Large Language Models as Pokémon Battle Agents: Strategic Play and Content Generation

Anonymous ACL submission

Abstract

Strategic decision-making in Pokémon battles presents a unique testbed for evaluating large language models. Pokémon battles demand reasoning about type matchups, statistical trade-offs, and risk assessment, skills that mirror human strategic thinking. This work examines whether Large Language Models (LLMs) can serve as competent battle agents, capable of both making tactically sound decisions and generating novel, balanced game content. We developed a turn-based Pokémon battle system where LLMs select moves based on battle state rather than pre-programmed logic. The framework captures essential Pokémon mechanics: type effectiveness multipliers, stat-based damage calculations, and multi-Pokémon team management. Through systematic evaluation across multiple model architectures we measured win rates, decision latency, type-alignment accuracy, and token efficiency. These results suggest LLMs can function as dynamic game opponents without domain-specific training, offering a practical alternative to reinforcement learning for turn-based strategic games. The dual capability of tactical reasoning and content creation, positions LLMs as both players and designers, with implications for procedural generation and adaptive difficulty systems in interactive entertainment.

1 Introduction

The intersection of artificial intelligence (AI) and competitive turn-based strategy has long been an area of innovation within computer science and digital media. In particular, the Pokémon battle system serves as a sophisticated environment for testing decision-making under complex constraints, including elemental type-hierarchies, variable statistics, and probabilistic outcomes. Traditional game AI in this domain has historically relied on finite state machines (FSMs) or minimax algorithms, which, while functional, often struggle with the vast state-space and the creative "meta-gaming" required for high-level competitive play. While Reinforcement Learning (RL) has enabled agents to optimize battle strategies through experience, these models often lack transparency and fail to adapt when the underlying "move-set" or "meta" changes unexpectedly. Meanwhile, Large Language Models (LLMs), built upon transformer architectures, have demonstrated advanced reasoning and contextual understanding (Vaswani et al., 2017). These capabilities suggest that LLMs may bridge the gap in Pokémon AI by making high-level strategic decisions based on a holistic understanding of the battle state. Despite progress in AI-driven gaming, most existing Pokémon simulators still rely on predefined heuristic logic or heavy reinforcement training, which lack the general reasoning to handle novel, user-generated content. The challenge addressed in this work is integrating LLMs as both strategic battle commanders and creative move designers within the structured, rule-based Pokémon environment. Specifically, we explore how LLMs can:

1. Execute Optimal Moves: Selecting moves and switches based on battle context (Health Points, Statistics, and Type-Matchups).

2. Expand the Move-Space: Generating novel, balanced, and type-consistent moves that adhere to the internal logic of the Pokémon franchise, effectively evolving the game's mechanics.

By moving beyond static scripts, this research positions LLMs as adaptive strategists capable of navigating the nuances of the Pokémon Battle engine. The system architecture combines a deterministic battle simulator with a generative LLM interface. During gameplay, the LLM receives a structured JSON-based game state representing the current battlefield. The model is then required to

reason through its move selection, simulating the "thinking" process of a competitive player. To evaluate the efficacy of this approach, we conducted LLM-vs-LLM tournaments to assess strategic depth across numerous automated matches, alongside human-agent benchmarks. We measured performance through win rates, "turns-to-win" efficiency, and type-exploitation accuracy, which is a metric quantifying how often the model correctly identifies elemental advantages. Furthermore, we analyzed the trade-off between reasoning depth (Chain-of-Thought) and operational latency. Finally, we subjected LLM-generated moves to a dual-validation pipeline: an LLM-based "creativity score" and a deterministic check for mechanical balance, ensuring that new content remains competitive without breaking the game's mathematical integrity. Our findings reveal substantial performance variation across different models. Gemini 2.5 Flash, with chain-of-thought reasoning enabled, achieved a 62% win rate while maintaining type-aligned move selection in 78% of decisions. Model-versus-model tournaments exposed deeper strategic differences: Grok 4 Fast dominated the field with near-perfect win rates and highly efficient, sub-6-turn victories. In the realm of move generation, we found that while GPT-5 Mini produced the most creative and thematically consistent moves (scoring 4.17/5 on creativity), Claude achieved superior mechanical balance, with 80% of its generations meeting all strict deterministic validity criteria. These results suggest that the choice of LLM significantly influences both the tactical "personality" and the mechanical stability of the Pokémon ecosystem.

2 Related Work

2.1 Reinforcement Learning in Game AI

Reinforcement Learning (RL) has been widely used to develop competitive agents in strategic games, with systems such as AlphaGo (Silver and et al., 2016) and StarCraft II agents (Vinyals and et al., 2019) demonstrating superhuman performance through large-scale self-play. Methods including Deep Q-Networks (DQN) (Mnih and et al., 2015) and Proximal Policy Optimization (PPO) (Schulman and et al., 2017) have become standard for learning optimal policies in fixed action spaces.

However, applying RL to Pokémon-style turn-based battles requires extensive training, carefully engineered rewards, and domain-specific abstractions. RL agents are also limited in adaptability and content generation, as they operate over predefined mechanics. In contrast, our work evaluates LLMs as zero-shot strategic agents that reason directly over symbolic battle states, such as type effectiveness, stats, and risk trade-offs, without task-specific training, offering a flexible alternative to RL for turn-based strategy games.

2.2 Procedural Content Generation with LLMs

LLMs have also been leveraged for procedural content generation (PCG) in games. Story2Game generates interactive fiction by populating worlds from story prompts (Fan and et al., 2024), while Microsoft's Muse produces game visuals and controller actions (Research, 2024). Other approaches include using transformer-based models for generating levels and abilities in platform games (Khalifa and et al., 2020). These methods highlight the ability of LLMs to expand the design space of games with new content, although ensuring mechanical balance remains a challenge. Pokémon moves, by contrast, must satisfy strict numerical constraints: a high power move cannot have 95% accuracy without breaking the game's balance. Our move-generation evaluation tests whether LLMs can navigate this tension between creativity and mechanical validity, producing content that feels fresh while remaining playable.

2.3 Strategic Decision-Making and Reasoning in Games

Strategic reasoning in games has been a significant focus. LLMs have been applied to game-theoretic contexts, demonstrating emerging capabilities in multi-agent decision-making (Baker and et al., 2024). Emotion-aware agents (Liu and et al., 2024) incorporate affective reasoning into strategic decisions, which can enhance realism in interactive gameplay. Additionally, research on multi-agent reasoning (Tan, 1993) explores coordination and adversarial strategies, providing a foundation for evaluating LLM-driven gameplay. While these studies establish that LLMs possess strategic reasoning capabilities, they typically evaluate performance in abstract game-theoretic scenarios or cooperative tasks. Pokémon battles introduce a different challenge: decisions must account for type effectiveness multiplicities, stat differentials, and move accuracy trade-offs simultaneously. Our work measures whether LLMs can apply general reasoning to domain-specific rules without explicit training,

using win rates and type-alignment percentages as proxies for tactical understanding.

2.4 Evolution of Competitive Pokémon AI

The application of LLMs to character behavior has traditionally focused on creating more engaging interactive experiences through believable human simulation. For example, (Park et al., 2023) introduced generative agents that simulate social behaviors like autonomously organizing events in a town environment. While these systems enable more fluid conversations and move beyond traditional static dialogue trees, they do not address the specific needs of competitive battle agents. Historically, Pokémon games have relied on rigid, predictable scripts that fail to account for complex human-like "meta-gaming," such as predicted switches or high-risk/high-reward tactical plays. This research examines whether LLMs can move beyond the limitations of traditional use of AI in Pokémon games to provide a more dynamic, human-like challenge.

2.5 Reasoning and Acting with Language Models in Pokémon Battles

The ReAct framework (Yao et al., 2023) highlights the importance of interleaving reasoning and action for effective decision-making in interactive environments. This paradigm is particularly relevant to Pokémon battles, where each turn requires reasoning over complex game state followed by a single, irreversible action.

In our system, the LLM receives a structured representation of the battle state and outputs an executable action each turn. Pokémon battles impose delayed rewards, probabilistic effects, and compounding strategic consequences, making them a demanding test of grounded reasoning. By measuring win rates, type-alignment accuracy, and decision efficiency, we assess whether LLMs can consistently translate general reasoning capabilities into effective domain-specific gameplay without reinforcement learning.

2.6 Prompt Engineering for Game AI

The performance of LLM-based game agents critically depends on prompt design and optimization. While manual prompt engineering remains prevalent, recent work has explored automated approaches to optimize prompts for specific tasks. The field recognizes that effective prompts must balance providing sufficient context about game mechanics and state while remaining concise enough to fit within model context windows.

For game-specific applications, researchers have found that few-shot learning with concrete battle examples significantly improves decision quality. The use of structured state representations, clear action spaces, and chain-of-thought prompting helps LLMs maintain strategic coherence across multiple turns of gameplay.

3 Methodology

3.1 Why Pokémon Battles?

The game demands contextual reasoning. Each turn presents multiple viable actions, but optimal play depends on type matchups, stat differentials, and probability assessment. A Water-type move deals double damage to Fire opponents but half damage to Grass. While this appears to be simple arithmetic, the LLM must retrieve this relationship and apply it correctly in context. Unlike chess, where piece values remain fixed, Pokémon strategy is highly situational: for example, a 60-power move with 100% accuracy often dominates a 120-power move with 70% accuracy when the opponent has low HP. Such judgment calls distinguish competent play from random action selection.

3.1.1 Deterministic and auditable mechanics

The mechanics are deterministic and auditable. Damage follows a fixed formula incorporating stats, type effectiveness, and accuracy. When an LLM makes a suboptimal choice—such as using a Fire-type move against a Water-type opponent—the failure can be traced to a specific reasoning error. Chain-of-thought prompting exposes this logic explicitly, unlike reinforcement learning agents that optimize win rates without transparent explanations.

3.1.2 Dual evaluation: decision-making and generation

Pokémon supports dual evaluation. Beyond decision-making, the move system also tests content generation. Creating a balanced move requires satisfying multiple constraints, including power–accuracy trade-offs, effect probability bounds, and stat-category alignment. This mirrors real-world game design, where procedural tools must generate mechanically valid and diverse content.

3.1.3 Turn-based structure

The turn-based structure removes latency as a confounding variable. Real-time games penalize slow responses regardless of decision quality. In contrast, Pokémon battles allow us to evaluate strategic competence without bias toward faster models.

Finally, Pokémon mechanics are well-documented and widely known. Type charts and damage formulas are available across wikis and strategy guides, making it likely that LLMs encountered this information during training. This setting allows us to test whether models can transfer latent knowledge into functional gameplay, a central question for general-purpose AI applied to specialized domains.

3.2 Explaining the Game

A *Pokémon* refers to a character in the game whose actions are controlled either by the user or the opponent. A battle takes place between two players, each having a team of Pokémon. The goal of the game is to defeat all Pokémon in the opponent's team. Each Pokémon has unique strengths and weaknesses, characterized by the following attributes:

- **Stats:** Determine performance in battles. These include:
  1. **Hit Points (HP)** – The damage a Pokémon can endure before fainting.
  2. **Attack (Atk)** – The power of offensive moves.
  3. **Defense (Def)** – Resistance to incoming attacks.
  4. **Speed (Spe)** – Determines the order of turns in a round.
- **Type:** Each Pokémon has a type (e.g., Fire, Water, Electric, Flying), which defines its strengths and weaknesses against others. For example, Fire is weak against Water but strong against Grass. These matchups determine the damage multipliers applied in battle.
- **Moves:** Each Pokémon can use up to three moves, each with a specific type, base damage, and accuracy probability.

In each turn of battle, both players first decide whether to attack or switch to their active Pokémon. If both choose to attack, the Pokémon with higher speed executes its move first. A Pokémon faints when its HP reaches zero. A player may switch Pokémon mid-battle but cannot attack during that turn.

3.3 Rationale for Dual Evaluation Methods

The use of dual evaluation methods is motivated by the need to assess orthogonal aspects of move quality. Mechanical constraints—such as power limits, accuracy thresholds, and type consistency—must be enforced objectively to ensure correct battle engine behavior and are therefore best handled through deterministic rules. In contrast, attributes such as thematic coherence, novelty, and creative expression require semantic judgment, which is well suited to LLM-based evaluation. By combining deterministic mechanical analysis with LLM-driven semantic assessment, the proposed pipeline provides a comprehensive evaluation framework that ensures generated moves are both mechanically sound and creatively meaningful within the game world.

4 Evaluation and Experimental Setup

4.1 Metrics

We evaluate model-driven gameplay using a mixture of *gameplay quality* and *mechanical correctness*:

- **Win rate (%)**: Fraction of battles won by the LLM agent over a set of episodes (higher is better).
- **Turns to win (mean)**: Average number of turns required to finish a match when the LLM agent wins (lower indicates more decisive play).
- **Type-alignment (%)**: Percentage of moves selected by the agent that exploit type advantage when available.
- **Token consumption**: Average number of input + output tokens consumed per decision (measures API cost).
- **Latency (ms)**: Round-trip time for each LLM decision (from prompt send to parsed response).
- **Human win rate & subjective satisfaction**: In human-vs-LLM matches, the human win rate and questionnaire scores for perceived difficulty.

- **Move Generation Balance and Validity**: The number of valid and balanced moves generated. (Determined using a mathematical function to check inverse relation between power, accuracy and PP)
- **Move Creativity and Originality**: Creativity and Originality scores given by an LLM as a judge out of 5.

All quantitative metrics report mean and standard deviation over repeated runs.

4.2 Experiments and Results

Below we enumerate the experiments conducted, each with objective, setup details, measured outcomes, and corresponding observations.

4.2.1 Experiment 1: Baseline Comparison (LLM vs. Random Player)

Objective. Quantify how much strategic value the LLM adds over simple baselines.

Setup. Ran 50 battles each for: (a) random move selector, (b) Gemini-Flash (Thinking-OFF), and (c) Gemini-Pro (Thinking-OFF). Identical pairings were used across conditions.

Player	Win Rate (%)
Random Player	18
Gemini-Flash (OFF)	62
Gemini-Pro (OFF)	71

Table 1: Baseline comparison of LLM opponents versus random move selector over 50 battles.

Observations. From this experiment, we observe that LLM-driven agents are able to make consistently stronger battle decisions than a random policy, achieving substantially higher win rates without requiring explicit domain-specific training. This demonstrates that even with thinking disabled, LLMs internalize enough structural knowledge of Pokémon mechanics to outperform naïve baselines reliably.

4.2.2 Experiment 2: Model Comparison (Gemini 2.5 Flash vs. Gemini 2.5 Pro)

Objective. Compare decision quality, latency, and token usage between two Gemini variants.

Setup. Ran 50 battles each for Gemini-Flash and Gemini-Pro under identical prompts. Token usage and latency were recorded per decision.

Model	Avg Tokens / Decision
Gemini-Flash	2168
Gemini-Pro	2290

Table 2: Comparison between Gemini-Flash and Gemini-Pro over 50 battles.

Observations. Gemini 2.5 Pro consistently consumed more tokens per decision than Gemini 2.5 Flash (approximately 5.6% higher). This increase in token usage did not correspond to a proportional improvement in baseline win rate, indicating diminishing returns in decision efficiency when moving to the larger model under identical prompting conditions.

4.2.3 Experiment 3: Thinking Mode Requirement (Chain-of-Thought ON vs. OFF)

Objective. Evaluate the effect of internal reasoning on decision quality, token consumption, and latency.

Setup. For Gemini-Flash and Gemini-Pro, 50 battles were run in both Thinking-ON and Thinking-OFF modes using identical seeds and scenarios.

Model / Mode	Avg Latency (s)
Gemini-Flash (OFF)	2.8
Gemini-Flash (ON)	3.5
Gemini-Pro (OFF)	3.3
Gemini-Pro (ON)	5.5

Table 3: Effect of Chain-of-Thought (Thinking ON/OFF) on LLM latency.

Observations. Enabling thinking mode increased latency by approximately 45% across models. While win rates improved when reasoning was enabled, disabling thinking led to a 15% drop in win rate and a 35% reduction in type-aligned move selection. This highlights a clear trade-off between decision quality and computational efficiency.

4.2.4 Experiment 4: Human vs. LLM Playtesting

Objective. Measure how human players perceive and fare against LLM opponents.

Setup. At least 30 participants each played 10 matches against Gemini-Flash and 10 matches against Gemini-Pro. Subjective ratings for difficulty were collected.

Opponent	Avg Difficulty (1–5)
Gemini-Flash (ON)	3.2
Gemini-Pro (OFF)	3.8
Gemini-Pro (ON)	4.0

Table 4: Human playtesting against LLM opponents (subjective metrics).

Observations. Human participants rated Gemini 2.5 Pro as noticeably more challenging than Gemini 2.5 Flash. However, the higher perceived difficulty was not always associated with increased enjoyment, suggesting that overly strong opponents may reduce player satisfaction. Gemini Flash with thinking disabled was rated as significantly easier, indicating its suitability for balanced gameplay scenarios.

4.2.5 Experiment 5: Move-Generation Quality (Batch Size 4)

Objective. Evaluate reliability of move generation when producing four moves per prompt.

Setup. Each model was evaluated over 30 trials, generating batches of four moves under identical prompts and scenarios.

Model	Validity (%)	Balanced (%)
Gemini Flash	88.3	56.7
Claude	89.2	70.8
GPT-5 Mini	86.7	72.5
DeepSeek V3	89.2	65.0
Grok 4	90.0	77.5

Table 5: Move-generation evaluation with batch size 4.

Model	Total Tokens
Gemini 2.5 Flash	83,514
Anthropic Claude	31,849
GPT-5 Mini	107,340
DeepSeek V3	26,802
Grok 4 Fast	55,121

Table 6: Token usage for move generation (batch size 4).

Observations. All models maintained high validity when generating batches of four moves; however, the ability to produce balanced moves varied substantially. Grok 4 and GPT-5 Mini generated the most balanced moves, while Gemini Flash struggled with stricter power–accuracy and power–PP constraints. Token usage differed significantly across models, with GPT-5 Mini incurring the highest cost.

4.2.6 Experiment 6: Move-Generation Quality (Batch Size 1)

Objective. Measure per-move reliability when generating a single move at a time.

Setup. Each model was evaluated over 30 single-move generations using the same states as Experiment 5.

Model	Validity (%)	Balanced (%)
Gemini Flash	100.0	36.7
Claude	100.0	80.0
GPT-5 Mini	100.0	66.7
DeepSeek V3	100.0	46.7
Grok 4	100.0	50.0

Table 7: Move-generation evaluation with batch size 1.

Model	Total Tokens
Gemini 2.5 Flash	48,046
Anthropic Claude	25,826
GPT-5 Mini	62,676
DeepSeek V3	21,657
Grok 4 Fast	41,213

Table 8: Token usage for move generation (batch size 1).

Observations. When generating one move at a time, all models achieved perfect validity. Differences in balance became more pronounced, with Claude and GPT-5 Mini performing best. Token efficiency improved across all models compared to batch generation, though relative differences in cost remained consistent.

4.2.7 Experiment 7: Move-Generation Creativity and Originality

Objective. Evaluate creativity and originality using an LLM judge.

Setup. Each model generated 30 moves evaluated on creativity and originality (1–5 scale).

Observations. GPT-5 Mini consistently produced the most inventive and original moves, while Gemini Flash and Claude tended to generate mechanically correct but stylistically conservative designs. This highlights a trade-off between strict rule adherence and creative diversity in LLM-driven content generation.

Model	Creativity	Originality	Overall
Gemini Flash	3.4	2.4	2.4
Claude	3.37	2.47	2.47
GPT-5 Mini	4.17	3.33	3.28
DeepSeek V3	3.5	2.5	2.5
Grok 4	3.57	2.63	2.6

Table 9: Creativity and originality scores assigned by an LLM judge.

4.2.8 Experiment 8: Cross-Model Battle Tournament

Objective. Compare strategic strength, efficiency, and battle pacing across LLM architectures.

Setup. A round-robin tournament was conducted among five models. Each pairing played up to 10 battles.

Observations. The tournament revealed substantial variation in strategic strength and efficiency across models. Grok 4 Fast dominated most matchups, producing near-perfect win rates and the shortest battles, often concluding in under six turns. In contrast, Claude and DeepSeek V3 tended toward longer engagements, frequently exceeding twenty turns, reflecting more conservative or less decisive strategies. Token-consumption patterns showed similar disparities, with Grok 4 Fast and DeepSeek V3 operating under substantially lower budgets compared to Gemini Flash and GPT-5 Mini. Overall, stronger strategic alignment correlated with shorter battles and more stable token usage.

5 Conclusion

The results of this study provide a definitive framework for selecting and configuring Large Language Models as autonomous agents within a competitive Pokémon environment. Our evaluation demonstrates that for high-stakes strategic play, prioritizing type-alignment and elemental exploitation is essential; consequently, we conclude that Thinking Mode (Chain-of-Thought) is a prerequisite for competent play, despite the associated increases in latency and token consumption. A primary objective of this research was to identify the "sweet spot" for AI difficulty that maintains player engagement without causing frustration or boredom. While Gemini 2.5 Pro demonstrated high difficulty, it did not provide a statistically significant improvement in strategic quality relative to the additional latency it introduced. Therefore, we identify Gemini 2.5 Flash as the optimal configuration for real-time play, offering a balanced 3.5/5 difficulty rating and superior responsiveness. Beyond individual performance, our cross-model tournament highlighted critical efficiency-performance trade-offs. Models such as Grok 4 Fast emerged as elite strategists, producing decisive, short-duration victories, whereas models like Claude and DeepSeek-V3 exhibited more conservative, prolonged engagement patterns. We also identified specific deployment challenges, such as GPT-5 Mini's high operational cost and DeepSeek-V3's occasional failures in maintaining strict JSON formatting. Finally, our dual-evaluation of move generation reveals that a model's "tactical personality" extends to its creative output. While GPT-5 Mini is the superior choice for inventive and original content, Claude remains the more reliable engine for ensuring mechanical balance and mathematical integrity. In summary, our research demonstrates that effectively integrating LLMs into the Pokémon ecosystem depends on aligning specific model architectures with their intended functions. While some models excel as strategic battle agents, others are better suited for balanced and creative content generation.

6 Limitations

This study evaluates LLM-based agents in a simplified Pokémon battle environment with small team sizes and restricted mechanics, which does not capture the full strategic complexity of competitive gameplay involving items, status effects, and long-horizon team planning. The agents operate in a zero-shot, prompt-engineered setting without learning or adaptation across battles, limiting their ability to refine strategies based on experience or opponent behavior. Several evaluation dimensions, particularly creativity and originality in move generation, rely on LLM-based judging, introducing a degree of subjectivity and potential model bias. Finally, reported latency and token usage reflect relative comparisons under specific deployment conditions and should not be interpreted as absolute performance guarantees across environments.

7 Acknowledgements

We thank the developers of publicly available Pokémon battle simulators and community-maintained resources for documenting Pokémon mechanics. We are also grateful for access to multiple commercial LLM APIs that enabled large-scale evaluation across different models.

Model	Claude	Gemini	GPT-5 Mini	DeepSeek V3	Grok 4 Fast
Claude	–	3–7–0	3–7–0	2–8–0	0–10–0
Gemini	7–3–0	–	5–5–0	6–4–0	8–2–0
GPT-5 Mini	7–3–0	5–5–0	–	7–3–0	4–6–0
DeepSeek V3	8–2–0	4–6–0	3–7–0	–	0–10–0
Grok 4 Fast	10–0–0	2–8–0	6–4–0	10–0–0	–

Table 10: Head-to-head win–loss–draw records across models.

Model	Claude	Gemini	GPT-5 Mini	DeepSeek V3	Grok 4 Fast
Claude	–	304,062	315,727	522,646	84,249
Gemini	368,062	–	622,712	398,234	467,621
GPT-5 Mini	374,023	458,938	–	429,762	384,390
DeepSeek V3	467,621	270,607	369,972	–	60,558
Grok 4 Fast	129,397	322,646	382,531	92,260	–

Table 11: Total token consumption per model across 10 battles per pairing.

Model	Claude	Gemini	GPT-5 Mini	DeepSeek V3	Grok 4 Fast
Claude	–	16.1	18.0	31.1	5.2
Gemini	16.1	–	21.3	15.5	31.1
GPT-5 Mini	18.0	21.3	–	21.2	16.0
DeepSeek V3	31.1	15.5	21.2	–	3.9
Grok 4 Fast	5.2	31.1	16.0	3.9	–

Table 12: Average battle duration (turns) across model pairings.

The authors also acknowledge the use of ChatGPT and Claude for assistance with improving the clarity, organization, and grammar of the manuscript. The paper remains an accurate and faithful representation of the authors' original ideas, methodology, and results.

References

John Baker and et al. 2024. Strategic reasoning in llms for game-theoretic scenarios. *Nature AI*.

Rui Fan and et al. 2024. Story2game: Llms for interactive fiction game generation. In *EMNLP*.

Ahmed Khalifa and et al. 2020. Pcgrl: Procedural content generation via reinforcement learning. In *IEEE Transactions on Games*.

Wei Liu and et al. 2024. Eai: Emotion-aware llm agents for strategic games. In *NeurIPS*.

Volodymyr Mnih and et al. 2015. Human-level control through deep reinforcement learning. In *Nature*.

Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, pages 1–22.

Microsoft Research. 2024. Muse: Generative ai for game design and interaction. https://www.microsoft.com/en-us/research/blog/introducing-muse-our-first-generative-ai-model-designed-for-gameplay-ideation/.

John Schulman and et al. 2017. Proximal policy optimization algorithms. In *NeurIPS*.

David Silver and et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature*.

Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *ICML*.

Oriol Vinyals and et al. 2019. Grandmaster level in starcraft ii using multi-agent reinforcement learning. In *Nature*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *Proceedings of the 11th International Conference on Learning Representations*.

Appendix

A Additional Methodological Details

A.1 Prompt Engineering

A crucial part of integrating an LLM into the game involved designing an effective prompting strategy that would enable the model to make consistent, interpretable, and contextually correct battle decisions.

Explaining common strategies Trivial strategies like switching when at a type disadvantage or low HP, to attack when in advantage, and to choose moves with high accuracy when both Pokemon have low HP were sent to LLM through the System Prompt.

Game State Serialization. To effectively communicate the current battle context to the LLM, the entire game state was serialized into a structured textual format. Each turn's state included the active Pokémon of both players, their remaining HP, type, and available moves along with damage and accuracy parameters. The information was represented in a tabular or JSON-like layout to maintain readability while ensuring deterministic parsing by the model.

Reasoning Control and Output Schema. The LLM was instructed to internally reason about move effectiveness and opponent strategy but return only the final structured decision in JSON format. This enforced consistency in parsing and allowed automatic validation before game application. Two modes of inference were supported: *thinking mode*, in which the model performed explicit reasoning before producing the answer, and *fast mode*, in which it produced direct responses without intermediate reasoning.

Evaluation-focused Prompt Design. All prompts were versioned and recorded to ensure reproducibility. Each variant was associated with its latency, token cost, and decision correctness metrics. This enabled a quantitative evaluation of prompt effectiveness and provided insights into how structured and adaptive prompting can influence model behavior in interactive gaming environments.

A.2 LLM vs. LLM Battle Setup

To evaluate the strategic decision-making capabilities of different large language models in adversarial scenarios, we implemented an automated AI-versus-AI battle system. This setup extends our core game engine to facilitate controlled experiments between different LLM agents without human intervention.

Battle Configuration Each battle involves two AI agents, each powered by a distinct LLM (Gemini 2.5 pro, GPT-5-mini, Claude 4.5 Haiku, DeepSeek-V3, Grok 4 Fast). At initialization, both agents are randomly assigned a team of three Pokémon from the available roster. The agents operate with identical system prompts and have access to the same game state information, ensuring that any performance differences reflect the underlying model capabilities rather than asymmetric information access.

Turn Execution Protocol The battle proceeds in discrete turns, with both agents simultaneously selecting actions without knowledge of their opponent's decision. Each aspect of the battle is same as the one made in the game engine.

Both agents invoke their respective LLM APIs during each turn to generate action decisions. API calls were made to each LLM, respecting their rate limits.

Action Processing Each turn, agents select between two action types: executing a move or switching to a different Pokémon. The execution of the move involves the calculation of damage using the standard Pokémon formula, which incorporates attack/defense statistics, type effectiveness multipliers, and critical hit mechanics. When a Pokémon's HP reaches zero, it faints, and the controlling agent must select a replacement from their remaining team members.

Victory Conditions and Metrics A battle concludes when all three Pokémon on one team have fainted. The system tracks token usage across all API calls for cost analysis, records the total number of turns, and logs all actions and outcomes for post-hoc analysis. This automated framework enables systematic evaluation of LLM strategic reasoning across multiple trials and model combinations.

A.3 Move Generation

After integrating an LLM-controlled opponent into the battle engine, we extended the system to automatically generate new Pokémon moves. The goal of this component was to evaluate whether large language models can produce game assets that are both mechanically valid within a turn-based battle system and thematically consistent with the Pokémon universe. More broadly, this experiment serves as a test of the content generation capabilities of LLMs in the context of structured game design. Move generation is guided by a structured prompting strategy that enforces a set of global constraints. These constraints define allowable numerical ranges for attributes such as power, accuracy, PP (PP refers to the maximum number of times a move can be used in a battle and it must be low for a high power move), and effect probabilities. The prompting framework also enforces consistency between a move's category and a Pokémon's offensive strengths and ensures alignment with the Pokémon's type. To enable automated evaluation and integration, the model is required to return each

generated move in a fixed, deterministic JSON format. In addition to these global constraints, the model is provided with Pokémon-specific contextual information, including type, Attack and Special Attack values, speed, and advantageous type matchups. This information encourages the generation of moves that are tailored to a Pokémon's strengths and strategic role, while remaining thematically appropriate. Together, these constraints and contextual cues enable the language model to generate novel moves that satisfy both mechanical requirements and narrative expectations, allowing them to integrate seamlessly into the existing battle system.

A.4 Evaluation of Generated Moves

Generated moves are evaluated using a two-stage procedure that separately assesses structural correctness and creative quality. This separation is motivated by the distinction between mechanical validity and thematic expressiveness, which represent complementary but fundamentally different evaluation dimensions.

A.4.1 Validity and Balance Evaluation

The first stage employs a deterministic evaluator implemented in the `MoveEvaluator` module. This evaluator verifies structural validity by checking the presence of all required fields and ensuring that numerical values fall within predefined bounds. Mechanical balance is assessed by enforcing expected trade-offs between power and accuracy, as well as between power and PP. Secondary effects are validated against permissible probability ranges, and type compatibility with the Pokémon's primary type is examined to ensure thematic consistency.

The evaluator outputs violations, warnings, and a numerical balance score ranging from 0 to 100. A move is classified as balanced if it contains no violations and achieves a score of at least 70. This deterministic evaluation ensures reproducibility and guarantees that generated moves conform to the constraints required for reliable integration into the battle engine.