Mission Engineering and Design using Real-Time Strategy Games: An Explainable-AI Approach

Article *in* Journal of Mechanical Design · October 2021

DOI: 10.1115/1.4052841

7 authors, including:

Adam Dachowicz
Purdue University
11 PUBLICATIONS **39** CITATIONS
SEE PROFILE

Kshitij Mall
Purdue University
27 PUBLICATIONS **156** CITATIONS
SEE PROFILE

Prajwal Balasubramani
Purdue University
6 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Apoorv Maheshwari
Quantitative Scientific Solutions LLC
28 PUBLICATIONS **120** CITATIONS
SEE PROFILE

Mission Engineering and Design using Real-Time Strategy Games: An Explainable-AI Approach*

Adam Dachowicz, Kshitij Mall, Prajwal Balasubramani, Apoorv Maheshwari, Jitesh H. Panchal, Daniel A. DeLaurentis†
Purdue University
West Lafayette, Indiana, USA 47907

Ali K. Raz
George Mason University
Fairfax, Virginia, USA 22030

Mission design is a challenging problem, requiring designers to consider complex design spaces and dynamically evolving mission environments. In this paper, we adapt computational design approaches, widely used by the engineering design community, to address unique challenges associated with mission design. We present a framework to enable efficient mission design by efficiently building a surrogate model of the mission simulation environment to assist with design tasks. This framework combines design of experiments (DOE) techniques for data collection, meta-modeling with machine learning models, and uncertainty quantification (UQ) and explainable AI (XAI) techniques to validate the model and explore the mission design space. We demonstrate this framework using an open-source real-time strategy (RTS) game called microRTS as our mission environment. The objective considered in this use case is game balance, observed through the probability of each player winning. Mission parameters are varied according to a DOE over chosen player bots and possible initial conditions of the microRTS game. A neural network model is then trained based on gameplay data obtained from the specified experiments to predict the probability of a player winning given any game state. The model confidence is evaluated using Monte Carlo Dropout Networks (MCDN), and an explanation model is built using SHapley Additive exPlanations (SHAP). Design changes to a sample game are introduced based on important features of the game identified by SHAP analysis. Results show that this analysis can successfully capture feature importance and uncertainty in predictions to guide additional data collection for mission design exploration.

*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.

†Corresponding Author

1 Introduction: A Framework for Mission Engineering and Design

Mission engineering and design (ME&D) is an important class of systems design problems that involves deliberate planning, analyzing, organizing, and integrating current and emerging operational and system capabilities to achieve desired mission outcomes [1]. ME&D represents a challenging design problem involving very large design spaces compared to other systems design problems, dynamic interactions between the mission environment and self-interested agent(s) operating in that environment, and continuous evolution of environmental features, agent characteristics, or both. As an example, consider a commander responsible for determining order of battle in a military scenario. This task represents a mission design challenge requiring multi-tiered and multi-dimensional analysis to select force assets and determine the underlying system-of-systems (SoS) architecture required for desired effects. The commander would be interested in designing a mission that is most likely to meet his/her objectives, while denying the enemy their objectives. Such a mission can be abstracted as a game played between two players (agents), where the goal of each agent is to design the mission (environmental factors, agent capabilities, etc.) to meet some design objective, such as maximizing their probability of winning the engagement subject to constraints on available assets.

In this paper, we propose a framework for addressing the ME&D design problem, and illustrate the framework by applying it in a real-time strategy (RTS) game environment. Mission designers already use simulation environments such as *AFSIM* [2–4], *Command* [5], and *Flashpoint Campaigns* [6, 7] to evaluate mission scenarios; RTS games provide a similar test bed to support ME&D by abstracting mission execution as a game played between agents. These games can capture complex interactions between intelligent agents and dynamic environments while still providing a controlled experimental setup. We use such an environment to simulate

many game *plays*, and then analyze the outputs of these plays to train a surrogate model that (i) maps game and agent parameters to a relevant mission design objective and (ii) provides actionable insights that will inform improved mission designs.

In practice the objective of the mission designer may incorporate any number of mission objectives, for instance by balancing acquisition and placement of assets in an engagement with cost constraints. The objective considered in this paper is the probability of winning for a given player, which is related to the notion of *game balance* used in the multi-player game community. For 2-player game scenarios like those considered in this paper, deviation from a probability of 0.5 could be considered a game (im)balance measure. The notion of game balance applied to ME&D is compelling: if one can predict when a mission/battle would be balanced, one can also determine the optimal means to imbalance the engagement (e.g., maximizing the probability of winning under resource constraints, increasing weapons systems strength or acquiring new systems to be most effective under cost constraints).

Assessing game balance (and eventually designing to maximize probability of win by creating imbalance) is challenging because of the high-dimensional design space, high computational cost of generating data by running the games, lack of clear metrics to quantify game balance, and difficulty interpreting the surrogate model's reasoning when iterating over new game designs. Our proposed framework addresses these challenges for generic mission design scenarios. A flowchart of the proposed framework is provided in Fig. 1. The specific approaches to each element implemented in this paper are called out in brackets in Fig. 1. The key components of this framework include i) a design of experiments (DOE) used to specify simulations (games) to run to generate training data, ii) a surrogate model built to predict a measure of game balance as a function of game features, iii) a method to estimate uncertainty in model outputs, and iv) a method to explain model outputs to a designer to guide future design iterations.

We also present an illustrative scenario where balance in an RTS environment is considered. To manage the game design space and specify the number of game simulations to run, our illustration specifies a DOE that varies game parameters to create a robust library of game plays. To address building the surrogate model, we consider only one game balance metric (probability of win) and leverage neural network approaches, specifically convolutional neural networks (CNNs), which are appropriate for learning the mapping between image-like game data extracted from our game plays and the relevant measure of balance. The DOE and/or the neural network architecture may them be refined to improve the accuracy of the model as needed. To better understand the relative impact of different mission parameters, and to guide sequential data acquisition [8, 9] to improve the game balance model, we employ SHapley Additive exPlanations (SHAP) analysis to help clarify model reasoning to designers and Monte Carlo Dropout Network (MCDN) analysis to estimate uncertainty in the output. With these methods, designers can investigate which input features contribute the most to variance in the model output. With these insights, designers can focus on the most relevant parameters by expanding their consideration in the DOE defining the next batch of games to run. In this way, the model may be improved through a sequential data acquisition process, leading to improved mission analysis and design over time. Furthermore, if the explanations are found to be incorrect, SHAP analysis can inform refinements to the DOE to obtain informative data for the surrogate and explanation models.

The structure of this paper is as follows. Section 2 provides a brief background on game balance, the game used in this study, and XAI. Section 3 demonstrates application of the approach to mission design using this game, and Section 4 presents results for a particular exemplar game design and changes to that design motivated by XAI analysis. Conclusions are then presented based on the results obtained from the devised framework.

2 Background

In this section, we provide background on literature relevant to the proposed framework. Games have been used in literature to understand and guide design decisions [10, 11]. In the following subsections, we discuss literature relevant to game balance. We also present background on microRTS, the game used in this study, and literature on XAI relevant to our modeling approach.

2.1 Game Balance

In the gaming industry, the idea of game balance reflects both how playable and enjoyable a game is for a player. The problem of game balance has been well-addressed by researchers in the game of Chess. Of the many balance-related questions, the impact of starting order in turn-based games like Chess is one that has been well studied [12, 13]. However, a small imbalance in Chess comes from the fact that a player with white pieces makes the first move, leading to a small edge over a player with black pieces. The Chess community continues to seek ways in which they can introduce rule changes to create imbalances and make the game more engaging [13]. Other balance related questions are still being explored to find ways for evaluating and controlling game balance.

Many efforts have been made to understand game balance and figure out a solution to reduce the need for human play-testing and iterative processes by implementing automation techniques. Jaffe [14] demonstrated a novel quantitative method using identical programmed agents instead of human players to assess game balance. The authors further proposed a "restricted game play" method, which allows perturbation of game parameters in only one dimension and assesses the corresponding changes to the game state [12]. Beyer et al. [15] proposed an integrated method to act as a stepping stone to eventually move away from manual balancing with an integrated balancing process. Leigh et al. [16] stated that mathematical proof for game balance may not be

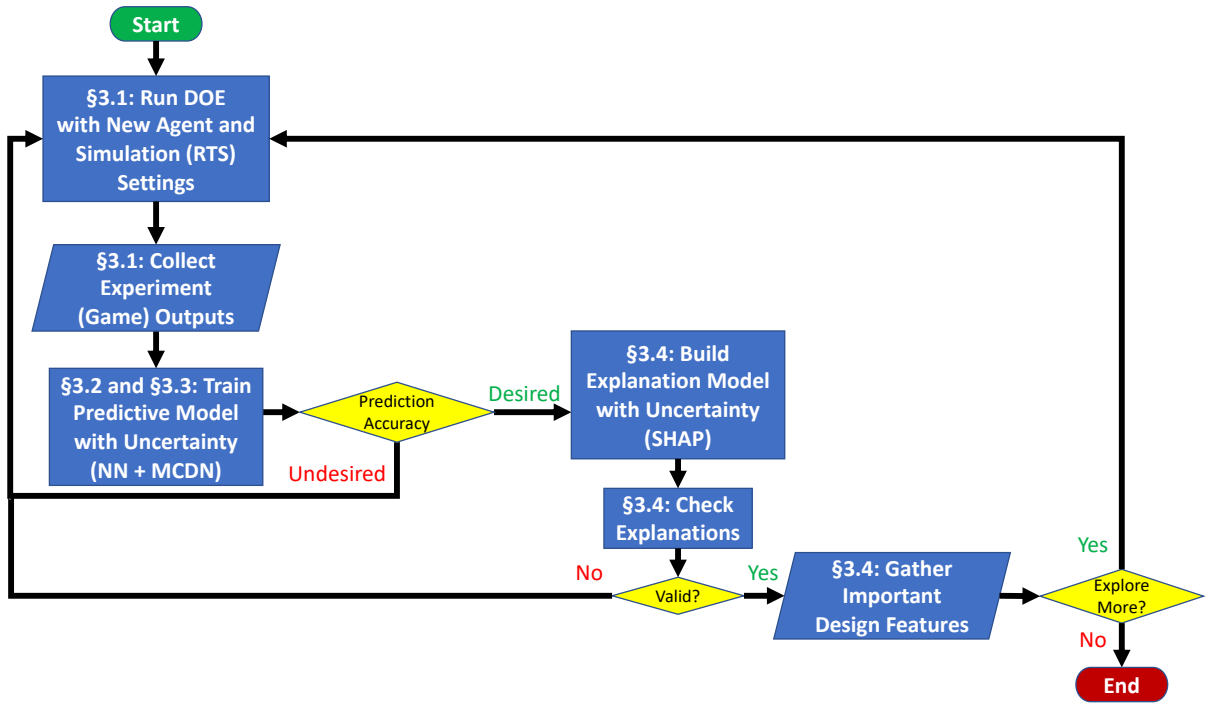

Fig. 1. Flowchart of the proposed ME&D framework.

possible with the current AI techniques, which can be validated using co-evolutionary algorithms. More recently, Xia and Anand [17] explored the space of multi-agent models intertwined with ecosystem mechanism to provide a more flexible way of game balance control. Our approach leverages the work of Jaffe [14] and Xia [17], by using restricted self-play games (games played between identical bots), to understand game balance in a two-player RTS setting.

Notions of game balance can also be used to investigate inherent imbalances in a game environment. microRTS, while being a very simple game, surprisingly has an inherent bias towards one of the two players involved. The root cause for this is to do with the game definitions and code structure dictating players' movements. While it may be difficult for the players or the designers to spot such biases, an ME&D design framework should be able to identify and quantify the initial imbalance so that such imbalances can be controlled for and/or corrected. This further shows the importance of game balance assessment in ME&D to identify any unexpected or unwarranted biases irrespective of how small or simple the implementation is. A brief overview of our game choice, microRTS, is presented in the following subsection. Note that our generic framework is applicable to more complex RTS games, such as StarCraft.

2.2 An Overview of microRTS

microRTS is a simple, open-source implementation of an RTS game by Ontanon [18]. It is developed in the Java programming language and is available via General Public License protocols for use and distribution. It is primarily designed to perform AI research and functions as a testbed to conceptualize and test learning algorithms. The advantage microRTS has over other high fidelity games like StarCraft or FreeCiv is its simplicity, which enables quick testing of theoretical ideas. Many researchers have used microRTS for successfully testing their hypotheses and ideas, not restricted to AI-based topics. The topics range from hierarchical task planning in networks to learning action probability models [19, 20]. Another key research area that microRTS is actively used in is the application of CNNs, Monte Carlo Tree Searches (MCTSs), and Reinforcement Learning (RL) methods to evaluate game states [21, 22]. The game's low barrier to entry and large developer community are key features that motivated us to choose it for this study. In its default configuration, microRTS is a two-player, deterministic, and real-time (i.e. players can issue actions simultaneously and independently) game. The game has two types of results: win-loss and draw. Win-loss is achieved when one player is no longer able to continue the game (make actions) due to loss of units, buildings, and resources. In this case, the other player is declared the winner. Draws can occur due to two different reasons: 1) when the time allotted for gameplay runs out, and 2) when both players have no units and resources to continue leading to a stalemate.

2.2.1 Map

The map in microRTS is a basic 2D array that stores information for each cell in the array. The array creates a graphical user interface (GUI) grid that is implemented in the source code allowing users to observe the gameplay in real-time. The size of the grid is variable according to the user's choice, but the code comes with a few standard settings; 8×8, 12×12, 16×16, 24×24, and 32×32 cells. We used the 16×16 map size that has ample player movement space

and high player interaction based on our preliminary testing. The map GUI also has a timer and player statistics indicator at the bottom left corner.

2.2.2 Agents, Objects, and Actions

microRTS has a total of six units that are summarized in Table 1 along with their associated functions. Objects are entities within the game that do not have an active role in the system but act as interactive elements of the environment. microRTS has only two objects built into the model; resources (minerals) and terrain. Actions are a set of available commands that a player (bot or human) can assign to their units to carry out during gameplay. microRTS has five actions that a player may utilize: move, attack, harvest, train, and construct.

Table 1. Agents, Objects, and Actions in microRTS

Units	Function
Base	Stores resources and trains workers
Barrack	Trains attack units
Worker	Harvests minerals and constructs buildings
Heavy (*H*)	High attack ability, low speed
Light (*L*)	Low attack ability, high speed
Ranged (*R*)	Long range attack ability

2.2.3 Bots

Bots are hard-coded players that perform a given strategy throughout the game. microRTS provides a variety of bots within the open-source package. The options range from simple heuristic bots to more complex bots using search algorithms and iterations (for eg., Monte Carlo Search). For this study we used the Monte Carlo Search bot that looks for *X* possible actions in its current game state and iterates *Y* possible future states (*X and Y* are controlled by input arguments to the bot). We used a value of 100 for both input arguments. The Monte Carlo Search bot then chooses the action with the highest reward from among its portfolio of actions. A game using bots with sophisticated algorithms takes around 2 orders of magnitude of game time as compared to a game with heuristic bots due to the deterministic nature of the latter. In order to reduce the variance in skill levels among the two players, our framework excludes human players. To further reduce the skill level difference, we conduct games where both players are the same bots i.e., self-play games.

2.2.4 Game Data

microRTS allows researchers to store and retrieve game data in files called traces. This file records information including every action, agent, object, and parameters involved in the game at each game timestep. Trace files are used to visualize the game. We primarily use this file to generate game frames to train our ML models. These frames are then used to generate the geographic and non-geographic data used as inputs to our models discussed in Section 3, as well as determine the winner of each game. For our purposes, the winner of the game is defined as the player with assets remaining on the field in the last frame of the game. It can also be used to gain deeper insights into specific games while looking for specific instances of game events.

2.3 Explainable Artificial Intelligence

One of the main contributions of this paper is the use of XAI in guiding mission design. The focus of many current AI researchers is on developing and using the third wave of AI (popularly known as the XAI) based on contextual adaptation that combines the strengths of the first and second waves of AI systems [23, 24]. In XAI, the machines construct underlying explanatory models for classes of real world problems that allow them to characterize real world phenomena [24]. There is a trade-off between the performance and interpretability of AI models [25]. Current second wave AI models are great in performance, but are also extremely hard to interpret [25, 26]. The purpose of a XAI system is to enable better understanding, management, and regulation of AI models. XAI also helps in debugging errors and in building trustworthiness.

For this study, post-hoc XAI analysis has been used to explain the ML models. There are several types of approaches in post-hoc analysis, including visualization, model simplification, text explanation, local explanations, explanations by example, and feature importance [25]. Since the model developed in this study is for a game involving many game features, the feature importance approach was selected. Several tools have been built to conduct post-hoc analysis, such as Integrated Gradients (IG) technique [27], Local Interpretable Model-agnostic Explanations (LIME) [28], Layer-wise Relevance Propagation (LRP) [29] and SHAP [30, 31].

In this study, we employed SHAP, which is a recently devised XAI approach with a robust Python implementation. SHAP is based on Shapley values first introduced by Lloyd Shapley in 1953. SHAP is considered a state-of-the-art post-hoc XAI method for reverse engineering the output of any predictive model. The strength of SHAP is that it identifies the feature importance for a prediction or for a model as a whole [32]. Before SHAP was devised, the researchers consistently faced the issue of using different XAI methods for different types of AI models. SHAP unites multiple additive feature importance methods to enforce the desirable properties of local accuracy, missingness, and consistency for the computed explanation coefficients [30]. In the following section, we expand on our approach to model game balance using SHAP in the ME&D context.

3 Approach

The remainder of this paper will focus on demonstrating our proposed ME&D design framework by applying the framework to assess the balance of game designs in microRTS. Using this framework, we approach the overall problem of balancing or unbalancing an RTS game by learning the map from game design (i.e., game features and rules), individual game states as the game progresses, and resources available to a player to the probability of a player winning the game. In this section, we lay out our framework for tackling this mapping. We begin by describing our definition of game balance and measure of this balance. We then describe our modeling approach for constructing the mapping between game parameters and game states, and the game balance measures.

For the purposes of this paper, we define game balance as follows:

Definition: A game is *balanced* if all players have an equal chance of winning the game given the game environment without any knowledge of the player *skill*.

The game environment includes 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. In our work, we utilize the following game balance metric in microRTS.

Predicted Game Outcome. Our model predicts the probability of all the game outcomes, i.e., probability of win for each player and probability of draw. These measures are the expectation of the game outcome given only the current frame, which may occur in similar games with different outcomes. Thus, these measures should be interpreted as the measure of *balance* at each game frame in favor of each outcome (a player winning or a draw). As these measures are computed *for each frame*, they change over the course of the game as the game evolves.

Our *modeling assumptions* in this paper are that:

1. **Skill is controlled.** The impact of differences in player "skill" on game balance may be controlled by giving each player identical strategies for play. This is achieved by considering games between identical player bots. We do not address models for learning optimal player strategies in this paper.
2. **Balance can be predicted at any time in a game.** Probability of win is a valid measure of game balance, and can be predicted as a function of features extracted from individual game states observed as games are played without considering time-dependent information.
3. **Model error is normally distributed.** This requirement allows the use of MCDN to estimate model uncertainty, as discussed in Section 3.3.

Note that in this paper, we will take the perspective of game designers that have control over the initial game states but will not intervene once a game play has started. Thus, even though our modeling approach is designed to generalize to analysis at any time during a game, here we will focus our analysis in greater detail on initial game frames.

We discuss the different blocks of our approach in the subsequent subsections. We start with a brief discussion on the DOE.

3.1 Design of Experiments (DOE)

Overview. To properly explore the game design space, we specify a DOE that defines a set of games to play with varying initial states. Unlike existing work studying the design of RTS bots [33–42] or the design of strategies for these bots [43], we aim to study the impact of game design changes while controlling for the impact of the player bots and their skill. We address controlling for player *skill* by considering only one type of player bot for this analysis, namely Monte Carlo AI bots implemented for microRTS. We add sufficient variety to the game data by varying the game parameters at the start of each game play. The parameters allowed to vary, and the allowable ranges, are summarized in Table 2. There are six parameters varied for each player (base x and y locations, worker health, worker damage, fighter unit cost, and production time), and two general parameters (terrain center and density). Parameters for each player were allowed to vary asymmetrically, yielding a total of 14 varied parameters.

Table 2. DOE with modified game parameters and ranges

Attribute	**Values Range**
Players (do not vary)	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)

Self-Play Games. We call games where player assets may differ, but players follow the same logic (i.e., players are identical bots) as *self-play games*. All games discussed in this paper are self-play games. We view this as a method for controlling player skill.

Stochastic Games. In this paper, we report results for bots using Monte Carlo algorithms to determine moves. This implies the bots are stochastic, meaning that game results are not determined by the starting conditions of the game. Thus, for the final model used to generate results reported here, data was extracted from multiple runs of each game specified in the DOE.

Specifying the DOE. We generated a Latin hypercube design to determine game parameters for each game. Using this design, we specified 5,000 unique games, each of which was run five times to generate a total of 25,000 gameplays. From each of these gameplays, 30 randomly-selected game

frames generated the data used to train the final game balance metric models.

Data Preprocessing. The training data was preprocessed as follows: (i) normalization of all image channel inputs to [0,1], (ii) standardization of non-geographic data, (iii) standardization and log-transform of the target remaining and total time data.

Game Terrain Generation. Each player starts in any cell in their 5×5 *starting* areas. The terrain attributes comprise of possible terrain states (presence or absence of impassable terrain feature) for a map with $16 \times 16 = 256$ available cells. These features were randomly generated using two parameters that control the location of likely terrain features and the probability of spawning terrain features around that location. These terrain features were generated to add variance to the training data set and provide an additional input to be analyzed using XAI techniques like SHAP (discussed in Section 3.5). Results discussed in Section 4 are for test maps that do not contain terrain features.

Let these parameters be denoted by $k_{\text{center}} \in [0, 1]$ and $k_{\text{density}} \in [0, 1]$, respectively. Let the center of the terrain features be denoted by $c = (c_x, c_y)$, with x- and y-coordinates as c_x and c_y, respectively. For all maps generated in this design, the permitted centers were constrained to the diagonal line $c_x = c_y$. The terrain was generated using the following steps.

1. Set the center location $c_x = c_y = \text{round}(k_{\text{center}} * n)$, where $n = 16$ is the map dimension in the x and y directions.
2. Generate a 2D array of all 0s, and assign a value of 1 for the entry (c_x, c_y) .
3. Apply a Gaussian blurring filter with a standard deviation of 2.5 pixels to the array.
4. Scale all pixel values by k_{density}.
5. For each pixel, (x, y), assign a terrain feature to that pixel with a probability equal to the pixel value.

The terrain features were further constrained by preventing spawning in the 5×5 starting areas for each player, and by rejecting any maps that made it impossible for players to access each other's bases. Example maps are provided in Fig. 2.

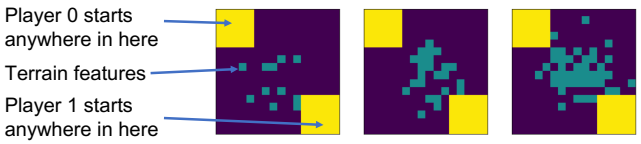

Fig. 2. Examples of randomly generated maps with different starting center and density parameters.

3.2 Winner Prediction Model

Given the training data, we developed a neural network model to predict the winner game balance measures (probability of Player 0 and Player 1 winning the game, and the probability of a draw). The model has the architecture specified in Fig. 3, and takes image-based features specified in Table 3 (captured in 15 separate 16×16 image channels) and non-image based features summarized in Table 4 (captured in 15 numerical features representing minimum distances between each player base and the units of each player, player unit health, player unit count, x- and y- locations for the centroid of player units, rate of player damage, and map terrain count). This data is extracted from microRTS trace data, and is limited to data that can be calculated at a specific game frame.

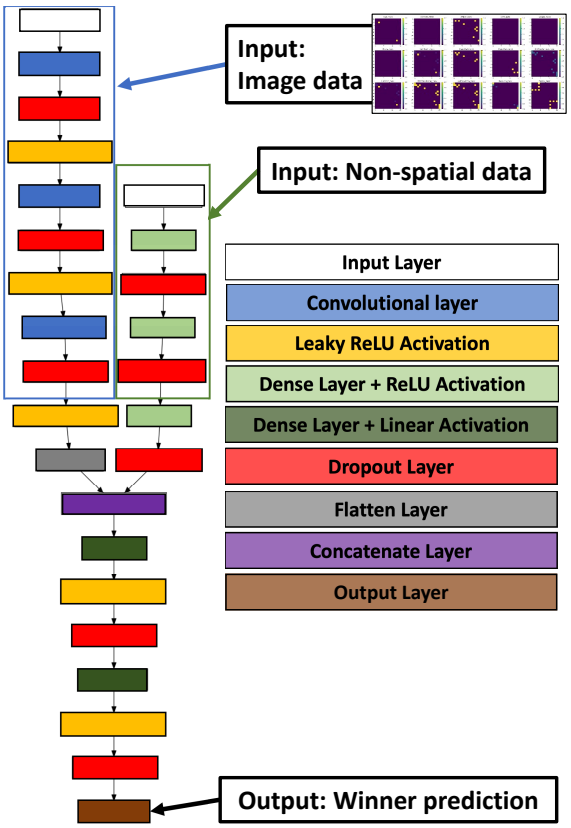

Fig. 3. Neural network architecture diagram.

Specific classes of neural nets, such as CNNs, are used for handling spatially structured data, such as image data. CNN is appropriate for considering the kinds of features relevant to the game balance metrics we consider: relative location of assets, resources, asset health and attack power, position of bases, and position of terrain features, if any. The convolutional branch of the model is heavily inspired from [22], which includes implementation of a CNN model for microRTS-generated image data. The model was implemented using Keras [44].

We also introduce non-image based features through additional dense layers. A primary motivation for this is to introduce features that are easy to interpret through XAI techniques like SHAP. A second dense branch takes as input the 15 non-geographic inputs discussed in the previous section.

Table 3. Geographic features extracted from each game frame (all pixels set to 0 unless otherwise specified)

Channel	Channel Description
Base Location	All pixels with base set to 1
Barracks Location	All pixels with barracks set to 1
Worker Location	All pixels with Worker set to 1
Light Location	All pixels with Light set to 1
Heavy Location	All pixels with Heavy set to 1
Ranged Location	All pixels with Ranged set to 1
Unit Cost	Pixel set to unit resource cost
Unit Damage	Pixel set to unit max damage
Unit Range	Pixel set to unit attack range
Unit Movement	Pixel set to unit movement time
Asset Health	Pixels assigned to current health of the corresponding asset
Player 0	All pixels with asset owned by 0 set to 1
Player 1	All pixels with asset owned by 1 set to 1
Resource Amount	Pixels set to current resources of corresponding asset
Terrain Location	Pixels with terrain features set to 1

Table 4. Descriptions and unique feature counts for non-geographic features extracted from each game frame.

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

These inputs are fed forward through three dense layers with 64, 32, and 16 nodes, respectively. These dense layers are each followed by rectified linear units (ReLUs).

The output of the convolutional layer is then flattened, and the outputs of these two branches are concatenated and further passed to two dense linear layers with 128 and 64 leaky rectified linear units (LReLUs), respectively. The output layer is fully connected with three units, one for each player and one for the draw case. To obtain outputs that may be interpreted as probabilities of players winning or a draw occuring, a softmax function is also applied to the output layer. Categorical crossentropy loss was used to train the winner model.

The winner model was trained using Adam optimization [45] over 62 epochs, with 90 percent of the data used for training and 10 percent used for testing and validation, and a 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.

Accuracy on the test dataset was computed at each epoch, and is plotted in Fig. 4. Model accuracy over the test dataset was 87.1% after 62 training epochs. We observe that the model converges quickly during training (in the first few epochs). This may be due to the decaying learning rate and the frequency of mid- and late-game cases in the training data that are more easily classified, leading to high average accuracy early in training. Note that in general, we observe that the model under-performs early in games, where important signals like difference in player health or unit counts (as indicated by SHAP results during game play) are not available. We consider this characteristic desirable: we want this surrogate model to capture variance in confidence over time and over different initial game states.

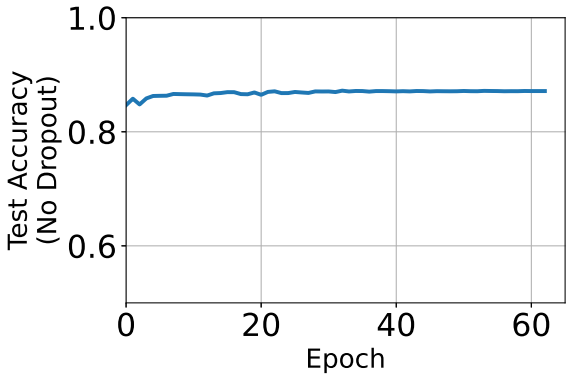

Fig. 4. Model accuracy on the test data as a function of epoch with no MCDN dropout applied at the test time.

3.3 Uncertainty Quantification (UQ)

UQ characterizes the robustness of the current model and, by identifying areas with low model confidence or high confidence in incorrect predictions, informs data collection for improving model performance. Possible source of uncertainty and/or bias in the predictive model include, for instance, (i) limitations in the design of experiments (such as not accounting for certain interactions between game design features or sparse sampling in high-variance regions of the design space), (ii) hidden undesired imbalances caused by

features of the game engine (such as those introduced by pathing algorithms for some MicroRTS bots), and (iii) poor choice of features used or neglected as input to the model. Uncertainty analysis for specific game states can help identify the first two sources, which can then be addressed by increased sampling or altering bot or game code. Uncertainty analysis over the full test dataset, or for incorrectly labeled data, can help identify the third type of source, which may be remedied by adding additional inputs to the model or updating model architecture.

We implemented UQ using Monte Carlo Dropout Networks (MCDNs) [46], a means of implementing UQ for CNN and other neural net models that are computationally more efficient than other Bayesian methods for estimating model uncertainty [46]. MCDNs are considered one of the state-of-the-art methods for implementing UQ in neural networks [47]. Dropout is a regularization technique that helps in preventing overfitting of the training data. On the other hand, Monte Carlo is a class of computational algorithms that depend on repeated random sampling to obtain a distribution of a numerical quantity of interest.

As the name indicates, MCDNs involve dropping out certain neurons in the neural network leading to a CNN model that is less sensitive to the input data. Thus, in MCDNs, there is a *dropout rate* of neurons, typically set between 0 and 0.5, which refers to the percentage of neurons switched off (50% of all neurons switched off in case of dropout rate of 0.5). While training the model on a single batch of data, the model's architecture is different each time due to a different set of neurons being dropped out. Therefore, the different networks with different neurons dropped out are treated as Monte Carlo samples from the space of all available models. The outcome obtained as a result is an average ensemble of many different neural networks.

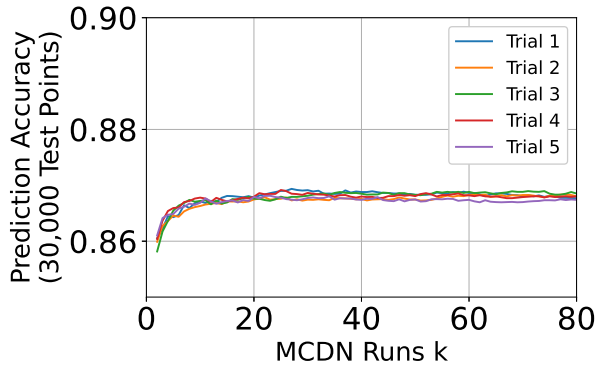

Fig. 5. Accuracy of winner prediction model with increasing MCDN runs for 30,000 test points. Results plotted for 5 trials.

MCDN was implemented in this study as follows. Dropout layers were added after each convolutional and dense layer in the network as seen in Fig. 3, with a dropout rate of 0.5. Importantly, dropout was permitted to occur during both training and evaluation. At test time when using the model to make game balance predictions, the model was run $k = 20$ times, each time with a different seed controlling the behavior of the dropout layers. Results from each of these runs were then averaged to produce the final prediction, and the standard deviation was also computed. This deviation provides an estimate of the model's uncertainty for the given input, and can provide useful insight into the model's confidence at different points in a game play. Model accuracy for test data was observed to reach a peak at around $k = 20$, with additional MCDN runs offering little improvement for the increased computational cost (see Fig. 5).

As the SHAP results are a function of the trained model, SHAP results were computed for each of the 20 runs and then averaged to provide a final SHAP result for the input. A brief discussion about SHAP is given below.

3.4 SHAP for Explainability

As previously described, SHAP falls under post-hoc analysis XAI methods that utilizes Shapley values to describe the feature importance of various features involved in any mission design. Shapley values utilize coalition game theory. A Shapley value captures the contribution of the corresponding feature towards the prediction of interest from among all features used as input for the model. In ME&D context, a Shapley value, for example, tells how the health or strength of a particular soldier contributes to the prediction of probability of win or loss of his/her army. Furthermore, SHAP involves *additive* explanations because the Shapley values sum to give the output generated by the learning model.

Figure 6 shows the implementation of SHAP to convert the complex model used in this project to a simple linear explanation model. The input data, z, is mapped to simpler inputs, z', using a mapping function, $h_z(z')$. Thus, z is converted to z', with elements taking values only of 0 and 1, corresponding to a feature being absent or present, respectively. The linear XAI model outputs $g(z')$ that has the form given by Eq. 2. Let z_S be the set of features in z with nonzero entries in z'. Then, the ϕ in Eq. 2 are attributions for each feature present in z', and the SHAP values are averages of these attributions over all possible ordered subsets z_S of z such that the values satisfy local accuracy, missingness, and consistency properties [30]. Note that SHAP values can have positive or negative values. The additive nature of explanations provided by SHAP is also captured in Eq. 2.

$$f(z) = g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \tag{2}$$

For ML models like those considered in this paper, "absent" input features are not permitted. So, the direct computation of model responses to some subset z_S of input features needed to compute SHAP values is impossible. Thus for SHAP, for a subset of input features z_S, the model response $f(z_S)$ is approximated as $E[f(z)|z_S]$ [30], which itself must

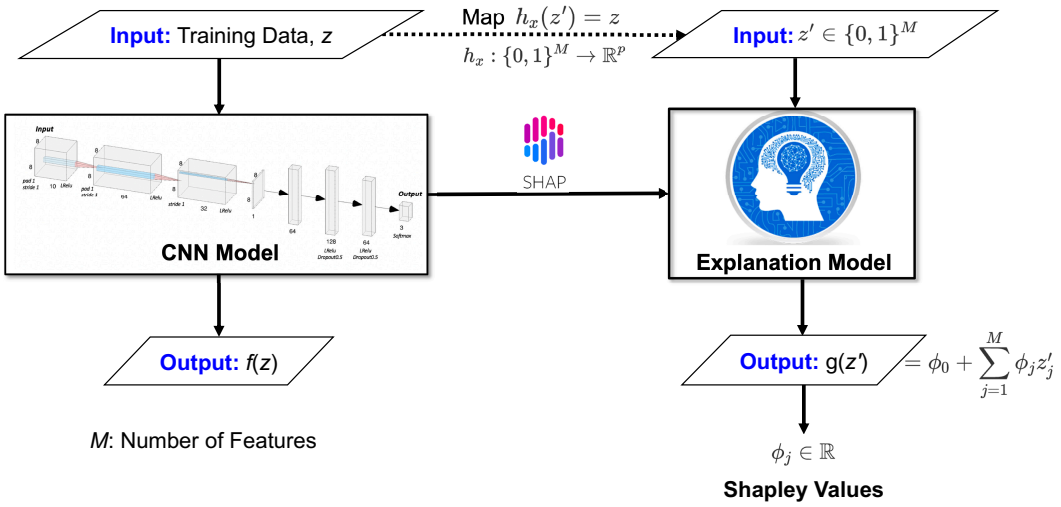

Fig. 6. Converting the CNN black box model to a more explainable model using SHAP.

be estimated for a given model and input data set. For this study, we employed the SHAP toolbox developed by Lundberg [48] for all SHAP analysis. To estimate Shapley values, from among the available methods, we chose *expected gradients*, an estimation method for differentiable models explained in detail by Erion et al. [49].

The positive and negative SHAP values are indicated by red and blue colors, respectively, in the plots generated by this toolbox. The prediction and SHAP results obtained for a sample gameplay is shown in the following section. Note that these results have UQ analysis embedded in them through MCDN, just as with the winner prediction results.

4 Results

In this section, we present an analysis of one game scenario analyzed using the methods discussed in Section 3. The goal of this analysis is to illustrate how the proposed framework can be applied to study the balance of this game and the important features contributing to game balance that a designer could change in future design iterations.

We begin by presenting prediction results for two example game plays (termed *perturbations* here to emphasize that new games are generated by perturbing the game parameters listed in Table 2) in Section 4.1, along with a brief discussion on interpreting SHAP results for game plays. In Section 4.2, we investigate one of these game plays in greater detail through XAI and UQ to determine game design changes that are likely to unbalance the game. We also discuss how these results may be used to identify regions of the game design space where more data should be collected to improve model performance. We present these analyses from the perspective of game designers that can vary initial game features (such as starting locations, unit strengths/health, and so on), and therefore focus our attention on model predictions for the first game frames in each perturbation.

As discussed above, we focus our analysis to probability of win as a measure of balance. However, note that we do not enforce a specific value of this probability as a threshold for defining a "balanced" game state. Rather, we say games are more or less balanced if the probability of win predictions for each player are more or less close to 0.5. A designer may enforce more specific thresholds or use deviation from win probability 0.5 as a design objective when iterating on game designs using this framework. We focus on presenting insights offered to designers using this framework, and do not comment on any specific objectives a designer may formulate for their use case. Furthermore, we do not view probability of a draw as an interpretation of balance, but rather as a possible end game state that a designer should keep in mind.

4.1 Winner Prediction for Perturbations

Representing winner prediction over game play. As discussed in Section 3, our game balance model takes in geographic and non-geographic data at each game frame as input, and yields player win/draw probabilities. Since we implement MCDNs at predict time, we can also estimate model uncertainty for each of these outputs.

As an illustration of these outputs, consider Fig. 7. Solid lines in each panel represent the *mean* prediction (y) at the given game time (x) for two sample game plays. Bands around these solid lines represent intervals of two standard deviations estimated from the MCDN results for that prediction. Specifically, Panel 7A are predictions for game frames sampled from a *balanced* game play defined by Perturbation 1 parameters (see Table 5 row 1 for specific game parameters), while Panel 7B results are sampled from an *unbalanced* game play (Player 0 has a greater worker maximum damage, while Player 1 has a greater worker health) defined by Perturbation 2. Perturbation 2 results are presented here to demonstrate outputs for an unbalanced game, and are not discussed further in this study for brevity. For reference, total health and unit counts for both players in the balanced game play are plotted in Fig. 8. Although these results rep-

resent only two sampled games, these figures display several strengths of this modeling approach as discussed below.

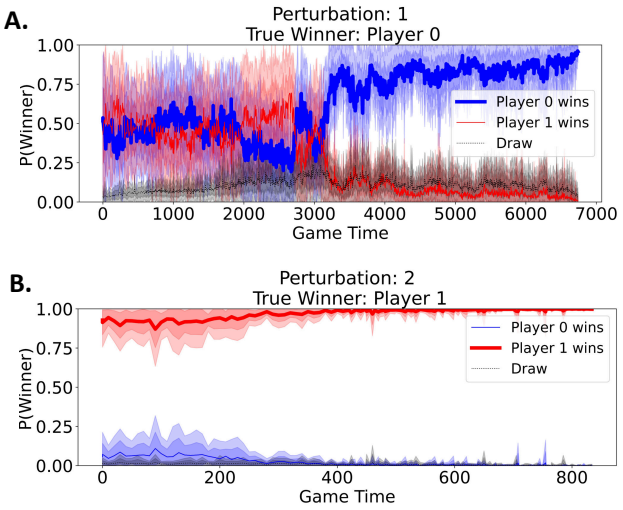

Fig. 7. Example winner predictions over two game plays: a "balanced" game with symmetrical game parameters for each player (A) corresponding to Perturbation 1, and a game unbalanced in favor of Player 1 (B) corresponding to Perturbation 2. Bands represent two standard deviations as computed from 20 MCDN runs.

Table 5. Perturbation input parameters. No terrain features were included in these games. Player 1 has Worker health 4, Worker maximum damage 1, and base location (13, 13) for all perturbations

Perturbation	P0 Worker Health	P0 Worker Damage	P0 Base Location (x, y)
1	4	1	(2, 2)
1a	3	1	(2, 2)
1b	4	2	(2, 2)
1c	4	1	(0, 0)

Studying initial probability of win and changes over game play. For the balanced game, the prediction that Player 0 will win is very close to the prediction that Player 1 will win at the start of the game. This is expected in a balanced scenario according to our notions of game balance. Later in the game, the balance shifts between Player 0 and Player 1 several times before the model's prediction shifts drastically towards Player 0 at around game time 3000. This prediction holds until Player 0 indeed wins the game. For the unbalanced game, we see the model more confidently predicts a winner (Player 1) at the start of the game, and this prediction holds until indeed Player 1 wins. A game designer may

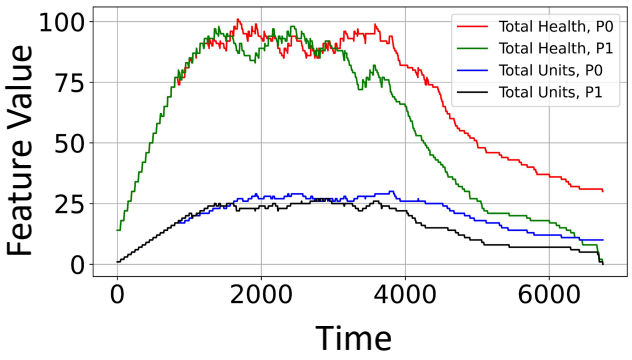

Fig. 8. Total asset health and unit counts for Player 0 and Player 1 during the Perturbation 1 game play represented in Figure 7A.

focus on regions relevant to their analysis when considering the next iteration of games to study: does the prediction at the first game frame agree with our expectations given the initial game parameters? Do swings in the prediction match with human expectations for shifts in the balance as the game evolves? How do these predictions map to expectations of the mission designer (e.g., does a reduction in enemy asset health as seen in Fig. 8 imply a large shift in balance)? In the case of the game represented in Panel 7A, a user could pair prediction results with the game inputs at different times and relevant SHAP results for predicting a win for Player 0 (see Figs. 9 to 11 and discussion in Section 4.2) to identify which game features or player assets the model *thinks* are important at a specific game time. This information could be used to identify game parameters that have out sized effects on the game balance measures under consideration by the designer, validate the model by ensuring predictions and importance align with human expectations about mission-level effects, or both. This is discussed in greater detail in the following subsections.

Analysis of uncertainty. Finally, consider how uncertainty changes over time in Fig. 7. For the balanced game, initial uncertainty is quite high, and this uncertainty shrinks as the game progresses and the model *picks* a winner. These results can be used to validate the model (we anticipate high uncertainty for frames early in a balanced game) and to identify game states that could deserve more study (if uncertainty is attributed to model under-fitting).

SHAP results for image inputs. Figure 9 shows the SHAP results for image-based data inputs corresponding to Perturbation 1. In Fig. 9, red pixels correspond to input features that contribute to a higher prediction that Player 0 will win, while blue pixels contribute to a lower prediction. The intensity of the color represents the magnitude of that contribution; white pixels add no impact in either direction. As seen in Fig. 9, the base locations of Players 0 and 1 have highly positive and negative contributions, respectively. The health of different units have smaller but still noticeable positive and negative contributions, respectively. The damage dealt by different units also have a lower but noticeable impact on Player 0's win prediction.

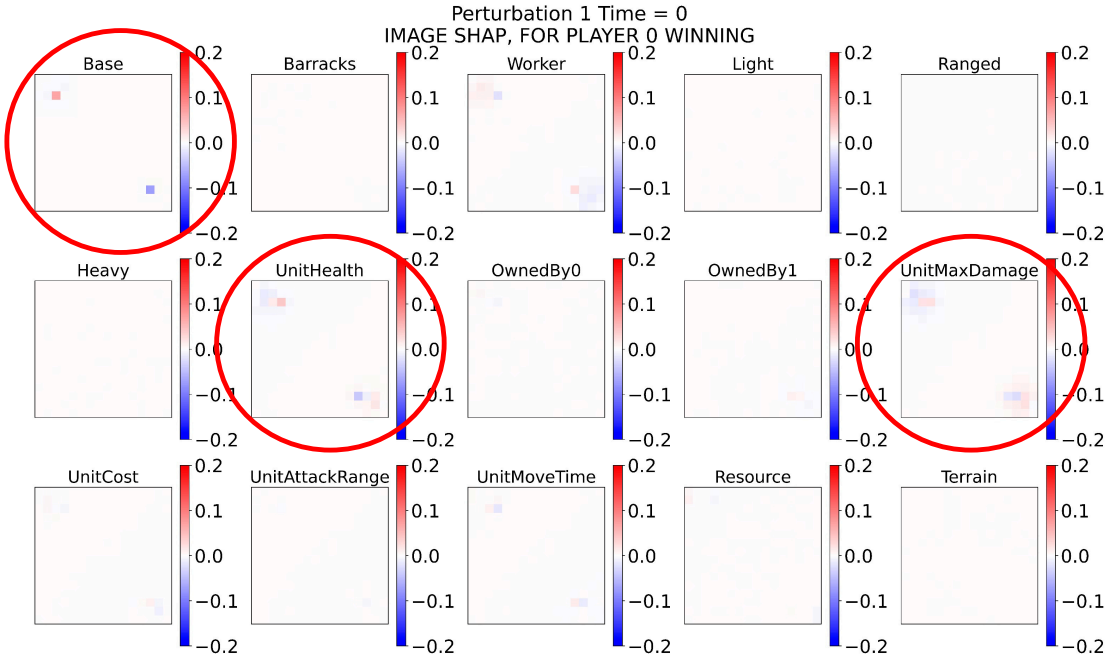

Fig. 9. SHAP results for geographic input data corresponding to the prediction that Player 0 will win the game defined in Perturbation 1 given the first frame inputs (channels representing the most significant information circled).

SHAP results for series inputs. Similar to image inputs, SHAP analysis can be conducted on non-image or series inputs corresponding to Perturbation 1 as shown in Fig. 10. Based on Fig. 10, the health of units of the two players, P0 and P1, have a major impact on the win probability for Player 0, with P0 health having a large negative contribution and P1 health having a large positive contribution towards P0's win prediction. The distance between units and bases of opposite players have the second highest impact while the number of units of the two players have the third highest impact on the probability of Player 0's win. Interestingly, for this balanced scenario, the relative effects of opposite features (P0 health vs. P1 health) have nearly-identical mean effect magnitudes but opposite effect signs, suggesting the model pays close attention to these features individually but their current balanced values lead to a small cumulative impact on the prediction.

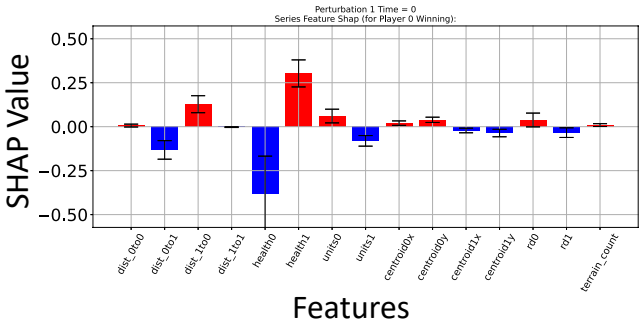

Fig. 10. SHAP results for non-geographic input data corresponding to the prediction that Player 0 will win the game defined in Perturbation 1 for the first frame inputs.

4.2 Detailed Perturbation Analysis

In this subsection, we provide a more in-depth analysis of Perturbation 1 as reported in Table 5. Note the baseline game ("Perturbation 1" in Table 5) is a game with symmetric parameter values for both players. This study is intended to highlight how the proposed game balance metric models and corresponding SHAP results can be used to diagnose model performance and game parameters most heavily influencing game balance, and thus should be emphasized by the mission designer in future DOE.

The perturbation under consideration represents a balanced game, where each player has identical worker health and damage output, and symmetric base placement, as previously described in Table 5. Both players have very high worker health, but equal damage output and symmetric base positioning. Reconsider the model predictions plotted in Fig. 7, where model predictions are plotted for one realization of a Perturbation 1 play. Note how the initial uncertainty in the winner predictions, represented as the bands around the winner prediction lines at time = 0 game steps, is quite high for this balanced game. For unbalanced games, one expects the model to favor one player over another, even at the first game frame (as can be seen at time = 0 game steps in Fig. 7(B)). This is intuitive as balanced games should be inherently more difficult to predict win for any player until the game has progressed.

SHAP analysis at the initial game frame. SHAP analysis is useful for games involving stochastic bots as results for the initial frame 1) are constant between game plays, and

2) should align with the model's *belief* about the balance of the initial game state. For this analysis, let us consider the prediction that Player 0 will win the game. Based on SHAP analysis of the geographic input features, the most important features contributing to a high probability of Player 0 winning include health of players, damage done by units on the map, and the position of each player's base as shown in Fig. 9. High absolute SHAP values for health and distance features in the series SHAP results for the first frame suggest the model pays close attention to these features as well, but note the results for each player appear to cancel each other out as shown in Fig. 10. These observations drove our decisions on which features to manipulate for Perturbations 1a, 1b, and 1c, detailed in Table 5.

Series SHAP results over game time. Before discussing the results of Perturbations 1a-1c, consider Fig. 11, which contains mean and uncertainty bands of SHAP results for the non-geographic features at each game frame. We can see how changes to unit health, unit counts, and damage done by each player have a varying impact at different stages of the game. Furthermore, SHAP reveals features that are important at points in the game where balance seems to shift. Consider the point in the example game for Perturbation 1 around time 3000 game steps; we can see high importance attributed to the total health of each Player as well as the rate of damage output by each Player and to a lesser extent the total number of units on the field. At the start of the game, we can see the relative distances between opposing player units and bases are also important, which drop off as the game progresses.

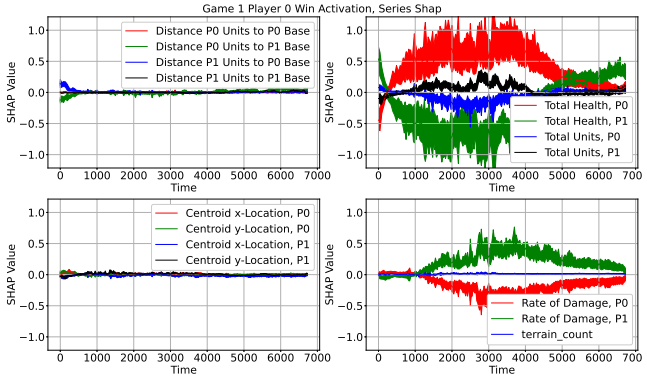

Fig. 11. Perturbation series input SHAP values with respect to predicting Player 0's win for a representative game play.

The average and standard deviation of the absolute value of the series SHAP results with respect to predicting win for Player 0 across the Perturbation 1 game play is represented in Fig. 12. Note that the main contributors include the total health of each player, the rate of damage of each player, and the total number of units under the control of each player.

Results for additional perturbations. Finally, consider Perturbations 1a, 1b, and 1c, summarized in Table 5. Each of these perturbations were run 60 times to observe Player 0 win/Player 1 win/draw ratios, and initial game frames from each of these perturbations were passed through the winner model to produce winner predictions. The results for these experiments, and a plot of winner predictions for sample game plays, are provided in Fig. 13. To gauge model performance for these perturbations, observed ratios of Player 0 win/Player 1 win/draw are reported, along with the corresponding model predictions at the first game frame and example winner prediction plots for a representative game play. This analysis can be used to verify the importance of features identified through SHAP analysis, and to check model performance for these new games.

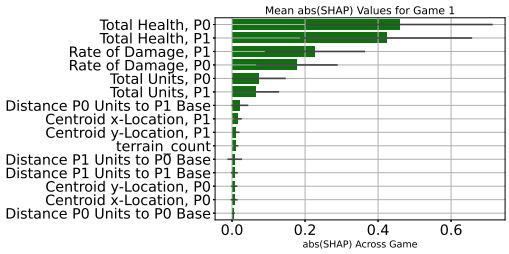

Fig. 12. Mean of absolute SHAP results for Perturbation series inputs with respect to predicting Player 0's win for a representative game play (lines represent 1 standard deviation).

Perturbations 1a and 1b were expected to improve Player 1's and Player 0's probability of win, respectively, which agrees with the most frequent winner observed in the 60 trial games corresponding to each perturbation. Note that according to the SHAP results, changes to damage output were expected to have a smaller impact on the win probability change relative to changes in health. This observation matches with results for the rankings of win probability provided by the model and the observed win ratios, although the difference in these predictions used to generate these rankings is quite small and requires further investigation through experimentation to validate the magnitude of the effect. Perturbation 1c altered P0's base location, and was anticipated to favor Player 0. In fact, Player 1 won more of the 60 trial games than Player 0, indicating an incorrect model response. This result may be due to a number of factors. First, the relative position of the base and resource nodes is also altered when the base location is changed. This may lead to differing amounts of time to gather resources and return them to the base to build additional units, and such a relationship is not captured by the current model. Second, differences in unit path logic, implemented by the bots used as player agents, may cause Player 1 units to reach the altered base location more quickly than Player 0's units can reach Player 1's base. In future iterations of the model, features capturing these signals may improve performance.

Thus, beyond informing design exploration, this SHAP analysis allows for more rapid surrogate model improvement at many levels: (i) model validation, (ii) model improvement through further data collection, and (iii) identification and validation of features driving system imbalance. The source of the proposed model's attention on base locations in the

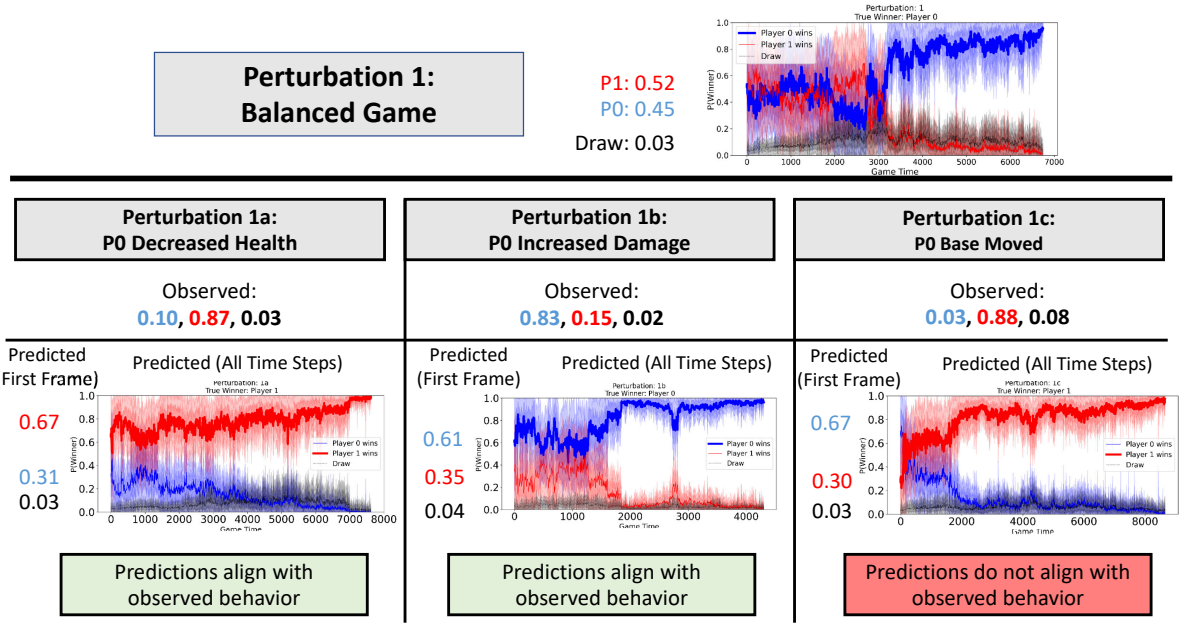

Fig. 13. Proposed counterfactual changes to the starting game state for the perturbation and the resulting predictions for perturbed starting game states. The player win and tie-game predictions for each perturbation and the true data for the same is presented. P0 Win - Blue; P1 Win - Red; Tie - Black.

example games requires further study, with additional experiments varying base location to validate this result and provide a higher-quality analysis of perturbations involving base location. This attention may be the result of a legitimately valuable signal for a player's probability of win coming from the relative base positions, but the current SHAP implementation must be expanded to give a more immediate way to study how these positions interact to generate more or less balanced games.

Closing the design loop using UQ and SHAP. Once the most sensitive parameters of a design environment are identified using XAI tools like SHAP, the designer can focus on making modifications to the game design based on these important features. While the importance of design features identified through SHAP would still need to be validated with the original simulation, the proposed method aims to identify good candidates efficiently (with SHAP) and identify regions of the game design space that are still underexplored (with uncertainty estimation using MCDN), so limited computational budget is not wasted on low-value game designs when running additional experiments. For the example we presented, health, position, and damage done by units are identified by the model as the most important features impacting the probability that a player wins. Furthermore, *optimal learning* [50] can be adopted to reduce the number of tests a designer performs when searching for higher-value game designs or generating more simulation data to further train the surrogate model. Incorporating optimal learning in our framework is planned for future work.

5 Conclusions

Motivated by the unique challenges inherent in Mission Engineering & Design (ME&D), we developed and demonstrated a framework to assist the mission design process. This framework incorporates an artificial intelligence (AI)-infused methodology and employs Real-time Strategy (RTS) games as a surrogate for real mission environments. The framework is used to construct a surrogate model of the mission environment to map relationships between mission design parameters and game balance/imbalance metrics (in this paper, probability of win) while managing the extensive design space and explaining the relationships between inputs and outputs to build trust with decision-makers. Overall, framework outputs allow mission designers to evaluate key mission outcomes, demonstrated in this paper via use of the RTS games. Though an RTS game is used for demonstration, the framework itself is generic and can be adapted for domain-specific complex simulation models for real-world applications.

A combination of Convolutional Neural Networks (CNN) and explainable AI (XAI) techniques produce predictions of game winner probability and the most influential factors on these probabilities, respectively. The resulting framework proved useful at uncovering game configurations and game play features that lead to game balance, as demonstrated by the perturbation analysis. The same information is used to suggest factors that could be most powerful to imbalance the game, which is the essential design objective for real mission engineering applications. Thus, the mission designers are better informed by XAI tools for decision making regarding specified objectives. Here, SHAP "closes the design loop" by informing the designer of the most important

game design features and their relative impacts on a prediction, and thereby provides an effective means to drive change in outcomes.

The proposed framework also includes methods to quantify uncertainty in the model predictions, essential not only to assess the confidence in the inferences but also to optimally learn the complex high-dimensional design space characteristic of ME&D problems. Thus, ongoing efforts to improve the framework include using tools like optimal learning to reduce the number of tests a designer has to perform to effectively map the design space. To make the framework more general, future work should also expand beyond self-play games. This work should include efforts to incorporate more complex information about the players themselves, including features that capture differing player skill or differences in strategies.

Acknowledgements

This report documents the work performed by a research team at Purdue University with the 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.

References

[1] Zimmerman, P., and Dahmann, D. J., 2018. "Digital Engineering Support to Mission Engineering". *21st Annual National Defense Industrial Association Systems and Mission Engineering Conference*, October, p. 22.

[2] West, C. T. D., and Birkmire, M. B., 2019. "Afsim". *WHAT WE OFFER,* **7**(3), p. 50.

[3] Hanlon, N., Garcia, E., Casbeer, D., and Pachter, M., 2018. "Afsim implementation and simulation of the active target defense differential game". In 2018 AIAA Guidance, Navigation, and Control Conference, p. 1582.

[4] Soleyman, S., and Khosla, D., 2020. "Multi-agent mission planning with reinforcement learning". In AAAI Symposium on the 2nd Workshop on Deep Models and Artificial Intelligence for Defense Applications: Potentials, Theories, Practices, Tools, and Risks.

[5] Bunyard, J., 2021. "Virtual training: Preparing future naval officers for 21st century warfare". *Center for International Maritime Security*.

[6] Doyle, S., 2020. "Gaming-defence. game of duty [wargaming]". *Engineering & Technology,* **15**(2), pp. 24–27.

[7] Goodman, J., Risi, S., and Lucas, S., 2020. "Ai and wargaming". *arXiv preprint arXiv:2009.08922*.

[8] Raina, A., McComb, C., and Cagan, J., 2019. "Learning to design from humans: Imitating human designers through deep learning". *Journal of Mechanical Design,* **141**(11).

[9] Bayrak, A. E., and Sha, Z., 2021. "Integrating sequence learning and game theory to predict design decisions under competition". *Journal of Mechanical Design,* **143**(5).

[10] Panchal, J. H., Sha, Z., and Kannan, K. N., 2017. "Understanding design decisions under competition using games with information acquisition and a behavioral experiment". *Journal of Mechanical Design,* **139**(9).

[11] Ren, Y., Bayrak, A. E., and Papalambros, P. Y., 2016. "ecoracer: Game-based optimal electric vehicle design and driver control using human players". *Journal of Mechanical Design,* **138**(6).

[12] Jaffe, A., Miller, A., Andersen, E., Liu, Y.-E., Karlin, A., and Popovic, Z., 2012. "Evaluating competitive game balance with restricted play". In Eighth Artificial Intelligence and Interactive Digital Entertainment Conference.

[13] Tomašev, N., Paquet, U., Hassabis, D., and Kramnik, V., 2020. "Assessing game balance with alphazero: Exploring alternative rule sets in chess". *arXiv preprint arXiv:2009.04374*.

[14] Jaffe, A. B., 2013. "Understanding game balance with quantitative methods". Thesis, University of Washington, July.

[15] Beyer, M., Agureikin, A., Anokhin, A., Laenger, C., Nolte, F., Winterberg, J., Renka, M., Rieger, M., Pflanzl, N., Preuss, M., and Volz, V., 2016. "An integrated process for game balancing". In 2016 IEEE Conference on Computational Intelligence and Games (CIG), pp. 1–8. ISSN: 2325-4289.

[16] Leigh, R., Schonfeld, J., and Louis, S. J., 2008. "Using coevolution to understand and validate game balance in continuous games". In Proceedings of the 10th annual conference on Genetic and evolutionary computation, GECCO '08, Association for Computing Machinery, pp. 1563–1570.

[17] Xia, W., and Anand, B., 2016. "Game balancing with ecosystem mechanism". In 2016 International Conference on Data Mining and Advanced Computing (SAPIENCE), pp. 317–324.

[18] Villar, S. O., 2020. microRTS - GitHub, June. original-date: 2015-03-12T18:16:53Z.

[19] Ontanón, S., and Buro, M., 2015. "Adversarial hierarchical-task network planning for complex real-time games". In Proceedings of the 24th International Conference on Artificial Intelligence, pp. 1652–1658.

[20] Ontanon, S., 2016. "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.

[21] Barriga, N. A., Stanescu, M., and Buro, M., 2017. "Game tree search based on nondeterministic action scripts in real-time strategy games". *IEEE Transactions on Games,* **10**(1), pp. 69–77.

[22] Stanescu, M., Barriga, N. A., Hess, A., and Buro, M., 2016. "Evaluating real-time strategy game states using convolutional neural networks". In 2016 IEEE Conference on Computational Intelligence and Games (CIG),

IEEE, pp. 1–7.

[23] Gunning, D., 2017. "Explainable artificial intelligence (xai)". *Defense Advanced Research Projects Agency (DARPA), nd Web,* **2**(2).

[24] Turek, M., 2018. "Explainable artificial intelligence (xai)". *Defense Advanced Research Projects Agency. http://web. archive. org/web/20190728055815/https://www. darpa. mil/program/explainable-artificial-intelligence*.

[25] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., García, S., Gil-López, S., Molina, D., Benjamins, R., et al., 2020. "Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai". *Information Fusion,* **58**, pp. 82–115.

[26] Das, A., and Rad, P., 2020. "Opportunities and challenges in explainable artificial intelligence (xai): A survey". *arXiv preprint arXiv:2006.11371*.

[27] Janzing, D., Minorics, L., and Blöbaum, P., 2020. "Feature relevance quantification in explainable ai: A causal problem". In International Conference on Artificial Intelligence and Statistics, PMLR, pp. 2907–2916.

[28] Ribeiro, M. T., Singh, S., and Guestrin, C., 2016. ""why should i trust you?" explaining the predictions of any classifier". In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 1135–1144.

[29] Binder, A., Montavon, G., Lapuschkin, S., Müller, K.-R., and Samek, W., 2016. "Layer-wise relevance propagation for neural networks with local renormalization layers". In International Conference on Artificial Neural Networks, Springer, pp. 63–71.

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

[31] Molnar, C., 2020. *Interpretable Machine Learning*. Lulu.com.

[32] Joung, J., and Kim, H. M., 2021. "Approach for importance–performance analysis of product attributes from online reviews". *Journal of Mechanical Design,* **143**(8), p. 081705.

[33] Lara-Cabrera, R., Cotta, C., and Fernández-Leiva, A. J., 2013. "A review of computational intelligence in rts games". In 2013 IEEE Symposium on Foundations of Computational Intelligence (FOCI), IEEE, pp. 114–121.

[34] Synnaeve, G., and Bessiere, P., 2015. "Multiscale bayesian modeling for rts games: An application to starcraft ai". *IEEE Transactions on Computational intelligence and AI in Games,* **8**(4), pp. 338–350.

[35] Richoux, F., 2020. "microphantom: Playing microrts under uncertainty and chaos". In 2020 IEEE Conference on Games (CoG), IEEE, pp. 670–677.

[36] Fernández-Ares, A., García-Sánchez, P., Mora, A. M., Castillo, P. A., and Merelo, J., 2016. "There can be only one: Evolving rts bots via joust selection". In European Conference on the Applications of Evolutionary Computation, Springer, pp. 541–557.

[37] Liu, S., Louis, S. J., and Ballinger, C., 2014. "Evolving effective micro behaviors in rts game". In 2014 IEEE Conference on Computational Intelligence and Games, IEEE, pp. 1–8.

[38] Fernández-Ares, A., García-Sánchez, P., Mora, A. M., and Merelo, J., 2012. "Adaptive bots for real-time strategy games via map characterization". In 2012 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, pp. 417–721.

[39] Garćıa-Sánchez, P., Tonda, A., Mora, A. M., Squillero, G., and Merelo, J., 2015. "Towards automatic starcraft strategy generation using genetic programming". In 2015 IEEE Conference on Computational Intelligence and Games (CIG), IEEE, pp. 284–291.

[40] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al., 2019. "Grandmaster level in starcraft ii using multi-agent reinforcement learning". *Nature,* **575**(7782), pp. 350–354.

[41] Ontanón, S., Synnaeve, G., Uriarte, A., Richoux, F., Churchill, D., and Preuss, M., 2013. "A survey of real-time strategy game ai research and competition in starcraft". *IEEE Transactions on Computational Intelligence and AI in games,* **5**(4), pp. 293–311.

[42] Tang, Z., Shao, K., Zhu, Y., Li, D., Zhao, D., and Huang, T., 2018. "A review of computational intelligence for starcraft ai". In 2018 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, pp. 1167–1173.

[43] Bayrak, A. E., McComb, C., Cagan, J., and Kotovsky, K., 2021. "A strategic decision-making architecture toward hybrid teams for dynamic competitive problems". *Decision Support Systems,* **144**, p. 113490.

[44] Chollet, F., et al., 2015. Keras.

[45] Kingma, D. P., and Ba, J., 2014. "Adam: A method for stochastic optimization". *arXiv preprint arXiv:1412.6980*.

[46] Gal, Y., and Ghahramani, Z., 2016. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning". In international conference on machine learning, pp. 1050–1059.

[47] Sicking, J., Akila, M., Wirtz, T., Houben, S., and Fischer, A., 2020. "Characteristics of monte carlo dropout in wide neural networks". *arXiv preprint arXiv:2007.05434*.

[48] Lundberg, S., 2021. A game theoretic approach to explain the output of any machine learning model., Jan. original-date: 2016-11-22T19:17:08Z.

[49] Erion, G., Janizek, J. D., Sturmfels, P., Lundberg, S., and Lee, S.-I., 2019. "Learning explainable models using attribution priors". *arXiv preprint arXiv:1906.10670*.

[50] Powell, W., 2012. *Optimal learning*. Wiley, Hoboken, New Jersey.

