continuous float between 0 and 1 inclusive.

- ***Positioning actions:*** For the 1st variable, the 1st 10 action values, starting with 0, correspond to distinct territories on the map, used in the 1st phase of the action cycle, the 31st action corresponds to moving to end the current phase, and the 32nd action for ending the turn, and actions between 11th and 30th are not legal. The 2nd variable corresponds to the proportion of spare players to be placed. The number of spare players is only bounded by the total number of cycles and can change dynamically. The proposed configuration can handle a large number of cycles.

- ***Transitory actions:*** Attack and fortification phases require troops to move from one place to another. The former required from conquered territory to opponent territory, for the latter both are conquered territories. For the 1st variable, actions between 11th and 30th inclusive, corresponding dedicated 20 unidirectional edges can be used, mapping to a specific form and to territory, 31st and 32nd action follows the same rules as positioning actions. The 2nd variable corresponds to the ratio of the troops to be sent. In case the proportion is 100%, the environment enforces restrictions to keep a single trooper and tries to send the remaining.

Action mask: The action space defines the legal actions for the agent per phase as a one-hot vector of length 32:

- ***Positioning actions:*** Its 10 values are set to 1 when the positioning is legal on the corresponding territory, i.e., either the player owns the territory, or the territory is unoccupied, and the player hasn't deployed its first troop, additionally the player has some spare troops, else these are 0. The 11th to 30th values are 0, and the 31st and 32nd actions are always legal, i.e., 1.

- ***Transitory actions:*** 1st 10 values are always 0 in the attack phase. Only those edge actions are legal where the "from" territory is owned by the player and has at least 2 troops, along with the "to" territory either occupied by an

opponent or empty. In the fortify phase, however, the "to" territory should be self-owned.

Game loop: The game loops through cycles sequentially through each player and all three phases for the players. Until either the time limit is reached or all players but one are eliminated from the map.

Model Architecture and Training pipeline

We employ causal transformer models with three heads per timestep: two action heads (a_t^1, a_t^2) and one value head (V_t). Action head a_t^1 outputs a 32-dimensional SoftMax distribution (aligned with the action mask), while a_t^2, predicts a single continuous value (0 to 1). (V_t) outputs an unbounded continuous value.

Given a context length k, the model receives sequences of states s_t, actions a_t^1, a_t^2, and return-to-go R_t (totaling $4k-3$ tokens up to s_t), predicting V_t and the distribution of . The most probable a_t^1 is chosen (randomly breaking ties). a_t^2 is then predicted autoregressively ,fig. 3, by updating the trajectory T_t and re-feeding it to the model, allowing the agent to act. This autoregressive step occurs once per timestep. Initially, return-to-go is set to the maximum possible return(110 in the case of Tiny Risk).

Notice, unlike DT (Chen et al. 2021), A2C-DT predicts the state value function (not return-to-go) (Zheng, Zhang, and Grover 2022; Yamagata, Khalil, and Santos-Rodriguez 2023; Wu, Wang, and Hamaya 2024; Liu et al. 2022) which is used for advantage estimation and loss calculation in self-play training. This approach can be extended to imitation learning. The model uses only the last predicted token, focusing on the current state in the sequence.

Training pipeline:

The training pipeline, fig. 2 consists of three phases: Experience, Data Preprocessing, and Model Update.

Experience phase: In each cycle, N trajectories for N episodes are recorded. For each episode, each agent maintains a context k running memory and tensors of episode observables. Agents observe state changes even during opponent turns, storing these trajectories (excluding opponent actions/masks) to differentiate timesteps with and without agent actions for loss calculation. While agents could potentially visualize opponent legal moves, we omit this to simplify the state space, expecting independent dynamics learning. Gradient analysis of opponent action influence is deferred to future work. At each timestep, normalized observables update the running memory for a_t^1, a_t^2, and V_t prediction. Initial training cycles use epsilon-greedy random legal action exploration, decaying over time. Episode records of observables, agent actions, and predicted values are then passed to data preprocessing phase.

Data Processing phase: This phase involves three steps: normalizing state observables per episode for stability; estimating discounted future return per action. Lastly, to minimize space utilization, each episode trajectory is converted to a data loader of a single trajectory that produces a sequence of k observables, rewards, and recorded predictions. A replay buffer of size X is maintained, replacing from oldest $X - N$ trajectories with new ones. This buffer acts as a data loader of data loaders, shuffling and providing episode data loaders during training.

Model update: The model trains on shuffled episode trajectories. Each trajectory yields a variable-sized batch (trajectory length $\times$ context length). Each batch is chunked into fixed-size mini-batches of size $W$for standardized updates. Mini-batch losses are normalized by the number of mini-batches per episode to account for varying trajectory lengths. A target network (delayed weights) predicts $V(s_{t+1})$ for critic loss estimation. Training model weights are copied to the target network every N episodes. AdamW optimizer with small weight decay and a cosine annealing scheduler (with warmup) is used for learning rate control.

Loss functions:

This section defines the Actor and Critic loss functions, adapted from Advantage Actor-Critic, calculated per mini-batch.

Actor Loss: Actor loss is computed only for mini-batch sequences ending with agent actions (predicted or explored). Mini-batches without agent actions only contribute to Critic loss. Due to the shared network backbone, Critic updates still improve feature representations. The A2C policy gradient is estimated as (Schulman et al. 2017; Mnih et al. 2016; Sutton et al. 1999), where τ is the trajectory history of length k:

$$\nabla_\theta J(\theta) \approx \sum \left[\nabla_\theta \log \pi_\theta(a|\tau) \cdot A(\tau, a)\right]$$

However, the action a is composed of autoregressively predicting a^1 and a^2. The joint probability of taking both actions a^1 and a^2, given trajectory τ, is:

$$\pi_\theta(a = \{a^1, a^2\}|\tau) = \pi_\theta(a^1|\tau) \cdot \pi_\theta(a^2|a^1, \tau)$$

Therefore, the log probability of the action reduces to:

$$\log \pi_\theta(a = \{a^1, a^2\}|\tau) = \log \pi_\theta(a^1|\tau) + \log \pi_\theta(a^2|a^1, \tau)$$

There are two policy gradients that can be estimated: one for $\hat{a}^1$ and $\hat{a}^2_{v2}$ predicted autoregressivly, and the second for the prediction of $\hat{a}^2$ given the actual action taken a^1. Let's denote the predicted actions as $\hat{a}^1$ and $\hat{a}^2$, and the actual actions as a^1 and a^2. Therefore:

$$LP_1 = \log \pi_\theta(a = \{\hat{a}^1, \hat{a}^2_{v2}\}|\tau)$$

$$= \log \pi_\theta(\hat{a}^1|\tau) + \log \pi_\theta(\hat{a}^2_{v2}|\hat{a}^1, \tau)$$

$$LP_2 = \log \pi_\theta(\hat{a}^2|a^1, \tau)$$

The total log probability (LP) is a weighted combination:

$$LP = \rho \cdot LP_1 + \beta \cdot LP_2$$

And the policy gradient becomes:

$$\nabla_\theta J(\theta) \approx \sum \left[\nabla_\theta LP \cdot A(\tau, a = a^1, a^2)\right]$$

The log probabilities can be expressed using cross-entropy (CE) and binary cross-entropy (BCE):

$$\log \pi_\theta(\hat{a}^1|\tau) = \text{CE}(\text{Logits}(\hat{a}^1), a^1)$$

$$\log \pi_\theta(\hat{a}^2_{v2}|\hat{a}^1, \tau) = \text{BCE}(p(a^2|\hat{a}^2_{v2}, \hat{a}^1), p(a^2|a^2, a^1))$$

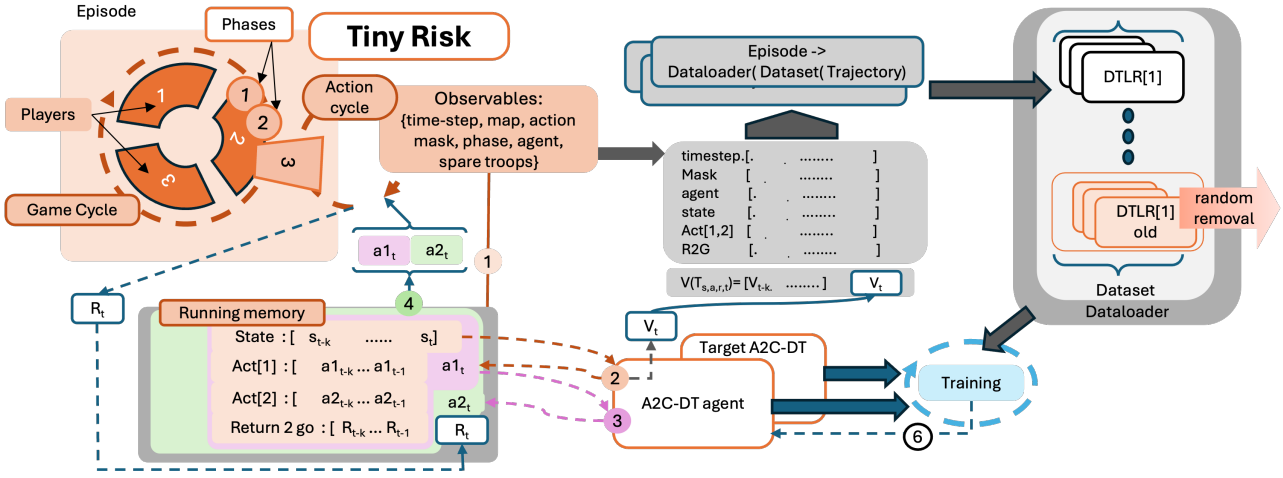


Figure 2: Workflow of A2C DT pipeline, depicting different stages training process

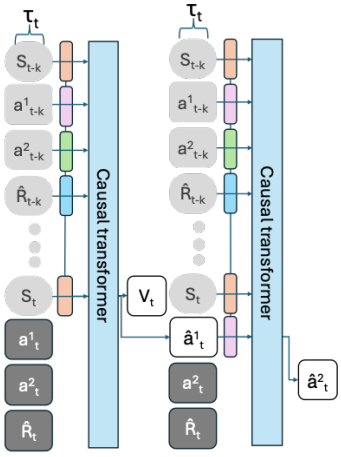


Figure 3: A2C-DT Autoregressive action prediction

$$\log \pi_\theta(\hat{a}^2|a^1,\tau) = \text{BCE}(p(a^2|\hat{a}^2,a^1), p(a^2|a^2,a^1))$$

Notice, the gradient of a_t^1 are retained a_t^2, The final policy gradient per minibatch is where M is total number of minibatch in an episode, and N is the number of sequences ending in the agent's action.

$$\nabla_\theta J(\theta) \approx \frac{1}{M}\sum_{i=1}^{W}\frac{1}{N_i}\left[\nabla_\theta LP^{(i)} \cdot A(\tau^{(i)},(a^1,a^2)^{(i)})\right] \tag{1}$$

Advantage A, is defined as(the discount return is precalculated in during data processing):

$$A(\tau^{(t)},(a^1,a^2)^{(t)}) = \sum_{k=0}^{T-t}\gamma^k r_{t+k+1} - V(\tau_t)$$

Critic Loss: Critic loss minimizes the difference between predicted value function $V_\theta(s)$ and actual returns. To account for rewards gained during opponent turns, critic loss predicts trajectory value at each timestep, initially defined as per (Mnih et al. 2016):

$$L_{\text{critic}} = \frac{1}{M \times W}\sum_{i=1}^{W}\left(R(\tau^{(i)}) - V_\theta(\tau^{(i)})\right)^2$$

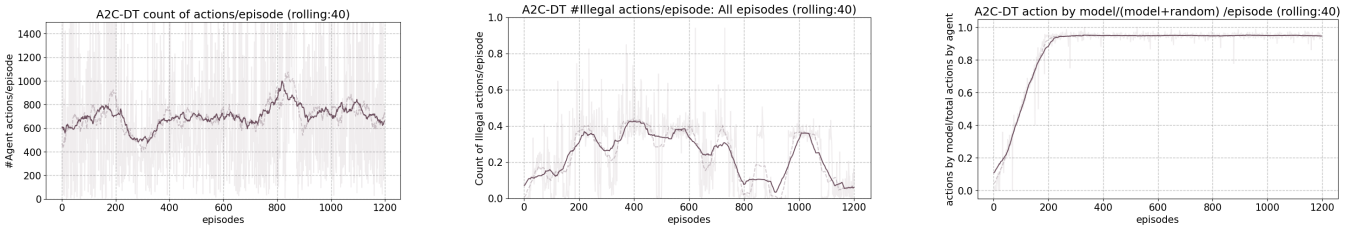
where $R(\tau^{(i)})$ is the actual discounted cumulative reward. To enhance stability (Barto and Duff 1993; Amiranashvili et al. 2018), we incorporate a one-step TD error term using a target network to predict value at t+1. Therefore, the following definition of the loss was used, where θ and ϕ represent the training and target models, respectively.

$$L_{\text{critic}} = \frac{1}{M \times W}\sum_{i=1}^{W}\Bigg[\lambda\left(r^{(i)} + \gamma V_\phi(\tau'^{(i)}) - V_\theta(\tau^{(i)})\right)^2 + \eta\left(R(\tau^{(i)}) - V_\theta(\tau^{(i)})\right)^2\Bigg] \tag{2}$$

Experiments and Results

This section covers the results of our preliminary experiments using A2C-DT on Tiny Risk. Most runs were conducted on a 3-player board with 10 territories. Our initial experiments focus on the learnability of the model, with A2C-DT competing against two random legal action agents with a time limit of 3000 steps per game.

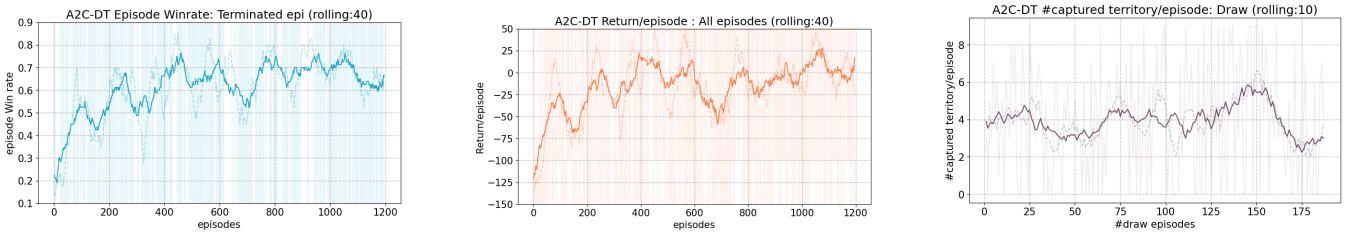
During training, 85% of episodes terminated, and the remaining 15% resulted in a draw. Close to the 200th episode, on average, 95% of the moves were made by the A2C-DT agent fig. 4c. A 5% action stochasticity was introduced to maintain environmental complexity. Note that as the branching factor per action cycle increases exponentially for a phase, the impact of 5% stochasticity per action cycle also increases similarly. As seen in fig. 4b, 4a, even with a penalty of -0.01 per bad action, the model learns to avoid



(a) Agent Action steps per episode

(b) Ratio of illegal moves per episode

(c) Ratio of agent to random actions



(d) Win rate: 1st position over time

(e) Return per episode: Draw & Terminated

(f) Count of captured territories : draw

Figure 4: A2C-DT behavioral characteristics and learning performance in 3 player TR, against random legal bots

them over time, even with a dynamic legality of the action space, which increases as the model tries to explore and take illegal actions. The number of total actions taken per episode also increases, suggesting the learning and execution of longer-term strategies.

As shown in fig. 4d, despite the stochasticity, the model achieves the highest 85% win rate in 1200 episodes against two random legal-action bots. In contrast, vanilla A2C and PPO models showed no visible improvement even after 5 million episodes and are therefore omitted due to the significant difference in training time and their poor performance. This suggests high sample efficiency for our model, a point we aim to validate in future work.

The model's returns exhibit instability, potentially due to four factors: **Cosine Annealing Learning Rate Scheduler**: While aiding escape from local minima, the cyclical nature of cosine annealing can introduce oscillations in prediction accuracy. Furthermore, cyclical learning rate increases can amplify optimizer momentum, causing overshooting and oscillations around optimal parameter regions. **High Gradient Variance:** The sparsity of rewards in Tiny Risk leads to large gradient fluctuations and, consequently, unstable prediction performance. **Off-Policy Updates:** The mini-batch-based data management and network updates introduce off-policy elements. This can cause instability, as the data collection policy may differ from the updated policy. **Limited Hyperparameter Tuning:** While transformers are generally robust across various configurations, the extensive training time required for validating each parameter set limited our exploration. Due to computational constraints, we used a smaller transformer model (3 blocks, dimension 64) and a replay buffer of only 20 episodes. Given the promising performance of this base model, further tuning is likely to yield improvements.

The results shown in fig. 4e, 4f, the moving average of Return per episode, including draws and terminated episodes, aligns with our earlier discussion on the increasing win rate, depicted by the rise in average rewards, alongside the oscillations explained previously. The average reward decreases in later episodes with more steps, resulting in draws. However, even in these drawn episodes, the average number of territories the agent controls increases.

This suggests the agent, aiming for a longer game, is attempting to expand its territorial control. While potentially a viable strategy, draws still result in a -100 reward, only partially offset by the +1 reward per territory occupied. The reward function, while defining the social dynamics of SSDs, also governs the learnability of effective strategies. Balancing these considerations within the reward function is a major design challenge and a key requirement for applying Tiny Risk to real-world problem reduction.

In experiments with a 10,000 timestep limit, the model began learning to create strongholds, leading to a higher frequency of draws. This, in turn, resulted in a substantial number of episodes with negative rewards, where the agent exhibited game suicide behavior. This phenomenon may indicate the limitations of a smaller model in navigating a highly complex environment, potentially due to restricted representational power. Larger models and longer training times, in conjunction with reward function redesign, could address this limitation.

Conclusion and Discussion

This research introduces Tiny Risk, a simplified variant of Risk designed to study cooperation in complex SSDs. Motivated by the limitations of existing MARL approaches in environments like Diplomacy and Civilization, TR balances complexity and tractability, enabling analysis of emergent

strategies and the potential of social learning algorithms. While retaining Risk's core strategic elements, TR constrains dynamics for manageable analysis.

To address the performance gap of existing MARL models in SSDs and explore social learning potential (second hypothesis), we developed A2C-DT, an A2C-based Decision Transformer architecture. A2C-DT learns from state change trajectories influenced by its own and opponents' actions, implicitly modeling opponent strategies without explicit modeling or prior human data. This framework is easily adaptable to incorporate human data in the future. Our results show A2C-DT's high sample efficiency, achieving an 85% win rate against random agents within 1200 episodes. This suggests effective social situation identification through observing opponent-driven trajectory changes, enabling efficient learning in SSDs. The transformer's attention mechanism likely facilitates handling the dynamic nature of strategy initiation.

Despite its success in this limited setting, A2C-DT exhibits performance oscillations, likely attributable to the cosine annealing learning rate schedule, high gradient variance, off-policy mini-batch updates, and computational constraints on hyperparameter tuning.

To fully address the second hypothesis, future work must explore techniques for optimizing model training, including advantage normalization, GAE, and alternative learning rate schedules to mitigate oscillations. Furthermore, we will incorporate algorithmic modifications to better capture the nuances of cooperation and competition in TR, studying dynamic relationship building, the impact of induced moral values in opponents, and dynamically changing environmental factors. This will involve evaluating A2C-DT against more sophisticated opponents, incorporating explicit communication, and analyzing emergent cooperative behaviors.

Experiments with longer game horizons revealed a behavioral bias, highlighting the limitations of smaller models in representing complex long-term strategies. This underscores the importance of representational capacity in solving SSDs, motivating future work with larger models and extended training.

To explore the impact of social group size on agent intelligence (third hypothesis), we will train A2C-DT in larger, denser Risk environments and assess transfer learning to smaller scenarios. Inspired by the cultural brain hypothesis, we hypothesize that experience in complex social settings will enhance performance in simpler ones. This research contributes to the development of more sophisticated, socially intelligent agents.

References

Adachi, H.; Kamekawa, H.; and Iwata, S. 1987. Shogi on n$\times$ n board is complete in exponential time. *Trans. IEICE*, 70: 1843–1852.

Agapiou, J. P.; Vezhnevets, A. S.; Duéñez-Guzmán, E. A.; Matyas, J.; Mao, Y.; Sunehag, P.; Köster, R.; Madhushani, U.; Kopparapu, K.; Comanescu, R.; et al. 2022. Melting Pot 2.0. *arXiv preprint arXiv:2211.13746*.

Amiranashvili, A.; Dosovitskiy, A.; Koltun, V.; and Brox, T. 2018. TD or not TD: Analyzing the role of temporal differencing in deep reinforcement learning. *arXiv preprint arXiv:1806.01175*.

Anthony, T.; Eccles, T.; Tacchetti, A.; Kramár, J.; Gemp, I.; Hudson, T.; Porcel, N.; Lanctot, M.; Pérolat, J.; Everett, R.; et al. 2020. Learning to play no-press diplomacy with best response policy iteration. *Advances in Neural Information Processing Systems*, 33: 17987–18003.

Asal, V.; Sin, S. S.; Fahrenkopf, N. P.; and She, X. 2014. The comparative politics game show: Using games to teach comparative politics theories. *International Studies Perspectives*, 15(3): 347–358.

Ashton, B. J.; Thornton, A.; and Ridley, A. R. 2018. An intraspecific appraisal of the social intelligence hypothesis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 373(1756): 20170288.

Babaeizadeh, M.; Frosio, I.; Tyree, S.; Clemons, J.; and Kautz, J. 2016. Reinforcement learning through asynchronous advantage actor-critic on a gpu. *arXiv preprint arXiv:1611.06256*.

Bakhtin, A.; Wu, D.; Lerer, A.; and Brown, N. 2021. No-press diplomacy from scratch. *Advances in Neural Information Processing Systems*, 34: 18063–18074.

Bakhtin, A.; Wu, D. J.; Lerer, A.; Gray, J.; Jacob, A. P.; Farina, G.; Miller, A. H.; and Brown, N. 2022. Mastering the game of no-press Diplomacy via human-regularized reinforcement learning and planning. *arXiv preprint arXiv:2210.05492*.

Bao, W.; and Liu, X.-y. 2019. Multi-agent deep reinforcement learning for liquidation strategy analysis. *arXiv preprint arXiv:1906.11046*.

Barto, A.; and Duff, M. 1993. Monte Carlo matrix inversion and reinforcement learning. *Advances in neural information processing systems*, 6.

Blomqvist, E. 2020. Playing the game of Risk with an AlphaZero agent.

Carr, J. 2020. Using Graph Convolutional Networks and TD ($lambda$) to play the game of Risk. *arXiv preprint arXiv:2009.06355*.

Chen, C.; Wu, Y.-F.; Yoon, J.; and Ahn, S. 2022. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*.

Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; and Mordatch, I. 2021. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34: 15084–15097.

Cosmides, L.; Barrett, H. C.; and Tooby, J. 2010. Adaptive specializations, social exchange, and the evolution of human intelligence. *Proceedings of the National Academy of Sciences*, 107(supplement_2): 9007–9014.

De Jonge, D.; and Sierra, C. 2015. NB 3: a multilateral negotiation algorithm for large, non-linear agreement spaces with limited time. *Autonomous Agents and Multi-Agent Systems*, 29(5): 896–942.

De Jonge, D.; and Zhang, D. 2017. Automated negotiations for general game playing.

Esslinger, K.; Platt, R.; and Amato, C. 2022. Deep transformer q-networks for partially observable reinforcement learning. *arXiv preprint arXiv:2206.01078*.

(FAIR)†, M. F. A. R. D. T.; Bakhtin, A.; Brown, N.; Dinan, E.; Farina, G.; Flaherty, C.; Fried, D.; Goff, A.; Gray, J.; Hu, H.; et al. 2022. Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, 378(6624): 1067–1074.

Fatima, S. S.; Wooldridge, M.; and Jennings, N. R. 2007. Approximate and online multi-issue negotiation. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, 1–8.

Fraenkel, A. S.; and Lichtenstein, D. 1981. Computing a perfect strategy for n$\times$ n chess requires time exponential in n. In *International Colloquium on Automata, Languages, and Programming*, 278–293. Springer.

Gopalakrishnan, A.; Irie, K.; Schmidhuber, J.; and van Steenkiste, S. 2023. Unsupervised learning of temporal abstractions with slot-based transformers. *Neural Computation*, 35(4): 593–626.

Gordon, D. 2013. Risk vs Diplomacy — cardboardrepublic.com. https://www.cardboardrepublic.com/classics/risk-vs-diplomacy. [Accessed 17-10-2024].

Gray, J.; Lerer, A.; Bakhtin, A.; and Brown, N. 2020. Human-level performance in no-press diplomacy via equilibrium search. *arXiv preprint arXiv:2010.02923*.

Han, M.; Zhang, L.; Wang, J.; and Pan, W. 2020. Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters*, 5(4): 6217–6224.

hasbrp.com. 1993. hasbro.com. https://www.hasbro.com/common/instruct/risk.pdf. [Accessed 17-10-2024].

Hu, S.; Zhu, F.; Chang, X.; and Liang, X. 2021. Updet: Universal multi-agent reinforcement learning via policy decoupling with transformers. *arXiv preprint arXiv:2101.08001*.

Hughes, E.; Leibo, J. Z.; Phillips, M.; Tuyls, K.; Dueñez-Guzman, E.; García Castañeda, A.; Dunning, I.; Zhu, T.; McKee, K.; Koster, R.; et al. 2018. Inequity aversion improves cooperation in intertemporal social dilemmas. *Advances in neural information processing systems*, 31.

Jacob, A. P.; Wu, D. J.; Farina, G.; Lerer, A.; Hu, H.; Bakhtin, A.; Andreas, J.; and Brown, N. 2022. Modeling strong and human-like gameplay with KL-regularized search. In *International Conference on Machine Learning*, 9695–9728. PMLR.

Jaques, N.; Lazaridou, A.; Hughes, E.; Gulcehre, C.; Ortega, P.; Strouse, D.; Leibo, J. Z.; and De Freitas, N. 2019. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International conference on machine learning*, 3040–3049. PMLR.

Kearns, M. J.; and Singh, S. 2000. Bias-Variance Error Bounds for Temporal Difference Updates. In *COLT*, 142–147.

Lee, H.-R.; and Lee, T. 2021. Multi-agent reinforcement learning algorithm to solve a partially-observable multi-agent problem in disaster response. *European Journal of Operational Research*, 291(1): 296–308.

Leibo, J. Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; and Graepel, T. 2017. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*.

Li, X.; Zhang, J.; Bian, J.; Tong, Y.; and Liu, T.-Y. 2019. A cooperative multi-agent reinforcement learning framework for resource balancing in complex logistics network. *arXiv preprint arXiv:1903.00714*.

Liu, H.; Huang, Z.; Mo, X.; and Lv, C. 2022. Augmenting reinforcement learning with transformer-based scene representation learning for decision-making of autonomous driving. *arXiv preprint arXiv:2208.12263*.

Marks, M. P. 1998. Using the Game of" Risk" to Teach International Relations. *International Studies Notes*, 11–18.

Melo, L. C. 2022. Transformers are meta-reinforcement learners. In *international conference on machine learning*, 15340–15359. PMLR.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T. P.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous Methods for Deep Reinforcement Learning. In *International Conference on Machine Learning*.

Muthukrishna, M.; Doebeli, M.; Chudek, M.; and Henrich, J. 2018. The Cultural Brain Hypothesis: How culture drives brain expansion, sociality, and life history. *PLoS computational biology*, 14(11): e1006504.

Ning, Z.; and Xie, L. 2024. A survey on multi-agent reinforcement learning and its application. *Journal of Automation and Intelligence*.

Olsson, F. 2005. A Multi-Agent System for playing the board game Risk.

Osborne, J. 2003. Markov Chains for the RISK Board Game Revisited. *Mathematics Magazine*, 76: 129 – 135.

Palanisamy, P. 2020. Multi-agent connected autonomous driving using deep reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–7. IEEE.

Paquette, P.; Lu, Y.; Bocco, S. S.; Smith, M.; O-G, S.; Kummerfeld, J. K.; Pineau, J.; Singh, S.; and Courville, A. C. 2019. No-press diplomacy: Modeling multi-agent gameplay. *Advances in Neural Information Processing Systems*, 32.

Perolat, J.; Leibo, J. Z.; Zambaldi, V.; Beattie, C.; Tuyls, K.; and Graepel, T. 2017. A multi-agent reinforcement learning model of common-pool resource appropriation. *Advances in neural information processing systems*, 30.

Qi, S.; Chen, S.; Li, Y.; Kong, X.; Wang, J.; Yang, B.; Wong, P.; Zhong, Y.; Zhang, X.; Zhang, Z.; et al. 2024. CivRealm: A learning and reasoning odyssey in Civilization for decision-making agents. *arXiv preprint arXiv:2401.10568*.

Ritz, F.; Ratke, D.; Phan, T.; Belzner, L.; and Linnhoff-Popien, C. 2021. A sustainable ecosystem through emergent cooperation in multi-agent reinforcement learning. In *Artificial Life Conference Proceedings 33*, volume 2021, 74.

MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ....

Robson, J. 1983. The Complexity of Go. volume 9, 413–417.

Rosati, A. G. 2017. Foraging cognition: reviving the ecological intelligence hypothesis. *Trends in cognitive sciences*, 21(9): 691–702.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R. S.; Barto, A. G.; et al. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12.

Tan, B. 1997. Markov chains and the RISK board game. *Mathematics Magazine*, 70: 349–357.

Tromp, J.; and Farnebäck, G. 2006. Combinatorics of go. In *International Conference on Computers and Games*, 84–99. Springer.

Van de Wiele, T.; Warde-Farley, D.; Mnih, A.; and Mnih, V. 2020. Q-learning in enormous action spaces via amortized approximate maximization. *arXiv preprint arXiv:2001.08116*.

Wen, M.; Kuba, J.; Lin, R.; Zhang, W.; Wen, Y.; Wang, J.; and Yang, Y. 2022. Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems*, 35: 16509–16521.

Wu, Y.-H.; Wang, X.; and Hamaya, M. 2024. Elastic decision transformer. *Advances in Neural Information Processing Systems*, 36.

Yamagata, T.; Khalil, A.; and Santos-Rodriguez, R. 2023. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*, 38989–39007. PMLR.

Zhang, H.; and Yu, T. 2020. AlphaZero. *Deep Reinforcement Learning: Fundamentals, Research and Applications*, 391–415.

Zheng, Q.; Zhang, A.; and Grover, A. 2022. Online decision transformer. In *international conference on machine learning*, 27042–27059. PMLR.

Appendix A.1 : Additional details on the Tiny-Risk Environment

A.1.1 Task Difficulty - State action space complexity

Risk shares similarities with almost unbounded state-action space complexity. In real-world scenarios, tasks often have prolonged natural or unnatural terminations, such as the death of an agent, aging, disease, natural calamities, or changes in economic policies. Similarly, by design, in the game Risk, there is no predefined termination condition in case a single player fails to conquer the entire world.

This implies the existence of at least three non-terminal gameplay scenarios. *First*, a purely defensive strategy focused on survivability, where an agent aims to outlast opponents by building an impenetrable stronghold. Theoretically, by deploying an infinite number of troops in a single territory over an infinite period, a player could create a practically impenetrable defense.

Furthermore, multiple players could adopt this strategy simultaneously. While this scenario is not a true stalemate, as it remains possible for other strong players to form an alliance and defeat such a player. Given an equal number of turns, an alliance of players would collectively possess more troops.

Additionally, gameplay can lead to a dynamic loop, where players are perpetually unable to gain sufficient strength to monopolize the game board. This is primarily because, unlike games like Go or Chess, the number of troops and territories owned in Risk are not fixed, *troops can be perpetually added and removed from the board*. These variable quantities are critical factors influencing game termination. Consequently, the action space is essentially unbounded.

By extension, the state space is not solely defined by territory ownership (e.g., for Tiny Risk: 3^{10} configurations for 10 territories and 3 players). Instead, it must also account for the varying number of troops in each territory. Even with a cap of 300 troops per territory in Tiny Risk, the total number of state configurations can be calculated as $(3 \times (300 + 1))^{10}$, which is approximately 3.6×10^{29} for a game board with 3 players and 10 territories (approximately 360 sextillion). Increasing the number of players by one increases the state space by a factor of approximately $17.75x$, and by two players by approximately $165.38x$. Similarly, increasing the number of territories by one increases the state space by approximately $903x$, and by two territories by approximately $815,409x$. The full game of Risk, if limited to a maximum of 300 troops per territory, 42 territories, and 3 players, would have approximately 1.3×10^{124} possible state configurations, which is about 3×10^{77} times larger than John Tromp's estimate for the state space of Chess. However, in a general, unrestricted game of Risk, the number of troops and even players can be further increased, exacerbating this complexity. Consider a player with 3 territories, 100 troops per territory, 2 possible actions per troop per territory, and 50 moves per turn. The *per-turn branching factor*, in this simplified scenario, is approximately $((2 \times 100)^3)^{50}$, and this factor increases dramatically with more moves, troops, and territories. This immense scale underscores the significant challenges Tiny Risk presents for traditional reinforcement learning approaches due to the vastness of its state and action spaces.

This potentially is one of the reasons we saw traditional models struggling even in a smaller setting of Tiny-Risk.

A.1.2 Task Difficulty - Non-Markovian Game Property

The game of RISK, particularly when considering the social dynamics between interacting agents (both human and AI), fundamentally violates the Markov property, posing significant challenges for the direct application of standard re-

inforcement learning (RL) algorithms. A Markov Decision Process (MDP), the cornerstone of many RL methods, relies on the assumption of memorylessness: the future state of the system depends solely on the current state and the agent's action. Past history is considered irrelevant once the current state is known. In social RISK, however, this assumption is demonstrably false, making the environment non-Markovian.

Partial observability and its violation The core issue is that the complete state of the game is not fully observable. While the board configuration (territory ownership and army placements) is visible to all, a crucial part of the state remains hidden: the internal states of the other agents. These internal states are not merely static parameters, but rather dynamic and evolving representations of:

- Reputation: Each agent holds internal beliefs about the other agents, quantifying their trustworthiness, aggressiveness, propensity for retaliation, and opportunism.
- Grudges and Alliances: Agents remember past interactions.
- Strategic Intentions: Agents have hidden goals and plans.

These internal states are interdependent and create feedback loops. An agent's action modifies not only the board state, but also the internal states of other agents. These altered internal states then influence the future actions of those agents, which feedback to affect the original agent.

Cyclical Dependency of Beliefs and Actions:

1 Agent A's Action: Agent A takes an action based on its internal beliefs and the board state.

2 Other Agents' Belief Update: Other agents (e.g., Agent B) observe A's action and update their own internal beliefs about A.

3 Other Agents' Action: Agent B, based on its updated internal beliefs about Agent A, takes an action.

4 Agent A's Belief Update: Agent A observes B's action and updates its own internal beliefs about B, completing the cycle.

Illustrative Example:

Consider two scenarios with identical board states: Bot A and Bot B are neighbors with equal armies.

- Scenario 1 (History: Neutral): No prior interaction. Bot B might fortify its borders.
- Scenario 2 (History: Betrayal): Bot A previously betrayed Bot B. Despite the identical board state, Bot B's internal state (high grudge, low trust) will likely make it attack Bot A preemptively.

This difference, from the same observable state but different histories and internal states, shows the non-Markovian nature. The future is not conditionally independent of the past, given the present observable state.

This presents a substantial obstacle for standard RL. Partially Observable MDPs (POMDPs) address hidden state, but make a crucial simplifying assumption: the underlying state transitions, while hidden, still follow a Markov process. This means that even though the agent doesn't see the true state, the probability distribution governing how that true state changes from one step to the next is static and depends only on the current true state and the agent's action.

In RISK, this is not the case. The "underlying state" (which includes all agents' internal beliefs) changes in a way that does not have a static probability distribution. The probability of an opponent attacking is not a fixed number; it depends on their dynamically changing internal state (reputation, grudge, etc.), which is influenced by the entire history of interactions and, crucially, by the learning agent's own actions. This creates a non-stationary environment where the rules of the game, from the agent's perspective, are constantly changing. The observation the agent receives, which is tied to the underlying interal state, no longer follows a simple, consistent probability distribution. Therefore the POMDP assumption is violated.

This requires sophisticated opponent modeling, potentially incorporating Theory of Mind, and handling non-stationarity and the complex feedback loops, posing a substantial challenge for traditional RL.

NOTE: While we restricted our competitors to be random legal agents, which highly restricts this challenge, it is expected during SSD experiment designs, the experimenters would develop rule-based or learning agents to mimic different human player personalities. We intend to develop and open-source reusable human personality agents in future work.

A.1.3 Task Difficulty - Non-stationary

The game of RISK presents a profoundly non-stationary environment for reinforcement learning agents, exceeding the challenges typically encountered in traditional non-stationary settings. In standard RL, non-stationarity often arises from changes in the environment's dynamics, such as shifting reward functions or transition probabilities that are external to the agent's actions. In RISK, however, the non-stationarity is endogenous, driven primarily by the interactions between multiple intelligent agents, each with their own evolving strategies and hidden internal states. Several key factors contribute to this:

Drastic State-Space Changes: Unlike environments where state transitions are primarily driven by the learning agent's actions or gradual environmental shifts, RISK's state space can undergo large, unpredictable changes between an agent's turns due to the actions of other players, in their turn. These changes are not simply stochastic environmental noise; they are the result of strategic decisions made by other agents, making them difficult to model as a fixed probability distribution. This is fundamentally different from, say, a changing weather pattern in a navigation task, where the changes, while non-stationary, are typically independent of the agent's choices. Additionally, in Risk this impact is further amplified when opponents change their strategies, i.e. during a game players can intentionally change their long term behaviour, which renders the current perception of others, based on historical interactions, meaningless.

Dynamic Player Count: The number of players in RISK is not fixed. As players are eliminated, the fundamental dynamics of the game change dramatically. A strategy that is

optimal in a six-player game may be disastrous in a three-player game, as the balance of power, potential alliances, and threat landscapes are completely altered. This is a discrete, structural change in the environment, not just a gradual parameter shift.

Asynchronous and Intermittent Reward Signals: In many RL environments, rewards are received directly after the agent's actions. In RISK, however, an agent receives rewards (e.g., gaining or losing territories and armies) throughout the entire round, even when it is not the agent's turn to act. This means that the total reward received after a single action is not solely a consequence of that action; it is also heavily influenced by the intervening actions of all other players, a factor that is both unpredictable and dependent on their evolving strategies. The length and complexity of these intervening actions compound this effect, creating a highly variable and delayed reward signal that is difficult to attribute to specific actions.

These sources of non-stationarity are particularly challenging because they are directly tied to the strategic behavior of other intelligent agents. The environment is not simply changing randomly or according to some pre-defined schedule; it is being actively reshaped by the decisions of opponents who are themselves learning and adapting. This creates a complex, interdependent, and highly dynamic learning problem that goes beyond the typical assumptions of non-stationary RL environments.

A.1.3 Design Choices and Simplifications

As described in the main text many simplifications are added to Tiny Risk, these simplifications are intentional and crucial for isolating and analyzing complex social interactions. Our primary goal is not to replicate the full operational complexity of a real-world environment but to provide a tractable and interpretable testbed for studying sequential social dilemmas. That is, by reducing extraneous complexities (e.g., detailed combat mechanics), Tiny Risk allows us to concentrate on how agents learn to navigate these social dilemmas, understand opponent strategies, and adapt. The simplified environment facilitates clearer analysis of agent behavior and promotes reproducibility, which is essential for foundational research in this area.

Simplified stochasticity model To improve the tractability of reinforcement learning in RISK, we introduce a simplified stochasticity model in our "Tiny RISK" environment, replacing the traditional dice rolls and card draws with two normal distributions. This design choice is motivated by several factors:

Reduced Complexity, Increased Focus on Strategic Interaction: The original RISK game incorporates multiple sources of randomness:

- Battle Resolution: Dice rolls determine combat outcomes, with a maximum of five dice (three for the attacker, two for the defender) and a complex probability distribution for troop losses.
- Card Rewards: Players receive cards that can be exchanged for variable numbers of troops, introducing uncertainty in resource acquisition.
- Card Inheritance: Eliminating a player grants the victor the defeated player's cards, adding an unpredictable influx of resources.
- Territory-Based Reinforcements: The number of troops received each turn depends on the number of territories controlled, which is itself subject to the actions of other players.

These intertwined stochastic elements make it exceedingly difficult to isolate the impact of strategic decisions from the influence of chance. The resulting variance in rewards significantly dilutes the credit assignment process, hindering effective learning. By replacing these with two normal distributions (please refer section 2 of the main text), we create a controlled source of randomness that allows us to focus on the core strategic challenges of the game: opponent modeling, long-term planning, and social interaction.

Analytical Tractability and Robustness: Normal distributions are well-understood and have desirable mathematical properties. A vast body of work exists on learning and optimization in the presence of normally distributed noise. This provides stronger theoretical guarantees for convergence and robustness compared to the complex, discrete, and multi-faceted randomness of the original game. The central limit theorem suggests that even the combined effects of multiple dice rolls tend to approximate a normal distribution, making this simplification a reasonable, albeit imperfect, approximation. Using the normal distribution, a defined mathematical equation, makes analysis easy.

Concentrated Stochasticity: Rather than having randomness injected at multiple points (battles, cards, territory control), we concentrate the stochasticity into a single, well-defined mechanism within the battle resolution. This allows for a clearer attribution of outcomes to strategic choices, while still maintaining an element of uncertainty that prevents the game from becoming fully deterministic. It helps focus the training procedure.

In summary, while the original RISK's randomness adds to its realism and excitement as a game for human players, this complexity hinders the application of RL techniques. Our simplification using normal distributions strikes a balance between retaining some stochasticity (essential for a non-deterministic game) and creating a more analytically tractable and learnable environment, allowing us to isolate and study the core strategic challenges of multi-agent interaction in RISK.

Micro-action and Reward Shaping A key aspect of our Tiny RISK environment is the configurability of the micro-action limit and the reward shaping parameters (action penalty and illegal action penalty). By varying these parameters, we can investigate their impact on agent learning, emergent strategies, and social dynamics. This enables us to explore not just how agents learn to play RISK, but also how the environment itself shapes the learning process.

To ensure computational tractability and encourage efficient learning, we introduce two key modifications to the standard RISK game structure in our Tiny RISK environment. These modifications are designed to streamline the game while preserving its core strategic elements:

Micro-Action Limit (Configurable): While the average RISK game length is around 15 full turns, significantly longer games are possible. To prevent computationally expensive "runaway" games and encourage efficient exploration, we impose a configurable limit on the total number of micro-actions an agent can take. A micro-action is defined as a single atomic action, such as attacking a territory or fortifying troops. The default limit is set to 3000.

Reward Shaping (Configurable): The original RISK game provides a sparse reward signal: +1 for winning and 0 (or -1) for losing. This makes learning extremely challenging. To facilitate learning, we introduce configurable reward shaping:

- Illegal Action Penalty: A small negative reward is given for each micro-action. The default value is -0.01, encouraging efficient solutions. Varying this value results in different strategic exploration and development. If an agent reaches the micro-action limit (primarily through repeatedly attempting illegal actions for 3000 steps), it accumulates a large negative penalty of -30, strongly discouraging unproductive behavior. This is an extreme case.
- Terminal Reward: The original win/loss reward for Risk is (+1/-1 or +1/0), however the size of this reward can be varied to study the associated impact on learning process and stability.

A.1.4 Uniqueness of Tiny Risk, and why is it a necessity: comparison with existing environments

Tiny Risk offers several key advantages for modeling real-world problems, compared to SSD environments heavily studied in literature, such as Harvest and Cleanup. Three crucial factors enhance its applicability:

Rich Social Dynamics: Unlike the primarily competitive environment of Harvest and the mixed cooperative-competitive setting of Cleanup, Tiny Risk offers a unique blend. While primarily competitive, the extended timeframe and single-reward structure incentivize cooperation due to high long-term uncertainty. These alliances differ significantly from the short-term cooperative behaviors seen in Cleanup's teaming variants. The longer interaction horizon in Tiny Risk allows for exploring nuanced social dynamics like deception, forgiveness, resentment, reciprocity, negotiation, niche behaviors (e.g., competitive exclusion), and even stubbornness. Multifaceted interactions across different territorial fronts, where players might cooperate with some opponents while competing with others, further enrich the diplomatic landscape. While Melting Pot 2.0's substrates exhibit some of these elements, no single environment combines them all as Tiny Risk does (Agapiou et al. 2022). Importantly, Tiny Risk's player-driven environmental changes promote transitions between different interaction types, unlike Harvest's static environment. Cleanup captures some of this dynamism, but its short reward horizons limit the complexity of emergent behavior.

Unified, Long-Term Objective: A key driver of Tiny Risk's social dynamics and uncertainty is its single, long-term reward structure. In contrast, the multiple in-game rewards of Harvest and Cleanup allow for partitioning the game into smaller, matrix-like subgames, as demonstrated in previous research. This translates to repetitive game dynamics, which, while useful, limits the emergent complexity compared to a single, overarching objective. This simplification may not adequately capture the complexities of real-world sub-tasks or long-term goal-oriented social interactions. These factors necessitate modeling evolving opponent strategies within each game episode.

State-Action Dynamics: Harvest, Cleanup, and similar environments compel action at every timestep within a defined ecosystem, assuming all players act with a bounded set of predefined actions. This model doesn't accurately reflect real-world scenarios for several reasons:

- Multitasking: Humans often work on multiple tasks concurrently, choosing to act or not on each task at different times. One person's activity doesn't preclude another's progress on a different task. Turn-based games like chess and Go, while simplifying simultaneous actions, still restrict players to single actions defined by the action distribution.
- Skill and Time Disparities: Real-world actors exhibit varying skill levels and time constraints. Some individuals can perform multiple actions while others manage only one. Furthermore, performance can fluctuate even within a single skill domain due to various factors.

Risk offers a more general model with long sequences of micro-actions per turn. Other players can observe the changing world state but not the opponent's specific actions. A simplified real-world analogy is a court trial, where the defense and prosecution take turns asking multiple questions, influenced by skill and circumstance, with limited explicit cooperation. Tiny Risk's action space offers further nuances:

- Dynamic Action Legality: Unlike Harvest's fixed legal actions and Cleanup's environment-dependent legality, Tiny Risk's legal actions depend on both the environment (territory ownership, troop deployments) and the game phase. The current phase dictates the possible action types.
- Context-Dependent Action Meaning: The same action can have different effects depending on the game phase. For example, reinforcing a territory versus attacking it. This context-based polysemy, abundant in real life (e.g., a back slap as praise or punishment), isn't fully captured in other environments, including Melting Pot. Forced task partitioning in Cleanup, while introducing some context, doesn't alter the fundamental meaning of actions.
- Action Complexity: Tiny Risk's actions incorporate degrees of application, reflecting human sophistication. The "right" amount of action, not just the type, matters. Physics-based environments, such as Pistonball, explore this concept but with limited action types.

Comparison with GO and Chess While games like Go and Chess have served as landmark challenges for AI, they represent a specialized case of turn-based gameplay that RISK significantly generalizes. Classic board games exhibit two key limitations. First, they possess a fixed action distribution, predetermining the complexity of each move.

While techniques like allowing sequences of micro-actions can vastly increase the effective complexity and enable near-human performance, this complexity remains fundamentally bounded by the game's inherent rules. Second, Go and Chess enforce a rigid alternation of turns, forcing a response after every single opponent action. This contrasts sharply with real-world scenarios, where agents often engage in multi-tasking, operate under flexible time constraints, and do not adhere to a predefined schedule of actions with fixed complexity. Furthermore, skill levels, both between different players and even within a single player across different tasks or time periods, exhibit considerable variance.

Appendix A.2: Additional details on A2C-DT

Motivation for A2C-DT

Motivation to use a transformer model: The choice of a Transformer architecture for our RISK agent, as opposed to traditional neural networks like multi-layer perceptrons (MLPs) or recurrent neural networks (RNNs), is motivated by several key advantages, particularly in the context of this complex, multi-agent, partially observable, and non-stationary environment:

- *Contextualized Representations and Implicit Opponent Modeling:* The self-attention mechanism allows the Transformer to create contextualized representations of states and actions. The representation of a particular action is influenced by all other states and actions within the context window, including those of other players. This provides a richer, more informative representation than feedforward networks (which treat each input independently) or RNNs (which have a limited, sequential context). This implicit consideration of other players' actions is a form of intrinsic opponent modeling, allowing the agent to better anticipate and react to opponent strategies, mitigating the challenges of partial observability. Since this representation/constructs created inherently in the network from observations and value-based optimization, they are potentially better than handmade Reference (Indirectly Related - Opponent Modeling): [Albrecht, S. V., & Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. Artificial Intelligence, 258, 66-95.]relationship features.
- Improved Sample Efficiency: By capturing long-range dependencies and creating richer representations within the context window, Transformers can often learn effectively from fewer training examples than RNNs or MLPs. This was observed in our experiment as well.
- Handling Non-Stationarity (Indirectly): While Transformers don't directly solve non-stationarity, their ability to capture long-range dependencies and contextual information within the context window makes them more robust to these changes. Additionally, by considering a wider context within the window, the agent is less likely to be thrown off by short-term fluctuations in opponent behavior.
- Parallelization: Unlike RNN, transformer allows for parallel processing, which helps in faster learning.

Existing model architectures Recent transformer models in RL often focus on representation learning or world models (transition/reward functions) as assistive frameworks, rather than pure reinforcement learning, (Chen et al. 2022; Gopalakrishnan et al. 2023; Melo 2022). For instance, the original offline DT (Chen et al. 2021) acts as a transition function model, stitching observed offline trajectories through supervised learning. While capturing causal sequential transitions, it doesn't address the value of individual actions through bootstrapped value functions, deviating from theoretical Bellman optimality, though furthering the recent successes of supervised sequential modeling. DTQN (Esslinger, Platt, and Amato 2022) and UPDET (Hu et al. 2021) explore Q-value learning with transformers, but transformers' sample inefficiency poses challenges in complex state-action spaces. A2C's on-policy learning, advantage function, and target network reduce variance and stabilize training, minimizing oscillations, (Babaeizadeh et al. 2016; Han et al. 2020). While not discounting the potential of more stable algorithms like PPO, we prioritized faster debugging and analysis in this initial work. Unlike MAT (Wen et al. 2022), DTQN, and UPDET, A2C-DT's critic updates use both TD loss and supervised sequential learning, modeling value changes from all changes in the environment, even when the model is not active. This is particularly useful to mitigate the non-stationary nature of single-delayed rewards multiagent environment. Furthermore, In A2C-DT we modeled causally dependent components of micro-actions from different distributions within a single framework, generalizable to any deep RL setting, which was either tackled by separate networks, independence assumption or learning to search for best action (Van de Wiele et al. 2020).

Overcoming limitations of offline Decision Transformer

While our work primarily focuses on Tiny-Risk environment, to better model a performant agent we derived A2C Decision Transformer (A2C-DT) architecture for online reinforcement learning in complex, multi-agent environments like RISK. Traditional Decision Transformer (DT) is typically trained offline on pre-collected datasets, often employing value pessimism to mitigate distributional shift issues inherent in offline RL. While effective in batch settings, these approaches can struggle with the limited and evolving state-action distributions encountered during online learning. Our A2C-DT overcomes these limitations by incorporating a process of updating batches of experiences using a replay-buffer and an actor-critic framework, enabling online training and active exploration. Unlike sequence-modeling-only approaches like the original DT, which rely solely on supervised learning to "stitch" together successful trajectories, our model integrates the strengths of both value-based learning (through the A2C component) and sequential modeling (through the transformer component). The actor-critic framework provides crucial value estimates for guiding policy improvement, mitigating the reliance on purely supervised imitation of past actions. The transformer architecture, with its attention mechanism, captures long-range dependencies and contextual information within the trajectory,

enabling a more informed "look-ahead" and "look-back" capability compared to traditional recurrent or feedforward networks. This combination allows for both effective exploitation of learned value functions and efficient exploration guided by the sequential context.

Extended reason for Value-based learning: A critical challenge in RL, particularly in games with long horizons and delayed rewards like RISK, is the credit assignment problem. This refers to the difficulty of determining which actions, within a long sequence, were responsible for the eventual outcome (win or loss).

The RTG method in the original DT, could face severe limitations in highly non-stationary long-horizon game like Risk:

- Delayed and Sparse Rewards: In RISK, the primary reward signal ($+1 \times k$ for winning, 0 or $-1 \times k$ for losing, where k is some constant) is only received at the very end of the game. This means that the RTG for actions taken early in the game is based on a single, distant outcome. The original paper explores this idea in a stationary robot environment problem. Further exploration and comparative analysis is required to ascertain the utility of offline DT in Risk-like environment.
- Noisy Signal: The RTG for an early action is an extremely noisy signal of that action's true value. A single good move early on can be followed by many mediocre or even bad moves, yet the final reward (and thus the RTG for all actions in the sequence) will be the same. Conversely, a poor early move might be compensated for by later good moves. The RTG would face difficulties in distinguish between these, added with high non-stationarity of the environment.

Finally, RTG based-learning in DT is effectively a Monte Carlo (MC) estimation. MC methods tend to exhibit higher variance than Temporal Difference (TD) methods (Sutton, Barto et al. 1998)(chapter 6,7), (Kearns and Singh 2000). This is because the MC update target (the full return) is a sum of many random variables (rewards), and the randomness accumulates over the entire episode. TD, by bootstrapping from the estimated value of the next state, relies on a shorter-term, less variable signal. This effect is prominent in long-horizon, stochastic environments with delayed reward like in RISK.

However, Off-Policy TD Learning methods (like Q-learning) can have significantly higher variance than on-policy methods. Therefore, to control the variance we chose on-policy A2C learning.

Design Modification in RL algorithms for Tiny Risk

The subsection describes the model modification used to adapt DDPG, A2C and PPO algorithms to work with Tiny-Risk's hybrid action space (Discrete, Continuous).

Dynamic N-step Return : A significant challenge in applying reinforcement learning to RISK is the game's long sequence length and cyclic, turn-based gameplay. The agent only takes consecutive actions during its own turn and remains inactive while other players act. However, the agent continues to receive rewards during these intervening periods (e.g., due to other players attacking its territories or the territories of its opponents). This asynchronous reward structure makes credit assignment difficult, particularly for one-step or fixed n-step TD learning methods. To mitigate this, we employ a dynamic n-step return during training.

Dynamic N-step return, $R_n(t)$ at timestep t, adapts to the asynchronous and cyclical nature of the game. The value of n represents the number of steps until the agent's next opportunity to act. During the agent's turn, n=1 for most micro-actions, reflecting the immediate effect of the agent's choices. However, for the final micro-action of the agent's turn (in the third phase), n becomes equal to the total number of micro-actions taken by all other players before the agent's next turn begins. If the agent is eliminated during the game, n capture the entire length of the gameplay post the agents's last action. This captures the cumulative effect of other players' actions on the agent's reward.

The dynamic N-step return is calculated as follows:

$$R_n(t) = \sum_{i=0}^{n-1} \gamma^i \times r(t+i) \tag{3}$$

where:

- $R_n(t)$ is the dynamic N-step return at timestep t.
- n is the dynamic step size, as described above.
- i the iterator.
- $\gamma$is the discount factor.
- $r(t+i)$ is the reward received at timestep t+i.

By using this dynamic N-step return, we provide a more accurate and informative training signal for the agent, bridging the gap between its actions and the delayed, asynchronous rewards influenced by other players. This improves credit assignment and facilitates more effective learning in this complex, multi-agent environment.

Notice, as calculating $R_n(t)$, requires a dynamic look ahead, the processed episode interactions are added to the replay buffer only after completion of each episode.

Loss functions - DDPG

TD_loss_critic: Adapting $R_n(t)$ in the Q' equation, we calculate the TD Q-value loss for the critic as follows:

$$DDPG\_TD\_loss_{critic} =$$

$$mse\Big[R_n(t) + \gamma^n \times Q_{target}(s_{t+n}, a'^1, a'^2),$$

$$Q_{current}(s_t, a^1, a^2)\Big]$$

where:

- $R_n(t)$ is the dynamic N-step return at timestep t.
- $\gamma$is the discount factor.
- s_{t+n} is the state observed at time t+n
- $a'^1, a'^2)$ are the on policy actions taken at time t+n.

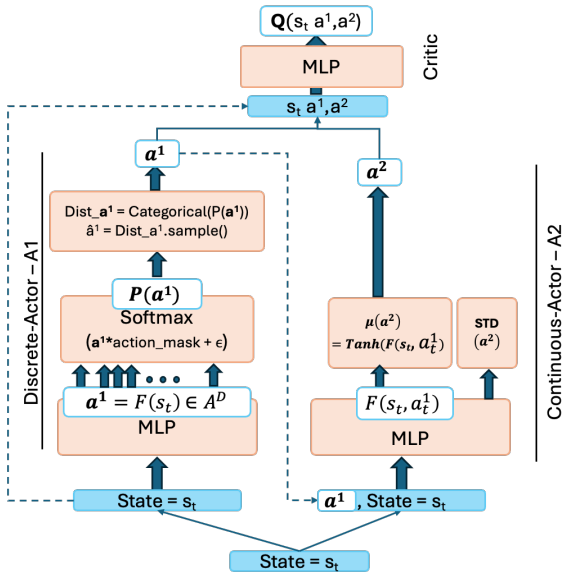


Figure 5: Modified DDPG actor-critic prediction flow and summarized architecture

- $Q_{target}(s_{t+n}, a'^1, a'^2)$, is the q value estimate at state-actions triplet s_{t+n}, a'^1, a'^2, using target critic network, and is 0 for the last state of the game.
- $Q_{current}(s_t, a^1, a^2)$ is the q value estimate at state-actions triplet s_t, a^1, a^2 using current network.

Policy loss: The policy loss is calculated for the actions taken, and it follows the same calculation as the original DDPG:

$$a^1 = Actor_1(s_t)$$
$$a^2 = Actor_2(s_t, a^1)$$
$$DDPG\ Policy\ loss = -Q_{current}(s_t, a^1, a^2)$$

Because the two sub-actions (a^1, a^2) are predicted by separate networks, and due to the nested dependency of the Q-function on a^2 (which itself depends on a^1), we use two distinct optimizers, one for each sub-action's network. Back-propagation requires two steps, retaining the computational graph, as the gradient flows to Actor_1 through two paths: directly via the Q-function and indirectly via Actor_2. Figure 5 details the modified prediction flow of the two different actor networks and the critic.

Loss functions - A2C

TD_loss_critic: Similar to DDPG we estimate the TD Loss for critic using $R_n(t)$, as follows:

$$V_{target}(s_t) = R_n(t) + \gamma^n \times V(s_{t+n})$$
$$V_{predicted}(s_t) = V(s_t)$$
$$A2C\_TD\_loss_{critic} =$$
$$mse\Big(V_{target}(s_t), V_{predicted}(s_t)\Big)$$

Notice, as A2C uses a state-value function (fig. 6) instead state-action-value function, its estimation is not action-dependent and therefore can be made for every time-step, even during the opponent's turn. However, we still use Dynamic n-step estimation in our calculations, as it enables faster information propagation while reducing biases present in 1-step TD learning (Kearns and Singh 2000). Also note, that a target critic network is not utilized in A2C (Mnih et al. 2016). For further learning stabilization, gradient clipping was employed.

Policy_loss: The two losses are calculated the two actor networks, respected, which are updated using a policy gradient method, guided by the advantage function. The advantage Adv is calculated as follows:

$$Adv_t = V_{target}(s_t) - V_{predicted}(s_t)$$

Note that A is detached from the computational graph to prevent gradients from flowing back through the critic during the actor updates.

The policy loss for each actor is then calculated as the negative of the expected log probability of the taken action, weighted by the advantage. An entropy regularization term is added to encourage exploration.

During training, Actor_1 (Discrete Action, a^1) outputs a probability distribution, $\mathbf{p}_1(s) \in [0,1]^{|\mathbf{A}_1|}$, over discrete actions $\mathbf{A}_1$, given state s, calculated using a softmax over network logits, with masking of invalid actions. action mask is 1 for valid actions, and 0 otherwise:

$$z = logits(s) \cdot actionmask(s) + \epsilon$$
$$p_{1,i}(s,a) = \frac{\exp(z_i(s))}{\sum_{j \in A_1} \exp(z_j(s))}$$

where $p_{1,i}(s,a)$ is the probability of action a (an integer index), and $z_i(s)$ are the logits. The log probability of the chosen action a^1 is:

$$\log p_1(a^1|s) = \log(p_{1,a^1}(s))$$

The policy entropy is:

$$H(p_1) = -\sum_{a \in A_1} p_{1,a}(s) \log(p_{1,a}(s))$$

Actor_2 (Continuous Action, a^2) outputs parameters of a Gaussian distribution (mean $\mu(s,a^1)$ and standard deviation $\sigma(s,a^1)$) over continuous actions A_2, conditioned on s and a^1. The log probability density of the chosen action a^2 is:

$$\log p_2(a^2|s,a^1) = -\frac{1}{2}\log(2\pi\sigma(s,a^1)^2) - \frac{(a^2 - \mu(s,a^1))^2}{2\sigma(s,a^1)^2}$$

The differential entropy of Actor_2's policy (for a univariate Gaussian) is:

$$H(p_2) = \frac{1}{2} + \frac{1}{2}\log(2\pi) + \log(\sigma(s,a^1))$$

The loss functions for actions are calculated as follows:

$$A2C_{L1}(\theta_1) = E_{s_t, a_t^1 \sim \pi_{\theta_1}}\big[-\log \pi_{\theta_1}(a_t^1|s_t) \cdot Adv(s_t)$$
$$-\beta H(\pi_{\theta_1}(\cdot|s_t))$$

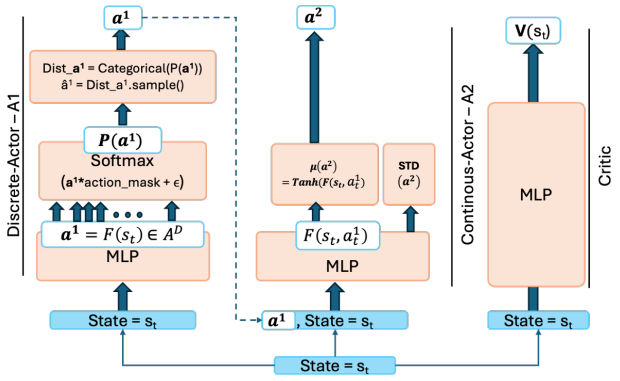


Figure 6: Modified A2C and PPO actor-critic prediction flow and summarized architecture

$$A2C_{L2}(\theta_2) = E_{s_t,a_t^1,a_t^2\sim\pi_{\theta_2}}\left[-\log \pi_{\theta_2}(a_t^2|s_t,a_t^1)\cdot Adv(s_t) - \beta H(\pi_{\theta_2}(\cdot|s_t,a_t^1))\right]$$

where

- $A2C_{L1}(\theta_1)$ and $A2C_{L2}(\theta_2)$ are the loss functions for Actor_1 and Actor_2, respectively, parameterized by their network weights θ_1 and θ_2.
- $E_{s_t,a_t^1\sim\pi_{\theta_1}}[\dots]$ denotes the expectation over states s_t and actions a_t^1 sampled from the policy π_{θ_1} of Actor_1. Similarly for Actor 2.
- $\pi_{\theta_1}(a_t^1|s_t)$ is the probability of taking action a_t^1 in state s_t under Actor_1's policy.
- $\pi_{\theta_2}(a_t^2|s_t,a_t^1)$ is the probability of taking action a_t^2 in state s_t and action of actor 1 a_t^1 under Actor_2's policy.
- $Adv(s_t)$ is the advantage function.
- $H(\pi_{\theta_1}(\cdot|s_t))$ is the entropy of Actor_1's policy in state s_t.
- $H(\pi_{\theta_2}(\cdot|s_t,a_t^1))$ is the entropy of Actor_2's policy in state s_t and action of actor 1 a_t^1.
- β is the 'entropy_coefficient', controlling the strength of the entropy regularization.
- Note that, expectation is being taken over s_t, a_t^1 for actor 1, and s_t, a_t^1, a_t^2 for actor 2.

Due to the nested dependency (Actor_2 depends on Actor_1), backpropagation is done in two steps, similar to DDPG:

- Actor_1 Update: $A2C_{L1}(\theta_1)$ is backpropagated, retaining the computational graph.
- Actor_2 Update:$A2C_{L2}(\theta_2)$ is then backpropagated.

Separate optimizers (actor1_optimizer,actor2_optimizer) are used. Gradient clipping is applied to both critic and actor networks.

Loss functions - PPO

This section details the modifications made to the A2C algorithm (described in the last subsection) to implement modified Proximal Policy Optimization (PPO). PPO constrains the policy update to prevent excessively large changes, enhancing stability. This is achieved via a clipped surrogate objective.

Old Policy: PPO introduces "old" policies, $\pi_{\theta_{1_{old}}}$ (Actor_1) and $\pi_{\theta_{2_{old}}}$ (Actor_2), representing the policies *before* the current update. Their parameters, $\theta_{1_{old}}$ and $\theta_{2_{old}}$, are updated periodically via a soft update:

$$\theta_{i_{old}} \leftarrow \tau\theta_i + (1-\tau)\theta_{i_{old}}$$

where $i \in \{1,2\}$ and τ is a hyperparameter.

Probability Ratio: A core element of PPO is the probability ratio, quantifying the change between the current and old policies. We calculate separate ratios for Actor_1 and Actor_2:

$$r_1(s_t,a_t^1) = \frac{\pi_{\theta_1}(a_t^1|s_t)}{\pi_{\theta_{1_{old}}}(a_t^1|s_t)}$$

$$r_2(s_t,a_t^1,a_t^2) = \frac{\pi_{\theta_2}(a_t^2|s_t,a_t^1)}{\pi_{\theta_{2_{old}}}(a_t^2|s_t,a_t^1)}$$

These ratios are computed in log-space for numerical stability. The old policy log probabilities are detached from the computational graph.

*Clipped Surrogate Objective:*PPO uses a clipped surrogate objective. The unclipped objectives for Actor_1 and Actor_2 are:

$$L_1^{CLIP}(\theta_1) = r_1(s_t,a_t^1)\cdot A(s_t)$$
$$L_2^{CLIP}(\theta_2) = r_2(s_t,a_t^1,a_t^2)\cdot A(s_t)$$

where $A(s_t)$ is the advantage function (calculated as in A2C). PPO *clips* the probability ratio:

$$r_{i\_clipped} = \text{clip}(r_i, 1-\epsilon, 1+\epsilon)$$

where $i \in \{1,2\}$ and ϵ is a hyperparameter. The clipped surrogate objectives are:

$$L_1^{CLIP}(\theta_1) = \min(r_1(s_t,a_t^1)\cdot A(s_t), r_{1\_clipped}(s_t,a_t^1)\cdot A(s_t))$$

$$L_2^{CLIP}(\theta_2) = \min(r_2(s_t,a_t^1,a_t^2)\cdot A(s_t), r_{2\_clipped}(s_t,a_t^1,a_t^2)\cdot A(s_t))$$

This clipping prevents excessively large policy updates.

Actor Loss: The final PPO actor losses, incorporating the clipped surrogate objective and entropy regularization, are:

$$PPO_{L1}(\theta_1) = E_{s_t,a_t^1\sim\mathcal{D}}\left[-L^{CLIP}1(\theta_1) - \beta H(\pi\theta_1(\cdot|s_t))\right]$$

$$PPO_{L2}(\theta_2) = E_{s_t,a_t^1,a_t^2\sim\mathcal{D}}\left[-L^{CLIP}2(\theta_2) - \beta H(\pi\theta_2(\cdot|s_t,a_t^1))\right]$$

Crucially, unlike in the A2C implementation where actions are resampled from the current policy during training,

the actions a_t^1 and a_t^2 used in the PPO loss calculations are the actions *taken in the past and stored in the replay buffer*, denoted by $\mathcal{D}$. These are *not* resampled from the current policies π_{θ_1} and π_{θ_2}. This is a defining characteristic of PPO and contributes to its stability.

These losses are minimized using the respective actor optimizers. The critic loss and the overall training loop structure (sampling, backpropagation, gradient clipping) remain the same as in A2C. The key difference is the clipped surrogate objective in the actor loss, which promotes more stable policy updates.

Connecting with A2C-DT

This section details the action prediction mechanism and loss function calculations for our A2C Decision Transformer (A2C-DT) model in the Tiny RISK environment. We build upon the foundational concepts of A2C and Decision Transformers, introducing several key modifications designed to improve learning efficiency, stability, and performance in this complex setting of Tiny-Risk.

Transformer-Based Action Prediction: Unlike standard A2C, which typically uses separate actor networks, our A2C-DT leverages the Transformer's inherent sequential processing capabilities for *direct* action prediction. The Transformer's hidden states, encoding the context of the input sequence (states, actions, rewards, and, importantly, returns-to-go), serve as the basis for action selection.

- **Input Sequence:** The Transformer receives a sequence of inputs representing the game history (within a context window), including states, actions (of all players), rewards, and returns-to-go.
- **Hidden States:** The Transformer processes this input, producing hidden states, $\mathbf{h} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_T]$.
- **Action Prediction:** Simple linear layers, applied to specific hidden states, predict the action distribution parameters:
  - **Discrete Action (a^1):** A linear layer, W_1, followed by a Softmax, maps the hidden state corresponding to the agent's turn, $\mathbf{h}_t$, to a probability distribution over discrete actions:

    $$\mathbf{p}_1 = \text{Softmax}(W_1 \mathbf{h}_t + b_1)$$

    where $\mathbf{p}_1 \in [0,1]^{|A_1|}$ is the probability vector, and b_1 is a bias term. Invalid actions are masked before the Softmax.
  - **Continuous Action (a^2):** A separate linear layer, W_2, maps the hidden state corresponding to the agent's turn, $\mathbf{h}_t$, to the parameters of a *Beta distribution*:

    $$\text{params}_2 = W_2 \mathbf{h}_t + b_2$$

    Where the parameters are then used to get α and β of the beta distribution.

    $$\text{Beta}(\alpha, \beta)$$

    Where the parameters are calculated as, $\alpha = \text{params}_2 * concentration$ and $\beta = (1 - \text{params}_2) * concentration$.

    We use a Beta distribution, rather than the Gaussian distribution used in our previous A2C/PPO implementations, because the continuous action in our setting (e.g., the proportion of troops to allocate) is naturally bounded between 0 and 1. The Beta distribution is well-suited to this constraint, providing a flexible and appropriate way to model the action distribution.

  Improvement: This integrated approach offers several advantages:

  - **Contextualized Decisions:** Actions are conditioned on a rich representation of the game history, unlike standard A2C where actions depend only on the current observation.
  - **Implicit Policy Representation:** The policy is implicitly encoded within the Transformer and the linear layers, simplifying the architecture.
  - **Better Long-Range Reasoning:** The Transformer's attention mechanism can capture long-range dependencies within the context window more effectively than RNNs.

Note unlike A2C and PPO we do not mask the illegal actions for A2C-DT, as we force the model to learn how to differential during training, via negative reward.

Critic Loss: We combine *both* a 1-step TD loss and a Monte Carlo loss:

$$V_{target}^{TD}(s_t) = r_t + \gamma V(s_{t+1})$$
$$L_{TD} = E\left[\text{smooth\_l1\_loss}\left(V(s_t), V_{target}^{TD}(s_t)\right)\right]$$
$$L_{MC} = E\left[\text{smooth\_l1\_loss}\left(V(s_t), G_t(s_t)\right)\right]$$
$$L_{Critic} = L_{TD} + L_{MC}$$

Where:

- $V(s_t)$ is the predicted value.
- r_t is the immediate reward.
- γ is the discount factor.
- $V_{target}^{TD}(s_t)$ is the 1-step TD target.
- $G_t(s_t)$ is the actual discounted return, starting from s_t.
- smooth_l1_loss is the Huber loss.

Improvement: This hybrid approach combines the benefits of both TD and MC learning:

- TD: Lower variance and faster initial learning due to bootstrapping.
- MC: Unbiased estimate of the true value function, providing a long-term correction to the TD estimates.
- Robustness: The Huber loss (smooth L1) is less sensitive to outliers than MSE, improving robustness in a stochastic environment.
- Forming internal representation of opponent's strategy: since we are learning from long-term opponent behaviour observed as trajectory input the DT, it learns to associate the value outcome and the past behavior.

Actor Loss: The actor loss is calculated only for timesteps where the agent itself took an action. We introduce a three-loss structure for the actors: The advantage function is:

$$Adv(s_t) = G_t(s_t) - V(s_t)$$

Note the use of full return $G_t(s_t)$ instead of TD target.

- $L_{Actor\_1}(\theta_1)$ **(Discrete Action):** This loss uses the *past* action $a^1_{t_\mathcal{D}}$ from the replay buffer:

$$L_{Actor\_1}(\theta_1) = E_{s_t, a^1_{t_\mathcal{D}} \sim \mathcal{D}} \Big[I(t \in \text{Agent's Turns}) \cdot \big(- \log \pi_{\theta_1}(a^1_{t_\mathcal{D}} | s_t) \cdot Adv(s_t) - \beta H(\pi_{\theta_1}(\cdot | s_t)) \big) \Big]$$

  The expectation is over states s_t and past actions $a^1_{t_\mathcal{D}}$ sampled from the replay buffer $\mathcal{D}$. The $\log \pi_{\theta_1}(a^1_{t_\mathcal{D}} | s_t)$ term is the log probability of the *past* action $a^1_{t_\mathcal{D}}$ under the *current* policy π_{θ_1}, given the current state s_t. This is computed by taking cross-entropy loss between the predicted probabilities and past action.
- $L_{Actor\_2}(\theta_2)$ **(Continuous Action, Conditioned on Re-predicted a^1_t):** This loss uses the *re-predicted* continuous action a^2_t from the current policy π_{θ_2}, conditioned on the current state s_t and the *re-predicted* discrete action a^1_t:

$$L_{Actor\_2}(\theta_2) = E_{s_t, a^1_t \sim \pi_{\theta_1}, a^2_{t_\mathcal{D}} \sim \mathcal{D}} \Big[I(t \in \text{Agent's Turns}) \cdot \big(- \log \pi_{\theta_2}(a^2_{t_\mathcal{D}} | s_t, a^1_t) \cdot A(s_t) - \beta H(\pi_{\theta_2}(\cdot | s_t, a^1_t)) \big) \Big]$$

  The expectation is over states and actions a^1_t sampled according to current policy π_{θ_1} and a^2_t is sampled from $\mathcal{D}$. The $\log \pi_{\theta_2}(a^2_{t_\mathcal{D}} | s_t, a^1_t)$ is computed using a BCE Loss.
- $L_{Actor\_2'}(\theta_2)$ **(Continuous Action, Conditioned on Past $a^1_{t_\mathcal{D}}{}'$):** This loss uses the *same* re-predicted continuous action a^2_t as L_{Actor_2}, *but* it is conditioned on the past discrete action $a^1_{t_\mathcal{D}}{}'$ taken from the *replay buffer*:

$$L_{Actor\_2'}(\theta_2) = E_{s_t, a^1_{t_\mathcal{D}}{}' \sim \mathcal{D}, a^2_{t_\mathcal{D}} \sim \mathcal{D}} \Big[I(t \in \text{Agent's Turns}) \cdot \big(- \log \pi_{\theta_2}(a^2_{t_\mathcal{D}} | s_t, a^1_{t_\mathcal{D}}{}') \cdot A(s_t) - \beta H(\pi_{\theta_2}(\cdot | s_t, a^1_{t_\mathcal{D}}{}')) \big) \Big]$$

  The expectation is over states and past actions $a^1_{t_\mathcal{D}}{}'$ sampled from the replay buffer $\mathcal{D}$. The action a^2_t is sampled from the current policy π_{θ_2} given s_t and $a^1_{t_\mathcal{D}}{}'$. The $\log \pi_{\theta_2}(a^2_t | s_t, a^1_{t_\mathcal{D}}{}')$ term is computed using a BCE Loss.

Where:

- $I(t \in \text{Agent's Turns})$ is an indicator function (1 if t is an agent's turn, 0 otherwise).
- $\mathcal{D}$ represents the replay buffer.
- $H(\pi_{\theta_1})$ and $H(\pi_{\theta_2})$ are the policy entropies.
- β is the entropy regularization coefficient.
- $a^{<i}_t$ represents the actions taken by actors *before* actor i

Backpropagation is performed in a multi-step process, retaining the computational graph as necessary. We use separate optimizers for each actor network and for the critic network.