DeceptionNet: A Hybrid Imitation + RL Agent for Social Deduction Games

Jerry John Thomas
Indian Institute Of Technology(IIT) Palakkad
jerryjohnthomastvm@gmail.com

Abstract

We present **DeceptionNet**, a hybrid Imitation Learning (IL) and Reinforcement Learning (RL) agent submitted to the NeurIPS 2025 MindGames Challenge. We evaluate our method on Secret Mafia, a multi-agent social deduction game, which requires dialogue generation, belief modeling, and strategic voting with Consistency Regularization in our RL pipeline. Our final submission relies heavily on IL and/or rule-based components, with PPO-based fine-tuning showing instability near the deadline. Our approach features a modular design where a trainable RL policy handles core game mechanics (voting, night actions) while lightweight LLM components process conversational context. This separation of concerns enables efficient training through imitation learning and enables targeted improvements to individual components.

1 Introduction

Social deduction games, such as Mafia or Among Us, represent a complex frontier for artificial intelligence, requiring agents to integrate natural language understanding, theory of mind, and long-term strategic planning under uncertainty. The primary challenge in this domain is bridging the gap between noisy, often deceptive conversational signals and actionable discrete policies. Purely end-to-end approaches frequently struggle to capture the nuanced social dynamics such as fluctuating trust and suspicion necessary for high-level play.

Our methodology is grounded in the hypothesis that a robust agent must explicitly model the game's hidden information before formulating a policy. Consequently, we propose a modular system that disentangles perception (listening), belief updates (reasoning), and action selection (policy). Diverging from trends that rely on end-to-end fine-tuning of LLMs which can be computationally prohibitive and unstable in sparse-reward Reinforcement Learning (RL) settings.

Critically, we *do not* fine-tune the LLM's weights. This design choice was motivated by compute constraints and gradient instability proves surprisingly effective. Our Stage 1 submission achieved competitive performance with **5-second average response time** versus 40+ seconds for competing LLM-based agents, demonstrating that strategic efficiency can outweigh raw model scale.

Key Insight. By treating the LLM as a frozen feature extractor rather than an end-to-end trainable component, we sidestep common failure modes (catastrophic forgetting, mode collapse, gradient instability) while retaining its natural language understanding capabilities. The result is a system that is simultaneously more performant, more interpretable, and orders of magnitude cheaper to train than monolithic alternatives.

We validate this approach through our performance in the MindGames competition. In Stage 1, our Imitation Learning (IL) backbone achieved a top-tier placement. Notably, this result highlighted the efficiency of our architecture: our rule-based perception mode operated with an average inference time of approximately 2 seconds, orders of magnitude faster than the 40-second average of competing submissions. Furthermore, due to a configuration oversight, our small LLM (Phi-3-mini-128k-instruct) agent competed in the unconstrained category against significantly larger models, yet remained competitive, demonstrating the robustness of our explicit belief modeling. While we were unable to stabilize PPO fine-tuning in Stage 2, our results suggest that separating perception from reasoning offers a robust path forward for social deduction agents. Our results suggest a broader lesson for multi-agent AI: *less coupling can yield more robustness*

Contributions

1. A novel modular architecture separating perception (LLM), reasoning (belief network), and action (RL policy) with clean interfaces.
2. Demonstration that small frozen LLMs can effectively augment trainable RL agents without gradient-based integration.
3. Competitive performance on the NeurIPS MindGames benchmark with **20× faster inference** than comparable LLM-native agents.
4. Open-source implementation enabling reproduction on consumer hardware (single laptop GPU) - `https://github.com/JerryJohnThomas/deceptionNet`.

2 Methodology

Our agent architecture follows a strict separation of concerns: perception (Listener), belief maintenance (BeliefNet), state fusion (StateBuilder), and action selection (Policy). This modular design avoids the instabilities of end-to-end LLM fine-tuning while enabling efficient training through imitation learning and structured probabilistic reasoning.

Figure 1 shows the architecture.

2.1 Hybrid Listener: Dual-Mode Perception

Social deduction games generate long, noisy conversational logs containing both honest coordination and deliberate deception. To extract actionable signals from this data, we implement a two-tier listener system with interchangeable backends.

2.1.1 Rule-Based Listener (Fast Mode)

The default configuration uses deterministic pattern matching for millisecond-scale inference. The rule-based listener extracts per-player signals through:

Keyword Analysis. Three curated lexicons capture social dynamics:

- **Accusation cues**: suspect, accuse, vote, eliminate, push
- **Support cues**: defend, clear, support, town, ally
- **Sentiment markers**: positive {good, trust, clear, safe} vs. negative {mafia, scum, lying, bad}

Structural Analysis. Regular expressions identify: Player mentions via bracket notation `[0-9]`, Speaker-target relationships in conversation flow, Temporal patterns (e.g., who speaks when during voting phases), **Behavioral Detection:** It uses Advanced heuristics such as Contradiction scores: When a player both accuses and defends the same target, Bandwagon indicators: Multiple accusers piling onto a single target, Mention frequency: Normalized attention each player receives

The rule-based mode achieves ∼**2 second average response time**, providing a 20× speedup over LLM-heavy competitors while maintaining competitive strategic performance.

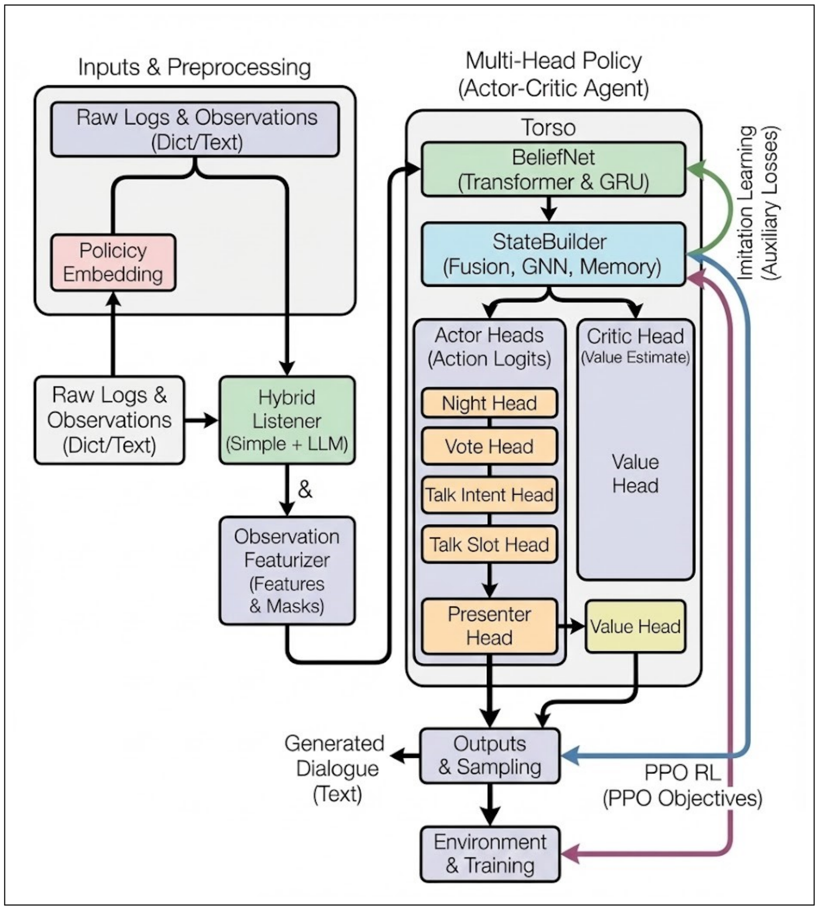
Inputs & Preprocessing
Raw Logs & Observations (Dict/Text)
Policicy Embedding
Raw Logs & Observations (Dict/Text)
Hybrid Listener (Simple + LLM)
&
Observation Featurizer (Features & Masks)
Multi-Head Policy (Actor-Critic Agent)
Torso
BeliefNet (Transformer & GRU)
StateBuilder (Fusion, GNN, Memory)
Actor Heads (Action Logits)
Night Head
Vote Head
Talk Intent Head
Talk Slot Head
Presenter Head
Critic Head (Value Estimate)
Value Head
Value Head
Imitation Learning (Auxiliary Losses)
Generated Dialogue (Text)
Outputs & Sampling
PPO RL (PPO Objectives)
Environment & Training

Figure 1: DeceptionNet Architecture.

2.1.2 LLM-Enhanced Listener (Semantic Mode)

For richer semantic understanding, we optionally augment rule-based features with a locally-hosted instruction-tuned LLM (Phi-3-mini-4k-instruct or Qwen-2.5-1.8B-Instruct, quantized to FP16). The LLM receives a structured prompt:

```
Summarize the following social deduction conversation
as JSON. Keys:  accuses (dict[str, list[int]]), defends
(dict[str, list[int]]), claims (dict[str, str]), tone
(dict[str, str in {confident, unsure, neutral}]),
contradictions (list[int]).
```

The model outputs structured data encoding:

- Explicit relational triples: ⟨speaker, relation, target⟩ (e.g., "Player 3 accuses Player 1")
- Confidence indicators per speaker (confident/unsure/neutral)
- Role claims extracted from natural language
- Multi-speaker contradiction patterns

Critical Design Choice. The LLM runs in **inference-only mode** with **no gradient computation**. We cache outputs for previously-seen conversation contexts (LRU cache, size 16), reducing redundant computation. This design avoids catastrophic forgetting from fine-tuning and enables rapid prototyping (swap models without retraining), and Maintains interpretability (features remain human-readable)

The LLM features are *blended* with rule-based outputs through additive augmentation, providing robustness: if the LLM fails to parse correctly, rule-based signals remain intact.

2.2 BeliefNet: Probabilistic Hidden State Tracking

Rather than feeding raw listener features directly to the policy, we maintain a structured belief representation modeling hidden information. The BeliefNet operates as a learned Bayesian filter, updating beliefs about opponent roles and trustworthiness over time.

2.2.1 Architecture

The BeliefNet processes three streams of information in parallel:

Per-Player Encoder. For each player, we combine 20 features including:

- **Status features**: alive, votes received, votes cast
- **Social signals**: mentions, sentiment, support, accusations
- **Behavioral patterns**: contradictions, bandwagon behavior, role claims
- **Role embedding**: 4-dimensional one-hot encoding if role is known

These features pass through a 2-layer MLP (with ReLU and LayerNorm) to produce 128-dimensional embeddings per player.

Global Context Encoder. Game-level features are processed separately: Round number and turn count (normalized), Ratio of players still alive, Estimated number of Mafia remaining, Phase encoding (night/day_talk/day_vote). This produces a 128-dimensional global context vector.

Conversation Encoder. The listener's 512-dimensional conversation embedding (byte-encoded representation of recent dialogue) is projected down to 128 dimensions to match the other streams.

2.2.2 Relational Reasoning

The Transformer allows the model to reason about patterns across multiple players simultaneously. The three encoded streams are combined and fed into a 2-layer Transformer encoder. Input is Player embeddings + global context + conversation context, architecture consists of 4 attention heads, 256-dim feedforward layers and its purpose is to capture relationships like ”Player 1 and Player 2 both accuse Player 3”

2.2.3 Temporal Integration

A GRU cell maintains memory across game rounds. At each timestep: Pool the Transformer outputs by averaging over alive players, Feed this pooled representation to the GRU, GRU updates its hidden state based on new observations and previous memory. This enables tracking patterns like ”Player 3 has been consistently suspicious across multiple rounds.”

2.2.4 Output Heads

Three prediction heads operate on the Transformer outputs to produce structured beliefs:

Role Predictor. A linear layer outputs logits over 4 roles: Villager, Mafia, Doctor, Detective.
Suspicion Scorer. A linear layer + sigmoid produces a score in [0, 1] indicating Suspicion.
Trust Scorer. Another linear layer + sigmoid produces a trust score in [0, 1] indicating perceived trustworthiness.

The resulting `BeliefState` dataclass contains:

- **player_embeddings**: 128-dimensional rich representations per player
- **global_hidden**: GRU hidden state (128-dim) tracking game history
- **role_logits**: Probability distributions over roles for each player

- **suspicion scores**: Continuous values [0, 1] per player
- **trust scores**: Continuous values [0, 1] per player

These structured outputs are consumed by the StateBuilder and policy heads for decision-making.

2.3 StateBuilder: Graph-Enhanced Fusion

The StateBuilder combines belief outputs with raw features to produce a unified representation for all policy heads. This module performs four operations: feature fusion, cross-attention, graph message passing, and memory integration.

2.3.1 Feature Fusion

The StateBuilder first processes two streams of information in parallel:

Player Stream. For each player, we concatenate:

- Belief embeddings (128-dim) from the BeliefNet
- Raw player features (14 dimensions) encoded to 128-dim
- Suspicion and trust scores (scalars)

Conversation Stream. We combine The listener's 512-dim conversation embedding and the Game-level context (round, turn, alive count, phase). This is also processed to 256 dimensions.

2.3.2 Cross-Attention Mechanism

We use cross-attention to let the conversation context interact with player representations. The conversation embedding acts as a "query" that asks: *"which players are most relevant to what's being discussed?"*

The attention output is blended back into both streams: The conversation embedding gets updated by averaging with its attention result, Player embeddings get updated by adding the attention output (broadcast to all players)

This allows the model to reason about statements like "Player 3 defended Player 1" by connecting conversational context to specific player states.

2.3.3 Graph Neural Network Layers

We model relationships between players using a social graph. The graph edges come from:

- **Voting patterns**: Who voted for whom (bidirectional)
- **Mention co-occurrence**: Players mentioned together frequently

Dead players are masked out so they don't receive updates. The two layers enable 2-hop reasoning: "Player 1 trusts Player 2, who defends Player 3, so maybe Player 1 should reconsider Player 3."

2.3.4 Memory State

Finally, we maintain episode-level memory using a GRU cell. We pool the player representations in two ways: Mean pooling and Sum Pooling (Total activation). These pools, along with the conversation embedding, are concatenated and fed to a GRU that tracks patterns across multiple rounds (e.g., "this player consistently defends accused targets").

The complete `SharedState` output contains:

- Player embeddings ($N \times 256$): graph-enhanced representations
- Conversation embedding (256): attention-refined context
- Memory state (256): episodic history
- Action masks indicating valid moves

2.4 Multi-Head Policy Architecture

The policy uses specialized heads for different action types. Each head is a small MLP that takes relevant parts of the shared state:

Table 1: Policy Head Specifications

Head	**Uses**	**Outputs**	**Masking**
Night	Player embeds + memory	Score per player	Role-dependent
Vote	Player embeds + conversation	Score per player	Alive $\setminus$ self
Talk Intent	Conversation + memory	8 intent types	None
Talk Target	Player embeds + conversation	Score per player	Alive only
Value	All three components	Scalar value	N/A

Design Rationale. Different actions require different context:

- **Night actions** depend on memory (strategic patterns over time)
- **Voting** depends on conversation (who said what recently)
- **Talk intent** depends on both (what to say given the discussion and history)

Talk Intent Types. We define 8 communication intents:

0. **Accuse [target]**: ”I suspect player X”
1. **Defend Self**: ”I'm innocent because...”
2. **Defend Other [target]**: ”Player X is trustworthy”
3. **Claim Role [target]**: ”I'm the Doctor, here's why...”
4. **Agree [target]**: ”I also suspect player X”
5. **Question [target]**: ”Player X, why did you...?”
6. **Filler**: ”We should be careful...”
7. **Silent**: Pass turn

Intents 0–5 require specifying a target player; the Talk Target head provides this.

Masked Sampling: Each head is a 2 layer MLP with ReLU and LayerNorm and each head produces raw scores (logits) for possible actions. We use masked softmax to ensure only valid actions can be selected. The mask sets impossible actions to zero probability before sampling. During training, we sample stochastically; during evaluation, we pick the highest-scoring valid action.

2.5 Presenter: Action-to-Language Generation

The presenter converts discrete policy outputs into natural language text. Like the listener, it supports two modes:

2.5.1 Template-Based Presenter (Default)

A lightweight generator using pre-defined templates with dynamic slot-filling: This mode achieves sub-second generation latency and was used in our Stage 1 competition submission.

$$\text{Accuse}(i) \to \text{“I find } [i]\text{'s statements inconsistent; I suspect them.”}$$
$$\text{Defend}(i) \to \text{“}[i] \text{ appears trustworthy based on their voting pattern.”}$$
$$\text{Claim(role)} \to \text{“I'm the } \texttt{role}\text{; I'll prove it through my actions.”}$$

Belief-Grounded Justifications. The presenter can optionally append evidence:

> *“I suspect [2] because their suspicion score is 78% and they contradicted earlier statements.”*

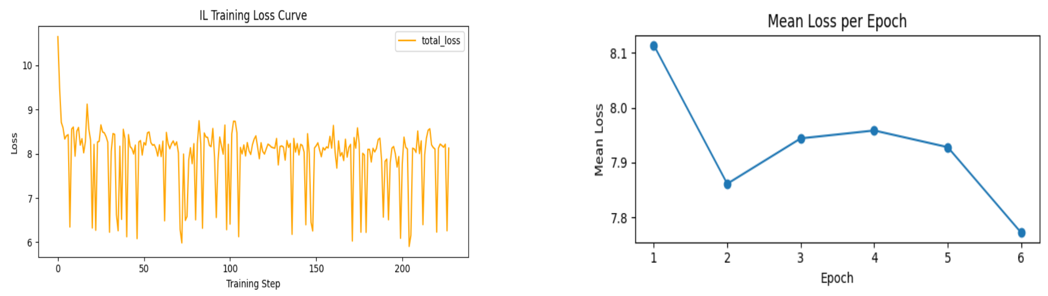

Figure 2: IL training dynamics: loss convergence (left) and epoch progression (right).

2.5.2 LLM-Based Presenter (Optional)

For more natural dialogue, we can substitute an instruction-tuned LLM (e.g., Mistral-7B-Instruct) that receives:

- Selected intent and target
- Belief state summary (suspicion/trust scores)
- Recent conversation context (last 5 turns)

The LLM generates multi-sentence responses constrained to the game's formatting rules. However, this mode incurs 30–50 second latency per turn, making it impractical for real-time play.

3 Training Methodology

We adopted a pragmatic two-stage training strategy: (1) a stable imitation-learning (IL) bootstrap using synthetic trajectories, and (2) exploratory PPO fine-tuning. Due to instability in reinforcement learning, our final submission relies entirely on the IL model.

3.1 Imitation Learning on Synthetic Trajectories

Because high-quality human gameplay logs are not publicly available, We generated synthetic demonstrations using rule-based heuristics implemented in code and derived a dataset of 300 samples of data points. These synthetic games capture core behaviors such as accusations, defenses, vote patterns, and role-specific night actions.

Each game is converted into (s_t, a_t) tuples containing: listener features, belief state, action masks, and the corresponding expert action. The policy is trained with standard cross-entropy losses over all four action heads (night, vote, talk intent, talk target).

This stage is deliberately simple: no complex objectives, no rollouts, and no opponent modeling. The goal is not optimality but *reliable and rule-consistent behavior*. Training converges quickly (20–30 minutes on a single consumer GPU) and produces a stable baseline agent capable of coherent play. Figure 2 shows the loss of IL.

3.2 Attempted PPO Fine-Tuning

We attempted to improve beyond the heuristic policy using self-play PPO. In practice, the method proved too brittle for this domain. Social deduction games exhibit long horizons, sparse rewards, and non-stationary multi-agent dynamics; small shifts in one component (e.g., the presenter) quickly destabilized learning.

During PPO fine-tuning, we observed:

- **Action drift**: the policy frequently produced invalid talk or vote targets.
- **Reward ambiguity**: most decisions have no short-term feedback.

- **Presenter–listener feedback loops**: occasional LLM leaks of role information created artificial training signals.
- **Self-play non-stationarity**: as the agent changed, the environment changed with it.

After several runs, performance consistently degraded from the IL baseline. For competition submission, we reverted to the imitation-learned model, which proved significantly more robust.

3.3 Final Configuration

For stage 1 IL based model was used, for stage 2 we tried to build PPO on top of it, however we were not able to stablise it.

4 Conclusion

We introduced a modular agent architecture for social deduction games, separating perception, belief tracking, and decision-making. A key outcome of this work is that a carefully engineered imitation-learning system, trained entirely on synthetic data, can achieve competitive performance without any end-to-end LLM fine-tuning.

Our attempts at PPO fine-tuning highlight broader challenges in multi-agent hidden-information environments: sparse rewards, non-stationary opponents, and unstable language-conditioned action spaces. These findings suggest that stable rule-guided imitation learning remains a strong baseline for such domains.

Despite its simplicity, our system delivered fast, coherent gameplay and achieved a strong result in the NeurIPS MindGames challenge. We hope this work provides a practical reference for researchers exploring hybrid symbolic–neural approaches to social reasoning games.

References

[1] B Sarkar et al. Training Language Models for Social Deduction with Multi-Agent Reinforcement Learning

[2] PD Patel et al. Modeling Trust and Deception in Multi-Agent Reinforcement Learning Using the Werewolf Game

[3] Brandizzi et al. Conversational Agents in Human-Machine Interaction: Reinforcement Learning and Theory of Mind in Language Modeling

[4] Chi et a;. DVM: Towards Controllable LLM Agents in Social Deduction Games

[5] Zhang et al.Enhance reasoning for large language models with reinforcement learning in the game werewolf

[6] Y Chi et al. AMONGAGENTS: Evaluating Large Language Models in the Interactive Text-Based Social Deduction Game

[7] Schulman et al. Proximal Policy Optimization Algorithms. arXiv 2017.

[8] Jiang et al. Mistral 7B. 2023.

[9] Leon Guertler et al. title=TextArena. 2025.