Fusing Level and Ruleset Features for Multimodal Learning of Gameplay Outcomes

Antonios Liapis, Daniel Karavolos, Konstantinos Makantasis, Konstantinos Sfikas, Georgios N. Yannakakis
Institute of Digital Games, *University of Malta* , Msida, Malta
{antonios.liapis, daniel.karavolos, konstantinos.makantasis, konstantinos.sfikas, georgios.yannakakis}@um.edu.mt

***Abstract*—Which features of a game influence the dynamics of players interacting with it? Can a level's architecture change the balance between two competing players, or is it mainly determined by the character classes and roles that players choose before the game starts? This paper assesses how quantifiable gameplay outcomes such as score, duration and features of the heatmap can be predicted from different facets of the initial game state, specifically the architecture of the level and the character classes of the players. Experiments in this paper explore how different representations of a level and class parameters in a shooter game affect a deep learning model which attempts to predict gameplay outcomes in a large corpus of simulated matches. Findings in this paper indicate that a few features of the ruleset (i.e. character class parameters) are the main drivers for the model's accuracy in all tested gameplay outcomes, but the levels (especially when processed) can augment the model.**

***Index Terms*—Machine Learning, Convolutional Networks, Shooter Games, Level Design, Character Classes, Orchestration**

I. Introduction

Over the last 30 years, digital games have demonstrated that they can be rich and multifaceted experiences. While games such as *Rockband* (Harmonix, 2017) put particular emphasis on the game's soundtrack and reward gameplay that matches a rhythm, other games such as *Dear Esther* (The Chinese Room, 2012) reward exploration of a visually stunning world and discovery of snippets of stories scattered within it. Regardless of each game's individual focus, most games are a fusion of six facets: visuals, audio, game rules, levels, narrative, and gameplay [1], [2]. Out of those facets, the first five must be designed and carefully tuned before the game becomes available, while *gameplay* is reliant on a community of players. Gameplay includes a player's strategies to win, their understanding of the game's embedded narrative and moral code, and the emotions that are triggered by scripted or unscripted in-game events. One of the most challenging tasks for a game developer, therefore, is the creation of content (e.g. levels, visuals and rules) that are harmonious and complementary and also reward the intended gameplay style and elicit the intended players' emotions [1].

If the task of *orchestrating* game content of different facets towards intended player experiences is the most challenging task for a human designer, is it possible to empower computers to perform orchestration? Research in artificial intelligence (AI) and games has often attempted to measure the player experience either in terms of game progression and outcomes [3], [4] or in terms of emotional responses [5] and gameplay motivations [6]. Game analytics [7] collect metrics of real players in a released game or during beta-testing, while simulations with artificial agents are used when end-users are unavailable or when a large volume of generated game content must be evaluated [8]. While AI-based simulations are less demanding (in terms of both resources and time) than human evaluations, they still have a substantial computational overhead. Moreover, testing each and every aspect of a game (or minor tweak) via simulations does not match how human designers create content. In a typical design iteration, humans use their past knowledge of this game or other similar games, their own preferences and their own playstyle to estimate the player experience of the game content currently designed, which they then tweak to better align to their predictions of how end-users will perceive its quality.

This paper focuses on how a computational designer can predict gameplay qualities and outcomes of content, without testing it in simulations. The computational model accounts for "objective" metrics of gameplay traces such as game balance and match duration, and attempts to predict these outcomes from the initial state of the game, accounting for multiple facets in a multimodal fashion. While there is extensive work in detecting patterns within a corpus of similar or dissimilar types of content [9] via unsupervised learning, the proposed model aims to match patterns across facets to specific gameplay outcomes, via supervised multimodal learning. Moreover, while work on affective computing in games [10] focuses on predicting players' emotions and other personal aspects of the player experience from human playtraces, this paper targets less subjective gameplay properties based on analytics and, more importantly, predicts gameplay outcomes from the initial designed game assets without actually playing the game.

The experimental method followed by this paper is largely exploratory, using a corpus of shooter game playtraces [11], [12] to find which features of two facets (levels and rules) can best predict quantifiable gameplay outcomes. Using a corpus of $2 \cdot 10^5$ simulated matches in a competitive two-player shooter game developed specifically for this line of research, a deep learning architecture is tested using levels and the two players' *character classes* as input, and one gameplay metric as output. Character classes are player roles chosen by the players before a game starts, and differentiate the avatar's weapon and attributes. Different character classes allow for different player strategies and playstyles, while in competitive e-sports the balance between classes must be constantly monitored based on the meta-game and corrected with patches. By modifying aspects of the ruleset (as parameters of character classes are constant) and the level, the computational models should be able to predict aspects of the gameplay facet based on a multimodal fusion of the level and rules facets, acting as a *surrogate* of cumbersome playthroughs.

This paper extends the work of [11]–[13] on surrogate-based procedural generation, focusing instead on the qualities of the surrogate model rather than on the generation of levels or character classes. This paper explores the impact of different inputs—especially level representations—on the predictive power of the deep-learning models. The two gameplay outcomes (kill ratio, match duration) of earlier work are augmented with measures of entropy on playtraces' heatmaps. Finally, this paper demonstrates how the models can explain their choices and aid the designer through visualizations on any set of player classes and level being designed.

II. Background Work on Deep Learning

Conventional machine learning methods are limited in their ability to process raw data. Shallow pattern recognition is a two-step process: (a) transform raw data to a suitable representation via a handcrafted feature construction process based on expert knowledge; (b) learn to make decisions based on the transformed data. In many cases, however, it is not known a priori which data representation is most appropriate for a learning task. Deep learning methods learn representations directly from the raw data via the composition of simple but nonlinear data transformations, thus bypassing any handcrafted feature construction process. Very complex functions can be learned by combining enough transformations, evidenced by the success of deep learning in visual recognition [14], natural language processing [15] and agent control [16].

Convolutional neural networks (CNNs) are deep learning models designed to process data that come in the form of multidimensional arrays and that exhibit spatially structured information, such as audio spectrograms, images and videos. While CNNs have a longer history, their success in the ImageNet competition [14] has made CNNs the dominant approach for almost all visual detection and recognition tasks [17], [18].

Deep learning has also been applied in multimodal learning tasks [19]. Multimodal learning involves relating information from multiple and, usually, heterogeneous information sources [20]. Multimodal learning exploits information fusion methods based either on feature-level fusion (early fusion) or score-level fusion (late fusion). In early fusion, information of different modalities is aggregated via simple element-wise averaging, product and/or concatenation [21]. In late fusion, high level representations for each modality are computed separately and then fused together via simple averaging or by stacking another learning model [22]–[24].

Deep learning has been applied in games for a broad variety of goals, such as agent decision-making based on an abstraction of the game state [25] or using the actual screen's pixels as input [16], [26]. In affective computing, deep

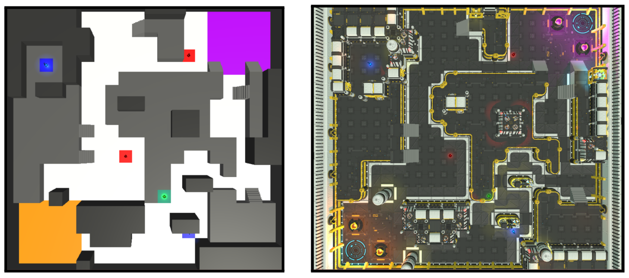

(a) 3D Level (b) In-game view

Fig. 1: Example level. In the 3D level, orange and purple areas are the bases of player 1 and 2 respectively. Red tiles are healing locations, blue and turquoise tiles are armor and double damage powerups respectively.

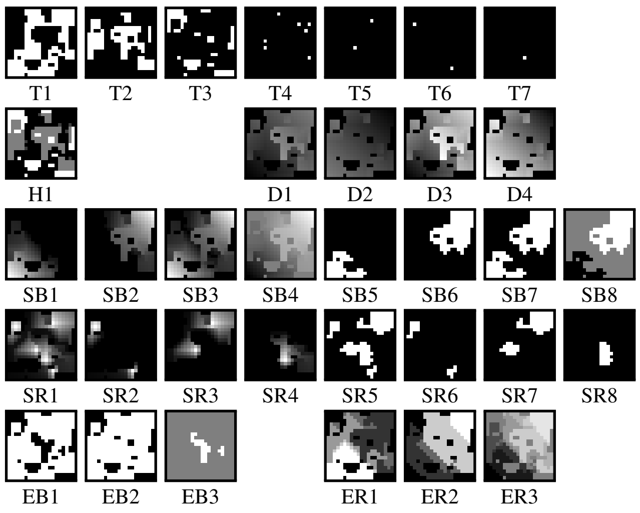
learning is often applied to predict the self-reported emotional states of players. For instance, gameplay and physiological data were combined as inputs to a CNN which predicted self-reported preferences [27], while in [28] gameplay logs (processed via a CNN) and the level (processed via another CNN) were fused in a final fully connected layer to predict players' feedback such as challenge or levels' creativity. In *procedural content generation via machine learning* (PCGML) [9], human-authored levels are often used as a corpus and the trained model decides which tile should be placed at some position based on which tiles precede it, e.g. via Long-Short Term Memory networks [29] or generative adversarial networks [30]. The line of research followed in this paper diverges from work in PCGML as it follows a supervised learning approach towards game outcomes computed via simulations; moreover, generation [11], [12] attempts to optimize the predicted gameplay outcome rather than exploiting learned patterns or the latent space.

III. Problem Description

As noted in the introduction, we use a first-person shooter game with typical deathmatch gameplay as the target of this study. Every match takes place in a level, with two players each controlling an avatar that competes for the most kills of their opponent. Every time an avatar's hit points (HP) reaches 0, the opponent scores a kill and the avatar respawns at the player's base. When the total number of kills reaches 20, the game ends and the player with the most kills is the winner.

The game is played in a level consisting of multiple floors. Players' bases are in opposite corners of the map; players start the game in their base and re-spawn there when killed. The level has walls which block movement and vision, and an elevated floor (first floor) which is only accessible from the ground floor via stairs. Avatars can jump from any first-floor tile to an adjacent ground-floor tile, and enjoy some cover from their elevated position. Therefore, staying on the first floor offers some tactical advantages such as a sniper location [31]. Any tile on the first or ground floor may have one of three types of powerups. The types of powerups are *healing* (which increases the avatar's hit points up to a maximum),

TABLE I: Input Channels: binary channels per tile type (T1-7) and heightmap (H1), distance from bases (D1-4), safety metric before and after high-pass filter between bases (SB1-8) and between powerups (SR1-8), exploration from base to base (EB1-3) and from base to resources (ER1-3).



armor (which adds temporary hit points to the avatar which are depleted first), and *double damage* (which doubles the player's weapon damage for a limited amount of time). When an avatar collides with a powerup, they receive its effect and this powerup can not be reused until some time has passed. The powerups can increase the survivability (healing, armor) or the threat (double damage) of the avatar.

The two competing avatars belong to different *character classes*, which are a common way of defining roles in shooter games such as *Team Fortress 2* or TF2 (Valve, 2007). Character classes in this testbed define the avatar's hit points and movement speed, as well as their weapon's characteristics. Six weapon parameters are included: damage (per bullet), accuracy (i.e. the size of the cone in which bullets are fired), rate of fire, clip size, the number of bullets per shot and weapon range. Weapon range discerns when AI agents should shoot, and is used during simulations (see Section III-A). There are three range values: "short", "medium" and "long".

A. Corpus Preparation

This paper explores the successes and failures of different machine learning models on a corpus of $2 \cdot 10^5$ data points. Each data point represents the simulation of one level with two character classes, and its gameplay outcomes. To produce the corpus, 10^5 shooter levels were generated via a combination of digger agents and cellular automata [32]. Every level in the corpus consists of 20×20 tiles, split into *cells* of 4×4 tiles for the purposes of the initial path generation and powerup placement. Details of the constructive algorithms used to generate the levels are in [11]. Each level in the corpus was tested by two character classes with random parameters within $[0, 1]$; these were normalized based on intuitive value ranges of classes in TF2, and are re-mapped to the original value range when performing the simulations. The generated pair of classes are tested in two simulations per level, swapping the bases of each character class in the second simulation. Simulations are performed by artificial agents controlled by behavior trees, adapted from the framework of [33]. The resulting dataset is $2 \cdot 10^5$ matches; it contains only matches that ended successfully with 20 kills within 600 seconds.

B. Input

The goal of this paper is to create a surrogate model based on machine learning, which can predict the gameplay outcomes based on a level and aspects of the ruleset (in this case players' character classes). The level and the class parameters are therefore the inputs to the model. Each player's class consists of 8 parameters as detailed above, and the parameters used as input to the network are normalized to $[0, 1]$ as detailed in Section III-A. Examples of archetypal classes of TF2 normalized to the value range used are in Table IV. Unlike previous work [11]–[13] which tested different network architectures while the input and output remained the same, this paper expands the types of inputs for levels and explores how they impact learning.

Levels can be parsed based on each individual tile's elevation and type (i.e. splitting ground-floor tiles, first-floor tiles, walls, stairs, healing, armor, double damage powerups) to produce 7 binary channels[1]. A simple addition to these channels is a "heighmap" which shows walls as 1, first-floor tiles as 0.5, ground floor tiles as 0 and stairs as 0.25 (i.e. between the first and ground floors). Four channels compute the distance of each tile from the base of one player (taking into account connectivity between first-floor and ground-floor and use of stairs). Per player, the path distance is normalized to an ad-hoc constant (80 tiles) or via min-max normalization, resulting in two channels per player. The ad-hoc constant normalization accentuates differences between players' distances and distinguishes between levels with long paths and levels with short paths. The min-max normalization shows more clearly which tile on the level is the furthest from one base. Based on relative proximity of each tile to each base, its *safety* score to that base can be computed: this metric is a real number within $[0, 1]$ and has non-zero values when tiles are much closer to one tile than another tile of the same type [34]. The safety matrix per base, and simple addition and subtraction operators on these matrices make up 4 channels, while binary versions of these matrices with safety values above an ad-hoc threshold of $C_S = 0.35$ [34] make up another 4 channels. Safety can also be calculated using the powerups as reference points, where safety channels per powerup have non-zero values if a tile is much closer to this powerup than to any other powerup. The aggregated version of these powerup safety channels (for all powerups, for healing only, for armor only and for damage only) make up 4 channels, and the same channels after thresholding values above $C_S = 0.35$

[1]While [11], [12] considered an 8th channel (reserved for "cover" tiles), we do not consider it in this study as it consists of zeros in the current corpus.

make up another 4 channels. Finally, exploration is measured based on a flood fill algorithm starting from one player's base until the other base is discovered [34]. This results in 3 channels (exploration from the player 1 base, exploration from the player 2 base, and their difference). The same process is repeated, averaging the exploration matrices from a player's base until a certain powerup is covered: this makes up another 3 channels (one per base and their difference). These different ways of processing the game levels result in 34 channels in total (see Table I), out of which 16 are binary and 18 have real values within $[0, 1]$.

In addition to these visualizations of level quality, a number of summary statistics can be calculated from the channels of Table I. Some of these statistics are introduced in [34], such as the average safety score of different powerups to one player's base or the exploration from one base to the other. Other metrics occur by simply adding all values in matrices of Table I, e.g. the number of armor powerups from the sum of matrix T6. Metrics are normalized to $[0, 1]$ in a reasonable fashion, e.g. the number of ground floor tiles is divided by 400 tiles in the level, the number of armor powerups is divided by 16 since the level generator places one powerup (if any) per cell, and the number of explored tiles is divided by the number of passable (non-wall) tiles. Among the statistics computed, the number of tiles of each type are computed in each of the four quadrants of the level, as well as the middle quadrant. In total, 90 summary statistics for the level (*level stats* for brevity) are computed; their impact on the predictive model (with or without level channels) is tested in Section IV.

C. Gameplay Outcomes

An important outcome of a playthrough is the winner of the match and the degree of winning; both are measured via a *kill ratio* (KR) as the number of kills of player 1 divided by the total number of kills of both players (i.e. 20 in these simulations). If KR≈0.5 then the matchup was balanced, while KR near 1 shows a clear advantage for player 1 and KR near 0 show a clear advantage for player 2. The duration of the match (t) is another intuitive metric, as a designer may wish for some matches to last longer. These gameplay outcomes have been used in past work as prediction targets and as targets for adapting character classes [12] and levels [11] towards balanced matches of a short, medium or long duration.

This paper explores two additional gameplay outcomes, focusing on the spatial behavior of players based on *heatmaps*. Specifically, the heatmaps of players' deaths and movement trajectories are used to compute an entropy score. The heatmap of the players' deaths consists of 16 cells on a 4×4 grid: each cell stores the number of deaths that occurred within the 5×5 tiles of that cell. Similarly, the heatmap of players' movement is 16 cells and each cell stores how many times each player was within that cell (their position is captured every second). With 16 values per heatmap, the entropy is calculated as:

$$H = -\frac{1}{\log_2 N} \sum_{c=1}^{N} \frac{p_c}{P} \log_2 \frac{p_c}{P} \tag{1}$$

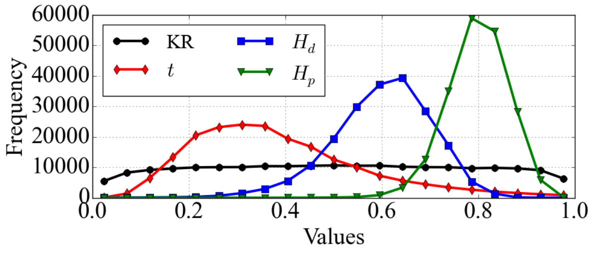


Fig. 2: Distributions of the four gameplay outcomes.

where N is the number of cells in the heatmap, p_c is the number of entries in a specific cell, and P is the total number of entries in the heatmap ($P = \sum_{c=0}^{C} p_c$).

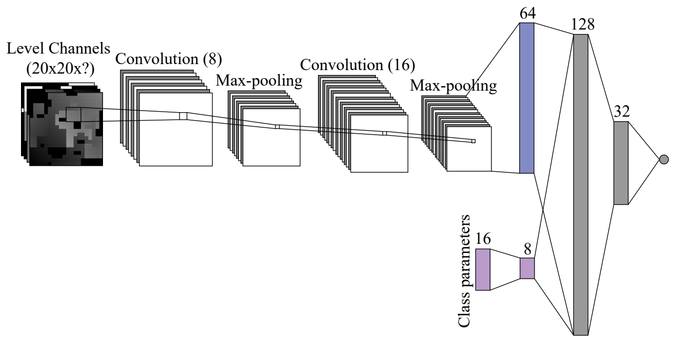
This results in two entropy scores: one for all death locations (H_d) and one for players' positions (H_p). These entropy scores do not distinguish between the two players; while it could be worthwhile to measure the entropy of positions of player 1 and even compare it with that of player 2, the aggregated entropy scores offer a more holistic view of the game's state.

D. Collected Data

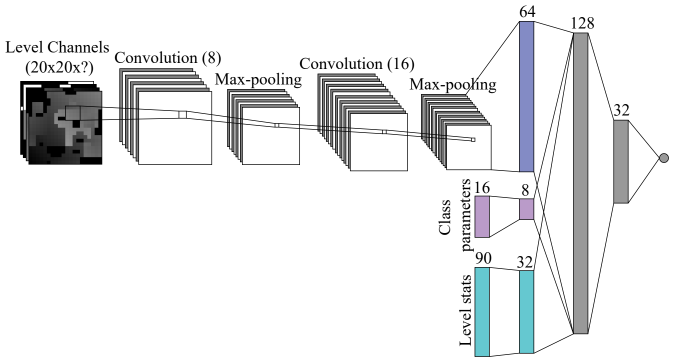
Based on Section III-C, four gameplay outcomes are considered for each simulated match. Since matches last between 150sec and 600sec, the t value is min-max normalized between $[150, 600]$. The distributions of these gameplay metrics in the corpus are presented in Fig. 2. While the KR metric is almost uniformly spread (dipping slightly in edge cases where a player won by a landslide), the distribution of t is skewed towards shorter matches, with an average of 323sec ($t = 0.385$). Finally, the distributions of H_d and H_p are skewed towards high values; this means that positions of deaths or player movements occurred within a few cells and are not spread uniformly.

E. Network Architecture

The network architecture is largely based on previous work [11], [12], [35], but its hyperparameters were adapted based on extensive preliminary experiments. The network follows a late fusion multimodal learning approach, with two or three separate information streams (see Fig. 3): one for the level, one for the pair of character classes, and an optional stream for the level stats. The level (parsed into binary or real-valued channels, as per Table I) is processed by two blocks of convolution (of size 5×5 without zero-padding) and max-pooling, with 8 and 16 filters respectively. The end-result of these convolutions is a flat vector of 64 features for the level. The 16 parameters of the character classes are passed to a single fully-connected layer of 8 nodes, the output of which is concatenated to the flat feature vector of the level. Finally, this combined feature vector (72 nodes) connects to a fully connected layer of 128 nodes which connects to another layer of 32 nodes, which then connects to one output that predicts one gameplay outcome (see Section III-C). In Fig. 3b, an alternative architecture which includes the level stats as input is shown, where the 90 level stats are passed as a vector to



(a) Architecture without level stats



(b) Architecture with level stats

Fig. 3: The surrogate model architecture, with two or three information streams: the level's channels (2D), the two character classes' parameters (vector), and in Fig. 3b also the level stats (vector). The model's output is one gameplay outcome.

a separate layer of 32 nodes which is concatenated with the hidden nodes from level channels and from class parameters. All nodes use an ELU activation function [36]. The output of each hidden layer is normalized via batch normalization [37].

IV. Results

Experiments in this paper explore how different channel sets and the inclusion of different information streams (such as the level stats) affect the performance of the regression-based predictive model. Finally, the paper suggests how the models can be used for visualization and designer assistance in a possible AI-assisted design tool [38]. This paper uses two accepted performance metrics for regression: (a) the mean absolute error, which measures the difference between predicted and actual gameplay output and (b) the R^2 metric (with typical ranges of $[0, 1]$) which measures how much of the variance in the data is explained by the model. All models in this section were trained for 30 epochs, while early stopping was used to prevent over-fitting. Reported results are averaged from 10 trials using the same 10% of the data for validation (using a hold-out procedure). Significant findings are based on the non-parametric Mann-Whitney U test with $\alpha = 0.01$.

A. Impact of Combined Level Channels

In order to explore how combinations of levels and weapons can better predict the different gameplay outcomes, the channels of Table I are used in full or in different subsets. Since earlier work [11] used only the naive tile-based channels T1 to T7 (notation henceforth T1-7), their performance will be used as a baseline. Different sets of channels were chosen based on authors' intuition and ranged from few core channels to the full set of Table I. The following sets of channels are reported:

$S1$	T1-7 (7 channels)
$S2$	D1-2, H1, SR1 (4 channels)
$S3$	D1-2, H1, SR1, ER1-2, EB3, SB4 (8 channels)
$S4$	D1-2, H1, SB3-4, SB7-8, SR2-4, SR6-8, EB1-3, ER3 (17 channels)
$S5$	all 34 channels of Table I
$S6$	T1-7, D1-2, H1, SR1 (11 channels)

Out of these channel sets, $S5$ and $S6$ include all channels of $S1$ (i.e. the original channels used as a baseline). $S2$, $S3$ and $S4$ have non-binary channels which do not include any of the channels in $S1$. The set $S6$ is the union of $S1$ and $S2$.

Table II shows the performance of the computational models for different sets of inputs and the four different game outcomes. While we treat $S1$ (without level stats) as the *baseline* used in past work, the table includes the indicative performance of a simple linear regression model which concatenates all 34 level channels ($S5$) and class parameters. Table II indicates that the naive binary channels used as a baseline ($S1$) are surprisingly powerful in predicting all gameplay outcomes, although for both entropy measures (H_d, H_p) the low R^2 indicates overfitting to the most common occurrences in the distribution. While small sets of channels chosen specifically for the rich information they contain ($S2$, $S3$) underperform rather substantially compared to the baseline, sets which combine $S1$ with other channels (i.e. $S5$, $S6$) significantly outperform the baseline in several target outcomes. The best performance is with $S5$ which combines all 34 channels of Table I, with a relative increase in terms of MAE of 2.2% for KR, 3.4% for t and 3.4% for H_p. It is surprising that the smaller channel set of $S6$ performs equally or better than the larger set of $S4$ (which however does not contain the channels of $S1$), which is further evidence that the information of $S1$ seems invaluable in predicting gameplay outcomes from levels. For remaining experiments in this paper, only the $S5$ channels will be explored (along with $S1$ as a baseline) as it is shown to be superior. When combining the level channels with quantifiable level stats, the models become more accurate than their respective ones without stats (using the same channels). The model with $S5$ channels and level stats has the best overall accuracy, especially for match duration (4.8% lower MAE and 5.2% higher R^2 than the baseline) and H_d (1.4% lower MAE and 6.4% higher R^2 than the baseline).

B. Individual Impact of Levels and Character Classes

It is worthwhile to understand which streams of information (the level stats, the full level channels, or the character class

TABLE II: Validation results for different inputs and outputs. Values significantly better (lower for MAE, higher for R^2) than the baseline ($S1$) are in bold. LR is linear regression.

	KR		t		H_d		H_p	
	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2
Without level stats (CNN of Fig. 3a)								
LR	0.127	0.690	0.131	0.166	0.092	0.006	0.0640	0.0021
S1	0.069	0.910	0.082	0.605	0.079	0.312	0.0340	0.555
S2	0.073	0.900	0.086	0.570	0.086	0.198	0.0398	0.410
S3	0.072	0.901	0.086	0.572	0.086	0.198	0.0400	0.405
S4	0.068	0.912	**0.080**	**0.621**	0.079	0.316	**0.0336**	**0.565**
S5	0.067	0.912	**0.079**	**0.627**	0.079	**0.323**	**0.0328**	**0.585**
S6	0.068	0.911	**0.080**	**0.624**	0.079	**0.32**	**0.0334**	**0.570**
With level stats (CNN of Fig. 3b)								
S1	**0.066**	0.914	**0.079**	**0.627**	**0.078**	**0.327**	**0.0331**	**0.578**
S5	0.067	0.914	**0.078**	**0.636**	**0.078**	**0.332**	**0.0327**	**0.587**

TABLE III: Validation results when streams of information are omitted or set to zero.

	KR		t		H_d		H_p	
	MAE	R^2	MAE	R^2	MAE	R^2	MAE	R^2
Character classes only (CNN of Fig. 3a)								
—	0.074	0.896	0.201	0.122	0.093	0.080	0.0535	0.034
Level channels only (CNN of Fig. 3a)								
$S1$	0.259	0.005	0.110	0.356	0.085	0.210	0.0366	0.487
$S5$	0.259	0.004	0.110	0.354	0.085	0.209	0.0365	0.487
Level stats only (CNN of Fig. 3b)								
—	0.257	0.012	0.113	0.326	0.086	0.186	0.0390	0.429
Level channels and level stats only (CNN of Fig. 3b)								
$S1$	0.258	0.011	0.109	0.368	0.084	0.219	0.0360	0.503
$S5$	0.258	0.012	0.109	0.367	0.084	0.217	0.0360	0.502
Level stats and character classes only (CNN of Fig. 3b)								
—	0.067	0.912	0.085	0.579	0.081	0.288	0.0364	0.502

parameters) contribute most to the models' predictions. To do this, we first test performance metrics for regression when the same architecture as in Section IV-A is trained on a modified corpus without certain streams of information. For example, the impact of character classes is tested by using the architecture of Fig. 3a or Fig. 3b and setting the character class vector as zero values. In this case, the prediction is made based on the level channels only or on the level channels and level stats only, respectively. Table III presents the MAE and R^2 metrics when different streams are omitted. While any combination of level channels and level stats seems unable to create useful predictive models for KR, the character classes are quite capable of predicting KR but are perform very poor when predicting heatmap metrics. The winner of a match seems to be primarily determined by the classes of the competing players while the level can only do that much to affect power differences. When level stats and class parameters are used without level channels, both MAE and R^2 values are comparable to the baseline ($S1$).

Further investigating the impact of individual character class parameters can also be informative. In this paper, the F statistic is used to test the significance of groups of character class parameters. It employs a linear regression model fitted on all 16 class parameters and a second linear regression model fitted on a subset of class parameters obtained after omitting one parameter from both classes (e.g. the HP of both player 1 and 2). The F statistic [39] is used to compute the probability (p values) for rejecting the null hypothesis that this class parameter has no impact on the sum squared error of the least squares fit of the linear regression model. Based on this method, we find significant effects ($p < 0.05$) between the same five class parameters in all gameplay outcomes: HP, damage, accuracy, number of bullets and range (also rate of fire for H_d). It is surprising that speed is not a strong factor for match duration, but otherwise the findings for the F statistic make sense, at least for KR and t. This analysis shows that while character classes' parameters or their non-linear combination can be a robust predictor for some gameplay outcomes, the level contains vital information that leads to much better predictive behavior when combined.

C. Explainability of the Model

As the success of machine learning applications grows, so does the call for explainable AI, e.g. visualizing the inner workings of black-box machine learning methods [40]–[42]. Understanding the decisions of an AI system makes it more accessible and allows designers to better explore the creative potential of such systems [38]. This can be achieved by showing which parts of the image have the most influence on the model's prediction, e.g. via Gradient-weighted Class Activation Mapping (grad-CAM) [43]. This method computes the gradient of an output node with respect to the nodes of a convolutional layer, given a particular input. By multiplying the input with the gradient, averaging over all nodes in the layer and normalizing the resulting values, we obtain a heatmap that shows how much each area of the input contributed to increasing the value of the output node. While grad-CAM is typically used in classification tasks, it also works for regression-based models which deal with floating point predictions; in that case, both the positive and the negative contributions are important.

As an example of how CAM can be applied to this particular problem, Fig. 4 shows the activation heatmap for the level of Fig. 1 in two matchups with archetypal classes of Team Fortress 2: Sniper versus Scout, and Heavy versus Scout (see the parameters of each class in Table IV). The 4×4 heatmaps show the last convolutional layer of Fig. 3a with $S5$ channels. Differences between e.g. Fig. 4a and Fig. 4e show that the class pairing affects the activation heatmap on the level, highlighting how the model acknowledges the interactions between the two facets. Red areas in Fig. 4 show which parts of the level lead to a lower value in the gameplay outcome and blue areas show which parts lead to a higher value. For example, Fig. 4a shows that the kill ratio of player 1 (Sniper) is negatively affected by the labyrinthine narrow paths of the right half of the level. Essentially, high activations (positive or negative) can tell a designer which part of the level to tweak.

The same principle can be applied to the class parameters: by computing the gradient of the output with respect to the input layer.[2] For the Sniper versus Scout matchup, the class pa-

[2]In this case, values are not normalized or averaged over all nodes in the layer as this would result in a single contribution value for all class parameters.

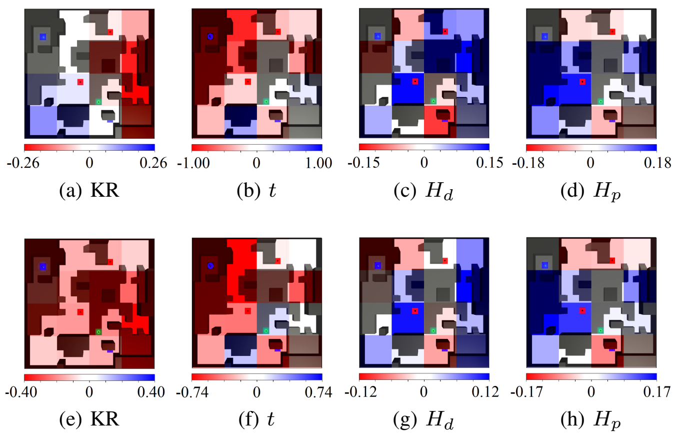


Fig. 4: Activation Maps: top row for the Sniper versus Scout matchup and bottom row for the Heavy versus Scout matchup.

TABLE IV: Normalized parameters for TF2 classes, used as an example for this paper (raw values in parentheses).

Normalized	Scout	Heavy	Sniper
HP	0.00 (125)	1.00 (300)	0.00 (125)
Speed	1.00 (133)	0.00 (77)	0.41 (100)
Damage	0.21 (60)	0.00 (36)	1.00 (150)
Accuracy	0.80 (80%)	0.60 (60%)	1.00 (100%)
Clip size	0.03 (6)	1.00 (200)	0.12 (25)
Rate of Fire	0.03 (1.6)	1.00 (40)	0.00 (0.67)
Bullets	1.00 (10)	0.00 (1)	0.00 (1)
Range	0.00 (short)	0.00 (short)	1.00 (long)

rameters activated (at 0.10 or above) are the Sniper's accuracy (-0.27 for t, 0.92 for KR) and the Scout's accuracy (-0.57 for KR). These numbers tell the designer that for longer matches a drop in the Sniper's accuracy is needed; for lower kill ratios for the Sniper, an increase in the accuracy of the Scout and a decrease in the accuracy of the Sniper are needed. These class parameter activations are even more straightforward to process and rationalize than the level heatmaps and can thus be a valuable tool. For completeness, in the matchup between Heavy and Scout the activated parameters (at 0.10 or above) are the Scout's accuracy (-0.32 for t, -1.20 for KR), the Heavy's HP (0.28 for KR), the Heavy's accuracy (0.55 for KR), and the Scout's damage (-0.12 for KR).

V. Discussion

The experiments in this paper were largely exploratory, in order to assess which representations of the level can best predict specific gameplay outputs. Findings indicate that a more holistic view of the level can lead to superior predictive models based on regression, although some naive binary channels are surprisingly effective in this task as well. Combining 2D representations with summary level statistics and classes wields more powerful models, and removing any of the three streams of information lowers the overall accuracy. The 2D level representation also offers useful feedback as a grad-CAM heatmap which can be integrated with a mixed-initiative tool [44], also because the system can predict gameplay outcomes of unseen, work-in-progress levels in milliseconds.

Obviously, this exploratory process of different datasets and models did reveal a number of pitfalls and limitations. On the one hand, low R^2 and low MAE statistics in some of the models point to overfitting. This can be explained from the skewed distribution of the data, especially when comparing the KR and the H_d distributions in Fig. 2 and their respective R^2 scores. Another reason for this behavior may be because the corpus contains samples that are very similar in terms of input features, but different in terms of output values. In such cases, a learning model must learn to map the same input to different outputs. The stochastic nature of playthroughs exacerbates this problem as the same level/class pair may result in different gameplay outputs.

On that topic, while the artificial agents are able to compete (winning or losing within the 5 minutes in 94% of cases), they navigate the level largely based on the location of powerups. In maps with few powerups, agents are expected to converge to the same locations and kill each other around those areas. On the other hand, human players are more likely to exploit areas that lack powerups but have tactical advantages, such as elevated platforms or cover due to walls. Therefore, a more elaborate AI agent or even human playtraces may lead to a more varied corpus and perhaps models that can learn it better.

Beyond the limitations in this paper's methods, several directions can be pursued in future work. Specifically, more complex gameplay outcomes as outputs could be used, such as the heatmap of players' movement (rather than its aggregated entropy score). A computational model which directly predicts a heatmap of an unseen level for two competing character classes could be beneficial both during the design process and as a cheap game analytics tool [7]. The models could be improved if the 2D level representations are transformed automatically into a lower-level representation (similar to the level stats), e.g. via an autoencoder. Finally, a more ambitious effort could test whether a broader corpus of dissimilar levels and dissimilar gameplay styles (such as team-based competition) could be used to train more *general* surrogate models that can predict gameplay in completely unseen games.

VI. Conclusion

This paper explored how different representations of the level facet (as more nuanced 2D visualizations and as summary statistics) can impact a surrogate model of gameplay which can predict the outcomes of a playthrough without simulating it. Focusing on shooter games and an extensive corpus which has been used for procedural content generation [11], [12], the trained models can fuse two facets of games (levels, rules) to predict a third facet (gameplay). Level summary statistics, but also a holistic set of level visualizations, prove to be effective predictors of four gameplay outcomes, although the character class parameters seem to be most important. Trials with explaining the impact of levels and rules on the output highlight how the models can become designer support tools.

VII. Acknowledgements

This project has received funding from the European Union's Horizon 2020 programme under grant agreement No 787476.

References

[1] A. Liapis, G. N. Yannakakis, M. J. Nelson, M. Preuss, and R. Bidarra, "Orchestrating game generation," *IEEE Trans. on Games*, vol. 11, no. 1, pp. 48–68, 2019.

[2] A. Liapis, G. N. Yannakakis, and J. Togelius, "Computational game creativity," in *Proc. of the intl. conf. on Computational Creativity*, 2014.

[3] P. L. Lanzi, D. Loiacono, and R. Stucch, "Evolving maps for match balancing in first person shooters," in *Proc. of the IEEE conf. on Comp. Intell. and Games*, 2014.

[4] D. Gravina and D. Loiacono, "Procedural weapons generation for Unreal Tournament III," in *Proc. of the IEEE conf. on Games, Entertainment, Media*, 2015.

[5] K. Karpouzis and G. N. Yannakakis, *Emotion in Games: Theory and Praxis*. Springer, 2016.

[6] A. Azadvar and A. Canossa, "UPEQ: Ubisoft Perceived Experience Questionnaire: a self-determination evaluation tool for video games," in *Proc. of the conf. on Foundations of Digital Games*, 2018.

[7] M. S. El-Nasr, A. Drachen, and A. Canossa, *Game analytics*. Springer London Limited, 2016.

[8] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Trans. on Comp. Intell. and AI in Games*, vol. 3, no. 3, 2011.

[9] A. Summerville, S. Snodgrass, M. Guzdial, C. Holmgård, A. K. Hoover, A. Isaksen, A. Nealen, and J. Togelius, "Procedural content generation via machine learning (PCGML)," *IEEE Trans. on Games*, vol. 10, no. 3, 2018.

[10] H. P. Martínez and G. N. Yannakakis, "Mining multimodal sequential patterns: A case study on affect detection," in *Proc. of the 13th intl. conf. on Multimodal Interfaces*, 2011.

[11] D. Karavolos, A. Liapis, and G. N. Yannakakis, "Using a surrogate model of gameplay for automated level design," in *Proc. of the IEEE conf. on Comp. Intell. and Games*, 2018.

[12] ——, "Pairing character classes in a deathmatch shooter game via a deep-learning surrogate model," in *Proc. of the FDG workshop on PCG in Games*, 2018.

[13] ——, "A multi-faceted surrogate model for search-based procedural content generation," *IEEE Trans. on Games*, 2019, accepted.

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[15] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of the intl. conf. on learning representations*, 2014.

[18] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. of the IEEE conf. on computer vision and pattern recognition*, 2016.

[19] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," in *Advances in neural information processing systems*, 2012, pp. 2222–2230.

[20] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *Proc. of the intl. conf. on machine learning*, 2011, pp. 689–696.

[21] E. Park, X. Han, T. L. Berg, and A. C. Berg, "Combining multiple sources of knowledge in deep cnns for action recognition," in *Proc. of the IEEE Winter conf. on Applications of Computer Vision*, 2016.

[22] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[23] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[24] V. Vukotić, C. Raymond, and G. Gravier, "Multimodal and crossmodal representation learning from textual and visual features with bidirectional deep neural networks for video hyperlinking," in *Proc. of the ACM workshop on Vision and Language Integration Meets Multimedia Fusion*, 2016, pp. 37–44.

[25] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

[26] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaskowski, "ViZDoom: A Doom-based AI research platform for visual reinforcement learning," in *Proc. of the Computational Intelligence in Games conf.*, 2016.

[27] H. P. Martínez and G. N. Yannakakis, "Deep multimodal fusion: Combining discrete events and continuous signals," in *Proc. of the ACM intl. conf. on multimodal interaction*, 2014.

[28] N. Liao, M. Guzdial, and M. Riedl, "Deep convolutional player modeling on log and level data," in *Proc. of the conf. on Foundations of Digital Games*, 2017.

[29] A. Summerville and M. Mateas, "Super mario as a string: Platformer level generation via LSTMs," in *Proc. of DiGRA & FDG*, 2016.

[30] V. Volz, J. Schrum, J. Liu, S. M. Lucas, A. Smith, and S. Risi, "Evolving mario levels in the latent space of a deep convolutional generative adversarial network," in *Proc. of the Genetic and Evolutionary Computation conf.*, 2018.

[31] K. Hullet and J. Whitehead, "Design patterns in FPS levels," in *Proc. of the conf. on Foundations of Digital Games*, 2010.

[32] N. Shaker, A. Liapis, J. Togelius, R. Lopes, and R. Bidarra, "Constructive generation methods for dungeons and levels," in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016, pp. 31–55.

[33] Opsive, "Deathmatch AI kit documentation," https://legacy.opsive.com/assets/DeathmatchAIKit/, 2016, accessed: 2019-03-24.

[34] A. Liapis, G. N. Yannakakis, and J. Togelius, "Towards a generic method of evaluating game levels," in *Proc. of the AAAI Artificial Intelligence for Interactive Digital Entertainment conf.*, 2013.

[35] D. Karavolos, A. Liapis, and G. N. Yannakakis, "Learning the patterns of balance in a multi-player shooter game," in *Proc. of the FDG workshop on PCG in Games*, 2017.

[36] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *Proc. of the intl. conf. on Learning Represenations*, 2016.

[37] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. of the intl. conf. on Machine Learning*, 2015.

[38] 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 *Proc. of the IEEE conf. on Comp. Intell. and Games*, 2018.

[39] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer series in statistics New York, 2001, vol. 1, no. 10.

[40] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *Proc. of the intl. conf. on Learning Representations*, 2014.

[41] A. Nguyen, J. Clune, Y. Bengio, A. Dosovitskiy, and J. Yosinski, "Plug & play generative networks: Conditional iterative generation of images in latent space," in *Proc. of the IEEE conf. on Computer Vision and Pattern Recognition*, 2017, pp. 3510–3520.

[42] D. Erhan, Y. Bengio, A. Courville, and P. Vincent, "Visualizing higher-layer features of a deep network," University of Montreal, Tech. Rep. 1341, Jun. 2009.

[43] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proc. of the IEEE intl. conf. on Computer Vision*, 2017, pp. 618–626.

[44] A. Liapis, G. Smith, and N. Shaker, "Mixed-initiative content creation," in *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016, pp. 195–214.