Efficient and Adaptive Pong Control via Gated Cell Ensembles

Kilian Merke, Julian Münzer

Final Project Report Machine Learning Project

22. January 2026

Abstract

In this report, we investigate the challenge of adaptive real-time control in the game of Pong, a deceptively simple environment characterized by non-stationary ball dynamics. We propose the Gated Cell Ensemble (GCE), a lightweight, online-adaptive classifier inspired by Mixture of Experts, designed to handle concept drift without the computational overhead of deep Reinforcement Learning. We compare the GCE against standard Decision Trees and incremental Hoeffding Trees in both offline and online settings. Our results indicate that while the GCE offers a novel architectural approach, it currently suffers from stability issues during online adaptation. The Adaptive Hoeffding Tree proved to be the most robust solution, significantly outperforming static baselines. Code available at `https://github.com/julsmzr/pong-ml`.

1 Problem Statement

Pong is a simple and well-known game where players choose among three actions: *UP*, *DOWN*, and *IDLE*. This simplicity makes it an ideal benchmark for machine learning algorithms aiming to autonomously learn game control [13].

The dominant method for such tasks is Reinforcement Learning (RL). While RL achieves outstanding performance when paired with neural networks, it often demands substantial computational resources and presents challenges in interpretation and debugging [15, 14].

Task Difficulty and Constraints. Despite its apparent simplicity, our Pong environment presents non-trivial control challenges due to physical constraints. The paddle moves at 420 pixels per second, while the ball starts at a base speed of 360 pixels per second. The ball accelerates by a factor of 1.1 on each paddle collision, resulting in an exponential speedup:

$$v_n = 360 \times 1.1^n \text{ px/s} \tag{1}$$

where n is the number of paddle hits. After approximately two hits, the ball speed ($\sim$435 px/s) exceeds the paddle speed, and after five hits it reaches approximately 580 px/s. This exponential acceleration creates a transition from purely reactive control in the early game to predictive control in later stages, as the agent can no longer simply track the ball position.

Additionally, the ball launch angle varies up to $\pm 60^\circ$ based on the paddle collision position, introducing trajectory uncertainty. The game runs at 60 FPS with no information delay so the full state vector is available immediately each frame. Unlike high-dimensional Atari benchmarks, our low-dimensional state space (5 features) enables the use of interpretable models while still presenting a meaningful control challenge.

To address the challenges of the constantly changing dynamics in real-time game control, it's crucial to develop a lightweight classifier capable of online adaptation. We propose the **Gated Cell Ensemble (GCE)**, an architecture inspired by Mixture of Experts (MoE) that partitions the input space through distance-based gating. Each cell specializes in a region of the state space, and predictions are aggregated only from activated cells. The ensemble supports incremental learning, allowing continuous adaptation during gameplay without full retraining. This design achieves computational efficiency through simple distance calculations while dynamically adjusting model capacity based on task complexity.

2 Literature Overview

Control in High-Dimensional Spaces. While Pong appears simple, it represents a continuous state-space control problem requiring rapid reaction times. Historically, it has served as a benchmark across various scientific fields, from psychology to computer science, for verifying the efficacy of agents in reactive environments [13, 3].

From Reinforcement to Imitation. Reinforcement Learning (RL) has traditionally been the standard for game control. However, RL suffers from high sample complexity and computational costs [14]. To mitigate this, recent research has shifted focus toward *Imitation Learning*, specifically Behavioral Cloning. This approach simplifies the control problem into a supervised classification task, allowing agents to learn policies directly from expert demonstrations without the instability of reward signal optimization [16, 4].

The Interpretability Gap. A significant drawback of neural-based RL and Imitation agents is their black box nature, making debugging and safety verification difficult [17]. While decision trees offer inherent interpretability, standard implementations (e.g., CART, C4.5) are ill-suited for online learning from continuous data streams involving non-stationary distributions.

Online Ensembles and Mixtures. To bridge the gap between adaptability and interpretability, researchers have proposed Online Decision Trees, such as the Hoeffding Tree, which updates incrementally [6, 8]. The Gated Cell Ensemble architecture proposed in this project shares theoretical grounding with Mixture of Experts systems. In MoEs, a gating network partitions the input space and assigns specific regions to specialized sub-models, offering a balance between model capacity and computational efficiency [11].

3 Proposed Method

This section details the formulation and design of our proposed solution. We first outline the shift from reinforcement learning to supervised imitation, followed by a formal definition of the Gated Cell Ensemble classifier and then explain our online training methodology.

3.1 Online Supervised Learning with an Oracle

We approach the Pong problem as a supervised classification task. For offline training, we utilize a Human and a PC player to generate a labeled dataset mapping states $\mathbf{x}$ to optimal actions y. For the online phase, we do not rely on standard imitation of a human; instead, we implement *Self-Supervised Learning with an Oracle*. We calculate a heuristic target, i.e. where the ball will land and use this "oracle" calculation to generate labels and reward signals for the agent in real-time. This allows us to train lightweight classifiers to mimic optimal strategies while maintaining the capability to handle online data streams effectively.

3.2 Gated Cell Ensemble Classifier

To address the challenges of real-time data streams, we propose the *Gated Cell Ensemble* (Figure 1). This classifier comprises multiple cells, each equipped with a *Gate* and a *Decision Element* (Figure 2).

The Gate $G(\mathbf{x})$ determines if a specific cell is appropriate for the current state vector. The Decision Element $D(\mathbf{x})$ generates a probability vector for the action classes. The final prediction is an aggregation of decision elements where the gate output meets a threshold. This structure allows the ensemble to partition the state space while remaining computationally efficient.

3.2.1 Gate Mechanism

Each cell's gate determines its fitness for a given input sample using a distance-based activation criterion. The gate maintains a learned vector $\mathbf{g} \in \mathbb{R}^{|F_g|}$, where F_g is a subset of input features assigned to the gate. Activation occurs when:

$$\|\mathbf{x}[F_g] - \mathbf{g}\|_2 < \tau \tag{2}$$

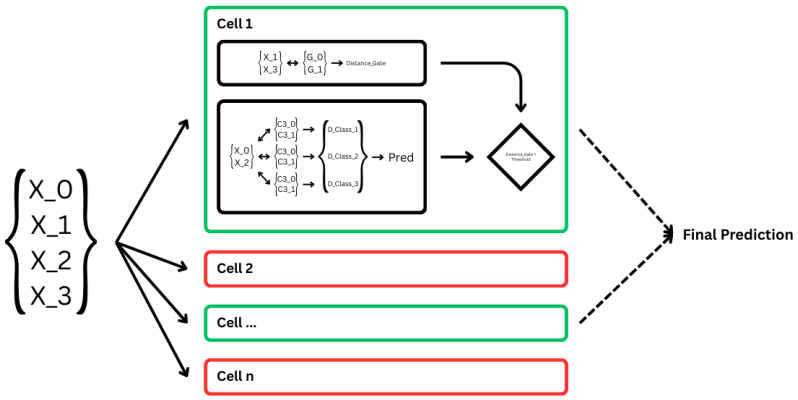

Figure 1: Schematic of the Gated Cell Ensemble Architecture

where τ is the boundary threshold (default: 0.5 for normalized features). Features are randomly partitioned between the gate and decision components at cell initialization, with each component receiving a disjoint subset. The gate learning rate is set to 0.01 to ensure stable adaptation.

3.2.2 Decision Element

When a gate activates, the corresponding decision element generates class predictions. Each decision element maintains a matrix $\mathbf{D} \in \mathbb{R}^{C \times |F_d|}$, where C is the number of classes (3 for UP/DOWN/IDLE) and F_d is the decision feature subset. Class distances are computed as:

$$d_c = \|\mathbf{x}[F_d] - \mathbf{D}_c\|_2 \quad \text{for } c \in \{1, \ldots, C\} \tag{3}$$

These distances are normalized to produce class weights, with the predicted class being $\arg\min_c(d_c)$. The decision learning rate is set to 0.05 for faster adaptation to action patterns.

3.2.3 Aggregation Mechanism

Only cells whose gates activate contribute to the final prediction. The aggregation computes the mean of normalized class-distance vectors from all activated cells:

$$\hat{y} = \arg\min_c \left(\frac{1}{|A|} \sum_{i \in A} \tilde{d}_c^{(i)} \right) \tag{4}$$

where A is the set of activated cells and $\tilde{d}_c^{(i)}$ is the normalized distance to class c from cell i. If no cells activate, the classifier falls back to a random prediction.

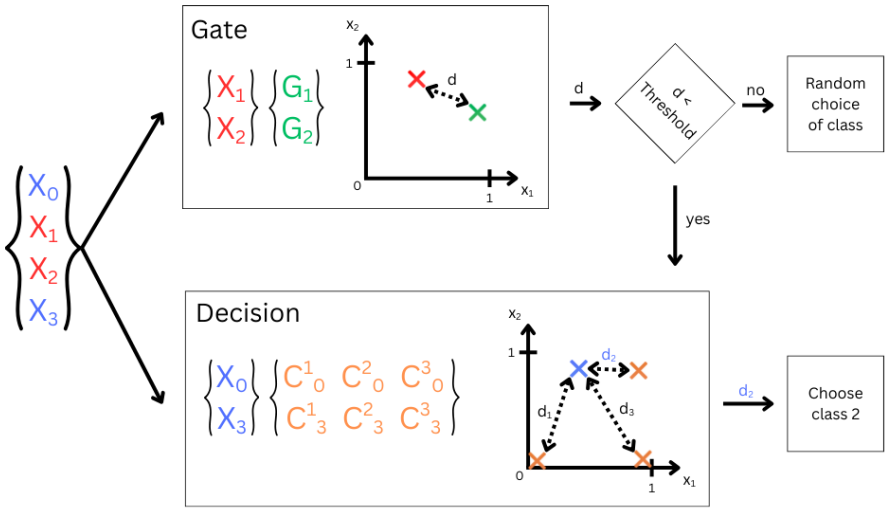

Figure 2: Schematic single cell for 4 input features and 3 output classes

3.2.4 Cell Lifecycle Management

The ensemble dynamically manages its cell population. It initializes with 3 and adds new cells when overall accuracy falls below the accuracy threshold(80%) after 1000 predictions. Cells are removed based on the following criteria:

- **Inactivity:** Lifetime exceeds 150 samples and action rate falls below 0.5%
- **Poor accuracy:** Lifetime exceeds 150 samples and accuracy falls below 20%
- **Suboptimal performance:** Lifetime exceeds 1000 samples and accuracy remains below the target goal
- **Redundancy:** Two cells share the same gate features and their gate vectors are within Euclidean distance 1.5

3.2.5 Online Learning Algorithm

The classifier processes samples sequentially, performing a forward pass followed by a backward pass. During the forward pass, each cell accumulates the input features in a buffer. During the backward pass, vectors are updated based on accumulated history with exponential decay ($\lambda = 0.9$). In our experiment we perform a backward after each forward path such that the exponential decay is irrelevant.

Algorithm 1 Gated Cell Ensemble Forward Pass

function FORWARD($\mathbf{x}$)
  $predictions \leftarrow []$
  for each *cell* in *cells* **do**
    $d_{gate} \leftarrow \|\mathbf{x}[F_g] - cell.\mathbf{g}\|_2$
    if $d_{gate} < \tau$ **then**
      for each class c in $\{1, 2, 3\}$ **do**
        $d_c \leftarrow \|\mathbf{x}[F_d] - cell.\mathbf{D}_c\|_2$
      end for
      $\tilde{\mathbf{d}} \leftarrow \text{normalize}([d_1, d_2, d_3])$
      $predictions$.append($\tilde{\mathbf{d}}$)
    end if
  end for
  if *predictions* is empty **then**
    return random class
  end if
  return $\arg\min(\text{mean}(predictions))$
end function

$$\Delta\mathbf{v} = \sum_{t=0}^{T} \lambda^t (\mathbf{x}_t - \mathbf{v}) \tag{5}$$

For correct predictions, vectors move toward the input: $\mathbf{v} \leftarrow \mathbf{v} + \eta\Delta\mathbf{v}$. For incorrect predictions, vectors move away: $\mathbf{v} \leftarrow \mathbf{v} - \eta\Delta\mathbf{v}$, where η represents the specific learning rate for either the gate or decision element.

3.2.6 Hyperparameter summary

To select proper hyperparameter for our proposed classifier GCE we used a grid search over multiple possible hyperparameters. The ranges arrived from experience we got in the developing process and from logical boundaries based

Similarity threshold \ Initial Cells	**3**	**5**	**7**	
0.5	3	2	3	2.67
0.75	2	1	0	1.00
1.0	2	2	4	2.67
1.25	4	2	1	2.33
1.5	**4**	2	4	**3.33**
1.75	2	2	0	1.33
2.0	2	2	1	1.67
Avg hits	**2.71**	1.86	1.86	2.14

Table 1: Grid search performance - Measured hits

Parameter	Value	Description
Initial cells	3	Number of cells at initialization
Gate boundary τ	0.5	Distance threshold for gate activation
Gate learning rate	0.01	Learning rate for gate vectors
Decision learning rate	0.05	Learning rate for decision vectors
Learning decay λ	0.95	Exponential decay for temporal weighting
Accuracy goal	0.65	Target accuracy for cell management
Initializer range	$[0, 1]$	Range for vector initialization
Min lifetime (pruning)	150	Minimum samples before pruning eligible
Min action rate	0.5%	Threshold for inactivity removal
Min Gate Distance	1.25	Threshold for similarity removal

Table 2: Gated Cell Ensemble hyperparameters

on the classifier structure. We mainly focused on two parameters: Number of initial starting cells and similarity threshold of cell gates. Both are crucial for final performance but difficult to estimate. The results of the grid search are shown in Table 1.

Table 2 summarizes the key hyperparameters of the Gated Cell Ensemble.

3.3 Online Training Strategy

Unlike offline training, which relies on a static dataset, our online strategy updates model parameters in real-time. This requires a mechanism to generate self-supervised signals from the game state transitions. We implement a trajectory-based target prediction system that feeds into a reward function, closing the loop with model-specific update rules.

3.3.1 Target Position Prediction

To generate a reliable reference signal, the agent must anticipate where the ball will intersect the paddle's x-coordinate ($x_p = 864$). Because the ball originates from the opponent, we can calculate the projected impact point y^* on the agent's side, accounting for velocity $\mathbf{v} = (\cos\theta, \sin\theta)$ and wall reflections.

Given the horizontal distance to the paddle $\Delta x = x_p - x_{ball}$, the unconstrained vertical displacement is $\Delta y = (\Delta x / v_x) \cdot v_y$. The raw target position is $\tilde{y} = y_{ball} + \Delta y$. We account for the top ($y = 0$) and bottom ($y = H$) game boundaries by determining the number of bounces $n = \lfloor |\tilde{y}| / H \rfloor$. The final target y^* is derived via reflection parity:

$$y^* = \begin{cases} \tilde{y} \mod H & \text{if } n \text{ is even} \\ H - (\tilde{y} \mod H) & \text{if } n \text{ is odd} \end{cases} \tag{6}$$

Figure 3 illustrates this trajectory prediction with a single wall reflection.

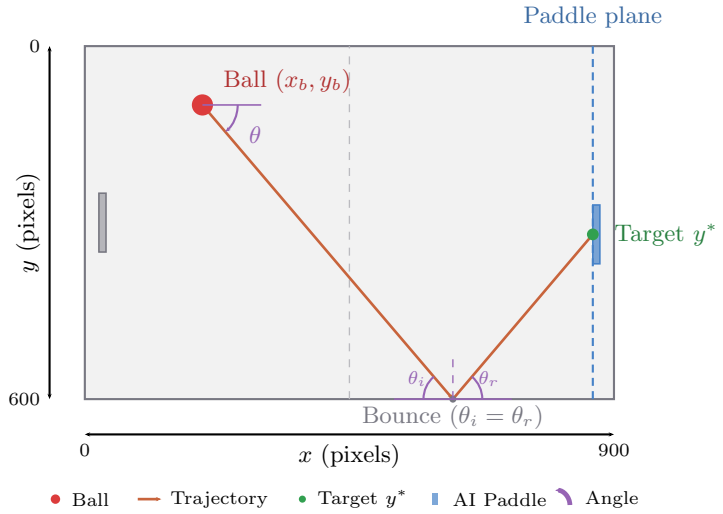

Figure 3: Trajectory prediction for the Pong AI. Given ball position (x_b, y_b) and velocity angle θ, the system computes the intersection point y^* at the paddle plane. Wall bounces follow the law of reflection ($\theta_i = \theta_r$).

3.3.2 Reward Signal Generation

We employ specific reward formulations tailored to the update mechanisms of the different adaptive models.

Continuous Reward (Hoeffding Tree). For the Hoeffding Tree, we compute a composite scalar reward $r(s_t, a_t)$ combining defensive success and positioning. The dominant component is the **Hit Bonus** ($r_{\text{hit}} = 10.0$), awarded when the paddle successfully returns the ball. A **Proximity Reward** encourages the agent to minimize the distance to the target y^* while the ball is approaching:

$$r_{\text{prox}} = \max\left(0, 1 - \frac{|y^* - y_{paddle}|}{300}\right) \tag{7}$$

Secondary components include a survival time bonus ($r_{\text{time}} = 0.01$) and an idle penalty ($r_{\text{idle}} = -0.1$) to discourage passivity. These continuous signals are discretized into optimal action labels ($\hat{a} \in \{\text{UP}, \text{DOWN}, \text{IDLE}\}$) to feed the tree's supervised update method.

Boolean Reward (Gated Cell Ensemble). The GCE utilizes a binary signal $r_b \in \{\text{True}, \text{False}\}$ indicating if the previous action improved the state. A move is considered beneficial if the paddle is within a tolerance threshold $\tau_{\text{thresh}} = 50\text{px}$ of the target, indicating a ball hit with $paddle\_height = 100px$, or if the distance to the target has decreased ($\Delta d < 0$):

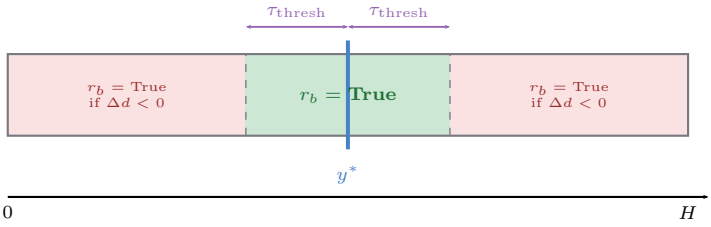

Figure 4: Boolean reward regions for GCE. The agent receives $r_b = \text{True}$ when the paddle is within threshold $\tau_{\text{thresh}} = 50\text{px}$ of target y^* (green), or when reducing distance to y^* in the outer regions (red).

$$r_b = (d_{t+1} < \tau_{\text{thresh}}) \vee (d_{t+1} < d_t) \tag{8}$$

Figure 4 visualizes the acceptance regions for this binary reward. When the ball moves away from the paddle, the target y^* defaults to the center of the screen ($H/2$) to center the agent.

3.3.3 Online Update Cycle

The learning process operates frame-by-frame. At timestep t, the model predicts action a_t given state $\mathbf{x}_t$. At $t+1$, the reward is computed, triggering the backward pass. For the Gated Cell Ensemble, the boolean reward r_b modifies the vector update direction defined in Equation 5:

$$\mathbf{v} \leftarrow \mathbf{v} + \eta \cdot \text{sign}(r_b) \cdot (\mathbf{x}_t - \mathbf{v}) \tag{9}$$

where $\text{sign}(r_b)$ is $+1$ for correct decisions (attracting the cell vector to the input) and -1 for incorrect decisions (repelling the vector). This creates a lightweight clustering effect where cells migrate toward state regions where they successfully predict the target trajectory.

4 Plan of Experiments

To evaluate the proposed Gated Cell Ensemble, we design an experimental framework comparing it against established baselines. This section details the pipeline, including data acquisition, model training, simulation-based metrics and statistical analysis.

4.1 Data Collection and Preprocessing

Environment. We have developed an interactive Pong game using Python and Pygame. This environment generates the data stream for training and serves as the simulation environment for evaluation.

Feature Vector. The state is represented by a vector $\mathbf{x} = [y_{paddle}, x_{ball}, y_{ball}, \theta_{ball}, v_{ball}]$. Note that for the agent, we exclude the opponent's paddle position to generalize the state space.

Data Fusion. To ensure robustness [4, 10], our training dataset contains games played with the left paddle as human-controlled while combining two sources for the right paddle:

1. **PC Player:** A programmatic agent moving the paddle aiming to reach $y_{paddle} \approx y_{ball}$.
2. **Human Player:** A human player introducing 'imperfect', yet corrective and predictive, data samples.

4.2 Algorithms and Models

PC Player. An algorithm designed to minimize the delta between paddle and ball height with a *stay idle* threshold of $|y_{paddle} - y_{ball}| \leq 10px$ to avoid excessive jitter.

Decision Tree. A standard implementation of a decision tree using scikit-learn [2].

HoeffdingTreeClassifier. An incremental decision tree algorithm designed for high-speed data streams.

Proposed Gated Cell Ensemble (GCE). Our custom classifier designed for stream-based imitation learning with online adaptation capability.

4.3 Experimental Loop

Training. We use the full dataset to offline train all models. The adaptive models then undergo 50 episodes of online-training where one episode is a game until a score of 5 is reached by one player. Afterwards these models compete in the simulation.

Simulation. We let the trained classifiers play games against the PC Player in a competition, setting a score limit of 5.

Metric Collection. We capture *Total Survival Time* as the primary metric. Secondary metrics include the number of successful ball returns, inference latency, and final score difference.

Metric Justification. Our choice of metrics follows established practices in game-playing agent evaluation. Survival time serves as the primary metric, consistent with episode length measurements used in the Arcade Learning Environment [1] and subsequent Atari benchmarks [12]. Ball returns (hits) quantify successful defensive actions and directly measure the agent's core objective. Finally, inference latency is critical for real-time control at 60 FPS, where each frame allows only 16.67ms for prediction. Together, these metrics capture defensive capability, policy quality, and computational efficiency.

4.4 Statistical Tests

Hypothesis. There exists a notable performance disparity among the various models.

Procedure. Following Demsar's methodology [5], we will perform a 10×5 repeated cross-validation both for offline pre-trained models on accuracy metrics and for online trained models on the simulation metrics. Significance will be assessed using the Friedman test [7], followed by a paired Wilcoxon post-hoc analysis with Hommel correction if significant differences are found [18, 9].

Offline Cross-Validation Protocol. We employ a frame-level data split strategy, where individual frames are randomly assigned to training or test sets regardless of their originating game session. While this approach risks some temporal correlation between consecutive frames, it maximizes data utilization and reduces variance in performance estimates. The 10×5 cross-validation consists of 10 repetitions of 5-fold cross-validation, yielding 10 total evaluations per model since we average each 5-fold evaluation. Each fold uses 80% of the data for training and 20% for testing. Final metrics are aggregated across all 10 evaluations for statistical testing.

Online Evaluation Protocol. Each model competes against the PC player for 20 consecutive games, with each match continuing until a score of 5 is reached. We conduct statistical analysis on the performance metrics derived from these sessions.

4.5 Additional Analysis

We plan to examine whether the models are trained to follow the ball y-coordinate strictly or if they adopt alternative strategies, such as ball vector prediction. Additionally, we will analyze the impact of hyperparameter tuning on the Gated Cell Ensemble.

Model	Accuracy (%)	Latency (μs)
Decision Tree	**88.12 ± 0.16**	**0.10 ± 0.01**
Hoeffding Tree	63.08 ± 0.34	10.10 ± 0.83
Gated Cell Ensemble	43.54 ± 0.45	102.88 ± 4.41

Table 3: Offline Cross-Validation Results (Mean ± Std. Dev.). Inference latency measured on CPU on a Macbook Air M4 with 10 CPU Cores. All results indicate real-time capability.

5 Results

This section presents the findings of our experiments. We report performance metrics for both the offline cross-validation on the static dataset and the online simulation against the algorithmic opponent. Statistical significance is verified using the procedures defined in the methodology. We employ a significance level of $\alpha = 0.05$ for all p-values.

5.1 Offline Cross-Validation

Table 3 summarizes the aggregated performance metrics from the 10×5 repeated stratified cross-validation. The Decision Tree serves as the baseline for classification accuracy and inference speed.

Statistical Analysis. We applied the Friedman test to the accuracy results across the folds. The test revealed a statistically significant difference in performance ($\chi^2_F = 100.0, p < 0.001$). A post-hoc Wilcoxon signed-rank test with Hommel correction indicated significant differences between all model pairs ($p < 0.001$ for all comparisons).

5.2 Online Simulation Performance

The models were evaluated in the interactive Pong environment. We compare the *Pretrained* variants (weights frozen after offline training) and the *Adaptive* variants (weights frozen after 20 episodes of training). Table 4 details the survival metrics and game scores.

5.3 Qualitative Analysis of Online Behavior

Beyond the aggregate metrics, we conducted an informal qualitative review of the gameplay footage to understand the failure modes of the agent. We observed that performance was highly inconsistent. In specific rallies, the agent returned the ball almost perfectly, executing fast maneuvers toward the perfect target_y position. However, these periods of successful play were frequently interrupted by episodes where the agent would almost freeze or exhibit erratic, non-strategic movement.

This behavior often occurred when the paddle started in a position slightly different from the training distribution, even if the ball trajectory was standard. This suggests that the model may be suffering from concept drift or poor generalization ability. The agent appears to overfit to specific spatial configurations, and when the online update shifts the decision boundaries, it occasionally leads to a collapse in policy rather than a gradual improvement.

5.4 Statistical Evaluation of Online Adaptation

To validate the performance differences, we analyzed the survival times and returns using the Friedman test followed by post-hoc comparisons.

Global Comparison. The Friedman test revealed a statistically significant difference in performance distributions across the models for both Survival Time ($\chi_F^2 = 46.56, p < 0.001$) and Average Returns ($\chi_F^2 = 49.49, p < 0.001$). However, no significant difference was found regarding the Goal Difference ($p = 0.48$).

Pairwise Analysis. Post-hoc analysis (Wilcoxon with Hommel correction) indicated that the Adaptive Hoeffding Tree significantly outperformed the Static Gated Cell Ensemble in terms of Survival Time ($p < 0.001$) and Returns ($p < 0.001$).

Contrary to our initial hypothesis, the Adaptive variants did not show statistically significant improvements over their Static counterparts in this specific experimental setup:

- **Hoeffding Tree:** The adaptive variant improved mean survival time ($30.65s \rightarrow 35.00s$), but this was not statistically significant ($p = 0.17$). Similarly, the improvement in returns was not significant ($p = 0.16$).
- **Gated Cell Ensemble (WF):** The adaptive variant showed mean deterioration ($23.18s \rightarrow 14.09s$), which was statistically significant ($p < 0.001$).

The standard Decision Tree (Static) remained a competitive baseline. It significantly outperformed the Static Gated Cell Ensemble in both Survival Time

Model	Variant	Survival Time (s)	Avg. Returns
Decision Tree	Static	30.37 ± 7.92	4.55 ± 2.33
Hoeffding Tree	Static	30.65 ± 7.19	4.75 ± 2.07
Hoeffding Tree	Adaptive	$\mathbf{35.00 \pm 10.56}$	$\mathbf{6.30 \pm 3.67}$
Gated Cell Ensemble	Static	23.18 ± 9.95	2.45 ± 2.65
Gated Cell Ensemble	Adaptive	14.09 ± 0.85	0.05 ± 0.22

Table 4: Online Simulation Results over 20 game sessions per model. Games were played until a score of 5 is reached by one player. *Returns* indicates the average number of paddle hits per game.

($p = 0.042$) and Returns ($p = 0.04$). When compared to the best performing model, the Adaptive Hoeffding Tree, the Decision Tree showed significantly lower Returns ($p = 0.037$), though the difference in Survival Time was not strictly significant ($p = 0.066$).

6 Conclusion

In this work, we introduced the Gated Cell Ensemble (GCE) as a candidate for adaptive real-time control. We compared its performance against established Decision Tree and Hoeffding Tree baselines in a high-velocity Pong environment.

Our experimental results indicate that in the current configuration, the Gated Cell Ensemble did not outperform the simpler baselines. The Adaptive Hoeffding Tree demonstrated the highest stability and return rates. While the Adaptive Hoeffding Tree showed trends toward higher mean survival times compared to its static version, these improvements were not statistically significant (Wilcoxon $p = 0.17$). However, it did achieve a statistically significant improvement in Average Returns compared to the static Decision Tree baseline.

We hypothesize that the catastrophic failure of the Adaptive GCE (0.05 returns) stems from instability in the vector update mechanism. The boolean reward signal ($+1/-1$) likely caused vector oscillation, where cell centroids were repelled out of the valid feature space, leading to a collapse in decision boundaries.

6.1 Future Outlook

While our proposed classifier, GCE, does not yet surpass existing classifiers, it demonstrates promising potential in identifying patterns across different states more effectively than random decisions. Future research should focus on continuing the training of GCE on various problems, ideally with extended training durations. Additionally, the effectiveness of online training capabilities should be investigated.

To enhance performance, a thorough re-evaluation of hyperparameters is essential, along with structural modifications to the model. Implementing a different selection algorithm for the 'expert' within the mixture of experts system could also improve both the performance and the stability.

Finally, it is crucial to examine why current methods such as the Decision Tree and Hoeffding Tree underperform in the specific context of the Pong problem. Insights from this analysis could inform future adjustments in our environment design.

References

[1] M. G. Bellemare et al. "The Arcade Learning Environment: An Evaluation Platform for General Agents". In: *Journal of Artificial Intelligence*

Research 47 (June 2013), pp. 253–279. ISSN: 1076-9757. DOI: 10.1613/jair.3912. URL: http://dx.doi.org/10.1613/jair.3912.

[2] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (Oct. 2001), pp. 5–32. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[3] R Michael Brown et al. "Gender and video game performance". In: *Sex roles* 36.11 (1997), pp. 793–812.

[4] The Viet Bui, Tien Mai, and Hong Thanh Nguyen. *MisoDICE: Multi-Agent Imitation from Unlabeled Mixed-Quality Demonstrations*. 2025. arXiv: 2505.18595 [cs.LG]. URL: https://arxiv.org/abs/2505.18595.

[5] Janez Demšar. "Statistical Comparisons of Classifiers over Multiple Data Sets". In: *J. Mach. Learn. Res.* 7 (Dec. 2006), pp. 1–30. ISSN: 1532-4435.

[6] Pedro Domingos and Geoff Hulten. "Mining high-speed data streams". In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2000, pp. 71–80.

[7] Milton Friedman. "The use of ranks to avoid the assumption of normality implicit in the analysis of variance". In: *Journal of the american statistical association* 32.200 (1937), pp. 675–701.

[8] Abhinav Garlapati et al. "A reinforcement learning approach to online learning of decision trees". In: *arXiv preprint arXiv:1507.06923* (2015).

[9] G. HOMMEL. "A stagewise rejective multiple test procedure based on a modified Bonferroni test". In: *Biometrika* 75.2 (June 1988), pp. 383–386. DOI: 10.1093/biomet/75.2.383. eprint: https://academic.oup.com/biomet/article-pdf/75/2/383/828082/75-2-383.pdf. URL: https://doi.org/10.1093/biomet/75.2.383.

[10] Borja Ibarz et al. *Reward learning from human preferences and demonstrations in Atari*. 2018. arXiv: 1811.06521 [cs.LG]. URL: https://arxiv.org/abs/1811.06521.

[11] Robert A Jacobs et al. "Adaptive mixtures of local experts". In: *Neural computation* 3.1 (1991), pp. 79–87.

[12] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (2015), pp. 529–533. DOI: 10.1038/nature14236. URL: https://doi.org/10.1038/nature14236.

[13] Benedikt Nork et al. "Machine Learning with the Pong Game: A Case Study". In: *Engineering Applications of Neural Networks*. Ed. by Elias Pimenidis and Chrisina Jayne. Cham: Springer International Publishing, 2018, pp. 106–117.

[14] Johan S. Obando-Ceron and Pablo Samuel Castro. *Revisiting Rainbow: Promoting more Insightful and Inclusive Deep Reinforcement Learning Research*. 2021. arXiv: 2011.14826 [cs.LG]. URL: https://arxiv.org/abs/2011.14826.

[15] Erika Puiutta and Eric MSP Veith. *Explainable Reinforcement Learning: A Survey*. 2020. arXiv: 2005.06247 [cs.LG]. URL: https://arxiv.org/abs/2005.06247.

[16] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. *A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning*. 2011. arXiv: 1011.0686 [cs.LG]. URL: https://arxiv.org/abs/1011.0686.

[17] Abhinav Verma et al. *Programmatically Interpretable Reinforcement Learning*. 2019. arXiv: 1804.02477 [cs.LG]. URL: https://arxiv.org/abs/1804.02477.

[18] Frank Wilcoxon. "Individual comparisons by ranking methods". In: *Biometrics bulletin* 1.6 (1945), pp. 80–83.