Toward Automated Game Balance: A Systematic Engineering Design Approach*

Daniel A. DeLaurentis
School of Aeronautics and Astronautics
Purdue University
West Lafayette, USA
ddelaure@purdue.edu

Jitesh H. Panchal
School of Mechanical Engineering
Purdue University
West Lafayette, USA
panchal@purdue.edu

Ali K. Raz
Systems Engineering and Operations Research
George Mason University
Fairfax, USA
araz@gmu.edu

Prajwal Balasubramani
School of Aeronautics and Astronautics
Purdue University
West Lafayette, USA
balasub9@purdue.edu

Apoorv Maheshwari
School of Aeronautics and Astronautics
Purdue University
West Lafayette, USA
amaheshw@purdue.edu

Adam Dachowicz
School of Mechanical Engineering
Purdue University
West Lafayette, USA
adachowi@purdue.edu

Kshitij Mall
School of Aeronautics and Astronautics
Purdue University
West Lafayette, USA
mall@purdue.edu

***Abstract*—This study proposes a framework to automatically assess and bring balance in real-time strategy (RTS) games. A three-layered framework comprising intelligent bots, deep machine learning, explainable artificial intelligence (XAI), uncertainty quantification (UQ), and optimal learning is presented. For preliminary analysis, we conducted a study using the microRTS game built specifically for advancing AI research. Data is generated through self play games between the intelligent bots, and game balance is measured through the predicted probability of each player winning a game. To demonstrate game re-balancing using this approach, a sample unbalanced game is shown along with proposed perturbations on important features identified using a popular XAI technique called SHapley Additive exPlanations (SHAP). Results indicate this framework enables efficient identification of game parameters causing imbalance and iterates over game parameters to restore balance. The three-layered framework is designed to be generic and applicable to more complicated RTS games, such as StarCraft II.**

***Index Terms*—Game balance, Game breaking, microRTS, Convolutional Neural Networks, SHAP, Explainable Artificial Intelligence, Real-time strategy games**

I. Introduction

Knowledge transfer from the gaming industry to engineering and design has gained attention in recent years as groundbreaking developments have been made with the advent of superhuman AI players such as AlphaStar [1], Pluribus [2], etc., that have beaten the best human players in the world. DARPA recently launched the *Gamebreaker* [3] program, looking specifically at real-time strategy (RTS) games and how methodologies from game design can be brought over to engineering design applications. This program has focused on the notion of *game balance* and how it can be affected through perturbations to game design.

One of the most important aspects of game design is to *balance* the game. This has been an active area of research in the gaming community for quite some time [4]–[7]. While a game designer would like to detect and keep the balance in a game to maintain the player's interest in the game, the implication of game balance is a bit abstract to directly apply to real-world applications. A game is *balanced* if all players have an equal chance of winning the game depending just upon the game environment and without any knowledge of the player skill levels. The game environment would include game rules and available actions to each player. Skill, on the other hand, accounts for a player's ability to select and then execute a particular sequence of actions out of all possible options. There are several aspects of a game that can be balanced; for this paper we chose win/loss for the players as a measure of balance.

For a simulation of a complex real-world system of interest to an engineer or mission designer (simulation is akin to a game, albeit less interesting), it is important to identify any inherent biases present in the simulation and strategies to effectively shift the balance of the simulation in the desired direction. Similar to the gaming industry, subject-matter experts (or skilled *players*) run the simulation for various conditions and are tasked with identifying such inherent biases

*This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA) under the *Gamebreaker* program with contract HR00112090069. The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

978-1-6654-3886-5/21/$31.00

(or *meta* strategies) that can lead to an unbalanced game. With increasing complexity of the simulations, this design space exploration requires extensive time and effort.

We propose leveraging this abstraction of engineering simulations as games to make use of innovative game-based research tools and relevant quality metrics, such as game balance. In this paper, we focus on the research question: *How do we automate game balancing while effectively exploring a large and complex design space?* We propose and demonstrate a framework that can automate the process of balancing a game by incorporating intelligent bots, advanced machine learning (ML) techniques, explainable artificial intelligence (XAI), uncertainty quantification (UQ), and optimal learning, which is the main contribution of this paper.

Game balance exploration and automating game balance is an active area of research, and is concerned with mitigating issues revolving around game design-space exploration [8]–[11]. A tool called the *Sentient Sketchbook* was developed (see [12]) to allow a game designer create game levels through a computer-aided sketching interface. This tool automates map evaluations, visualizes them on-screen and proposes alternative designs, thereby supporting the game designer immensely. This tool, based on genetic algorithms, is built for designing game maps catering to different game balance requirements. However, the Sentient Sketchbook suggests changes to the maps, but not to the properties of the player agents. The framework devised in this study includes modifying properties of both the map and players for achieving game balance. XAI has also been recently studied in the game industry to explain phenomena like game cheating [13], for visualization of games played by AI players [14], and for mixed-initiative co-creation [15]. SHapley Additive exPlanations (SHAP) [16] is a popular XAI tool that has started finding application in the game industry [13].

For this study, we chose microRTS [17] as our gaming platform since it is primarily designed to perform AI research. microRTS functions as a testbed to conceptualize and test learning algorithms. The simplicity of microRTS enables quick testing of theoretical ideas as compared to other high fidelity games like StarCraft II. Playing microRTS under uncertainty and with stochastic bots have been of recent interest to some game designers [18], [19]. Researchers have used microRTS successfully for testing their hypotheses and ideas. The topics, are not only limited to AI algorithmic development, ranges from hierarchical task planning in networks to learning action probability models [20], [21]. Further, researchers have used Monte Carlo Tree Searches (MCTS) and Reinforcement Learning (RL) methods to evaluate game states [22], [23].

The three-layered framework is presented in the next section. An example of a highly unbalanced game is then shown in the results section, where certain perturbations (*patches*) are introduced to bring back balance in the game. The final sections of this study discuss future work and conclusions.

II. Methodology

Our technical approach for detecting, maintaining, and bringing back balance is illustrated in Fig. 1. This systematic, three-layered, artificial intelligence (AI)-infused framework, which we call *Learn to Gamebreak (L2G)*, comprises the *Game Balance Layer*, *Tournament Layer*, and *Player Layer*. In this framework, each layer has a distinct role that separates the primary tasks of deriving inferences regarding game balance, managing the extensive design space, and identifying players to compete in multiple games.

A. Player Layer

This layer consists of the different player bots that compete against each other and generate the required game plays to train our model. *microRTS* [17], a simple implementation of an RTS game, was built for testing and conceptualizing AI for gameplay through experiments. This implementation has fostered a large research community that provide open-source algorithms for bots. The player bots for RTS games, like microRTS, range widely in their skill levels and strategy space, from simple rule-based bots to sophisticated Reinforcement Learning (RL) bots. In addition, each player, depending upon its type, can be subject to a varying set of resources and unit attributes (e.g., availability of weapons with varying performance, health, survivability, etc.). In the L2G framework, all players are AI-bots that execute a programmed strategy (stochastic, deterministic, or learned) and are subject to player parametric attributes that dictate their abilities. The results in this paper are from the games played using the Monte Carlo AI bot that explores future action spaces and game states, and then takes the best action.

B. Tournament Layer

This layer conducts multiple parallel games between the player AI agents and manages the individual game environment by varying game design features and player attributes. The features of game design and player attributes are based on an experimental design that provides a structured approach to manage an extensive design space. Initial sets of games (tournaments) are conducted in parallel to obtain initial data sets using experimental design techniques, like Latin Hypercube sampling. The tournament, along with information about the uncertainty in both model outcomes (supervised learning for prediction and XAI for post-hoc feature importance analysis), can then be used to sequentially determine the next set of tournaments to execute. This approach helps manage the extensive design space and generates targeted data sets that maximize learning of the game balance and identification of feature sets. In addition to generating game data sets for learning game balance, another primary role of the Tournament Layer is to validate the game balance predictions derived from the AI models in the Game Balance Layer and the response of these models to any perturbations and unforeseen inputs.

The game parameters varied in this study are specified in Table I, and include i) player unit health, 2) player unit damage, 3) player initial base location, and 4) placement of

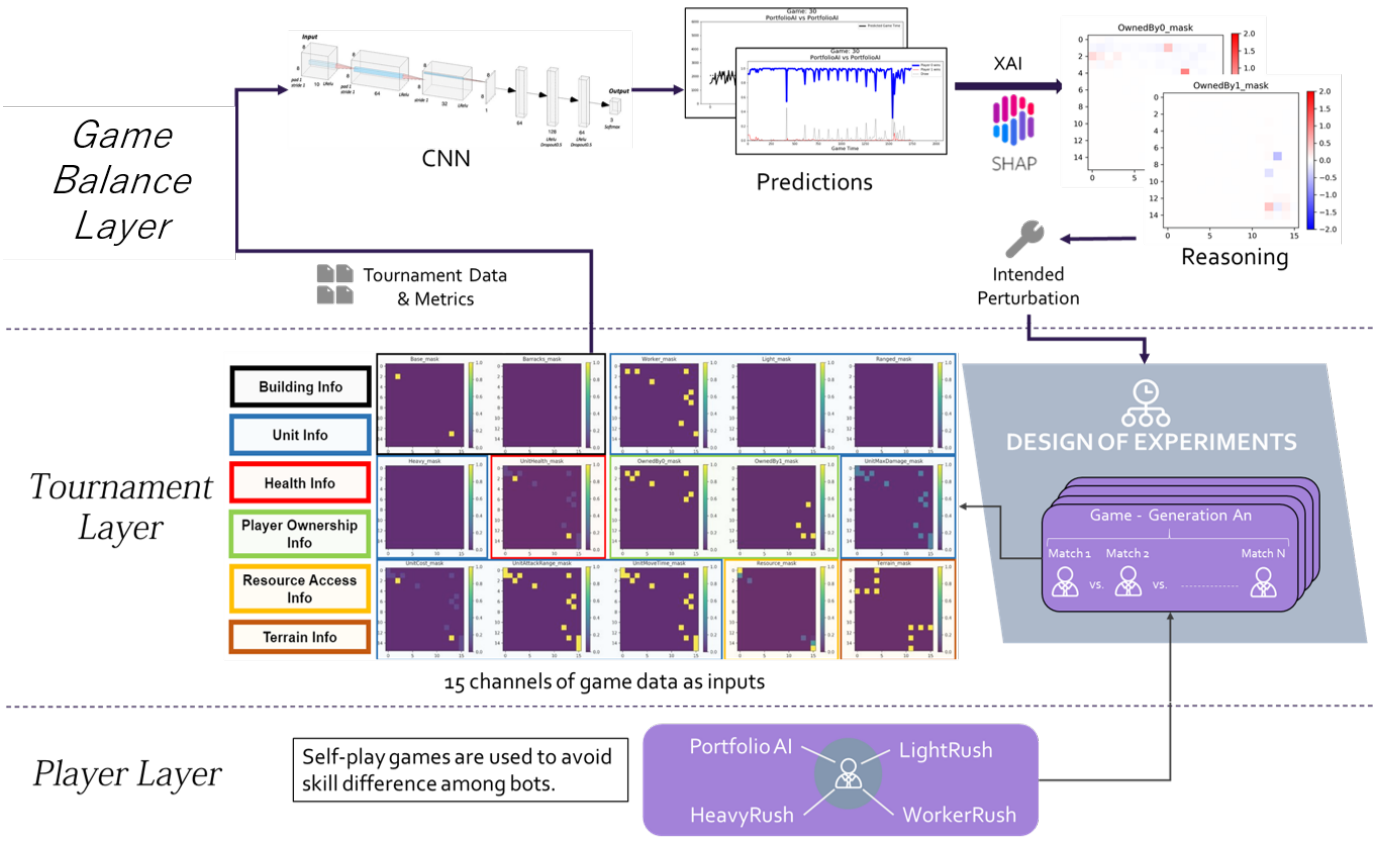

Fig. 1. Proposed three-layered L2G framework to assess, maintain or restore balance in a game.

TABLE I
DOE WITH MODIFIED GAME PARAMETERS AND RANGES

Attribute	Values Range
Players	Monte Carlo AI
Base Starting Location	5 × 5 starting area
Worker Unit Health	1-4 (int)
Worker Unit Damage Output	1-3 (int)
Light and Heavy Unit Resource Cost	1-4 (int)
Base Production Time	70-130 (Game Frames)
Terrain Center, k_{center}	0-1 (float)
Terrain Density, k_{density}	0-1 (float)

terrain features. Note that we only consider games played between identical bots in this analysis to control for differences in player *skill*; we refer to these games as *self play games*.

Data is gathered at each *frame* of a gameplay, so that a game designer can examine the inherent *balance* of game states at the start of the game, or during the early-, mid-, or late-game of example matches. Both spatial and non-spatial data are gathered. Spatial features like relative unit position, health, and available resources for each player are tracked as shown in the *Tournament Layer* of Fig. 1 (center). This data is captured in 15 image channels tracking unit locations and the corresponding feature value. 15 additional non-spatial features, capturing information regarding total unit counts, total health of all assets for each player, and damage rates for each player (specified in Table II) are also tracked. These data are then used as input to the game balance prediction model that makes up the Game Balance Layer, discussed next.

TABLE II
NON-SPATIAL FEATURES EXTRACTED FROM EACH GAME FRAME

Feature	Count
Minimum Distance Between Units and Bases	4
Total Health	2
Total Units	2
Unit Location Centroids	4
Damage Rate (rd)	2
Terrain Count	1

C. Game Balance Layer

This layer provides the overall solution for the balancing task by collecting, processing, and fusing information to determine the map between game design features and the probability of win for each player. The two primary constituents of this layer are (i) supervised AI techniques that direct the learning of the game balance based on available game data, and (ii) post-hoc analysis XAI techniques to examine the trained supervised learning model and identify feature

sets within the input space (i.e., game design, environment characteristics, player attributes, etc.) that impact the winning or losing outcome of a player. By identifying these feature sets, the Game-balance Layer can yield the combination of factors within the input space that will maintain and/or shift the game balance to one player's advantage/disadvantage. This learning of the game balance and identification of significant feature sets is an iterative process subject to the available game data provided by the Tournament Layer. Figure 2 summarizes the interactions between the Tournament Layer and Game-balance Layer, and also within the Game-balance Layer.

In this study, we have implemented a neural network model as the supervised learning technique and paired these with SHAP [16] to implement the post-hoc analysis methods for XAI. The network architecture consists of two input heads: one takes in image-like data of shape (16, 16, 15) representing the fifteen 16x16-pixel image channels extracted at each game frame, while the other takes in the fifteen scalar features capturing the non-image data. The input image data is propagated through three convolutional layers with leaky ReLU activations. This image data branch architecture was motivated by the architecture successfully employed by [22], who implement a CNN model that takes microRTS-generated image data as input. The non-image input data is propagated through three dense layers with ReLU activations. The outputs of both these branches are then flattened and concatenated, and then passed through an additional two dense layers with leaky ReLU activations until finally passed to the model output layer. The output layer is fully connected with three units representing the three possible outcomes: Player 0 wins, Player 1 wins, or a draw occurs. To obtain outputs that may be interpreted as probabilities of players winning or a draw occurring, a softmax function is also applied to the output layer.

Categorical crossentropy loss was used to train the model, and the model was trained over 62 epochs with decreasing learning rate LR given by

$$\mathrm{LR}(E) = 0.001 \exp\left(1 - 0.1\ E\right), \tag{1}$$

where E is the epoch number.

We have also made use of Monte Carlo Dropout Networks (MCDNs) [24] for model output uncertainty estimation to supplement this post-hoc analysis. These dropout networks are implemented by adding dropout layers with dropout probability $p = 0.5$ in front of all trainable, weighted layers in the network, and keeping these dropouts even at predict time. By evaluating the model $k = 20$ times for each prediction, each time with a different seed generating the dropout parameters, we generate 20 samples of the output and SHAP results, from which we can estimate uncertainty. For details of the network architecture and MCDN used in this study, please refer [25].

Finally, we make use of SHAP analysis to provide post-hoc explainability of model outputs. Briefly, SHAP is a model explanation technique that provides *local, additive explanations* of a model's output with respect to the input features. Given a model, training data set, and a point in the model's input space, for each input feature a SHAP value may be computed that captures its additive contribution to the corresponding model output under the Shapley requirements of local accuracy, missingness, and consistency (for full details, please see [16]). For this study, we employed the SHAP toolbox developed by Lundberg [26] for all SHAP analysis. To estimate SHAP values, we use an implementation of the *expected gradients* method, described in detail by Erion et al. [27].

III. Results

In this paper, our goal is to help a game designer in developing a game that is maximally balanced for both players. Here, as a motivating example, we will consider a game designer responsible for balancing an RTS game that has applied our framework to study causes of balance or imbalance. For the bots (players) specified in the Player Layer, a set of 25,000 example games have been run by the Tournament Layer, varying the game parameters under the game designer's control. Finally, data from these games is passed to the Game Balance Layer, from which balance predictions, along with uncertainty estimates, can be made for example games.

A. Game Balance Prediction

Given the trained model, a game designer can take an example game replay and run each frame of the game through the model. This produces game balance predictions, along with uncertainty and SHAP results, at each time step. Example game balance prediction outputs for balanced and imbalanced game plays are provided in Fig. 3. Note the model is more uncertain at the initial frames of the game, and becomes more confident in the result as the game progresses. This provides some confidence that the model is behaving correctly, as we anticipate games are *harder* to call early in the game.

A game designer can use these results to draw multiple conclusions. Looking at the first frame of a game (game time = 0), the designer can estimate how balanced the starting conditions are. For instance, for the game that generated Fig. 3(A), the game appears initially quite balanced; 3(B) indicates a game that heavily favors Player 1. Alternatively, a game designer may look at interesting points in a game, such as around game time 3000 in the balanced game, to determine what player choices or game events drive swings in the balance prediction, or around game time 300 in the imbalanced game to determine what game states seem to collapse the prediction to near-certain that Player 1 will win. These studies can then be paired with SHAP results to help determine the drivers of (im)balance more concretely, as discussed next.

B. Utility of SHAP and Uncertainty Quantification

For the rest of this section, we will consider the imbalanced game scenario used to generate Fig. 3(B). In this game, Player 1 had increased health (4 vs. 2), while Player 0 had increased unit maximum damage (2 vs. 1), and Player 0's base was located closer to the center of the map. No terrain features were spawned for this example game scenario. We have already observed that the model predicts a heavily imbalanced

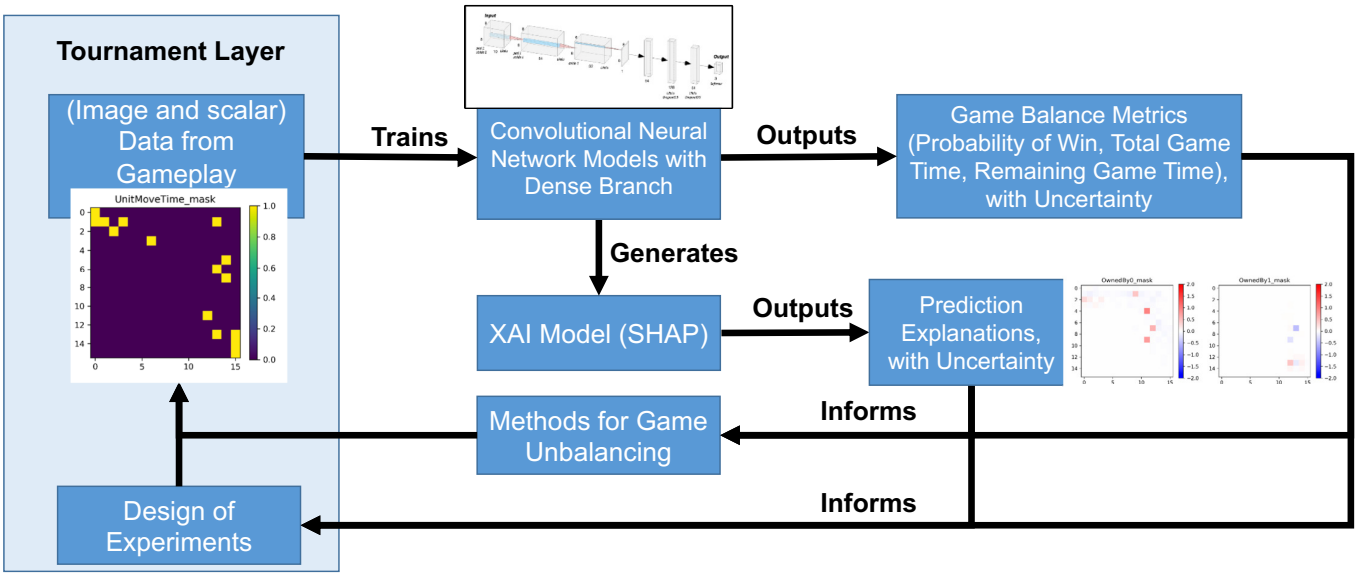

Fig. 2. Interactions between and operations in the tournament and game balance layers.

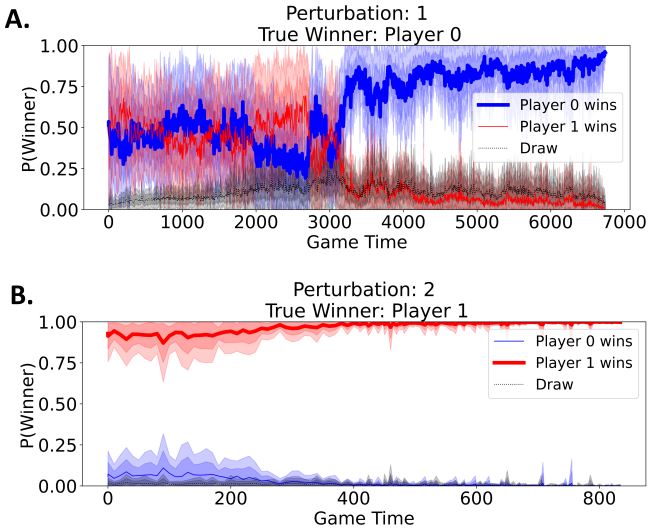

Fig. 3. Player win and draw predictions for each frame of two example games, one balanced game (A) and another imbalanced (B).

game in favor of Player 1. In practice, for 60 trial games run using this game scenario, Player 1 won all but 2 of the trials (Player 0 won the remaining 2 games, and no draws occurred). As we are interested in balancing this game, let us consider SHAP results that relate input spatial and non-spatial features to the prediction that the underdog Player 0 will win.

SHAP results for the spatial features extracted at the first game frame, corresponding to the prediction that Player 0 will win, are plotted in Fig. 4, while results for the non-spatial features for the same prediction are plotted in Fig. 5. Note that results in these figures correspond to the contribution of the feature to the prediction that player 0 will win, given inputs from the first game frame. Each box in Fig. 4 corresponds to an input channel. For both SHAP plots, features (represented as pixels in Fig. 4 and as bars in Fig. 5) contribute positively to the prediction that Player 0 will win if they are colored red, and negatively if they are colored blue, with the intensity of the color or height of the bar, respectively, corresponding to the magnitude of the effect. From these plots, it can be seen that there is a strong negative SHAP value corresponding to the location of Player 1's base, and smaller but still negative values corresponding to the health of Player 1's units and damage output of Player 1. Especially in Fig. 5, it can be seen that the current health of Player 0's units contributes very strongly against Player 0's win prediction.

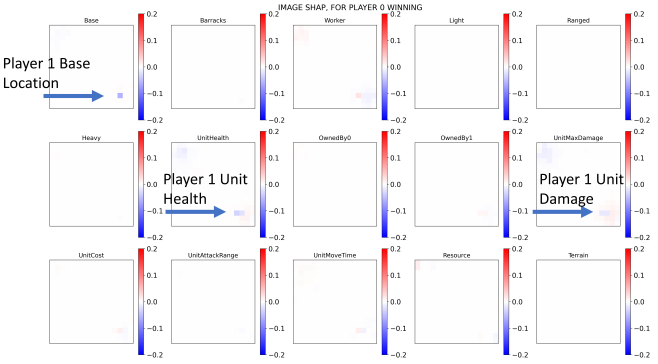

Fig. 4. SHAP results for spatial features for the imbalanced game.

A game designer may look at these results and reasonably determine that, assuming the model is correct, key factors unbalancing the game in favor of Player 1 include the location of Player 1's base, the unit health difference between the

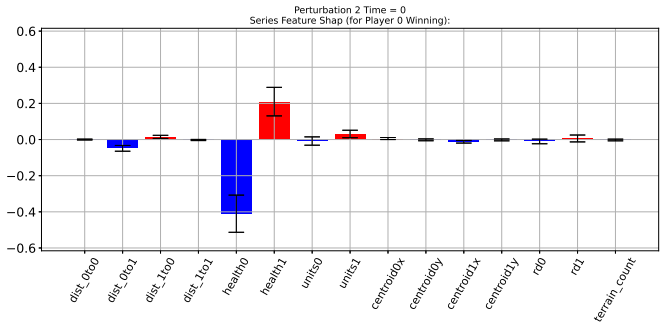

Fig. 5. SHAP results for non-spatial features for the imbalanced game.

players, and perhaps the relative damage rate of each player. We can then vary these factors first to see if we can indeed make the game more balanced. If we are successful, we have efficiently increased game balance by identifying the most important game parameters, while if we are unsuccessful, we have efficiently identified a region of the game design space where our model is incorrect and needs additional training data.

C. Imbalanced to Balanced Games

Given these results, the game designer decides to run three *perturbation* experiments. These experiments are: (a) vary base position to make positions symmetric, (b) increase Player 0 units' health, and (c) increase Player 0 maximum unit damage.

Consider Fig. 6, which reports model predictions for an example game play of the original unbalanced game (top) and predictions for example game plays for each of the varied game parameter settings (bottom). For the original unbalanced game, at time 0 the model predicts Player 0 and Player 1 have 6% and 93% chance of winning, respectively. Thus, the observed win-loss ratio for Player 0 was 3% / 97%. For experiment (a), we observe the model predicts a slightly more balanced game (predicted win ratios of 18% / 79%), and we also observe a slightly more balanced win ratio over 60 trial games (observed win ratios of 15% / 85%). Similarly for experiment (b), we observe a predicted win ratios 10% / 88%, and observed win ratios of 20% / 80%. Finally, for experiment (c) we observe a predicted win probability identical to the original case, while we do see a more balanced observed win ratio.

There are several interesting conclusions to be drawn from this analysis. First, while none of the games are completely balanced, we do see an increase in both predicted and observed game balance as measured by win ratio over the first two experiments. To bring the game into further balance, this same analysis could be run over these perturbed game settings to further identify causes of imbalance and propose additional perturbation experiments.

Second, note the disagreement between the observed win ratios for experiment (c) and the predicted. This perhaps indicates the model has not been exposed to the necessary data in this region of the game design space to make correct predictions. This is especially worrying, since the model makes these predictions with very low uncertainty relative to experiments (a) and (b). The designer may then choose to run additional experiments with game parameters similar to those in experiment (c) to get more training data to improve the model. Given enough time to generate data, it may also be worth gathering additional data similar to experiments (a) and (b) to bring the predictions for those games at time 0 closer to empirically observed win rates.

Third, note that for experiment (a), where the model prediction is the most balanced, the uncertainty at early stages of the game is higher than that of the original games or the other experiments. This agrees with the observation from Fig. 3 that the more balanced game has higher uncertainty earlier in the game. This suggests uncertainty in these first set of game frames may indeed be a secondary measure of game balance: more uncertain predictions indicate the game winner is harder to determine and thus the game is more likely to be balanced.

IV. Future Work

Apart from self play games, data from games played between different bots, or *cross play games*, should be incorporated to more fully explore the space of likely game plays. A novel aspect of the Tournament Layer in the L2G framework is the potential to use the theory of optimal learning [28] to conduct games between agents for efficient learning of game balance. Future study should also involve more advanced optimal learning techniques to fully automate iterations over game parameters, as well as incorporate other metrics of game balance, such as game time, resource or health attrition, or winner prediction model uncertainty. More complex game environments such as StarCraft II will be used to test and improve the generic three-layered framework proposed in this study.

V. Conclusions

In this paper, we presented a unique three-layered framework to assess and manage balance in real-time strategy (RTS) games. The L2G framework involves intelligent bots, self play tournaments, machine learning (ML) model, explainable AI (XAI), and uncertainty quantification (UQ). We conclude that L2G framework enables prediction of game balance. It also identifies the key features that impact the game balance using SHapley Additive exPlanations (SHAP). This study utilizes an example self play game in microRTS between Monte Carlo AI bots to demonstrate the utility, effectiveness, and systematic design support provided by this framework. The game features, when perturbed properly, can restore balance in the game. Analysis of uncertainty through Monte Carlo Dropout Networks (MCDNs) in the game balance predictions and corresponding SHAP results drive the data requirements needed to improve the ML models. SHAP in conjunction with UQ, as used in this study, can support further optimal learning tools to evolve the tournament layer of the L2G framework to

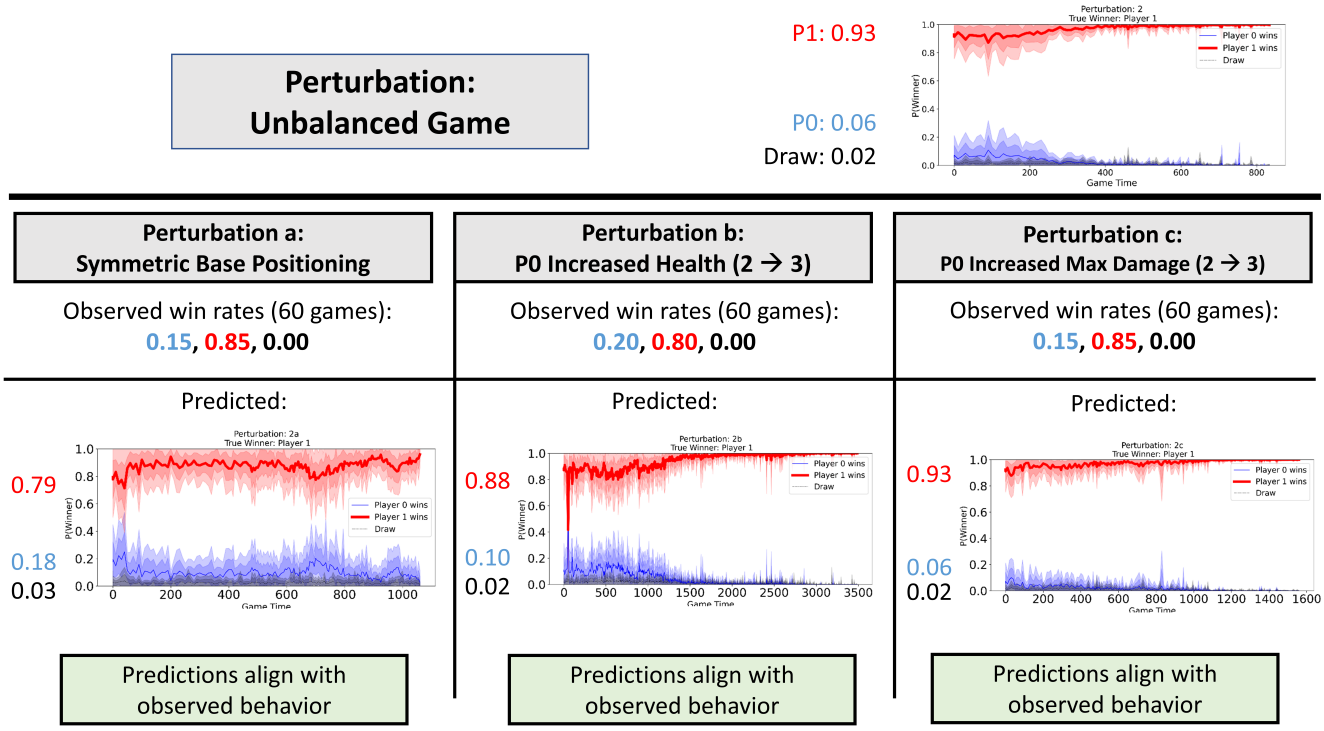

Fig. 6. An imbalanced game where player 1 had increased health, player 0 had increased unit damage, and player 0's base was located closer to the center of the map. The model correctly predicts the game is imbalanced in favor of player 1, who won 97% of 60 trial games. Three perturbations to the game parameters are proposed. For each, features were selected to be changed based on insights from SHAP. Changes (a) and (b) improve the balance as expected, but do not bring the game into complete balance.

more efficiently assess and automatically restore game balance for a given RTS scenario.

Acknowledgment

This paper documents the work performed by a research team at Purdue University for the DARPA *Gamebreaker* program under contract HR00112090069.

References

[1] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[2] N. Brown and T. Sandholm, "Superhuman ai for multiplayer poker," *Science*, vol. 365, no. 6456, pp. 885–890, 2019.

[3] "Gamebreaker AI Effort Gets Under Way." [Online]. Available: https://www.darpa.mil/news-events/2020-05-13

[4] T. J. Tijs, D. Brokken, and W. A. IJsselsteijn, "Dynamic game balancing by recognizing affect," in *International Conference on Fun and Games*. Springer, 2008, pp. 88–93.

[5] G. Andrade, G. Ramalho, H. Santana, and V. Corruble, "Automatic computer game balancing: a reinforcement learning approach," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, 2005, pp. 1111–1112.

[6] ——, "Extending reinforcement learning to provide dynamic game balancing," in *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games, 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2005, pp. 7–12.

[7] G. Andrade, G. L. Ramalho, A. S. Gomes, and V. Corruble, "Dynamic game balancing: An evaluation of user satisfaction." *AIIDE*, vol. 6, pp. 3–8, 2006.

[8] M. Morosan and R. Poli, "Automated game balancing in ms pacman and starcraft using evolutionary algorithms," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2017, pp. 377–392.

[9] P. Beau and S. Bakkes, "Automated game balancing of asymmetric video games," in *2016 IEEE conference on computational intelligence and games (CIG)*. IEEE, 2016, pp. 1–8.

[10] V. Volz, G. Rudolph, and B. Naujoks, "Demonstrating the feasibility of automatic game balancing," in *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016, pp. 269–276.

[11] J. Pfau, A. Liapis, G. Volkmar, G. N. Yannakakis, and R. Malaka, "Dungeons & replicants: automated game balancing via deep player behavior modeling," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 431–438.

[12] A. Liapis, G. N. Yannakakis, and J. Togelius, "Sentient sketchbook: computer-assisted game level authoring," in *8th International Conference on the Foundations of Digital Games, Chania*. ACM, 2013.

[13] J. Tao, Y. Xiong, S. Zhao, Y. Xu, J. Lin, R. Wu, and C. Fan, "Xai-driven explainable multi-view game cheating detection," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 144–151.

[14] H.-T. Joo and K.-J. Kim, "Visualization of deep reinforcement learning using grad-cam: How ai plays atari games?" in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–2.

[15] J. Zhu, A. Liapis, S. Risi, R. Bidarra, and G. M. Youngblood, "Explainable ai for designers: A human-centered perspective on mixed-initiative co-creation," in *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2018, pp. 1–8.

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in neural information processing systems*, 2017, pp. 4765–4774.

[17] S. O. Villar, "microRTS - GitHub," Jun. 2020, original-date: 2015-03-12T18:16:53Z. [Online]. Available: https://github.com/santiontanon/microrts

[18] F. Richoux, "microphantom: Playing microrts under uncertainty and chaos," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 670–677.
[19] A. Ouessai, M. Salem, and A. M. Mora, "Improving the performance of mcts-based μrts agents through move pruning," in *2020 IEEE Conference on Games (CoG)*. IEEE, 2020, pp. 708–715.
[20] S. Ontanón and M. Buro, "Adversarial hierarchical-task network planning for complex real-time games," in *Proceedings of the 24th International Conference on Artificial Intelligence*, 2015, pp. 1652–1658.
[21] S. Ontanon, "Experiments on learning unit-action models from replay data from rts games," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 12-1, 2016.
[22] M. Stanescu, N. A. Barriga, A. Hess, and M. Buro, "Evaluating real-time strategy game states using convolutional neural networks," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 2016, pp. 1–7.
[23] N. A. Barriga, M. Stanescu, and M. Buro, "Game tree search based on nondeterministic action scripts in real-time strategy games," *IEEE Transactions on Games*, vol. 10, no. 1, pp. 69–77, 2017.
[24] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.
[25] A. Dachowicz, K. Mall, P. Balasubramani, A. Maheshwari, A. K. Raz, J. H. Panchal, and D. A. DeLaurentis, "Mission engineering and design using real-time strategy games: An explainable-ai approach," *Journal of Mechanical Design*, 2020, [Revision Stage].
[26] S. Lundberg, "A game theoretic approach to explain the output of any machine learning model." Jan. 2021, original-date: 2016-11-22T19:17:08Z. [Online]. Available: https://github.com/slundberg/shap
[27] G. Erion, J. D. Janizek, P. Sturmfels, S. Lundberg, and S.-I. Lee, "Improving performance of deep learning models with axiomatic attribution priors and expected gradients," *arXiv preprint arXiv:1906.10670*, 2019.
[28] W. B. Powell and I. O. Ryzhov, *Optimal learning*. John Wiley & Sons, 2012, vol. 841.