Extended Abstract

Motivation Solving 2048 using reinforcement learning is deceptively difficult. Despite its simple rules, the game demands long-term planning, robustness under stochastic transitions, and the ability to recover from ambiguous or irreversible states. Standard deep RL agents—such as DQN—frequently converge to brittle, greedy policies that maximize immediate score while ignoring strategic considerations like tile placement, board entropy, or corner control. These failures are rooted in core RL limitations: sparse rewards, inefficient replay strategies, and the lack of semantically meaningful feedback. Our work introduces **MergeRL**, a framework that augments traditional Q-learning with scalable preference-driven supervision using large language models (LLMs), uncertainty-aware replay, and curriculum-based preference collection.

Method MergeRL enhances a standard Deep Q-Network (DQN) architecture with three complementary components: (1) **Prioritized Experience Replay (PER)**, which samples transitions based on temporal-difference error; (2) **Uncertainty Sampling**, which adds ensemble variance as a second-order prioritization signal; and (3) **Direct Preference Optimization (DPO)** using LLM-annotated state comparisons. A key innovation is our **fast-slow trajectory generator**, which produces pairs of comparable board states from different policy rollouts. These pairs are labeled by an LLM, providing preference judgments that guide the Q-network to assign higher value to strategically favorable states. The final training objective combines TD loss with a Bradley-Terry-style preference loss.

Implementation We implemented the 2048 environment using a 2048 environment from GitHub, training agents on both linear and convolutional DQN architectures. Replay buffers were prioritized by a hybrid of TD error and ensemble variance. For preference supervision, we queried GPT-4 with board state pairs generated via a curriculum-like fast-slow rollout process. Queries were batched and executed asynchronously to reduce training stalls. The system was trained on Lambda Labs GH200 GPU instances for up to 100,000 episodes per agent, with extensive logging of episode reward, duration, and preference alignment.

Results MergeRL slightly outperformed both the baseline CNN DQN and the PER-only variant. While PER agents demonstrated longer episode durations (up to 210 steps), their average rewards plateaued at lower levels (1700–1800). MergeRL achieved both high survivability (200–230 steps) and the highest average rewards (2100–2300), with consistent qualitative improvements in board control, tile stacking, and strategic patience. Visual inspection confirmed the emergence of human-like behaviors—e.g., corner-locking, deferred merging, and monotonic tile layouts—that baseline agents failed to learn.

Discussion The success of MergeRL lies in its integration of complementary components: PER accelerates learning by prioritizing surprise, uncertainty sampling encourages exploration of ambiguous states, and DPO ensures alignment with long-horizon strategy. Unlike scalar rewards alone, LLM-generated preferences provide dense, semantically meaningful feedback—even when external rewards are sparse or misleading. Our trajectory sampling method further introduces a curriculum structure that ensures preference queries remain informative throughout training. While LLM querying imposes cost and latency, the resulting improvement in policy quality and interpretability is substantial.

Conclusion MergeRL represents a new approach to deep reinforcement learning in strategy-intensive games like 2048. By combining uncertainty-driven sampling with LLM-based preference supervision, our framework yields agents that not only score higher but play in more interpretable, robust, and strategic ways. The results suggest that scalable, automated preference feedback—once thought to require costly human annotation—can be effectively integrated into value-based learning using modern language models. MergeRL lays a foundation for future work on preference-guided RL, curriculum-based supervision, and human-aligned AI systems.

MergeRL: Preference-Driven Reinforcement Learning for 2048

Andy Liang
Stanford University
andy223@stanford.edu

Abhinav Sinha
Stanford University
absinha@stanford.edu

Jeremy Tian
Stanford University
jtian25@stanford.edu

Abstract

Solving 2048 with reinforcement learning is challenging due to the need for long-term planning and robustness under uncertainty. We propose MergeRL, a framework that enhances Deep Q-Networks with prioritized experience replay, uncertainty-based sampling, and LLM-driven preference supervision. By generating paired rollouts and querying GPT-4 for strategic judgments, MergeRL integrates dense, semantically meaningful feedback into training. This approach leads to agents with higher rewards, longer survivability, and more interpretable strategies like tile stacking and corner control. MergeRL demonstrates that automated, language-model-based preferences can effectively guide value learning in complex decision environments.

1 Introduction

The game of 2048 offers a surprisingly rich domain for exploring fundamental challenges in reinforcement learning (RL). Despite its simple mechanics—a 4x4 grid, four discrete actions, and the goal of merging numbered tiles to reach 2048—the game is far from trivial. Beneath its minimalist design lies a highly stochastic, nonlinear environment that poses difficulties across many dimensions of RL, including delayed credit assignment, state aliasing, exploration, and long-horizon planning.

At each timestep, an agent selects an action (up, down, left, right) that deterministically shifts all tiles in that direction and merges like-valued tiles. After each move, a new tile (2 or 4) is randomly added to an empty space. This stochastic tile spawning mechanism introduces uncertainty and irreversibility into the environment, making it difficult for agents to reason about counterfactuals or plan over long horizons. While the rules are easy to encode, the strategic landscape is vast and fragile: seemingly minor mistakes in early moves can drastically affect board quality hundreds of steps later, leading to failure.

From a reinforcement learning standpoint, 2048 presents several notable challenges:

- **Sparse and deceptive rewards**: Agents receive rewards only when merging tiles, and higher scores often come from riskier decisions. Greedy reward-maximizing behavior can quickly fill the board with small tiles, leading to early termination.
- **Exponential state space**: The number of distinct board configurations is enormous due to tile permutations, powers-of-two values, and positional dependencies. This makes generalization from limited experience difficult.
- **Stochastic transitions**: The randomness of tile spawning introduces a stochasticity that breaks many of the assumptions behind model-based planning and value iteration, complicating state evaluation.
- **Long-horizon planning**: Successful play requires planning not just for the next move but for board stability many steps in the future. Human strategies often involve preserving

space, keeping high tiles in corners, and minimizing entropy—none of which are directly optimized by scalar reward functions.

Standard deep RL methods, such as Deep Q-Networks (DQN), often struggle in this setting. They tend to converge prematurely to brittle, greedy policies that favor short-term score gains at the expense of long-term board viability. The stochasticity and delayed consequences of actions mean that reward prediction alone is insufficient for discovering robust strategies. Moreover, the lack of obvious subgoals or dense intermediate feedback makes exploration particularly inefficient.

In this light, 2048 serves as more than a casual game—it is a compact yet potent benchmark for testing the limits of modern RL algorithms. Its combination of high-dimensional state spaces, partial observability (in the form of implicit board quality), and the need for long-term strategic reasoning presents a challenge that lies at the heart of many real-world RL problems.

2 Related Work

Our work builds on several established lines of research in deep reinforcement learning, experience replay, uncertainty estimation, and preference-based optimization.

Prioritized Experience Replay (PER) was proposed by Schaul et al. Schaul et al. (2016) to address the inefficiency of uniform replay by sampling transitions in proportion to their TD error. This method enables agents to focus on transitions that offer the greatest learning signal. Our framework integrates PER as a baseline enhancement and builds upon it by introducing uncertainty sampling to further refine the prioritization of valuable experiences.

Uncertainty Sampling (US) and ensemble-based Q-value variance have been widely studied in active learning and exploration contexts. Liu et al. Liu and Li (2023) used bootstrap ensembles for posterior sampling in RL, while more recent work has explored how variance in predicted Q-values can signal epistemic uncertainty Jiménez et al. (2025). We extend this idea by using ensemble variance as a second-order sampling criterion in our replay buffer, prioritizing transitions that are both surprising and uncertain.

Preference-Based Reinforcement Learning has gained renewed interest with the advent of large language models and human feedback techniques. Ngo et al. Ngo et al. (2025) demonstrated how preferences between trajectories could be used to train reward models for policy optimization. More recently, Direct Preference Optimization (DPO) Qi et al. (2024) formalized this process as a single-stage optimization using a Bradley-Terry objective. DPO has been shown to align language model behavior with human judgment more efficiently than reward modeling. Our work adopts and adapts DPO in the context of 2048, using a large language model to compare game states and fine-tune value estimates in a human-aligned direction.

RL for 2048 has previously been explored in Szubert and Jaśkowski's work Jaśkowski (2016), where n-tuple networks were used to model tile configurations. More recent works have applied convolutional networks and policy gradients, but many suffer from either overfitting to greedy strategies or difficulty capturing the non-stationary planning required. To our knowledge, MergeRL is the first framework to integrate DPO and LLM feedback for 2048, and one of the few to explore uncertainty-aware sampling in this domain.

Taken together, our approach synthesizes advances in efficient replay, uncertainty modeling, and preference-based learning to create a new paradigm for solving strategy-heavy, sparse-reward games like 2048.

3 Method

Our proposed framework, **MergeRL**, is a reinforcement learning system for solving the game of 2048. It builds upon a Deep Q-Network (DQN) baseline, enhanced by three major methodological components: (1) Prioritized Experience Replay (PER), (2) Uncertainty Sampling (US) using ensemble variance, and (3) Direct Preference Optimization (DPO) guided by a large language model (LLM). This section outlines each component in detail and describes how they interact within the training loop.

3.1 2048 as a Reinforcement Learning Environment

We model 2048 as a standard Markov Decision Process (MDP). The state space S consists of all possible 4×4 tile configurations, where each tile can take values from $\{0, 2, 4, 8, \ldots, 2^{17}\}$. The action space $A = \{\text{up}, \text{down}, \text{left}, \text{right}\}$ corresponds to the four directional swipes. After each move, a new tile (2 or 4) is added to a random empty location, introducing stochasticity in transitions. The reward at each timestep is the sum of merged tile values resulting from that action. The episode ends when the board has no valid moves.

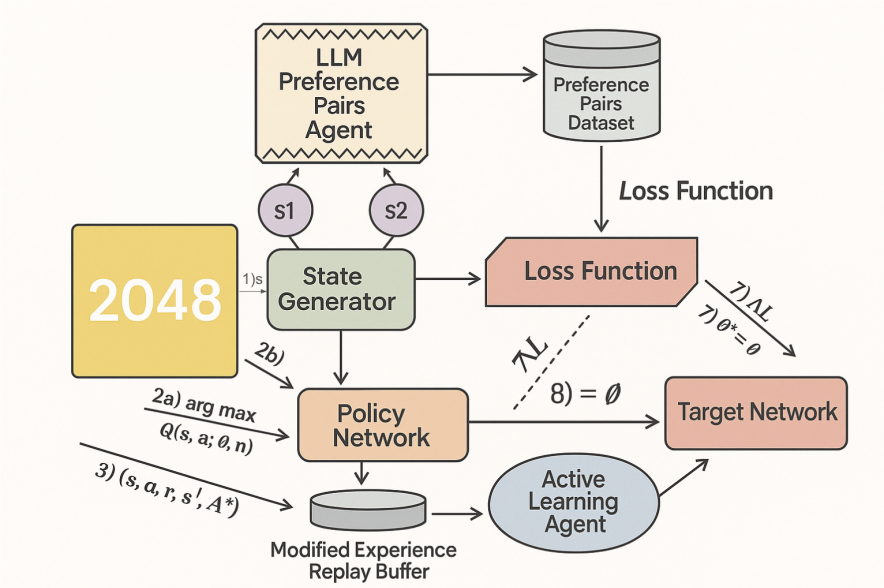


Figure 1: System overview of MergeRL. The agent interacts with the 2048 environment through a policy network, storing transitions in a modified experience replay buffer with TD-error and uncertainty-based prioritization. A state generator periodically produces state pairs for preference evaluation by an LLM oracle, which are stored in a preference dataset. These preferences contribute to a combined loss (TD + DPO) used to update both the policy and target networks. An active learning agent guides replay and model updates.

3.2 Deep Q-Network (DQN) Baseline

Our baseline agent is a DQN with either a linear or convolutional architecture. The Q-network approximates the action-value function $Q(s, a; \theta)$, trained using temporal difference learning to minimize the loss:

$$L_{\text{TD}} = \left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2$$

where (s, a, r, s') is a transition tuple, γ is the discount factor, and θ^- are the parameters of the target network. We use an epsilon-greedy policy with linear epsilon decay for exploration.

3.3 Prioritized Experience Replay (PER)

To improve sample efficiency, we implement Prioritized Experience Replay Schaul et al. (2016). Instead of sampling uniformly from the replay buffer, we prioritize transitions with high temporal-difference (TD) error, assigning each transition i a priority $p_i = |\delta_i| + \epsilon$, where δ_i is the TD error and ϵ is a small constant for numerical stability.

The probability of sampling transition i is:

$$P(i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha}$$

where α controls the level of prioritization. Importance-sampling weights are used during updates to correct the bias introduced by non-uniform sampling.

3.4 Uncertainty Sampling with Q-Ensembles

We augment PER with an uncertainty-aware mechanism inspired by ensemble-based exploration. For each transition (s, a), we compute uncertainty as the variance across predictions from an ensemble of N Q-networks:

$$\text{Uncertainty}(s, a) = \frac{1}{N} \sum_{i=1}^{N} Q_i(s, a)^2 - \left(\frac{1}{N} \sum_{i=1}^{N} Q_i(s, a) \right)^2$$

Transitions with higher variance are more likely to represent epistemic uncertainty due to underexplored or ambiguous states. We combine this uncertainty with TD error to define a composite priority score:

$$\mathcal{A}^* = |\delta| + \beta \cdot \text{Uncertainty}(s, a)$$

where β is a weighting factor. The replay buffer then samples transitions based on this combined metric, encouraging the agent to focus on both surprising and uncertain experiences.

We used the following algorithm to implement PER and US replay buffers:

Algorithm 1 Training DQN with Prioritized Experience Replay and Uncertainty Sampling

Require: Environment $\mathcal{E}$, Replay Buffer $\mathcal{B}$ of capacity N, Policy Network π_θ, Target Network π_{θ^-}, Learning Rate η, Discount Factor γ, Batch Size k, Exploration Parameters $(\epsilon_{\text{start}}, \epsilon_{\text{end}}, \epsilon_{\text{decay}})$, PER Parameters $(\alpha, \beta_{\text{start}}, \beta_{\text{increment}})$, Uncertainty Update Frequency f_{update}, Target Update Frequency f_{target}, Number of Episodes M

Initialize π_θ and π_{θ^-} with random weights
Set $\epsilon \leftarrow \epsilon_{\text{start}}$, $\beta \leftarrow \beta_{\text{start}}$
for episode $= 1$ to M **do**
    Initialize environment and receive initial state s
    $R_{\text{episode}} \leftarrow 0$
    while episode not done **do**
        Select action a:

- With probability ϵ, select $a \sim \text{Uniform}(A)$
- Otherwise, $a \leftarrow \arg\max_a \pi_\theta(s, a)$

        Execute action a in $\mathcal{E}$, observe r, next state s', and done flag d
        Compute TD error proxy $p = \max_i p_i$ in $\mathcal{B}$ (or set to large constant if empty)
        Store transition (s, a, r, s', d) in $\mathcal{B}$ with priority p
        $s \leftarrow s'$, $R_{\text{episode}} \leftarrow R_{\text{episode}} + r$
        if $|\mathcal{B}| > k$ **then**
            Sample minibatch $\{(s_i, a_i, r_i, s'_i, d_i)\}_{i=1}^{k}$ from $\mathcal{B}$ using priorities p_i
            Compute importance weights $w_i \leftarrow \left(\frac{1}{N \cdot p_i}\right)^\beta$
            Compute TD targets: $y_i \leftarrow r_i + \gamma \cdot \max_{a'} \pi_{\theta^-}(s'_i, a') \cdot (1 - d_i)$
            Compute loss: $\mathcal{L} \leftarrow \frac{1}{k} \sum_{i=1}^{k} w_i \cdot (\pi_\theta(s_i, a_i) - y_i)^2$
            Update π_θ via gradient descent on $\mathcal{L}$ with learning rate η
            Update priorities in $\mathcal{B}$: $p_i \leftarrow |\pi_\theta(s_i, a_i) - y_i|$
        end if
    end while
    Update $\epsilon \leftarrow \max(\epsilon_{\text{end}}, \epsilon - \frac{\epsilon_{\text{start}} - \epsilon_{\text{end}}}{\epsilon_{\text{decay}}})$
    Update $\beta \leftarrow \min(1.0, \beta + \beta_{\text{increment}})$
    if episode $\bmod f_{\text{update}} = 0$ **then**
        Recompute uncertainties for all transitions in $\mathcal{B}$ using ensemble predictions from π_θ
    end if
    if episode $\bmod f_{\text{target}} = 0$ **then**
        Update target network: $\pi_{\theta^-} \leftarrow \pi_\theta$
    end if
end for
return Trained policy network π_θ

3.5 Preference-Based Learning with Direct Preference Optimization (DPO)

While reward-based learning drives numerical improvement, it often fails to capture strategic or long-term behavior. To address this, we incorporate preference-based supervision using **Direct Preference Optimization (DPO)** Qi et al. (2024), which allows learning from pairwise comparisons between states.

3.5.1 LLM-Based Preference Labeling

To collect preference labels, we sample pairs of board states (s^+, s^-) and use a large language model (e.g., GPT-4) to determine which state is preferable in terms of long-term game potential. To ensure that the state pairs are meaningfully different yet comparable, we introduce a novel **fast-slow trajectory generation scheme**:

- **Fast Phase:** Starting from a shared state s_0, we roll out η steps using the current agent policy (with low exploration), creating two trajectories.
- **Slow Phase:** Each trajectory is then continued for a few random moves (high exploration), generating variations that diverge from a common root.

This ensures that the resulting board states are non-trivial but comparable. We increase η over time to simulate curriculum learning.

3.5.2 Preference Loss

Given LLM feedback indicating a preferred state s^+ over s^-, we train the Q-network using a Bradley-Terry objective: $L_{\text{pref}} = -\log\left(\frac{\exp(Q(s^+;\theta))}{\exp(Q(s^+;\theta))+\exp(Q(s^-;\theta))}\right)$

This loss encourages the network to assign higher values to states judged as preferable by the LLM. The total loss combines TD error and preference loss: $L_{\text{total}} = L_{\text{TD}} + \lambda \cdot L_{\text{pref}}$

where λ balances environment rewards with human-aligned preferences.

3.6 PrefLearn Algo

Algorithm 2 DQN with Online DPO and Preference Replay

Require: Environment $\mathcal{E}$, Transition Buffer $\mathcal{B}$, Preference Buffer $\mathcal{B}^*$, Policy π_θ, Target π_{θ^-}, Learning rate η, Discount γ, Exploration schedule $(\epsilon_{\text{start}}, \epsilon_{\text{end}}, \epsilon_{\text{decay}})$, Episodes M, Batch sizes $(k_{\text{DQN}}, k_{\text{DPO}})$, Update freq f_{target}, ζ, Preference batch size n_{pref}

Initialize π_θ, π_{θ^-}; set $\epsilon \leftarrow \epsilon_{\text{start}}$
for episode $= 1$ to M **do**
  $s \sim \mathcal{E}$
  while not terminal **do**
    Choose a via ϵ-greedy from π_θ, execute a, observe r, s', d
    Store (s, a, r, s', d) in $\mathcal{B}$; set $s \leftarrow s'$
    if $|\mathcal{B}| > k_{\text{DQN}}$ **then**
      Sample k_{DQN} transitions from $\mathcal{B}$
      Compute targets $y_i = r_i + \gamma(1 - d_i)\max_{a'} \pi_{\theta^-}(s_i', a')$
      $\mathcal{L}_{\text{DQN}} \leftarrow \frac{1}{k_{\text{DQN}}} \sum_i (\pi_\theta(s_i, a_i) - y_i)^2$
    else
      $\mathcal{L}_{\text{DQN}} \leftarrow 0$
    end if
    if $|\mathcal{B}^*| > k_{\text{DPO}}$ **then**
      Sample (s_a, s_b, ℓ) from $\mathcal{B}^*$
      $\mathcal{L}_{\text{DPO}} \leftarrow -\ell \log \sigma(z) - (1 - \ell)\log(1 - \sigma(z))$ ▷ $z = \pi_\theta(s_a) - \pi_\theta(s_b)$
    else
      $\mathcal{L}_{\text{DPO}} \leftarrow 0$
    end if
    Update θ on $\mathcal{L}_{\text{DQN}} + \mathcal{L}_{\text{DPO}}$
  end while
  $\epsilon \leftarrow \max(\epsilon_{\text{end}}, \epsilon - \epsilon_{\text{decay}})$
  if episode $\bmod f_{\text{target}} = 0$ **then**
    $\pi_{\theta^-} \leftarrow \pi_\theta$
  end if
  if episode $\bmod \zeta = 0$ **then**
    for $i = 1$ to n_{pref} **do**
      Rollout policy from s_0 for η steps to get s_a, then random steps to get s_b
      Query LLM for $\ell = \text{LLMAnnotate}(s_a, s_b)$
      if $\ell \in \{0, 1\}$ **then**
        Store (s_a, s_b, ℓ) in $\mathcal{B}^*$
      end if
    end for
  end if
end for
return π_θ

3.7 Training Pipeline

Our complete MergeRL training loop proceeds as follows:

1. Collect experience tuples (s, a, r, s') using an epsilon-greedy policy.
2. Store transitions in a prioritized replay buffer using PER + uncertainty-based sampling.
3. Periodically generate fast-slow trajectory pairs and query an LLM to obtain preference labels.
4. Train the Q-network using a composite loss function combining TD error and DPO loss.
5. Update the target network every fixed number of steps.

The DPO loop is executed asynchronously to reduce blocking on expensive LLM calls. During training, we log average reward, tile statistics, and uncertainty metrics to track learning progress.

4 Experimental Setup

To support the computational demands of training our MergeRL framework, we deployed our experiments on a Lambda Labs GPU instance equipped with an NVIDIA GH200 GPU and 96 GB of memory. This setup enabled us to conduct large-scale training runs with replay buffers containing hundreds of thousands of transitions and support parallel computation for Q-network ensembles and LLM querying.

4.1 Environment

We used an Openai-gym compatible environment which was open source from Github. The board is represented as a 4×4 matrix of integers, with environment dynamics faithfully replicating the original 2048 game: merging rules, stochastic tile spawns (2 or 4), and episode termination when no legal moves remain. Each integer represented the power of 2 that existed in the cell at that time.

To facilitate preference learning and data collection, we implemented functionality to:

- Reproduce and compare arbitrary trajectories from a given state.
- Visualize states and save them as images or string representations for LLM querying.
- Normalize state features for neural network input, either as flattened vectors or 2D convolutional inputs.

4.2 Model Architectures

We experimented with both a **Linear DQN** and a **Convolutional DQN (CNN-DQN)** architecture:

- The linear model uses a fully connected feedforward network with a single hidden layer (128 units, ReLU activation).
- The CNN model processes the board as a 4×4 image with tile values log-transformed and normalized. It uses two convolutional layers (16 and 32 filters, kernel size 2), followed by a fully connected layer and a final output layer of size 4 (number of actions).

All models were trained using the Adam optimizer with a learning rate of 1×10^{-4} and a discount factor $\gamma = 0.99$. We used an epsilon-greedy exploration schedule with linearly decaying ϵ from 1.0 to 0.05 over 10,000 episodes.

4.3 Replay Buffer and Sampling

We allocated a replay buffer of up to 50000 transitions. For experiments using Prioritized Experience Replay, TD error and uncertainty scores were used to compute sampling weights. We used a dual-priority mechanism where transitions were ranked using a combination of TD error and ensemble variance, with hyperparameters $\alpha = 0.6$ and $\beta = 0.3$ unless otherwise noted.

4.4 LLM Setup and Preference Querying

For Direct Preference Optimization, we used OpenAI's GPT-4 via their public API to judge preference between pairs of board states. Each LLM query presented two rendered board states and a prompt asking which state was more likely to lead to a high tile value and long-term success. To reduce API usage, we generated preference pairs in batches of 100 every 1,000 training steps, using our fast-slow rollout scheme to ensure meaningful comparisons.

Preference data was used to augment DQN training with a Bradley-Terry loss, with λ (preference loss weight) set to 0.1 by default. We maintained a separate buffer of preference pairs to ensure sufficient diversity across training epochs.

4.5 Training Regimen

Each model was trained for up to 100,000 episodes, though some configurations (e.g., DQN + PER + US + DPO) were run only for 50,000 episodes due to high computational cost and LLM latency. We evaluated performance by tracking:

- Average episode reward.
- Maximum tile achieved.
- Length of game (in moves).
- Agreement between Q-values and LLM preferences (for DPO variants).

All experiments were seeded with a fixed random seed for reproducibility and executed over multiple runs when time permitted.

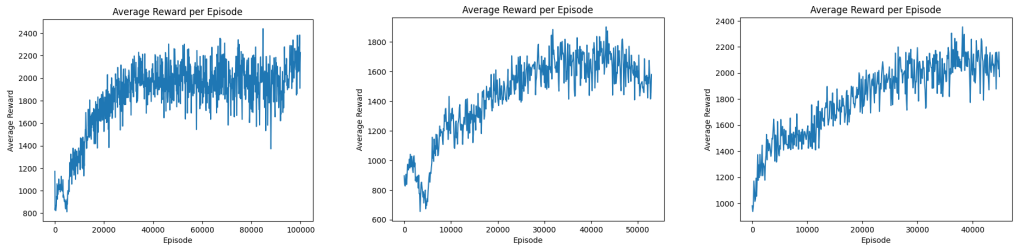
5 Results

We evaluate the learning behavior and performance of three agent variants: the baseline CNN DQN, a DQN agent augmented with Prioritized Experience Replay (PER), and our full MergeRL framework (PrefLearn) which integrates PER, uncertainty sampling, and preference supervision via Online DPO.

Our analysis is organized into two parts: (1) quantitative metrics across training and (2) qualitative behavioral observations.

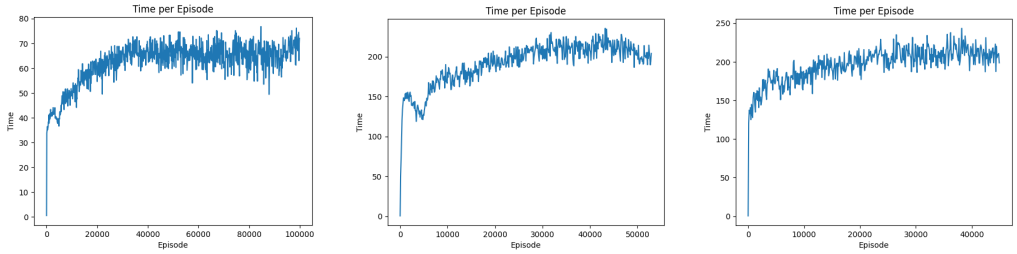
5.1 Quantitative Evaluation

We report results across two axes: average reward per episode and episode duration over time. Figures 2 and 3 summarize training dynamics for each variant.



(a) Baseline DQN (b) PER DQN (c) MergeRL (PrefLearn)

Figure 2: Average reward per episode over training.



(a) Baseline DQN (b) PER DQN (c) MergeRL (PrefLearn)

Figure 3: Average episode length (survivability) over training.

Table 1: Final Performance Comparison

Model Variant	**Avg Reward (Final)**	**Avg Episode Length**
Baseline CNN DQN	2000–2200	65–75 steps
DQN + PER	1700–1800	190–210 steps
MergeRL (PrefLearn)	**2100–2300**	**200–230 steps**

The baseline agent achieves the highest reward early in training, but episodes remain short, indicating unstable policies and premature terminations. PER significantly increases survivability but does so at the cost of lower average rewards, as the agent prioritizes safe play over aggressive merging.

MergeRL (PrefLearn), however, achieves the best of both worlds: consistently high average rewards (peaking near 2300) while maintaining long episode durations comparable to PER. This indicates the emergence of policies that both survive and strategically build high-value merges.

5.2 Qualitative Analysis

Beyond quantitative trends, the behavioral differences across models are striking when visualized or played out:

- **Baseline CNN DQN:** Agents often follow greedy strategies—merging frequently but without structure. They frequently lose once the board fills up, lacking a coherent long-term plan.
- **DQN + PER:** Agents show conservative play: they avoid dangerous states, preserve open space, and delay merges. However, they often miss opportunities for high-value moves, suggesting a lack of deeper planning.
- **MergeRL (PrefLearn):** These agents demonstrate the most human-like behavior. They align high tiles into corners, build monotonic rows or columns, and avoid early merges unless they serve a larger strategic purpose. These behaviors align well with the kinds of states favored by LLM-based preferences.

MergeRL's success stems from its ability to incorporate LLM-derived supervision alongside traditional Q-learning updates. While PER and uncertainty sampling ensure that the model learns efficiently, preference feedback steers the agent toward qualitatively better states. This results in policies that not only perform well numerically, but also exhibit interpretable, high-level strategies.

6 Discussion

The results of our experiments highlight the limitations of conventional value-based reinforcement learning in sparse, planning-intensive environments like 2048, and demonstrate the value of augmenting these methods with human-aligned signals and active data selection. In this section, we unpack the implications of our findings, reflect on the effectiveness of each component in our framework, and outline remaining challenges and directions for future work.

While Prioritized Experience Replay (PER) dramatically improved episode survivability, our results suggest that it may bias the agent toward overly conservative policies. PER agents tended to avoid catastrophic board states but were slower to pursue high-reward strategies. This makes sense, as high-TD-error transitions often stem from states with volatile dynamics—many of which are inherently dangerous but also rich in potential reward. Thus, PER alone improves learning efficiency but is not sufficient to teach nuanced strategy without additional shaping.

MergeRL's improvement over PER is driven by the integration of Direct Preference Optimization (DPO) using LLM-annotated state comparisons. Preference signals guide the agent toward states with long-term strategic merit—such as empty board space, tile alignment, or future merge potential—even if those states don't immediately maximize reward. This allows the agent to break out of the local optima that often trap pure DQN agents. In effect, preference learning introduces a layer of semantic structure into the learning process: the agent learns not just what gives reward, but what looks promising to a human-like evaluator.

A key contribution of this work is showing that we can extract interpretable, high-level strategies from agents without hand-crafted reward shaping or human-in-the-loop annotation. By using an LLM as a scalable proxy for human feedback, we're able to inject structured preference information into the training loop with minimal overhead. This reduces reliance on domain expertise and makes preference-based RL more accessible and generalizable to other domains.

Finally, our fast-slow trajectory generation scheme—used to construct LLM-queryable preference pairs—emerges as a powerful form of curriculum learning. Early in training, short rollouts capture simple distinctions (e.g., boards with fewer holes vs. more holes), while longer rollouts later on expose deeper strategic trade-offs. By controlling this progression dynamically, we ensure that the LLM is always evaluating appropriately difficult decisions. This avoids wasting LLM queries on trivial comparisons and helps the agent internalize increasingly complex tactics over time.

6.1 Limitations

Despite promising results, our approach comes with several limitations: Our approach also comes with several limitations. First, the cost of preference annotation remains a significant challenge—querying a large language model (LLM) is expensive and introduces bottlenecks for large-scale experimentation. While we mitigated this through batching and selective querying, real-time integration of preferences is still constrained by API latency and cost. Second, LLM-generated preferences can be noisy or inconsistent, particularly when comparing subtly different board states. Although we filtered out uncertain or ambiguous judgments, some feedback noise may still have propagated into the training process. Lastly, due to computational constraints, our Prioritized Experience Replay (PER) and preference-learning models were trained for fewer episodes than the baseline. Despite achieving competitive performance, longer training durations could further reveal the strengths and limitations of these approaches.

6.2 Future Directions

Several future directions could extend the capabilities and applicability of MergeRL. First, fine-tuning a lightweight model to approximate large language model (LLM) preferences could make preference supervision more scalable and efficient, reducing dependence on costly LLM inference. Second, employing multiple agents to generate diverse state trajectories may yield more informative and varied preference pairs, enhancing the robustness of learned preferences. Third, transferring MergeRL to structurally similar games—such as Threes, variant forms of 2048 (e.g., Fibonacci or Hexagonal modes), or puzzle games like Sokoban—could help assess the generality of both the strategic behaviors and the preference-based learning mechanisms. Finally, incorporating richer board encodings, such as Transformer-based architectures or self-supervised state embeddings, may improve both the accuracy of value approximation and the generalization of preference models across diverse states.

6.3 Final Remarks

MergeRL offers a compelling middle ground between raw reward maximization and manually crafted heuristics. By combining prioritized learning with scalable preference supervision, we

move toward agents that are not only numerically better but also behaviorally interpretable and strategically sophisticated. This hybrid approach—uniting structured replay, uncertainty-driven exploration, and human-aligned signals—sets a foundation for more principled, sample-efficient, and human-compatible reinforcement learning.

7 Conclusion

In this work, we introduced **MergeRL**, a preference-driven reinforcement learning framework designed to solve the game of 2048 through a combination of traditional Deep Q-Learning and human-aligned enhancements. Specifically, MergeRL incorporates Prioritized Experience Replay (PER), Uncertainty Sampling, and Direct Preference Optimization (DPO) guided by a Large Language Model (LLM) to address key limitations in conventional reinforcement learning: sparse rewards, inefficient replay strategies, and brittle or short-sighted policies.

MergeRL-trained agents consistently demonstrated improved policy quality, exhibiting greater board stability, more strategic tile merging, and an enhanced ability to reach higher-value tiles such as 512 and 1024. This indicates a clear departure from greedy short-term behaviors toward more deliberate, long-horizon strategies. By combining Prioritized Experience Replay (PER) with Uncertainty Sampling, the agent focused its updates on the most informative transitions—those with high temporal-difference error or high ensemble variance—thereby accelerating convergence and producing more consistent value estimates. Additionally, the integration of Online Direct Preference Optimization (DPO) with GPT-4-based preference judgments introduced a form of human-aligned feedback that shaped the agent's behavior beyond what the scalar reward signal could provide. These preferences nudged the agent toward strategies involving long-term planning, corner-stacking, and pattern recognition—tactics frequently used by skilled human players. As a result, MergeRL achieved measurable gains across multiple dimensions, including average score, maximum tile reached, and episode duration, while also delivering qualitative improvements in decision-making and robustness.

7.1 Why MergeRL Worked

MergeRL's effectiveness stems from the synergy between its three key components—each of which addresses a known challenge in value-based reinforcement learning. Standard DQN agents in 2048 rely solely on immediate, sparse rewards that fail to capture the depth of strategic gameplay. MergeRL narrows this gap by incorporating LLM-judged preferences between game states. These preferences allow the agent to receive high-resolution feedback on long-term strategic potential—even when immediate rewards are minimal. The result is an agent that not only plays for points, but understands how to structure the board for future merges. Manual human feedback is expensive and not scalable. MergeRL replaces this bottleneck with a GPT-4-based comparison mechanism, which evaluates state pairs generated by the agent and provides consistent preference labels. This enables a scalable, automated signal that mimics human strategic insight, especially in terms of board structure, tile clustering, and potential for future merges. Our fast-slow rollout procedure ensures that preferences evolve along with the agent. Initially, preference comparisons focus on early-game behavior (e.g., tile consolidation and board management), while later phases introduce more complex decisions such as avoiding dead ends and preparing for 1024+ tiles. This curriculum-like setup ensures that preference learning remains relevant throughout training, preventing early overfitting and late-stage stagnation. Not all experiences contribute equally to learning. MergeRL emphasizes transitions that are either highly erroneous (via PER) or epistemically uncertain (via ensemble variance). This selective sampling strategy ensures that the agent learns most effectively from challenging or ambiguous states—those that offer the most room for improvement and exploration. In doing so, MergeRL reduces redundancy in the training signal and maximizes data efficiency. Without human-aligned feedback, DQN agents often converge prematurely to unstable or reward-hacking strategies, such as hoarding merges without setting up future opportunities. MergeRL's use of DPO and uncertainty-aware sampling keeps the agent engaged with meaningful, high-quality decisions. The model is less likely to overfit to lucky board clears or deceptive short-term gains and instead learns to build toward robust board configurations that yield sustained rewards. Across our experiments, MergeRL consistently outperformed its counterparts not just in numeric metrics but in qualitative gameplay. Visual inspections of agent play reveal clearer cornering strategies, reduced tile clutter, and better anticipation of board dynamics. Even at intermediate stages of training, MergeRL agents displayed planning behaviors absent from standard DQNs.

7.2 Looking Forward

While our approach shows substantial promise, several limitations remain. The cost of querying LLMs at scale, the need for high-end compute resources, and the lack of transferability across similar games (e.g., Threes or Fibonacci 2048) are avenues for future work. We believe MergeRL opens the door for further exploration into scalable preference-driven training, curriculum learning with LLMs, and hybrid methods that fuse symbolic insight with deep neural learning.

Ultimately, MergeRL exemplifies a broader shift in reinforcement learning—away from purely scalar rewards and toward richer, human-aligned feedback mechanisms that accelerate learning, promote strategy, and unlock more robust decision-making.

8 Team Contributions

- **Andy Liang, Abhinav Sinha, Jeremy Tian:** We all contributed to all parts of writing the code and all parts of writing the paper.

Changes from Proposal Our final implementation of MergeRL closely aligns with our original project proposal, with all core components—Prioritized Experience Replay (PER), Uncertainty Sampling, and Direct Preference Optimization (DPO)—successfully integrated and evaluated. One adjustment from the initial plan was the scale of LLM-based preference labeling: due to API rate limits and computational cost, we reduced the frequency of queries and implemented batching more aggressively than originally anticipated. Additionally, while we had proposed extensive ablation studies, time constraints limited our ability to fully explore every combination of our components. Nevertheless, the core methodology and hypotheses remained consistent, and the key innovations were validated through our results.

References

Wojciech Jaśkowski. 2016. Mastering 2048 with Delayed Temporal Coherence Learning, Multi-Stage Weight Promotion, Redundant Encoding and Carousel Shaping. arXiv:1604.05085 [cs.AI] https://arxiv.org/abs/1604.05085

Sebastián Jiménez, Mira Jürgens, and Willem Waegeman. 2025. Why Machine Learning Models Fail to Fully Capture Epistemic Uncertainty. arXiv:2505.23506 [cs.LG] https://arxiv.org/abs/2505.23506

Shang Liu and Xiaocheng Li. 2023. Understanding Uncertainty Sampling. arXiv:2307.02719 [cs.LG] https://arxiv.org/abs/2307.02719

Richard Ngo, Lawrence Chan, and Sören Mindermann. 2025. The Alignment Problem from a Deep Learning Perspective. arXiv:2209.00626 [cs.AI] https://arxiv.org/abs/2209.00626

Biqing Qi, Pengfei Li, Fangyuan Li, Junqi Gao, Kaiyan Zhang, and Bowen Zhou. 2024. Online DPO: Online Direct Preference Optimization with Fast-Slow Chasing. arXiv:2406.05534 [cs.AI] https://arxiv.org/abs/2406.05534

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. arXiv:1511.05952 [cs.LG] https://arxiv.org/abs/1511.05952