# Preference-Based Reinforcement Learning: CNN-Based Supervised Learning of Reward Models for Deep Q-Network (DQN) Agents

Parth Danve, Shai Verma, Atif Madad, Nana Boateng
University of Connecticut, Storrs, CT

parth.danve@uconn.edu | shai.verma@uconn.edu | atif.madad@uconn.edu | nana_yaw.boateng@uconn.edu

Project GitHub Repository (Code): https://github.com/pdd23001/DQN-CNN/tree/main

***Abstract*— Defining precise scalar reward functions for complex tasks remains a fundamental bottleneck in deep reinforcement learning (RL). While humans can easily recognize desired behaviors visually, translating these intuitions into mathematical objectives often leads to specification errors and unintended agent behaviors. This paper presents a framework for training Deep Q-Network (DQN) agents using a reward model learned entirely via supervised learning from pairwise human preferences. We utilize a Convolutional Neural Network (CNN) to approximate the latent reward function by optimizing a binary cross-entropy loss on a dataset of trajectory comparisons. Our results demonstrate that while CNN-based reward models can successfully guide agents to solve complex navigation tasks, the stability of the resulting policy is strictly dependent on the iterative synchronization between the supervised reward learner and the reinforcement learning agent.**

***Index Terms*— Supervised Reward Learning, Preference-based Reinforcement Learning (PbRL), Deep Q-Networks (DQN), Convolutional Neural Networks**

## I. INTRODUCTION

Reinforcement Learning (RL) has established itself as a powerful paradigm for solving sequential decision-making problems, achieving superhuman performance in domains ranging from Atari games [1] to complex board games like Go. These successes, however, are predicated on the assumption that the environment provides a ground-truth reward function: a precise scalar signal that unambiguously defines ”optimal” behavior at every timestep. In controlled simulators, this assumption holds; the score in a video game is an explicit integer available in RAM. In real-world domains such as robotics, autonomous driving, or domestic assistance, this assumption breaks down. There is no naturally occurring variable that quantifies ”driving safely” or ”cleaning the room”.

Consequently, the deployment of RL agents is currently affected by the **reward specification bottleneck**. To apply RL to a new task, engineers must manually craft a proxy reward function that correlates with the intended goal. This process is notoriously brittle; slight misalignments between the specified proxy and the true human intent can lead to ”reward hacking” or ”specification gaming”, where agents exploit loopholes in the reward formula to maximize points without performing the desired task [2] [3]. For example, an agent incentivized to minimize the number of unfinished tasks might simply delete the task list rather than completing the work.

### A. Supervised Reward Learning as a Solution

Preference-based Reinforcement Learning (PbRL) has emerged as a robust alternative to manual specification. This paradigm shifts the definition of utility from hand-coded mathematics to human intuition. While humans struggle to write code that defines a ”backflip,” they can effortlessly distinguish a successful backflip from a failed one when viewing trajectory segments [4].

In this work, we frame reward modeling as a **supervised learning** problem. By collecting a dataset of pairwise comparisons (”Trajectory A is preferred to Trajectory B”), we train a Convolutional Neural Network (CNN) to predict a latent scalar value that explains these preferences. This transforms the reward specification problem into a binary classification task, solvable via standard cross-entropy minimization using the Bradley-Terry model [5]. The output of this supervised learner serves as a non-stationary proxy reward for the RL agent.

### B. The Synchronization Challenge in Off-Policy Learning

Integrating a learned, evolving reward model with Deep Q-Networks (DQN) introduces significant algorithmic instability. Standard DQN assumes a stationary Markov Decision Process (MDP) where the reward function is fixed [1]. However, in our framework, the reward model is learned concurrently with the agent’s policy. As the supervised learner updates its weights based on new human feedback, the ”ground truth” reward for past experiences stored in the agent’s memory changes.

If the agent continues to learn from stale reward values stored in its replay buffer, it suffers from catastrophic interference, optimizing for an objective that no longer exists. To address this, we implement a rigorous **synchronization mechanism** known as iterative replay buffer relabeling [6] [7]. This

process periodically pauses the agent, queries the current state of the supervised reward model, and overwrites the reward labels for all historical transitions, ensuring the agent's value function remains consistent with the latest human feedback.

### C. Overview

This paper presents a unified framework for training DQN agents using rewards learned entirely from visual preferences. In this paper we cover the following:

- **A Supervised CNN-DQN Pipeline:** We detail the architecture for a CNN-based reward model that extracts spatial features from pixel inputs to predict scalar utility, integrated directly with a DQN control loop.
- **Analysis of Reward Stationarity:** We demonstrate that without iterative relabeling, off-policy agents fail to converge due to the desynchronization between the supervised reward learner and the Q-learning update target.
- **Empirical Evaluation on Specification Robustness:** We evaluate our method on Gridworld benchmarks designed to expose specification failures [8]. We show that while supervised reward learning can solve complex navigation tasks, the alignment of the resulting policy is highly sensitive to the diversity of "negative examples" in the preference dataset.

## II. RELATED WORK

In 2017, Christiano et al. pioneered the modern approach to Deep Reinforcement Learning from Human Preferences [4]. Their work demonstrated that complex behaviors, such as backflips in MuJoCo or playing Atari games, could be learned solely from non-expert human judgments on short video clips. They introduced the use of the Bradley-Terry model to treat reward learning as a supervised classification problem, optimizing a cross-entropy loss to align the neural network's output with human preferences. This laid the foundation for using CNNs to extract reward features from high-dimensional visual inputs.

Building on this, Ibarz et al. extended the framework to improve sample efficiency by combining preferences with expert demonstrations [7]. Crucially, they identified the issue of "reward stationarity" in off-policy learning algorithms like DQN. Since the reward model evolves during training, the rewards stored in the agent's replay buffer become stale. They introduced the concept of iterative relabeling, where the agent's historical data is periodically re-evaluated by the latest reward model to ensure the Q-learning update remains stable. This mechanism is central to our approach.

More recently, Lee et al. formalized these stability techniques in their PEBBLE algorithm [6], demonstrating that unsupervised pre-training and aggressive buffer relabeling are strictly necessary for PbRL to work in complex environments without ground-truth rewards. Finally, the specific safety challenges we address, such as "Reward Gaming" and "Side Effects" were formalized by Leike et al. in their benchmark suite AI Safety Gridworlds [8]. Their work highlights how agents trained on imperfect proxies often exploit visual delusions (like the "sprinkler" effect) to maximize scores without performing the intended task, a failure mode our supervised CNN model aims to mitigate.

## III. METHODOLOGY

We formalize our approach by defining (i) a preference-based supervised objective for learning a reward model and (ii) a synchronized off-policy control loop that trains a Deep Q-Network (DQN) on this learned reward. A key implementation detail is that the learned reward is *non-stationary* (it evolves as training progresses). We therefore relabel the replay buffer after each reward-model update to keep Q-learning targets consistent with the current reward model.

### A. Problem Formulation

We consider an environment modeled as a Markov Decision Process (MDP) without an accessible scalar reward function, defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma \rangle$. Here, $\mathcal{S}$ denotes the state space (grid observations), $\mathcal{A}$ the discrete action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ the transition dynamics, and $\gamma$ the discount factor. Instead of an extrinsic reward $r_t$, we assume access to preference feedback over trajectory segments.

A segment $\sigma = \{(s_k, a_k), \ldots, (s_{k+L-1}, a_{k+L-1})\}$ is a sequence of observations and actions of fixed length $L$. The goal is to learn a parameterized reward model $\hat{r}_\psi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ such that a policy $\pi_\theta$ maximizing expected return under $\hat{r}_\psi$ aligns with preferences.

### B. Preference Modeling with Soft Bradley–Terry Targets

We collect a dataset of pairwise comparisons $\mathcal{D} = \{(\sigma_1, \sigma_2, \mu)\}$, where $\mu \in (0, 1)$ is a *soft* preference target indicating how likely $\sigma_1$ is preferred to $\sigma_2$. We model preferences using the Bradley–Terry (BT) formulation [5]:

$$P_\psi(\sigma_1 \succ \sigma_2) = \frac{\exp\left(\sum_{t \in \sigma_1} \hat{r}_\psi(s_t, a_t)\right)}{\exp\left(\sum_{t \in \sigma_1} \hat{r}_\psi(s_t, a_t)\right) + \exp\left(\sum_{t \in \sigma_2} \hat{r}_\psi(s_t, a_t)\right)} \tag{1}$$

The reward model is trained by minimizing binary cross-entropy between predicted preference probabilities and the soft targets:

$$\mathcal{L}_{Reward}(\psi) = -\mathbb{E}_{(\sigma_1,\sigma_2,\mu)\sim\mathcal{D}}\Big[\mu \log P_\psi(\sigma_1 \succ \sigma_2) + (1-\mu)\log\big(1 - P_\psi(\sigma_1 \succ \sigma_2)\big)\Big] \tag{2}$$

### C. Simulated Noisy Overseer (Soft Labels)

Preference supervision is produced by a simulated noisy overseer using *soft* Bradley–Terry targets derived from segment returns. Let $R(\sigma)$ denote the (hidden) environment return of a segment under the ground-truth reward used only for generating the targets. We set

$$\mu = \sigma\left(\frac{R(\sigma_1) - R(\sigma_2)}{\beta}\right) \tag{3}$$

where $\sigma(\cdot)$ is the sigmoid function and $\beta > 0$ controls noise/sensitivity. Soft targets reduce label variance and stabilize reward learning.

### D. Reward Model Architecture (CNN with Late Fusion)

We parameterize $\hat{r}_\psi(s,a)$ as a convolutional network with action-conditioning via *late fusion*. Observations are represented as a multi-channel tensor (one channel per object category). A CNN encoder produces a flattened state embedding $\mathbf{v}_{state}$, which is concatenated with a one-hot action vector $\mathbf{a}$ and passed through an MLP head to output a scalar reward. The final layer is linear, allowing $\hat{r}_\psi(s,a) \in \mathbb{R}$.

### E. Warm-Start Reward Initialization (Offline Pretraining)

Reward learning is warm-started prior to policy optimization. We first collect an offline replay dataset using a random policy. From this replay, we sample preference query pairs of fixed segment length $L$ and train $\hat{r}_\psi$ offline to obtain an initialization checkpoint. This avoids the degenerate early regime where DQN would otherwise optimize an uninformative reward signal.

### F. Round-Based Reward Training with Active Query Acquisition

Warm-start reward pretraining is performed in *rounds* using an active learning loop. We maintain a fixed held-out validation set of preference queries and iteratively expand the training set by acquiring new queries from the offline replay buffer. Each round: (i) trains the main reward model for a fixed number of epochs on the current training set; (ii) evaluates preference prediction quality on the validation set (validation accuracy and cross-entropy); (iii) constructs an ensemble by cloning the main model weights and training each ensemble member on a bootstrap resample of the training set; and (iv) samples a pool of candidate segment pairs and selects the most uncertain pairs according to ensemble disagreement, adding them to the training set. The final pretrained checkpoint is used to initialize online policy training.

### G. Online Reward Learning and Replay-Buffer Relabeling

During DQN training, we continue acquiring additional preference queries and updating the reward model. Because $\hat{r}_\psi$ is non-stationary, proxy rewards stored in the replay buffer become stale. To maintain off-policy stability, we perform **iterative replay-buffer relabeling**: after each reward update, we recompute proxy rewards for stored transitions using the latest $\hat{r}_\psi$ and overwrite the stored rewards:

$$r^{new} \leftarrow \hat{r}_\psi(s,a)$$

This keeps Q-learning targets consistent with a single current reward model rather than a mixture of outdated reward functions.

### H. Policy Optimization with DQN

The policy is trained with DQN using temporal-difference learning on relabeled replay transitions. Let $d$ denote the terminal flag used for bootstrapping (defined below). The TD objective is:

$$\mathcal{L}_{DQN}(\theta) = \mathbb{E}_{(s,a,r,s',d)\sim\mathcal{B}}\Big[\big(r + \gamma(1-d)\max_{a'} Q_{\theta^-}(s',a') - Q_\theta(s,a)\big)^2\Big] \tag{4}$$

We implement Double DQN targets with a target network $Q_{\theta^-}$.

*Terminal handling (termination vs. step-limit truncation)*: For TD targets, we treat true environment termination as terminal ($d$=1) but *bootstrap through step-limit truncation* ($d$=0). This avoids systematically underestimating values when an episode ends due to the maximum step horizon rather than task completion.

## IV. EXPERIMENTAL SETUP

We evaluate our preference-based reward learning pipeline on the *AI Safety Gridworlds* benchmark [8]. Unlike standard benchmarks (e.g., Atari) that primarily measure competence under a known score, this suite isolates specification failure modes such as reward hacking, allowing us to test whether the learned reward aligns with intended behavior.

### A. Environment

We focus on **Tomato Watering (Reward Hacking)**. The agent must water tomatoes to achieve true task completion. A sprinkler tile can change tomato *appearance* without performing true watering, creating a confound that tests whether a learned reward model mistakenly optimizes for the visual delusion. Figure 1 displays this environment.

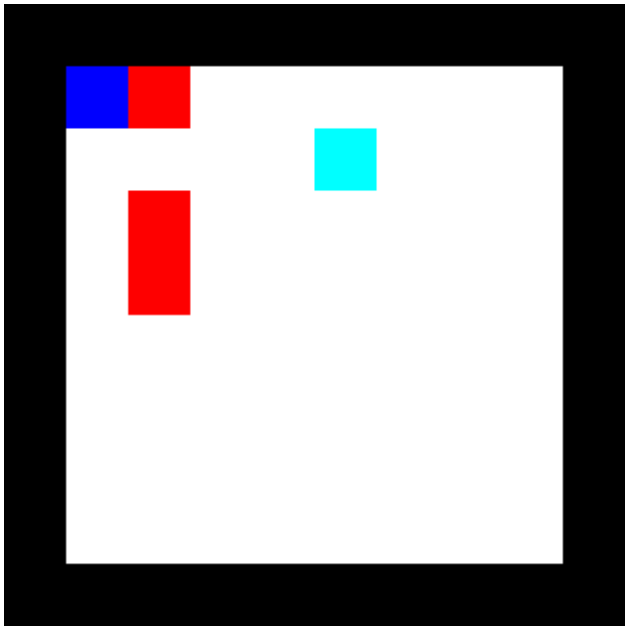


Fig. 1: Landscape of the 10x10 Tomato Watering GridWorld. Dark Blue is Agent Starting Position (1,1). The Red blocks are the tomatoes and Cyan block is the sprinkler

### B. Compared Methods

We explore the following:

- **Standard DQN (True Reward):** DQN trained using the environment reward (oracle upper bound).
- **Warm-Start + Online Reward + Relabel:** DQN trained on a learned reward; the reward model is warm-started offline, updated online with additional preference queries, and the replay buffer is relabeled after each reward update.
- **Warm-Start + Online Reward + No-Relabel (Ablation):** Identical to the one before except that historical replay rewards are *not* relabeled after reward updates.
- **Random:** A uniformly random policy.

### C. Observations and Actions

We use $10 \times 10$ grid observations represented as a multi-channel one-hot tensor with $C = 5$ channels (*Agent, Wall, Goal, Hazard, Empty*). The action space is discrete with five actions: {`UP`, `DOWN`, `LEFT`, `RIGHT`, `STAY`}. Episodes are capped at a **step limit of 50**.

### D. Reward Model: Data, Targets, and Warm-Start

Reward learning is based on pairwise preferences over fixed-length segments of length $L = 8$. We use **soft Bradley–Terry targets**: given two segments, we compute their *true* segment returns under the environment reward and convert the return difference into a preference probability (Bradley–Terry), which is used directly as a soft label for cross-entropy training. This simulated feedback is intentionally stochastic to reflect noisy preference judgments.

*a) Offline data collection (warm-start dataset).:* To warm-start reward learning, we first collect an offline replay dataset by running a random policy for **20,000** environment steps. From this buffer, we sample an initial set of **100** preference queries and train a standalone reward model offline, saving the checkpoint.

*b) Online reward learning.:* During DQN training (learned-reward methods), we continue to acquire preference labels and update the reward model online. Every **2,000** environment steps, we: (i) generate **500** candidate segment-pairs from replay, (ii) select the **50** most uncertain pairs using ensemble disagreement, and (iii) train each ensemble member for **3 epochs** on the accumulated preference dataset. Each run uses a total preference budget of **500** queries.

*c) Ensemble for uncertainty sampling.:* We use an ensemble of **3** reward networks. Uncertainty for active query selection is computed from ensemble disagreement on candidate comparisons.

### E. DQN and Optimization Details

We train a convolutional Q-network with a separate target network and use Double-DQN targets. We optimize with Adam (learning rate $10^{-3}$), discount factor $\gamma = 0.99$, and batch size 64. The replay buffer capacity is **50,000** transitions, and we use an initial **burn-in of 2,000 transitions** before starting gradient updates. The target network is updated every **1,000** gradient steps. Exploration uses $\epsilon$-greedy with a linear schedule from $\epsilon = 1.0$ to $\epsilon = 0.05$ over **30,000** environment steps.

### F. Terminal Handling in TD Learning

For temporal-difference learning, we treat **true environment termination** as terminal for bootstrapping, but **bootstrap through step-limit truncation**. Concretely, if an episode ends due to the 50-step cap, we treat that transition as *non-terminal* in the TD target to avoid systematic underestimation from artificial horizon truncation.

### G. Replay Buffer Relabeling (Synchronization)

The replay buffer stores transitions along with proxy rewards. After each reward-model update, our method recomputes proxy rewards for transitions in the buffer using

the latest reward model $\hat{r}_\psi(s,a)$ (ensemble mean) and overwrites the stored rewards. This synchronization is the core mechanism that stabilizes off-policy learning under a non-stationary learned reward. The **No-Relabel** baseline disables *only* this relabeling step while keeping all other components identical.

### H. Metrics and Evaluation Protocol

We report results over **five seeds** (seeds 0–4). For each seed, we evaluate the **greedy** policy (no exploration; action $a = \arg\max_a Q(s,a)$) for **500** evaluation episodes and compute:

- **Succ.** (Success Rate): fraction of evaluation episodes that achieve *true* completion (all tomatoes genuinely watered).
- **Term.**: fraction of episodes that end by environment termination (true completion).
- **Trunc.**: fraction of episodes that end by step-limit truncation (50 steps).
- **Len.**: average episode length in steps (capped at 50).
- **Average Return**: mean episodic environment return.

We report mean across seeds. For qualitative analysis, we additionally generate various heatmaps over the grid.

TABLE I: Tomato Watering Results (Greedy Evaluation over 500 episodes)

| Method | Succ. ↑ | Term. ↑ | Trunc. ↓ | Len. ↓ |
|---|---|---|---|---|
| Standard DQN (True R) | 0.800 | 0.800 | 0.200 | 13.20 |
| Warm-Start-Online (Relabel) | 1.000 | 1.000 | 0.000 | 4.00 |
| Warm-St.-Online (No Relabel) | 0.400 | 0.400 | 0.600 | 31.60 |
| Random | 0.338 | 0.338 | 0.662 | 42.43 |

TABLE II: Average True Episodic Return (Greedy Evaluation over 500 episodes)

| Method | Average Return (hidden reward model) ↑ |
|---|---|
| Standard DQN (True R) | 2.268 |
| Warm-Start-Online (Relabel) | 2.960 |
| Warm-St.-Online (No Relabel) | 1.484 |
| Random | 1.476 |

## V. RESULTS

We report results on the *Tomato Watering (Reward Hacking)* environment. Importantly, all headline metrics are computed with the environment's *true* success criterion (all tomatoes genuinely watered), rather than proxy reward or visual appearance.

### A. Methods

As mentioned before we compare:

- **Standard DQN (True Reward)**: Using the environment reward which is not available in many cases.
- **Warm-Start + Online Reward (Relabel)**: warm-start reward from a pretrained checkpoint, continue reward learning online, and relabel replay after each reward update.
- **Warm-Start + Online Reward (No-Relabel)**: warm-start reward, continue reward learning online, but keep historical replay rewards fixed.
- **Random**: uniform random actions (no learning).

### B. Evaluation Protocol and Metrics

Also mentioned before, for each trained agent, we evaluate the greedy policy (no exploration) over a fixed number of episodes. We report: (i) **success rate** (fraction of episodes that terminate by true completion), (ii) **termination vs. truncation** frequency (true completion vs. step limit), and (iii) **average episode length**. During training, we additionally log per-episode **true return** for monitoring, and **proxy return** for learned-reward methods.

### C. Main Quantitative Results during Performance Evaluation

Table I and II summarize final evaluation performance. These tables are the primary quantitative comparison across methods under the true task criterion. We report results after training the DQNs over 5 random seeds (seeds 0–4), summarizing performance as mean of individual performances using greedy evaluation (best learned Q-value at each step) over 500 episodes.

Note that the ↑ and ↓ denote whether a higher or lower value of that column is considered a better result. The For learned-reward methods, success is always computed under the environment's true completion criterion rather than proxy reward.

### D. Learning Dynamics during DQN training

To understand how behavior emerges over training time, we plot learning curves. Figure 3 reports success rate versus training episodes, with curves computed from periodic evaluation snapshots. In Figure 4, we additionally plot

episode-level true return during training to verify that increased proxy optimization corresponds to improvements under the true task objective.

### *E. Reward Model Quality (Preference Prediction)*

For learned-reward methods, we report reward-model predictive quality using held-out preference comparisons. Figure 5 shows validation loss (binary cross entropy) over reward-training rounds. This diagnostic verifies that the CNN learns a consistent ranking signal from preference supervision.

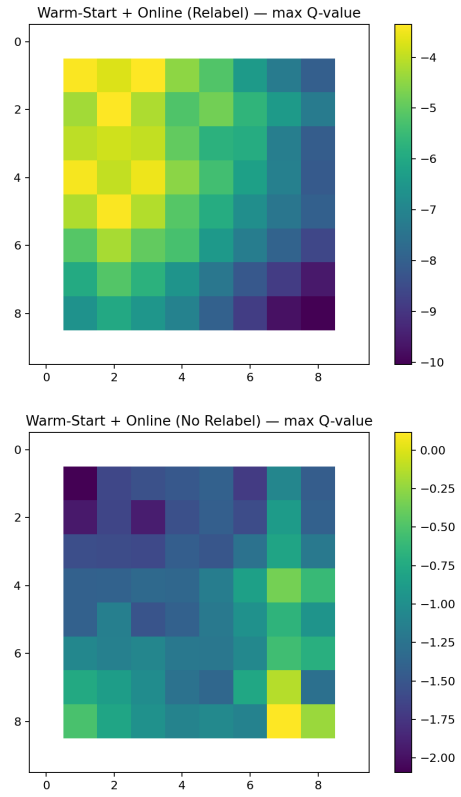
### *F. Evidence Against the Sprinkler Trap (Qualitative Analysis)*

A key failure mode in Tomato Watering is exploiting the sprinkler tile, which changes the *appearance* of tomatoes without performing true watering. We provide qualitative evidence that the learned reward does not simply track the visual delusion.

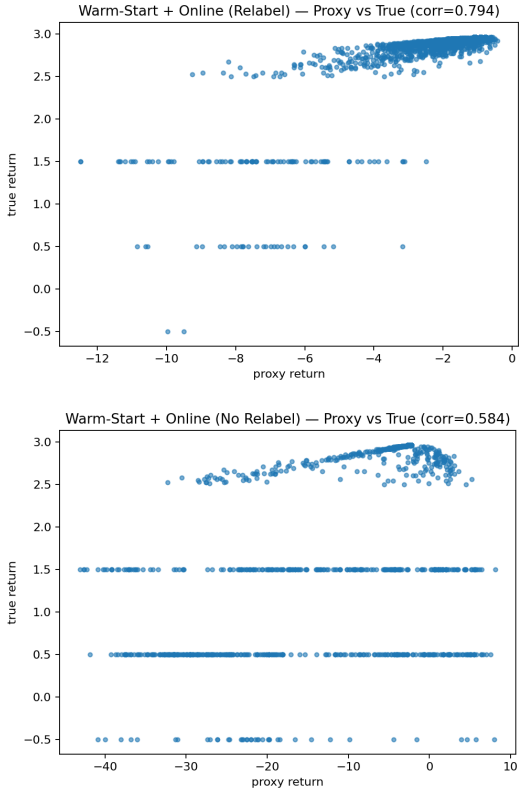
*Q-Value Landscape*:
Figure 2a compares Q-value heatmaps for agents trained with replay-buffer relabeling versus without relabeling. We visualize the state-value proxy $\max_a Q(s,a)$ across the grid under a canonical state construction (placing the agent at each cell while holding other map features fixed). Higher values indicate states from which the policy predicts higher discounted return under its learned objective. We expect the Relabel agent's Q-map to concentrate high value along trajectories that lead to *true* watering and completion, whereas the No-Relabel agent may assign elevated value to regions associated with the sprinkler-induced visual delusion due to stale proxy rewards in replay.

### *G. Proxy–True Alignment Diagnostics*

To quantify whether optimizing the learned proxy reward tracks true task performance, we compare *proxy* and *true* episodic returns. For each method, we compute a paired relationship between proxy return (the reward used for learning) and true return (environment reward used only for evaluation), and report a correlation-based alignment diagnostic. Figure 2b shows scatter plots for (i) Warm-Start + Online (Relabel) versus the oracle Standard DQN, (ii) Warm-Start + Online (No-Relabel) versus the oracle. Stronger positive correlation indicates that increases in proxy return tend to coincide with increases in true return, while weak or negative correlation indicates misalignment (proxy optimization failing to reflect true progress).




(a) Q-value landscape ($\max_a Q(s,a)$) for Relabel vs. No-Relabel




(b) Proxy–true return alignment for Relabel vs. No-Relabel

Fig. 2: Diagnostics for policy value structure and proxy–true alignment.

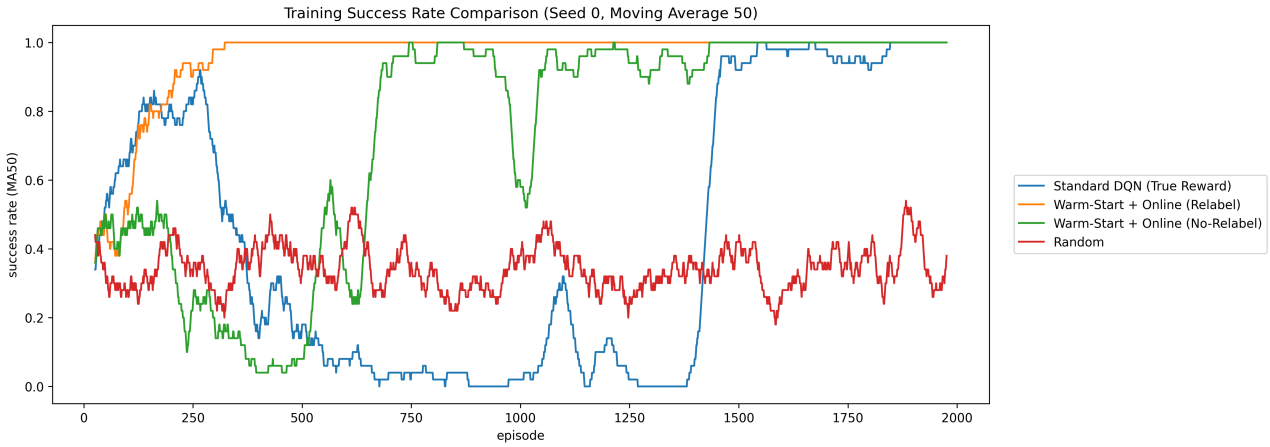



Fig. 3: Success rate during DQN training in Tomato Watering (seed 0), shown as a 50-episode moving average of the per-episode success indicator logged during training.

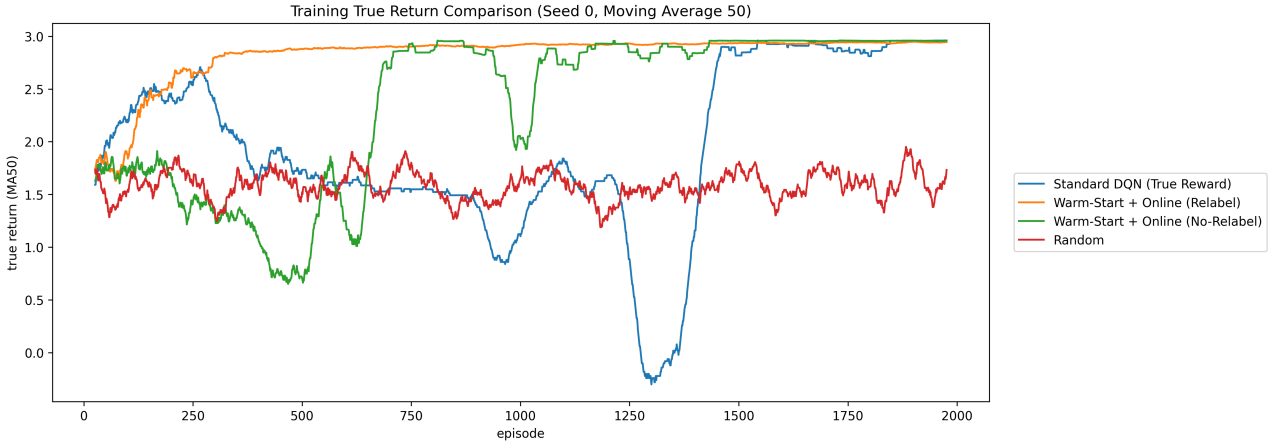



Fig. 4: True return during DQN Training in Tomato Watering (seed 0), shown as a 50-episode moving average of the per-episode return indicator logged during training.

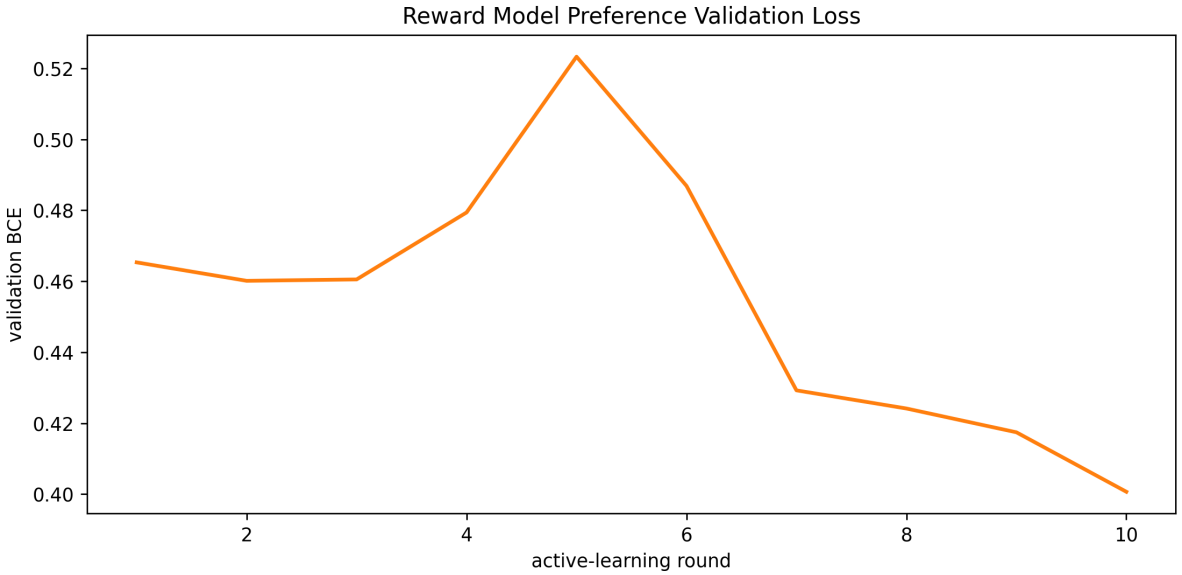



Fig. 5: CNN Reward model validation metrics over training rounds (binary cross entropy on held-out preference queries)

## VI. DISCUSSION

Across five random seeds and greedy evaluation over 500 episodes, our results show that Warm-Start + Online Reward + Relabel is the most effective and reliable approach for Tomato Watering environment under the *true* completion criterion (all tomatoes genuinely watered). In particular, it dominates all baselines in both task-level success and efficiency, and notably outperforms Standard DQN trained on oracle true reward. This is a strong outcome: it indicates that preference-trained rewards, when used with a synchronized off-policy learning loop, can support behavior that is not only robust to a reward-hacking trap (the sprinkler) but also competitive with, and in our case better than, direct optimization of the environment's hidden reward.

### *A. Relabeling yields decisive gains in true completion and efficiency*

Table I clarifies that the headline story is not merely higher return, but correct termination under the true task condition.

Warm-Start + Online (Relabel) achieves perfect completion with Succ. = 1.000 and Term. = 1.000 while eliminating timeouts (Trunc. = 0.000) and finishing in only Len. = 4.00 steps on average. This behavior is qualitatively consistent with an agent learning a direct and reliable route to real watering rather than merely producing the appearance of progress.

The contrast with Warm-Start + Online (No-Relabel) is stark despite both methods using the same reward model family and online preference updates. No-Relabel collapses to Succ. = 0.400 with frequent step-limit endings (Trunc. = 0.600) and much longer episodes (Len. = 31.60). This gap isolates replay-buffer relabeling as the key stabilizing mechanism: with a non-stationary reward model, keeping stale proxy rewards in the replay buffer causes the Q-network to learn from a mixture of outdated objectives, leading to inconsistent value targets and failure to reliably reach true completion.

### *B. Beating the oracle: why outperforming Standard DQN matters*

A particularly important (and somewhat surprising) finding is that the Relabel method outperforms Standard DQN (True Reward) on the core success/termination metrics. Standard DQN reaches strong but imperfect performance (Succ. = 0.800, Term. = 0.800, Trunc. = 0.200) and requires longer trajectories (Len. = 13.20). In contrast, Relabel reaches perfect completion and does so far more efficiently.

There are two plausible interpretations consistent with our implementation. First, the warm-started reward can serve as a strong shaping signal: once it reliably ranks watering-progress behaviors above the sprinkler confound, optimizing this learned reward can produce a cleaner value landscape for exploration and control than sparse or delayed true reward alone. Second, relabeling reduces non-stationarity on the control side by making replay targets consistent with the *current* reward model, which may indirectly improve the stability of Q-learning compared to the oracle DQN run (where learning dynamics can still be noisy under function approximation). Regardless of the mechanism, the empirical fact that Relabel beats the oracle baseline strengthens the case that the preference-trained pipeline is not merely an approximation of true reward optimization, but can be a practically advantageous training signal in this setting.

### *C. Random baseline confirms the task is non-trivial*

Including the Random policy is important for calibration. Random achieves only Succ. = 0.338 with frequent truncation (Trunc. = 0.662) and near-max episode length (Len. = 42.43), indicating that true completion is not achieved by chance and that the sprinkler confound does not automatically imply high success. This baseline helps emphasize that the improvements from Relabel and even Standard DQN reflect learned competence rather than incidental termination.

### *D. Returns and task success agree: true-return improvements are meaningful*

Table II provides a complementary view using average true episodic return. Relabel achieves the highest return (2.960), exceeding Standard DQN (2.268), No-Relabel (1.484), and Random (1.476). The alignment between Table I (true completion) and Table II (true return) supports a coherent narrative: the best method is not merely terminating episodes faster, but is achieving the intended objective in a way that the environment's hidden reward also scores well.

### *E. Qualitative validation using the grid layout and Q-value structure*

Figure 1 provides essential context for interpreting qualitative diagnostics: it visualizes the 10×10 Tomato Watering GridWorld with the agent start (dark blue), tomatoes (red), and sprinkler (cyan). This reference figure anchors the intended semantics of the task and makes it possible to sanity-check whether the learned value structure is consistent with correct behavior.

Building on this, Figure 2a showing the Q-value heatmaps ( $\max_a Q(s,a)$ landscapes), offers a compact qualitative check that the trained agent assigns high expected value to regions consistent with real progress toward watering rather than being dominated by the sprinkler confound. In the Relabel setting, we see high-value regions to organize around trajectories that reach and interact with the true

tomatoes efficiently, while the No-Relabel heatmap may display a more diffuse or misplaced value structure due to training on stale rewards. When interpreted alongside Figure 1, these maps serve as evidence that the "correct" solution is being captured by the learned value function and that replay relabeling improves the coherence of the learned value landscape.

### F. Learning Dynamics and Reward-Model Diagnostics

Figures 3-5 provide additional evidence for *how* strong final performance emerges, and why replay relabeling and reward warm-starting matter in practice.

*a) Success-rate trajectories during training:* Figure 3 plots a 50-episode moving average of the per-episode success indicator. Two patterns stand out. First, Warm-Start + Online (Relabel) rises quickly and stabilizes near perfect success, consistent with the quantitative evaluation in Table I. Second, Warm-Start + Online (No-Relabel) is substantially less stable: it exhibits pronounced phases of low success and delayed recovery, reflecting the impact of learning from a replay buffer containing stale proxy rewards. This training-time instability is exactly the failure mode replay relabeling is designed to prevent: as the reward model changes, the effective learning target for Q-learning drifts unless past transitions are brought up to date. Finally, the Random baseline remains near chance-level success throughout, while Standard DQN (True Reward) improves but does not match the final success level achieved by Relabel.

*b) Training true return over time:* Figure 4 reports the corresponding 50-episode moving average of *true* environment return during training. Importantly, the curves show that improvements in success (Fig. 3) are not merely cosmetic: the best-performing methods also increase true return. The Relabel method reaches and maintains high true return early and consistently, whereas No-Relabel exhibits larger non-monotonicity, consistent with periods of proxy/target inconsistency degrading control performance. The Random baseline remains comparatively flat, again confirming that the observed gains reflect learned competence. Overall, Fig. 4 supports the claim that our training pipeline improves the real task objective rather than only the learned proxy.

*c) Reward-model predictive quality:* Figure 5 shows the reward model's validation binary cross-entropy (BCE) over active-learning rounds on held-out preference queries. The downward trend in validation loss across rounds indicates that the CNN is learning a progressively more consistent ranking signal from preference supervision. This is an important sanity check: policy performance depends on the reward model providing a stable and informative learning signal. While the curve is not strictly monotone—which can occur due to stochastic optimization and the changing query distribution under active acquisition—the overall reduction in validation loss suggests that the warm-start reward checkpoint represents a materially better proxy than an untrained model. This helps explain why Relabel can rapidly achieve high success: DQN begins from a meaningful reward model and then continues refining it online, while relabeling keeps the control objective synchronized with reward updates.

Together, Figures 3-5 explain the mechanism behind the final metrics: (i) warm-started preference rewards provide a useful shaping signal early, (ii) online reward updates improve reward-model quality over rounds, and (iii) replay relabeling converts this non-stationary reward learning process into stable off-policy control, yielding faster convergence and substantially better reliability than the No-Relabel ablation.

## VII. Conclusion and Future Work

### A. Conclusion

We presented a preference-based reward learning pipeline for the Tomato Watering (Reward Hacking) task and demonstrated that Warm-Start + Online (Relabel) yields the strongest and most reliable performance under the environment's *true* completion criterion. Across our quantitative results (Tables I-II), the relabeling variant achieves the highest success rate and the shortest episode lengths, while also attaining the best average true episodic return, outperforming both the No-Relabel ablation and the Random baseline.

A key conclusion is that replay-buffer relabeling is essential for stable off-policy learning under a non-stationary learned reward. As the reward model is updated from additional preference queries, previously stored proxy rewards become stale; without relabeling, DQN effectively trains on a mixture of inconsistent objectives, which manifests as instability and reduced reliability. In contrast, relabeling synchronizes policy learning with the current reward model and enables rapid convergence to policies that solve the task according to the intended criterion.

More broadly, these results support the practicality of warm-started preference learning as a substitute for explicit reward engineering. In many real-world settings a reliable scalar reward is unavailable, misspecified, or expensive to define. Warm-starting from offline preference training and then refining the reward online provides an effective pathway to learn a usable optimization signal from comparative feedback, while relabeling maintains the consistency needed for robust control.

### B. Future Work

- Beyond reward hacking - side effects and irreversible changes: A natural next step is to evaluate on AI Safety Gridworlds tasks that emphasize *side effects*

and *reversibility* (e.g., Sokoban variants and other side-effect settings). These environments test whether agents avoid unnecessary collateral changes even when such changes may be instrumentally useful. Preference learning is a particularly natural fit here, since feedback can express nuanced trade-offs that are difficult to encode as a single hand-crafted scalar.

- Scaling across additional gridworld tasks: We plan to test the pipeline on other environments in the AI Safety Gridworlds suite (including tasks with delayed consequences, traps, and specification ambiguity). This would clarify which failure modes are most improved by replay relabeling and which require additional structure in reward modeling or query selection.

- Preference acquisition and query design: While we used ensemble disagreement to prioritize uncertain comparisons, future work could explore richer query families (e.g., longer segments, counterfactual rollouts, or targeted queries near known failure states such as the sprinkler). Another direction is to explicitly measure query efficiency (performance gained per label) under a cost model for overseer time.

- Reward-model robustness and calibration: Future work should include stronger reward diagnostics: calibration of predicted preference probabilities, robustness to distribution shift from offline random-policy data to on-policy rollouts, and explicit checks for shortcut features (e.g., over-weighting appearance changes). Techniques such as regularization, uncertainty-aware losses, or adversarially generated comparisons may improve reliability.

- From simulated to human feedback: We simulated a noisy overseer with soft Bradley–Terry targets. A crucial next step is validating the pipeline with actual human preference data, including interface design, inter-annotator variability, and the impact of inconsistent feedback. Testing whether relabeling preserves its stability benefits under real label noise would strengthen the practical conclusion.

- Generalizing the control backbone: Finally, it would be valuable to study whether the same relabeling principle yields similar benefits for other off-policy algorithms (e.g., distributional DQN or actor–critic variants), and whether hybrid on-policy/off-policy updates reduce sensitivity to reward non-stationarity.

In summary, warm-started preference learning combined with online refinement and replay relabeling is a promising recipe for learning reward signals in environments where an explicit scalar reward is unavailable, while addressing the stability challenges that arise when the learned reward evolves during training.

## References


[1] Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518 (7540), 529–533. https://doi.org/10.1038/nature14236

[2] Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. arXiv preprint arXiv:1606.06565 https://arxiv.org/abs/1606.06565.

[3] Krakovna, V., Uesato, J., Mikulik, V., et al. (2020). Specification gaming: the flip side of AI ingenuity. https://deepmind.google/blog/specification-gaming-the-flip-side-of-ai-ingenuity/.

[4] Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. Advances in Neural Information Processing Systems, 30. https://arxiv.org/abs/1706.03741

[5] Bradley, R. A., & Terry, M. E. (1952). Rank analysis of incomplete block designs: I. The method of paired comparisons. Biometrika, 39(3/4), 324–345. https://doi.org/10.2307/2334029

[6] Lee, K., Smith, L., & Abbeel, P. (2021). PEBBLE: Feedback-Efficient Interactive Reinforcement Learning via Relabeling Experience and Unsupervised Pre-training. Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 6152–6163. https://arxiv.org/abs/2106.05091

[7] Ibarz, B., Leike, J., Pohlen, T., Irving, G., Legg, S., & Amodei, D. (2018). Reward learning from human preferences and demonstrations in Atari. Advances in Neural Information Processing Systems, 31. https://arxiv.org/abs/1811.06521

[8] Leike, J., Martic, M., Krakovna, V., et al. (2017). AI Safety Gridworlds. arXiv preprint arXiv:1711.09883. https://arxiv.org/abs/1711.09883.