Predicting Game Difficulty and Engagement Using AI Players

SHAGHAYEGH ROOHI, Aalto University, Finland
CHRISTIAN GUCKELSBERGER, Aalto University, Finland
ASKO RELAS, Rovio Entertainment, Finland
HENRI HEISKANEN, Rovio Entertainment, Finland
JARI TAKATALO, Rovio Entertainment, Finland
PERTTU HÄMÄLÄINEN, Aalto University, Finland

This paper presents a novel approach to automated playtesting for the prediction of human player behavior and experience. We have previously demonstrated that Deep Reinforcement Learning (DRL) game-playing agents can predict both game difficulty and player engagement, operationalized as average pass and churn rates. We improve this approach by enhancing DRL with Monte Carlo Tree Search (MCTS). We also motivate an enhanced selection strategy for predictor features, based on the observation that an AI agent’s best-case performance can yield stronger correlations with human data than the agent’s average performance. Both additions consistently improve the prediction accuracy, and the DRL-enhanced MCTS outperforms both DRL and vanilla MCTS in the hardest levels. We conclude that player modelling via automated playtesting can benefit from combining DRL and MCTS. Moreover, it can be worthwhile to investigate a subset of repeated best AI agent runs, if AI gameplay does not yield good predictions on average.

CCS Concepts: • Human-centered computing → User models; • Computing methodologies → Modeling and simulation.

Additional Key Words and Phrases: Player Modelling, AI Playtesting, Game AI, Difficulty, Player Engagement, Pass Rate Prediction, Churn Prediction, Feature Selection

ACM Reference Format:

1 INTRODUCTION

The development of a game typically involves many rounds of playtesting to analyze players’ behaviors and experiences, allowing to shape the final product so that it conveys the design intentions and appeals to the target audience. The tasks involved in human playtesting are repetitive and tedious, come with high costs, and can slow down the design and development process substantially. Automated playtesting aims to alleviate these drawbacks by reducing the need for...
human participants [10, 22, 56]. An active area of research, it combines human computer interaction (HCI), games user research, game studies and game design with AI and computational modeling.

In this paper, we focus on the use of automated playtesting for player modelling, i.e. “the study of computational means for the modeling of a player’s experience or behavior” [64, p. 206]. A computational model that predicts experience and/or behavior accurately and at a reasonable computational cost can either assist game designers or partially automatize the game development process. For instance, it can inform the selection of the best from a range of existing designs, or the procedural generation of optimal designs from a potentially vast space of possibilities [39, 59, 63].

Recently, researchers have started to embrace AI agents in automated playtesting, based on advances in deep reinforcement learning (DRL) which combines deep neural networks [20] with reinforcement learning [57]. Reinforcement learning allows for an agent to use experience from interaction to learn how to behave optimally in different game states. Deep neural networks support this by enabling more complex mappings from observations of a game state to the optimal action. DRL has enabled artificial agents to play complex games based on visual observations alone [35], and to exhibit diverse [52] and human-like [2] gameplay. DRL agents can thus be employed in a wide range of playtesting tasks such as identifying game design defects and visual glitches, evaluating game parameter balance, and, of particular interest here, in predicting player behavior and experience [2, 15, 21, 24, 32].

In this paper, we introduce two extensions to an automated playtesting algorithm developed in prior work [46], in which we used DRL game-playing agents for player modelling, more specifically to predict player engagement and game difficulty. In particular, this algorithm combined simulated game-playing via deep reinforcement learning (DRL) agents with a meta-level player population simulation. The latter was used to model how players’ churn in some game levels affects the distribution of player traits such as persistence and skill in later levels. Originating from an industry collaboration with Rovio Entertainment, we evaluated our algorithm on Angry Birds Dream Blast [18], a popular free-to-play puzzle game. We operationalized engagement as level churn rate, i.e., the portion of players quitting the game or leaving it for an extended duration after trying a level. Game difficulty was operationalized as pass rate – the probability of players completing a level. Understanding and estimating churn is especially important for free-to-play games, where revenue comes from a steady stream of in-game purchases and advertisement; hence, high player churn can have a strong negative impact on the game’s financial feasibility. Well-tuned game difficulty is essential in facilitating flow experiences and intrinsic motivation [17, 49, 50, 58]. These two factors are not independent: previous work has linked excessive difficulty to player churn [6, 45, 47].

Our existing algorithm strongly appeals to games industry applications as it is capable of modeling individual player differences without the costly training of multiple DRL agents. However, the pass and churn rate predictions leave room for improvement, in particular the pass rate predictions of the most difficult levels. Our two extensions overcome these shortcomings, and are informed by the following research questions:

**RQ1** Considering the success of Monte Carlo Tree Search (MCTS) methods to steer AI agents in challenging games like Go [54], could MCTS also improve how well AI players can predict human player data?

**RQ2** Can one improve the AI gameplay features used by the population simulation, e.g., through analyzing correlations between AI gameplay statistics and the ground truth human data?

To answer these questions, we have implemented and compared $2 \times 2 \times 3 = 12$ approaches resulting from combining the two predictive models introduced in our previous work [46] with either DRL or MCTS agents (RQ1) and three feature selection approaches (RQ2). Similar to the original paper, we apply and evaluate our extension in Angry Birds Dream Blast [18]. We find that combining DRL with
MCTS, and using AI gameplay features from the best agent runs, yield more accurate predictions in particular for the pass rates. These improvements contribute to the applicability of human player engagement and difficulty prediction in both games research and industry applications.

2 BACKGROUND

We next provide the formal basis for readers from varied disciplines to understand our contributions and related work. We first formalize game-playing, the basis of our player modelling approach, as a Markov decision process. We then introduce deep reinforcement learning and Monte Carlo tree search as the two central AI techniques in our approach.

2.1 Markov Decision Processes

Game-playing can be considered a sequential decision-making problem. A Markov decision process (MDP) [4] represents a formal framework to describe such a problem, modelling the possible interaction between an arbitrary agent and its environment over time. At each point of the interaction, the agent can receive a reward from the environment. The problem consists of finding a policy, i.e. a set of rules associating each state with an action, that maximizes the reward which the agent can accumulate over time [57, p. 70].

Formally, the environment state at time-step $t \geq 0$ is denoted $s_t \in S$. An agent’s action $a_t \in A$ can influence the future state of the environment $s_{t+1}$ determined by the environment dynamics given as a conditional probability distribution $p(s_{t+1}|s_t, a_t)$. The actions available to an agent can depend on the current environment state. An action, leading to a transition between states, produces an immediate reward signal $r_{t+1} \in \mathbb{R}$, determined by the reward function $\rho(s_{t+1}, a_t, s_t) = r_{t+1}$. The Markov assumption implies that $s_t$ must encode all relevant information about the past agent-environment interaction that matters for how the process unfolds in the future [44, pp. 1-2].

Here, agent and environment are given by the player and game world, respectively, and the problem consists of finding a policy to score highly in the game. The game mechanics and scoring function fulfil the role of the environment dynamics and reward function, respectively.

2.2 (Deep) Reinforcement Learning

While an agent interacts with an MDP environment, they collect experience in the form of state, action, reward, and next state sequences $(s_t, a_t, r_{t+1}, s_{t+1})$. Reinforcement learning (RL) allows an agent to solve an MDP by learning a policy $\pi(a_t|s_t)$ that maximizes the expected cumulative future reward $\mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t r_t \right]$ from this experience. The discount factor $\gamma \in [0, 1]$ can be used to give near-future rewards higher priority. For simple problems, the required functions such as the policy can be represented by lookup tables. However, this becomes unfeasible for problems with large state and action spaces, as present in most video games. Deep reinforcement learning (DRL) comes to the rescue by approximating these functions with deep neural networks.

In this paper, we use Proximal Policy Optimization (PPO) [51], a popular, state-of-the-art DRL algorithm. It is an example of policy gradient methods, which parametrize the policy $\pi_\theta$ with $\theta$, the weights of a deep neural network. In the most basic of such methods, the policy is optimized based on the objective $J(\theta) = \mathbb{E} \left[ \pi_\theta(a_t|s_t)A(s_t, a_t) \right]$. The advantage function $A(s_t, a_t)$ captures the additional reward an agent could obtain by choosing the specific action $a_t$ over any action in the state $s_t$. Gradient ascent on this objective thus directs the policy towards actions that produce more reward in a given state. PPO contributes a modified objective function which limits the size of policy updates, allowing for smoother learning. The algorithm has been used in many applications, ranging from robot control to computer animation and game playing. However, PPO specifically and DRL more generally may require extensive multi-day training runs to learn, which is why this paper also investigates Monte Carlo tree search as an alternative.
2.3 Monte Carlo Tree Search

Monte Carlo tree search (MCTS) is an alternative method to solving MDPs. It estimates the optimal action by building a tree of possible future (game) states and rewards, with each tree node corresponding to the state resulting from an explored action. The root of the tree is the current state. The algorithm iterates over four steps [9, 64]:

1. **Select:** Traverse down the tree using the Upper Confidence Bound for Trees (UCT) formula until at a node with non-explored actions is hit. The selection of a child node is based on:

   \[
   \argmax_{s_{d+1} \in C(s_d)} \frac{V(s_{d+1})}{N_{s_{d+1}}} + c \sqrt{\frac{\ln N_s}{N_{s_{d+1}}}}
   \]  

   The UCT formula balances the exploration of nodes about which little is known, with the exploitation of nodes that have already shown to yield high reward. Here, \(C(s_d)\) denotes the children set of the under-inspection node \(s_d\) at depth \(d\), and \(N_{s_d}\) represents how often this node has already been visited. Moreover, \(s_{d+1}\) denotes a child node, \(V(s_{d+1})\) its value, and \(N_{s_{d+1}}\) how often it has been visited so far. The \(c\) parameter adjusts the exploration-exploitation trade-off, and we adopt the common value [29] of \(\sqrt{2}\) for our experiments.

2. **Expand:** The selected node is expanded by taking a random non-explored action. The resulting game state is added as a new child node of the selected node.

3. **Simulate rollout:** A forward simulation is used to advance the game from the added child with random actions until a terminal state is hit (level passed) or a move budget is consumed. The value of the added child \(s_{d+1}\) at depth \(d + 1\) is updated to \(\hat{V}(s_{d+1}) = \sum_t \gamma^t r_t\), and the child’s visit number increases to one \(N_{s_{d+1}} = 1\).

4. **Backpropagate:** The value estimates \(\hat{V}(s_{d+1})\) resulting from the simulation are propagated from the added child \(s_{d+1}\) up to the root \(s_0\), updating node visiting counts and values as:

   \[
   N_s \leftarrow N_s + 1, \quad \hat{V}(s_d) \leftarrow r_d + \gamma \hat{V}(s_{d+1}), \quad V(s_d) \leftarrow V(s_d) + \hat{V}(s_d).
   \]  

In contrast to DRL, MCTS requires no time-consuming and computationally expensive training; its estimation can be stopped at any time, and it can thus be used with arbitrary and dynamically changing computational budgets. MCTS is useful for long-term planning and has shown promising results in game-playing agents [23, 25, 43]. It has gained additional popularity after being used to reach superhuman performance in the game Go [54].

3 RELATED WORK

There exist numerous approaches to automated playtesting for player modelling, differing in what types of games they are applicable to, and what kind of player behavior and experience they are able to predict. We first discuss exclusively related work that employs the same state-of-the-art AI techniques that our research relies on – see Albaghajati and Ahmed [1] for a more general review. We follow this up by a more specialized discussion of how difficulty and engagement relate, and how these concepts can be operationalized. To this end, we also consider computational and non-computational work beyond the previously limited, technical scope.

3.1 AI Playtesting With Deep Reinforcement Learning and Monte Carlo Tree Search

Many modern automated playtesting approaches exploit the power of deep neural networks [20], Monte Carlo tree search [9], or a combination of these two methods. These technical approaches have emerged as dominant, as they can deal with high-dimensional sensors or controls, and generalise better across many different kinds of games. Progress in simulation-based player modelling
cannot be considered separately from advancements in AI game-playing agents, and we hence also relate to seminal work in the latter domain where it has strongly influenced the earlier.

Coming from a games industry perspective, Gudmundsson et al. [22] have exploited demonstrations from human players for the supervised training of a deep learning agent to predict level difficulty in match-3 games. A deep neural network is given an observation of the game state (e.g., an array of image pixel colors) as input and trained to predict the action a human player would take in the given state. The agent’s game-playing performance is used as operationalization of level difficulty. For the purpose of automatic game balancing, Pfau et al. [42] similarly employ a deep neural networks to learn players’ actions in a role-playing game. They analyze the balance of different character types by letting agents controlled by these neural networks play against each other. The drawback of such supervised deep learning methods for player modelling is that they require large quantities of data from real players, which might not be feasible to obtain for smaller companies or for a game that is not yet released.

A less data-hungry alternative is deep reinforcement learning (DRL, Section 2.2), where an AI agent learns to maximize reward, e.g. game score, by drawing on its interaction experience. This makes DRL a good fit for user modeling framed as computational rationality, i.e., utility maximization limited by the brain’s information-processing capabilities [11, 19, 27, 31]. Interest in DRL for playtesting has been stimulated by Mnih et al.’s [35] seminal work on general game-playing agents, showing that a DRL agent can reach human-level performance in dozens of different games without game-specific customizations. Bergdahl et al. [5] applied a DRL agent for player modelling to measure level difficulty, and to find bugs due to which agents get stuck or manage to reach locations in unanticipated ways. Shin et al. [53] reproduce human play more time-efficiently by training an actor-critic agent [34] based on predefined strategies. Despite such advances, training a game-playing agent with DRL remains expensive: reaching human-level performance in even relatively simple games can take several days of computing time.

An alternative to facilitate automated playtesting for player modelling is to utilize a forward planning method such as MCTS (Section 2.3), simulating action outcomes to estimate the optimal action to perform. Several studies [23, 25] have engineered agents with diverse playing styles by enhancing the underlying MCTS controller with different objective functions. Holmgard et al. [25] have showed how each playing style results in different behavior and interaction with game objects in various dungeon maps. Such agents could be employed in various player modelling tasks, e.g. to evaluate which parts of a level design a human player would spend most time on. Similarly, Guerrero-Romero et al. [23] employ a group of such agents to improve the design of games in the General Video Game AI (GVGAI) framework [41] by analysing the visual gameplay and comparing the expected target behaviour of the individuals. Poromaa [43] has employed MCTS on a match-3 game to predict average human player pass rates as a measure of level difficulty. Keehl and Smith [28] have proposed to employ MCTS to assess human playing style and game design options in Unity games. Moreover, they propose to compare differences in AI player scores to identify game design bugs. As a last example, Ariyurek et al. [3] have investigated the effect of MCTS modifications under two computational budgets on the human-likeness and the bug-finding ability of AI agents. Although MCTS, in contrast to DRL, does not need expensive training, the high runtime costs restrict its application in extensive game playtesting.

It is possible to negotiate a trade-off between training and runtime cost by combining MCTS and DRL. Presently, such combinations represent the state of the art in playing highly challenging games such as Go [54], using learned policy and value models for traversing the tree and simulating rollouts. However, such a combination has not yet been used to drive AI agents for player behavior and experience prediction. This observation is captured in research question (RQ1), leading to the first extension to existing work contributed in this paper.
3.2 Operationalizing Engagement and Difficulty

Our focus is on combining DRL and MCTS to predict pass and churn rates as measures of game difficulty and engagement, and to model the relationship between these two game metrics. This relationship has been discussed primarily from a game design and games user research point of view. Still, it remains difficult to pin down as engagement is considered one of the most complex player experiences [8]. Based on a review of user engagement research within and beyond video games, O’Brien and Toms define it as “a quality of user experience characterized by attributes of challenge, positive affect, endurability, aesthetic and sensory appeal, attention, feedback, variety/novelty, interactivity, and perceived user control” [38], and hence identify challenge as one key influence on user engagement. Game designers often highlight the importance of carefully matching game challenge and player skill [50, 58] to achieve a state of flow [36], which is in turn assumed to be evoked by, and to perpetuate, high levels of engagement [12]. Achieving such an engaging challenge-skill trade-off is non-trivial; Lomas et al. [33] for instance have studied an educational game and found that higher engagement was associated with lower game difficulty. An additional link between difficulty and engagement is postulated by self-determination theory (SDT), which explains human intrinsic motivation through the satisfaction of basic psychological needs. One of these is the feeling of competence, which is affected by the challenges experienced in performing a certain task [48]. In recent years, SDT has become increasingly popular in game research [49, 60].

We account for the complexity of engagement as player experience [8] by operationalizing it in terms of player churn, which can be triggered by any of the underlying factors [38]. Existing computational work has considered the churn probability of an individual player [6, 47], or the average churn rate measured over a population of players and a part of the game such as a single game level [45, 46]. Predicting the churn probability of a particular player allows game developers to offer personalized incentives to continue playing. Predicting the average churn rate of a game level in contrast can be used, amongst others, to detect and correct less engaging levels before they are released to the players.

Difficulty has been operationalized based on features extracted from static game content such as gap size and the player’s maximum jump length in a platformer game [55], or the board layout and pieces in puzzle games [61]. While these static measures are fast to compute, they do not generalize well beyond at most narrow game genres. Moreover, they are insensitive to difficulty arising from the interaction of the player and game facets such as the level design and mechanics during play. As solution, difficulty has also been operationalized by extracting features from simulated gameplay, such as the times the player dies [62] or the achieved score [26, 37]. While the earlier feature does not generalize well, a game’s score, if provided at all, can be stronger affected by certain types of challenge than others, e.g. by performative, cognitive [14] and decision-making [16] challenge, and less by e.g. emotional challenge [7, 13]. Crucially, while we similarly choose to extract features from simulated gameplay, operationalizing difficulty as pass rate, as similarly proposed by Poromaa [43], is sensitive to all these challenge types and agnostic with respect to the game having a score. It thus has a higher potential to be applicable across many games.

Kristensen et al. [30] predict difficulty in a match-3 game based on metrics extracted from static game content such as gap size and the player’s maximum jump length in a platformer game [55], or the board layout and pieces in puzzle games [61]. While these static measures are fast to compute, they do not generalize well beyond at most narrow game genres. Moreover, they are insensitive to difficulty arising from the interaction of the player and game facets such as the level design and mechanics during play. As solution, difficulty has also been operationalized by extracting features from simulated gameplay, such as the times the player dies [62] or the achieved score [26, 37]. While the earlier feature does not generalize well, a game’s score, if provided at all, can be stronger affected by certain types of challenge than others, e.g. by performative, cognitive [14] and decision-making [16] challenge, and less by e.g. emotional challenge [7, 13]. Crucially, while we similarly choose to extract features from simulated gameplay, operationalizing difficulty as pass rate, as similarly proposed by Poromaa [43], is sensitive to all these challenge types and agnostic with respect to the game having a score. It thus has a higher potential to be applicable across many games.
distribution, thus moving it closer to the human ground truth, by using the best, i.e., shorter, runs only. In their specific study, they found that the number of moves a DRL agent needs for completing a level has the highest correlation with human player pass rates if one uses the maximum found among the top 5% best runs. As part of the second extension contributed in this paper, we test their hypothesis by modifying the AI gameplay features used in our previous approach [46] (see RQ2).

4 ORIGINAL METHOD

Previously [46], we trained a DRL agent using the Proximal Policy Optimization (PPO, Section 2.2) algorithm [51] on each level of Angry Birds Dream Blast [18], which is a commercial, physics-based, non-deterministic, match-3, free-to-play mobile game. Below, we briefly summarize this approach to provide the required background for understanding our extensions.

4.1 AI Problem Formulation

To facilitate the DRL (Section 2.2) training, we [46] have adopted the popular game AI practice of formulating a game as a Markov Decision Process (MDP, Section 2.1). The state space resulted from combining a $84 \times 84 \times 3$ RGB pixel array rendering of the game, capturing what the player can observe during play, with a numerical vector, representing game state details such as the goals left to achieve. The action space included $32 \times 32$ possible positions in a discretized game screen to select the closest available match or booster. The rewards included a small penalty for using moves and a larger penalty for losing. Moreover, they comprised bonuses for winning, clearing goals, unlocking locks, and advancing spatially through the level, amongst others. In addition to the extrinsic rewards coming from the environment, we let the agent compute an intrinsic curiosity reward [40] to facilitate efficient exploration.

We have parametrized our DRL with an experience budget of 10240 and a batch size of 1024. For each level, we provided the agent with 4 times the moves available to human players in the same level. Agent experience was collected over a time horizon of 1024. Further hyperparameters are given by a decaying learning rate of 0.0003, a discount factor of 0.99, a generalized advantage estimation parameter of 0.95, a PPO clipping parameter of 0.2, a PPO entropy coefficient of 0.005, and a curiosity reward coefficient of 0.02.

4.2 Data Collection

We [46] have used the trained DRL agent to extract 16 features from simulated gameplay in each level: average and standard deviations of level pass rate, as well as statistics (average, standard deviation, min, max, 5th, 10th, 25th, 50th, 75th percentiles) of both cleared level goals and moves left when passing the level. To fit our predictive models, we moreover obtained a ground truth dataset of average pass and churn rates by 95266 human players, collected from the first 168 levels of Angry Bird Dream Blast. This is a live game which is constantly being updated and improved, and the data corresponds to a particular version that is no longer online. An individual player’s pass rate as an operationalization of difficulty was computed as one divided by the number of required attempts, evaluating to zero if the player never passed a level. The average churn rate as a measure of engagement was defined as the portion of players that stopped playing for at least seven days after trying the level at least once.

4.3 Pass and Churn Rate Prediction

We [46] previously put forward two methods for predicting the human pass and churn rates. Our “baseline model” uses simple linear regression to map the input features to pass and churn rates. Our “extended model” employs a meta-level player population simulation (Figure 2), modelling individual players with the attributes skill, persistence, and the tendency to get bored. For the
latter model, the baseline pass rate prediction was first normalized over the levels to produce a level difficulty estimate. This and each simulated player’s skill attribute were used to determine whether the player passes a level. If a player did not pass, they were assumed to churn with a probability determined by their persistence. Players are moreover assumed to churn randomly based on their boredom tendency. In summary, the population simulation of each level uses a DRL difficulty estimate and current player population as its input, and outputs pass and churn rates for the present level, as well as the remaining player population for the next level (Figure 1). This way, we were able to model how the relation of human pass and churn rates changes over the levels.

We next introduce two extensions to our original method, based on the research questions put forward in the introduction. We investigate the effect of combining DRL with MCTS to predict human players’ pass and churn rates (RQ1). We moreover explore a different feature selection method for the pass and churn rate predictive models, based on analyzing the correlation between AI gameplay statistics and human pass rates (RQ2).
5 PROPOSED METHOD

In this paper, we extend our previous approach [46] by complementing the DRL game play with MCTS, and by improving the features extracted from the AI gameplay data. By utilizing the same high-level population simulation, reward function, and DRL agents trained for each level, we allow for a direct comparison with the previous work.

Two observations in the original work provided initial motivation for our extensions: 1) churn predictions were more accurate if using ground truth human pass rate data instead of simulated predictions, and 2) the simulated pass rate predictions were particularly inaccurate for the most difficult levels. We hypothesize that employing MCTS in addition to DRL can improve the predictions, as the combination has been highly successful in playing very challenging games like Go [54]. We moreover adopt Kristensen et al.’s [30] hypothesis that the difficulty of the hardest levels could be better predicted by the agent’s best case performance rather than average performance. To implement this, we employed multiple attempts per level and only utilized the data from the most successful attempts as gameplay features. We next describe the two resulting extensions in detail.

5.1 Monte-Carlo Tree Search

To use MCTS for pass and churn rate prediction, we first had to identify which variant of the algorithm allows an AI agent to successfully play even the hardest levels of our game. We elaborate on this selection first, followed by a detailed description of our agent implementation.

5.1.1 MCTS Variants. It showed early that MCTS with no modifications requires prohibitively many moves to pass a level and does not produce better results than our original DRL approach [46]. To address the problem, we determined the five hardest game levels based on the average human pass rate, compared the following MCTS variants, and selected the best for further investigation.

- **Vanilla MCTS:** The basic algorithm described in section 2.3. This variant do not use any reward discounting ($\gamma = 1$).
- **DRL MCTS:** In contrast to Vanilla MCTS, this variant does not select random actions for rollout but samples them from a DRL game-playing policy, trained similarly as in our previous work [46]. This is a generally applicable technique inspired by AlphaGo [54]. Decreasing randomness in action selection reduces variance in the backpropagated values, which can result in better node selection and expansion. An alternative view is that MCTS improves on the results of an imperfect DRL agent.
- **Myopic MCTS:** In contrast to Vanilla and DRL MCTS, this variant gives a larger weight to near-future rewards by discounting with $\gamma = 0.9$ in the rollout value computation $\hat{V}(s_{d+1}) = \sum_t \gamma^t r_t$. The value estimate backpropagation to the tree root, $\hat{V}(s_d) \leftarrow r_d + \gamma \hat{V}(s_{d+1})$, also uses discounting.
- **DRL-Myopic MCTS:** This version combines both previously outlined modifications.

5.1.2 MCTS Implementation Details. Our MCTS implementations use the same moves available for human players. At each decision step, the algorithm is iterated 200 times, resulting in a tree depth of approximately 4. We assign a maximum of 10 rollout moves to each simulation step; a simulation is thus stopped either when exceeding 10 moves or when reaching a terminal state. A discount factor $\gamma = 0.9$ was used for the myopic variants.

To reduce the computing cost, we assume that the game dynamics are deterministic, i.e., the same action from the same current state always results in the same next state. This allows reusing the subtree below the selected action when moving to the next game state and restarting the tree search. In reality, the game is highly random, but this is compensated by running the agent multiple times for each level, which is anyway needed to collect more reliable gameplay statistics.
Table 1. Performance of different MCTS variants on the five hardest levels in the original dataset. The Myopic variants discount future rewards, and the DRL variants sample rollout actions from a DRL policy. Overall, our DRL-Myopic MCTS outperforms the other MCTS variants and the DRL agent from our previous work [46].

<table>
<thead>
<tr>
<th>Method</th>
<th>Pass rate</th>
<th>Average moves left ratio</th>
<th>Average cleared goal percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>116 120 130 140 146</td>
<td>116 120 130 140 146</td>
<td>116 120 130 140 146</td>
</tr>
<tr>
<td>Vanilla MCTS</td>
<td>0 0 0.15 0 0</td>
<td>0 0 0.04 0 0</td>
<td>0.24 0 0.67 0.29 0.4</td>
</tr>
<tr>
<td>DRL MCTS</td>
<td>0 0.1 0.55 0 0</td>
<td>0 0.02 0.08 0 0</td>
<td>0.2 0.1 0.82 0.29 0.02</td>
</tr>
<tr>
<td>Myopic MCTS</td>
<td>0.2 0.3 0.05 0.05 0.05</td>
<td>0.02 0.03 0.01 0 0</td>
<td>0.56 0.78 0.72 0.58 0.76</td>
</tr>
<tr>
<td>DRL-Myopic MCTS</td>
<td>0.6 0.4 0.15 0.2 0.45</td>
<td>0.06 0.04 0.01 0.01 0.05</td>
<td>0.82 0.7 0.78 0.59 0.89</td>
</tr>
<tr>
<td>DRL (Roohi et al. [46])</td>
<td>0 0.002 0.006 0 0</td>
<td>0 0.0015 0.0046 0 0</td>
<td>0.13 0.43 0.33 0.31 0.08</td>
</tr>
</tbody>
</table>

5.1.3 Selecting the Best MCTS Variant on Hard Game Levels. To select the best MCTS variant, we ran each candidate 20 times on each level. One run used 16 parallel instances of the game. This process took approximately 10 hours per level on a 16-core 3.4 GHz Intel Xeon CPU.

Table 1 shows the average performance of these four MCTS variants over 20 runs on the five hardest levels of the dataset. DRL-Myopic MCTS performs best. We thus select this variant for comparison against our original DRL-only approach [46], and for testing the effect of the feature extension described in Section 5.2.

5.2 Feature Selection: Correlating AI Gameplay Statistics With Human Data

In addition to implementing the MCTS variants as alternatives to using DRL only, we have also tested and compared various feature selection strategies to extend our original approach [46]. The latter uses a total of 16 input features for the pass rate predictor, computed over multiple AI agent runs per level. However, this amount of features is fairly high, which may lead to the pass rate prediction model overfitting the human pass rate data. In this paper, we focus on those features that are most highly correlated with the ground truth human pass rates. We have implemented and compared the following three feature selection strategies:

- **F16**: The original 16 features from our previous work [46].
- **F3**: A subset of 3 out of the 16 original features with the highest Spearman correlations with human pass rates: AI pass rate ($\rho=0.80$), cleared goals percentage ($\rho=0.60$), and the ratio of moves left to maximum allowed moves ($\rho=0.74$), computed as averages over 1000 runs of the DRL agent per level.
- **F3P**: Similar to F3, but with averages computed from a subset of the best rather than all runs, characterized by a high number of moves left after passing a level.

The last selection strategy, F3P, is inspired by Kristensen et al. [30], who have observed a stronger correlation between human ground truth data and features extracted from best runs (see Section 3). To back this up and to identify the best sample size, we computed the Spearman correlation between the ground truth human pass rate and the average of the DRL agent’s performance over different percentages of the agent’s best runs (Figure 3). The results indicate that human pass rates are predicted best from AI pass rates if the latter are calculated over all AI runs. However, the average moves left and average cleared goals features correlate most with human pass rates if calculated from the best AI runs. This supports Kristensen et al.’s [30] hypothesis for at least some, although not all, investigated gameplay features.

Based on the results in Figure 3, we used the top 15% runs for computing the average moves left, and top 5% for computing the average cleared goals percentage features. Pass rate was averaged over all runs. To compute the averages, we performed 20 MCTS runs and 1000 DRL runs per level. MCTS is much more computationally expensive, and the limited number of runs means that the...
6 RESULTS

We compare the performance of the original [46] DRL model and our DRL-enhanced MCTS extension, denoted as DRL-Myopic MCTS in Section 5.1.1. We also assess the impact of the three different feature selections strategies introduced in Section 5.2, including F16 from the original approach. To enable a fair comparison with previous work, we combining these algorithms and feature selection strategies with the two pass and churn rate predictors from the original work, i.e., the baseline linear regression predictor and extended predictor with player population simulation. Taken together, this yields $2 \times 3 \times 2 = 12$ combinations to evaluate.

Table 2 shows the mean squared errors (MSEs) of pass and churn rate predictions computed using the compared methods and 5-fold cross validation on our original ground truth data. Figure 4 illustrates the same using box plots. The Baseline-DRL-F16 and Extended-DRL-F16 combinations are the original approaches analyzed previously [46]. For each fold, the extended predictor’s errors were averaged over five predictions with different random seeds to mitigate the random variation caused by the predictor’s stochastic optimization of the simulated player population parameters. We find that our Extended-MCTS-F3P configuration produces the overall most accurate predictions, and the proposed F3P features consistently outperform the other features in pass rate predictions.

Figure 5 shows scatter plots of churn and pass rates in all the 168 levels in our dataset. It affords a visual comparison between ground truth human data, the Extended-DRL-F16 configuration that corresponds to the best results in the original work [46], and our best-performing Extended-MCTS-F3P configuration. Both simulated distributions appear roughly similar, but our Extended-MCTS-F3P yields slightly more realistic distribution of the later levels (121-168), with the predicted pass rates not as heavily truncated at the lower end as in Extended-DRL-F16.

7 DISCUSSION

We have proposed and evaluated two extensions to our previous player modeling approach [46], in which difficulty and player engagement are operationalized as pass and churn rate, and predicted
Table 2. Means and standard deviations of pass and churn rate predictions mean squared errors, computed from all 12 compared configurations using 5-fold cross validation. The best results are highlighted using boldface. The Baseline-DRL-16 and Extended-DRL-16 configurations correspond to the original approaches developed and compared in our previous work [46].

<table>
<thead>
<tr>
<th>Predictor type</th>
<th>Agent</th>
<th>Features</th>
<th>Pass rate MSE</th>
<th>Churn rate MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>DRL</td>
<td>F16</td>
<td>$\mu = 0.01890, \sigma = 0.00514$</td>
<td>$\mu = 0.00014, \sigma = 0.00003$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F3</td>
<td>$\mu = 0.02041, \sigma = 0.00453$</td>
<td>$\mu = 0.00012, \sigma = 0.00003$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F3P</td>
<td>$\mu = 0.01742, \sigma = 0.00368$</td>
<td>$\mu = 0.00012, \sigma = 0.00003$</td>
</tr>
<tr>
<td></td>
<td>MCTS</td>
<td>F16</td>
<td>$\mu = 0.02040, \sigma = 0.00370$</td>
<td>$\mu = 0.00013, \sigma = 0.00003$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F3</td>
<td>$\mu = 0.01919, \sigma = 0.00305$</td>
<td>$\mu = 0.00012, \sigma = 0.00003$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F3P</td>
<td>$\mu = 0.01452, \sigma = 0.00207$</td>
<td>$\mu = 0.00012, \sigma = 0.00003$</td>
</tr>
<tr>
<td>Extended</td>
<td>DRL</td>
<td>F16</td>
<td>$\mu = 0.01953, \sigma = 0.00482$</td>
<td>$\mu = 0.00008, \sigma = 0.00002$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F3</td>
<td>$\mu = 0.01984, \sigma = 0.00548$</td>
<td>$\mu = 0.00008, \sigma = 0.00002$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F3P</td>
<td>$\mu = 0.01755, \sigma = 0.00464$</td>
<td>$\mu = 0.00007, \sigma = 0.00003$</td>
</tr>
<tr>
<td></td>
<td>MCTS</td>
<td>F16</td>
<td>$\mu = 0.02048, \sigma = 0.00405$</td>
<td>$\mu = 0.00008, \sigma = 0.00002$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F3</td>
<td>$\mu = 0.01925, \sigma = 0.00369$</td>
<td>$\mu = 0.00008, \sigma = 0.00002$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>F3P</td>
<td>$\mu = 0.01419, \sigma = 0.00216$</td>
<td>$\mu = 0.00007, \sigma = 0.00002$</td>
</tr>
</tbody>
</table>

via features calculated from AI gameplay. We firstly used a game-playing agent based on DRL-enhanced MCTS instead of DRL only. We secondly reduced the set of prediction features based on correlation analysis with human ground truth data. We moreover calculated individual features based on different subsets of AI agent runs.

Overall, both our proposed enhancements consistently improve prediction accuracy over the original approach. The F3P features outperform the F16 and F3 ones irrespective of the agent or predictor, and the MCTS-F3P combinations outperform the corresponding DRL-F3P ones. The improvements are clearest in pass rate prediction: Our Extended-MCTS-F3P configuration improves the best of our previous approaches [46] by approximately one standard deviation, thus overcoming a crucial shortcoming of that work. We remark that the baseline churn rate predictions only slightly benefit from our enhancements, and the extended churn rate predictions are all roughly similar, no matter the choice of AI agent or features. We also replicate our previous results [46] in that the extended predictors with a population simulation outperform the baseline linear regression models.

A clear takeaway of this work is that computing feature averages from only a subset of the most successful game runs can improve AI-based player models. In other words, whenever there is a high variability in AI performance that does not match human players’ ground truth, an **AI agent’s best-case behavior can be a stronger predictor of human play** than the agent’s average behavior. As detailed below, these extensions must be tested in other types of games to evaluate their generalizability.

### 8 LIMITATIONS AND FUTURE WORK

The amount of MCTS runs and the reward discount factor $\gamma$ have a considerable impact on the computing cost of our model. To deliver the best possible predictions at a low cost, it would be valuable to investigate in future work how changes to these parameters affect the accuracy of pass and churn rate predictions.

While the evaluation of our approach on a commercially successful game demonstrates it applicability beyond mere research prototypes, it did not afford a comparison with existing AI playtesting approaches. We consider such a comparison of the prediction accuracy and computational costs on the same game an important next milestone.
Predicting Game Engagement and Difficulty Using AI Players

Fig. 4. Boxplots of the cross-validation Mean Squared Errors for all 12 compared configurations. The configurations corresponding to our original approach [46] are shown in blue. Our proposed F3P features consistently outperform the other features in pass rate predictions, and our Extended-MCTS-F3P configuration produces the overall most accurate predictions.

Fig. 5. Human and predicted pass and churn rates over four groups of 168 consecutive game levels, comparing our original [46] best configuration (Extended-DRL-F16) to our best-performing extension (Extended-MCTS-F3P) here. The colors and shapes correspond to level numbers. Our predicted pass rates are slightly less truncated at the lower end.

Our focus on a single game and data from only 168 levels moreover limits our findings with respect to the generalizability of our approach. For future work, it will hence be necessary to evaluate our models on larger datasets and several games from different genres.
Accurate player experience and behavior models can be used to assist game designers, or facilitate automated parameter tuning during design and runtime. For the future, we plan to test our method in the development process of a commercial game. More specifically, we aim to employ our algorithm to detect new game levels with inappropriate difficulty, and investigate how changes to those levels would reduce player churn rate before presenting the levels to human playtesters or end users.

9 CONCLUSION

Our work advances the state-of-the-art in automated playtesting for player modeling, i.e., the prediction of human player behavior and experience using AI game-playing agents. Specifically, we have proposed and evaluated two enhancements to our recent approach [46] for predicting human player pass and churn rates as operationalizations of difficulty and engagement on 168 levels of the highly successful, free-to-play match-3 mobile game Angry Birds Dream Blast [18]. The required ground truth data can be obtained without refraining to obtrusive methods like questionnaires.

By comparing against our original DRL-based approach, we demonstrate that combining MCTS and DRL can improve the prediction accuracy, while requiring much less computation in solving hard levels than MCTS alone. Although combinations of MCTS and DRL have been utilized in many state-of-the-art game-playing systems, our work is the first to measure its benefits in predicting ground-truth human player data. Additionally, we propose an enhanced feature selection strategy that consistently outperforms the features used in our previous work.

Moreover, in evaluating this feature selection strategy, we replicate Kristensen et al.’s [30] findings that an AI agent’s best-case performance can be a stronger predictor of human player data than the agent’s average performance. This shows in our extended predictor with stochastic population simulation, but has also been verified by computing the correlations between the human ground truth data and AI gameplay variables. This finding is valuable, as it should be directly applicable in other AI playtesting systems by running an agent multiple times and computing predictor features from a subset of best runs. We hope that our findings and improvements inspire future research, and foster the application of automated player modelling approaches in games industry.

10 ACKNOWLEDGMENTS

We thank the anonymous reviewers for their thorough and constructive comments. CG is funded by the Academy of Finland Flagship programme "Finnish Center for Artificial Intelligence" (FCAI). AR, HH and JT are employed by Rovio Entertainment.

REFERENCES


Predicting Game Engagement and Difficulty Using AI Players


Marc Van Kreveld, Maarten Löffler, and Paul Mutser. 2015. Automated puzzle difficulty estimation. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, Tainan, Taiwan, 415–422. https://doi.org/10.1109/CIG.2015.7317913


Received February 2021; revised June 2021; accepted July 2021