Deep Convolutional Player Modelling on Log and Level Data

Nicholas Liao, Matthew Guzdial, Mark Riedl

[illegible]

Mark Riedl, Faculty Advisor

Sonia Chernova

Sonia Chernova, Second Faculty Reader

Deep Convolutional Player Modeling on Log and Level Data

Nicholas Liao
Georgia Institute of Technology
nliao7@gatech.edu

Matthew Guzdial
Georgia Institute of Technology
mzguzdial@gatech.edu

Mark Riedl
Georgia Institute of Technology
riedl@cc.gatech.edu

ABSTRACT

We present a novel approach to player modeling based on a convolutional neural net trained on game event logs. We test our approach and a hybrid extension over two distinct games, a clone of *Super Mario Bros.* and *Gwario*, a human computation version of *Super Mario Bros.: The Lost Levels*. We demonstrate high accuracy in predicting a variety of measures of player experience across these two games. Further we present evidence that our technique derives quality design knowledge and demonstrate the ability to build a more general model.

CCS CONCEPTS

•**Human-centered computing** → *HCI theory, concepts and models;* •**Applied computing** → *Computer Games;*

KEYWORDS

player modeling, deep neural nets, convolutional neural nets, super mario bros.

ACM Reference format:
Nicholas Liao, Matthew Guzdial, and Mark Riedl. 2017. Deep Convolutional Player Modeling on Log and Level Data. In *Proceedings of , , ,* 9 pages.
DOI: 10.1145/3102071.3106346

1 INTRODUCTION

Player modeling is the field associated with the problem of learning to predict player experience. A common machine learning approach involves a designer picking out a set of super-features to summarize the player's performance (e.g. total enemies killed, total number of deaths, etc), writing code to pull these values from game logs (timestamped records of button presses and in-game events occurrences. from a particular playthrough), and mapping these features to player experience measures (e.g. fun, challenge, etc). This mapping is then used to predict on novel player experiences. Despite successful experimental applications, player modeling goes unused in most modern games, with game companies preferring to model players in aggregate with player analytics [2, 3]. A difficulty in adopting player modeling systems is acquiring accurate accounts of player experience. The most common approach is to forgo collecting data, and rely on some theory or categorization of player experience. This represents an additional burden on designers.

One reason that designers choose not to pursue player modeling might be the difficulty in designing appropriate super features to summarize player experience. For example, a particular game might log when an enemy of a particular type is killed. One could imagine using a single super-feature that treated all enemy types equally (e.g. # enemies killed), one super-feature for each enemy type, or super-features that split the enemies according to some characteristic (e.g. # flying enemies killed and # ground enemies killed). In addition, with the recent diversification of the video game industry, techniques must account for larger variability in a player preferences[10]. Old assumptions can now only be used for specific demographics. A novel technique that could learn super-features to track without access to the game engine could afford more developers access to player modeling.

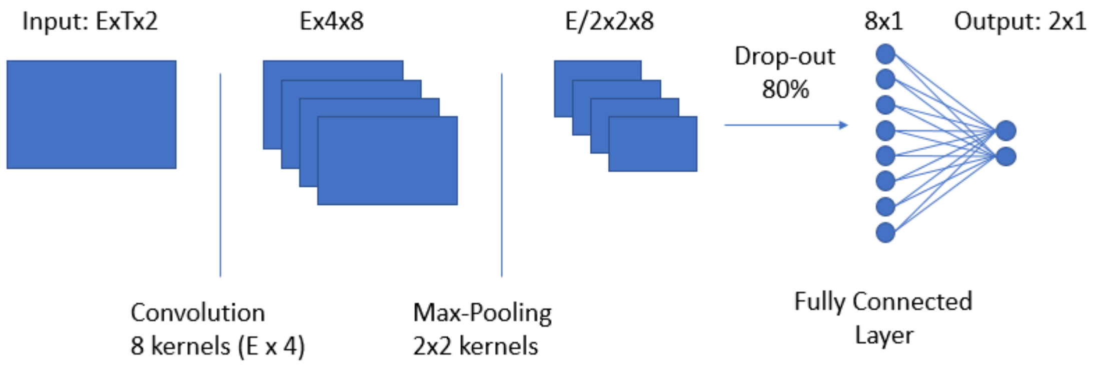
In this paper we present techniques to automatically rank player experiences from game event logs and level structure information based on self-reported rankings. We examine the applications of a convolutional neural net (CNN), for its ability to learn a "set of features" to track automatically, cutting back on designer authoring burden. While CNNs are typically applied to images, we find them appropriate to this task as they perform well in domains where local structure has high-predictive value (e.g. killing an enemy impacting a player's perception of a level's difficulty). We evaluate this technique in two games, a *Super Mario Bros.* clone and a related platformer. We demonstrate that this technique, along with a complementary prior technique [9], can accurately predict player experience. The primary contribution presented in this paper is an approach to model players automatically from game event logs based on pairwise rankings.

The rest of this paper is organized as follows. We begin with a description of prior related work. In section 3 we discuss the specifics of our neural network architecture. In section 4 we overview the three evaluations we ran: two evaluations on two distinct games (sections 5 and 6) and an evaluation on potential generalizability of the system (section 7). In total we present a novel approach to player modeling that cuts back on designer burden and shows great success in conjunction with prior methods.

2 RELATED WORK

Yannakakis et al. [21] describe player modeling as "the study of computational means for the modeling of player cognitive, behavioral, and affective states which are based on data (or theories)". Most commonly player modeling is applied to the problem of player customization, adjusting elements such as difficulty to tailor an individual user experience [4][3][16][1]. While many varied approaches have made use of the term player modeling, we focus on the body of work to learn a model to predict a player's subjective experience based on data from prior players. Thus we identify the primary characteristics that differ between player modeling approaches (outside of the games they are implemented in) as: (1) the set of game data used to make the subjective experience prediction, (2) the kind of subjective experience (or artificial [9, 18])

Figure 1: A visualization of our presented convolutional neural network architecture. A single CNN layer empowers this model that scans 3-step sequences of events from each pair of game logs and makes predictions on which presented game log would be ranked higher according to self reports.



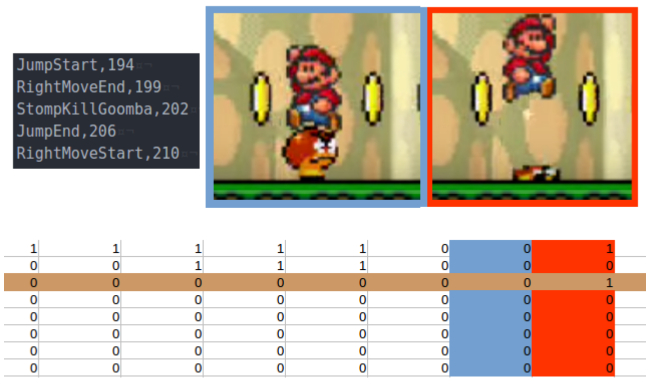
data collected, and (3) the machine learning approach used to learn a mapping between game variables and predicted subjective experience. We identify a set of prior player modeling work relevant to this paper and describe how our work differs from each.

Drachen et al. [6] created a player modeling system in the game *Tomb Raider: Underworld* trained on a set of hand-defined variables (times help command used, level completion time, and number/cause of death) extracted from game event logs, which are files that represent the sequence of actions taken and events that occurred during play. They make use of self organizing maps [17], a type of artificial neural network as the basis of their model. Rather than train their model to predict subjective player experience, they use their model to categorize different styles of player experience based on their chosen variables. We differ from Drachen et al. in our use of a CNN to automatically learn what sequences of events are predictive, the use of a deep neural network architecture (CNN), and training on self-reports.

Summerville et al. [16] take a novel approach to player modeling in *Super Mario Bros.* They extract the path a player takes through the level from gameplay footage and use this path and the level architecture to train a long-short term memory recurrent neural network (LSTM RNN) to generate new levels that are more likely to afford similar player paths. While this work more cleanly fits into the field of experience-driven procedural content generation [22], the LSTM RNN does learn an implicit model for predicting player experience. In addition, this work parallels our own in its use of a deep neural network architecture and not requiring a set of hand-defined variables.

Shaker et al. [13] present a general player modeling system applied to *Super Mario Bros.* and a first person shooter game called *Sauerbraten*. They hand define a wide set of super-features summarizing game log events (e.g. total enemies killed), but make use of an unsupervised approach to pick from this set. They make use of a model based on an artificial neural network architecture constructed via an evolutionary process, and predict player experience based on self reports. This work is the most similar to our own, but we differ in training directly on game event logs without

Figure 2: Example of the game log matrix and their corresponding ticks. The third row records when the player stomps on a Goomba.



```
JumpStart,194
RightMoveEnd,199
StompKillGoomba,202
JumpEnd,206
RightMoveStart,210
```

1	1	1	1	1	0	0	1
0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

hand defined super-features and utilizing a deep neural network architecture (CNN).

There are two prior, relevant applications of CNNs to games outside of the field of player modeling. Guzdial et al. [9] proposes a technique for level modeling using a computer agent that plays through the levels. They have shown that CNNs can independently determine what parts of a level determine player enjoyment, effectively selecting its own features. We utilize the Guzdial et al. system in a "hybrid" approach in conjunction with our novel log-based method. Mnih et al. [12] have further shown that CNNs can capture player strategies and behaviors.

3 SYSTEM OVERVIEW

In this section we overview our technique to learn a predictive model of player experience from game log and level structure. We begin by describing our general approach for learning features from low-level logs via a convolutional neural net approach, then extend

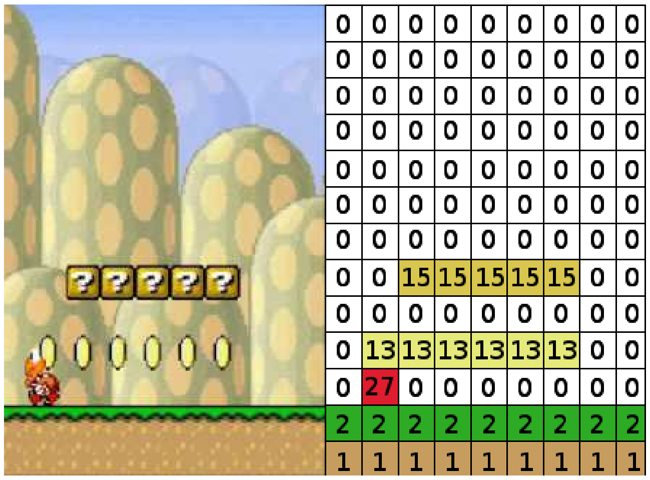


Figure 3: Example of the level matrix. Each block or enemy has a unique identifier.

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	15	15	15	15	15	0	0
0	0	0	0	0	0	0	0	0
0	13	13	13	13	13	13	0	0
0	27	0	0	0	0	0	0	0
2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1

it in a hybrid system with a previous process [9]. Our approach requires the following steps. First we acquire player rankings across the set of subjective experience measures we wish to model (e.g. difficulty, engagement, etc). We make use of rankings rather than ratings, due to the greater consistency of rankings [20].

3.1 Log Network

This subsection describes our "log" network, the convolutional neural net (CNN) approach based entirely on game logs as input. Given that we make use of a neural network architecture all game logs must be of the same shape. We therefore format all information as a matrix.

We present an illustration of our game log matrix in Figure 2. The rows of the matrix correspond to different events in the logs and columns corresponding to the number of time steps (called "ticks" in games) needed to complete the level. Since CNNs require a fixed input dimension and completion time varies among players, the tick numbers are normalized to fit within the predefined size. Our CNN architecture is trained on two game logs (representing two different levels a single player played) and set as it's target the ranking the player reported across a particular feature (e.g. "level 1 was more fun than level 2"). In this fashion a unique CNN is trained for each feature present in the self-reported rankings. For clarity to the readers we note there may be different rankings for the same levels (level 1 is more fun than level 2. level 2 is more frustrating than level 1), thus it is insufficient to make a prediction of player experience solely on level information. We hypothesize that the different events the player experienced in each level might account for any variation in rankings.

We present an example of our game log representation in Figure 2. Suppose a game has E unique game events, and a player took T ticks. The game log maps to an $E \times T$ Matrix. Each column contains 1's and 0's, signifying the presence or absence, respectively, of an event type that tick. A game log is transformed from a sequence of events and the timesteps at which they occur by taking each event and timestep pair and setting the value of that event row to 1 for that timestep. Thus the matrix has 1's representing an event (row) occurred at time (column) with 0's at all other indexes.

After transforming each player's game logs into a matrix there are 2-dimensional matrices with a consistent T value (given that the same events occur across all levels), but differing values of E (as players will take varying amounts of time to finish the level). A deep neural network requires that all input be of a consistent size. We experimented with a set of different techniques to get a consistent size including cutting all logs off after a given amount of time, and setting all matrices E values to the maximum possible time (leaving most matrices to have many columns of 0's at the end). However, we found the most success with normalizing all matrices to the same E value of 1000 time steps (a value slightly lower than the lowest actual completion time in the dataset). We anticipate that this was successful as most events occurred across several frames, and therefore this allowed the CNN to capture more interactions between events.

We now discuss our CNN architecture, visualized in Figure 1. As with other neural networks, training is composed of gradient descent and backpropagation. The core component of the CNN, the convolution layer, scans subregions of the input to find patterns. A filter is a fix-sized "picture frame", that moves across the input, creating the subregions. These techniques were originally used to analyze images. For a facial recognition task, the filter looks for eyes, mouths, etc. by comparing the relationships between nearby pixels. [11][5] We find CNNs to be appropriate for the task of learning a model of player experience from events due to their relational awareness in other contexts.

We connect the convolution layer to a max-pooling layer, a dropout layer, and a fully connected layer for prediction. The max pooling layer reduces the dimensionality of the input, and focuses in on the parts of the trace that help determine preference. After training, we can examine this layer to find what the player finds challenging, enjoyable, etc. To reduce overfitting, the dropout layer randomly disables some of its nodes, and its connections. It has also been shown to help nodes specialize into capturing different aspects of the input. [15]

Since the CNN can capture relational information, we make use of a filter that captures four columns at a time, representing a sequence of four actions. In a level, we would expect blocks close to each other in space to be related to one another in some way. In a player log, a sequence of logs may represent some high level action like stomping on an enemy.

Our first technique takes only log matrices as its input, feeds it through the CNN architecture as discussed above, and predicts solely on the events that occur to the player. Two players may play through the same levels and rank them opposite of one another. While the level technique would feed the same information in and have contradicting datapoints, the log matrix would have unique inputs, specific to each player's playthrough.

3.2 Hybrid Network

Our second technique is a hybrid of our system and the Guzdial et al. system, with both neural network architectures combining into a final fully connected layer. This "hybrid" therefore represents a combination of log and level information, with the ability to make decisions based on both individual systems. We see this as an extension of previous techniques by integrating our log information. We describe the level information approach briefly here but for more detail see [9].

For levels, we make use of a similar matrix representation. Each level can be broken up into its underlying grid system, where a grid space can only be occupied by one object at a time. As seen in Figure 3, each unique foreground object, whether it be blocks, collectibles, or enemies, is mapped to a unique identifying number. The location in the matrix corresponds to its position in the level.

The log and level matrices go through their own convolution, max-pooling, and dropout layers. They are then connected together by a fully-connected layer to be used for prediction.

4 EVALUATION OVERVIEW

We ran a total of three evaluations of our system. We applied our system on two games, a *Super Mario Bros.* clone called Infinite Mario [19] and a Mario-derivative focused on performing human computation *Gwario* [14]. We note that in *Gwario*, the player must collect specific sets of items as opposed to only finding the end of the level (i.e. the player's objectives are different). We make use of two separate games in our evaluation in order to demonstrate the generalizability of our model, and focus on Mario-like games due to the popularity of Mario as a baseline. We ran a final third evaluation to further address the question of generalizability.

For the two game evaluations we drew on preexisting datasets from human subject studies of each game. Given that we did not run any novel human subject study for this paper, certain elements were out of our control, notably the size of the subject pool and the ranking questions that were asked. Therefore the two datasets differ in length and both asked each individual player to rank the play experiences according to slightly different subjective experience measures. Therefore the Infinite Mario dataset has rankings on the measures of fun, frustration, challenge, creativity, design, and style as it was primarily interested in questions of judging player experience on specific levels. The Gwario dataset on the other hand has rankings for frustration, challenge, and fun/engagement as it focused explicitly on player experience.

In addition to the two techniques that we test, we also include results from a baseline. Guzdial et al. [9] made use of a CNN-based approach to predict an aggregate player score of a Mario level based on the level architecture, as shown in Figure 3, and a small set of hand-defined variables (e.g. number of deaths to enemies, number of deaths to gaps, number of enemies killed, and time to complete a level divided by its width). We include it as our baseline as it represents an approach that foregrounds the level in making decisions about player experience. Further, as it is a component part of our "hybrid" approach, if it beats out that extension of our system that would demonstrate a failure of our log-based CNN approach for player modeling.

5 SUPER MARIO BROS. EVALUATION

For our first evaluation we applied our system to a clone of *Super Mario Bros.* called "Infinite Mario". We drew on a dataset from a study previously conducted in the game engine, which we describe briefly below but for more detail see [7]. Ultimately we ran a ten cross-fold analysis on the engine between the three experimental systems. In the following subsections we discuss the evaluation setup (including a description of the game), discuss the results of our ten cross-fold analysis, and give examples of the learned features of our CNN.

5.1 SMB Evaluation Setup

Super Mario Bros. is a 2D platformer game originally created for the Nintendo Entertainment System. The player controls a single character and tries to get past enemies and reach the end of the level, only being able to run right and left, jump, and shoot fireballs (with a power-up). An open source copy called *Infinite Mario* was used so that logs could be extracted.

We adapted the dataset used by Guzdial et al. in [8]. Seventy-five players were asked to play Level 1-1 from the original Super Mario Bros, and then two other levels from a pool of 15 artificially generated levels. After, players were asked to rank the three levels based on fun, frustration, challenge, level design, and creativity of the levels.

For each player, we took permutations of two level logs, and labeled the pair with a classification of "Level 1 was more X" or "Level 2 was more X" where X was fun, frustating, challenging, well designed, or creative. This resulted in 6 data points per person, or 450 data points. By adding the reverse of the reported ranking into the dataset, we have guaranteed that half the dataset is of class 1, and the other half class 2. Therefore, we would anticipate a pure random system to perform at around 50% accuracy.

5.2 SMB Results

We report results over the five categories in tables 1 and 2. In the tables we use "Log" to indicate our system, "Level" to indicate the prior system largely reliant on level structure [9], and "Hybrid" to indicate the combined architecture of the two prior systems. In addition to accuracy, we report the coefficient of determination, a measure of predictability of the label, based on the input data. We use the Wilcox test to evaluate statistical significance in output between pairs of systems.

For the challenge and frustration categories we found nearly identical results, with our hybrid system significantly more accurate than the level system and the level system more accurate than our log system ($p < 0.05$). Thus we ended up with a total ordering of hybrid>level>log. However, we note that all systems beat the absolute baseline of 50% for the prediction task.

The creativity category stood out as while our hybrid system was still significantly more accurate ($p < 0.01$) than either of the other two systems, our log system was significantly more accurate than the level system ($p < 0.01$). We further note that creativity provided the lowest overall models in terms of their predictive quality, with none of the systems reaching R^2 values above 0. This indicates that the mean of the testing data does a better job of "explaining" the data than the predictions.

The Design and fun categories offer similar results, with both representing a small variation in the hybrid>level>log total order. For design, both level and hybrid were found to have significantly more accurate results than log ($p < 0.01$), but the Wilcoxon-Mann

Table 1: Mean and median accuracies across 10 folds. Mario

	Challenge		Creativity		Design		Frustration		Fun	
	mean	median	mean	median	mean	median	mean	median	mean	median
Level	74.44%	74.44%	53.33%	54.44%	77.56%	76.67%	76.22%	77.78%	63.33%	62.22%
Log	65.78%	64.44%	64.67%	64.44%	75.11%	73.33%	69.33%	68.89%	64.89%	65.56%
Hybrid	**83.11%**	**82.22%**	**71.33%**	**70.00%**	**81.11%**	**80.00%**	**81.55%**	**83.33%**	**81.56%**	**81.11%**

Table 2: Mean and median R^2 across 10 folds. Mario

	Challenge		Creativity		Design		Frustration		Fun	
	mean	median	mean	median	mean	median	mean	median	mean	median
Level	0.068	0.045	-0.478	-0.376	0.197	0.193	0.110	0.216	-0.355	-0.392
Log	-0.254	-0.285	-0.274	-0.251	0.098	0.135	-0.084	-0.054	-0.308	-0.347
Hybrid	**0.399**	**0.324**	**-0.065**	**-0.007**	**0.308**	**0.281**	**0.31**	**0.338**	**0.267**	**0.235**

Figure 4: Maximally activated visualizations of four of the eight filters for challenge.

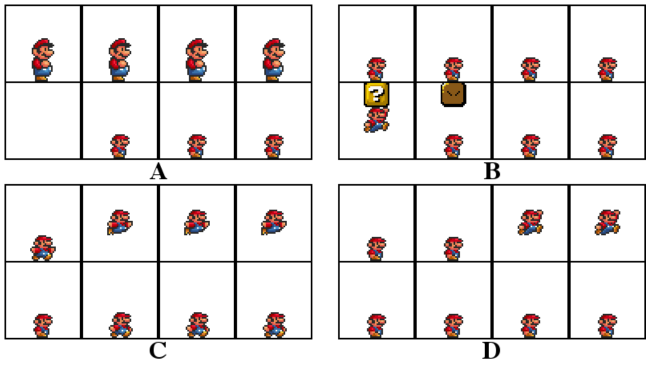
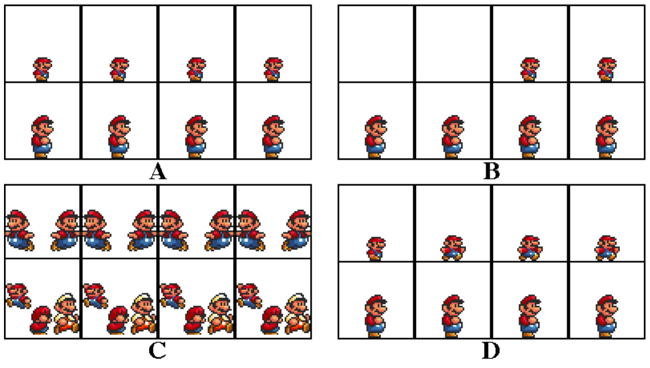


Figure 5: Maximally activated visualizations of four of the eight filters for frustration.



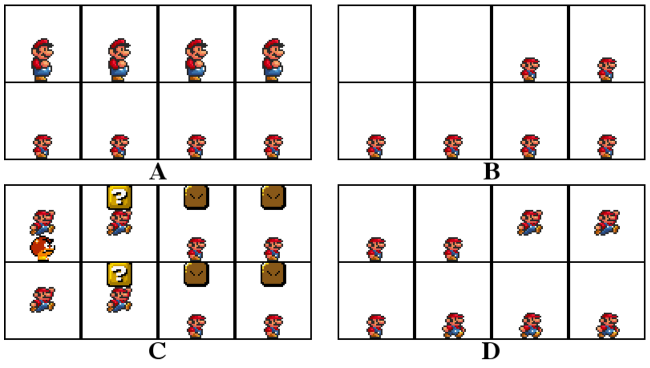
Whitney U paired test was unable to find any such ordering between hybrid and level. Similarly, with fun the hybrid system was found to have significantly more accurate results than the level and log systems ($p < 0.01$), but no ordering was found between level and log.

In all cases but design, the hybrid system performs significantly better than either of its two constituent systems. This suggests that the two constituent systems (log and level) offer complimentary information towards making predictions of player experience, rather than one being strictly better than the other. In addition, these results demonstrate that some types of information are more predictive to certain measures of player experience. For example, the creativity labels were predicted more accurately given access to logs of events, suggesting players reflected on the comparative experiences when deciding on this ranking.

5.3 SMB Learned Actions

One of the strengths of CNNs is in their ability to learn useful features from training data. To evaluate the CNN's ability to extract useful super-features from raw game logs, we visualize the pairs of comparative event sequences that maximally activate the learned filters our "log" CNN trained to predict. We visualize four of the

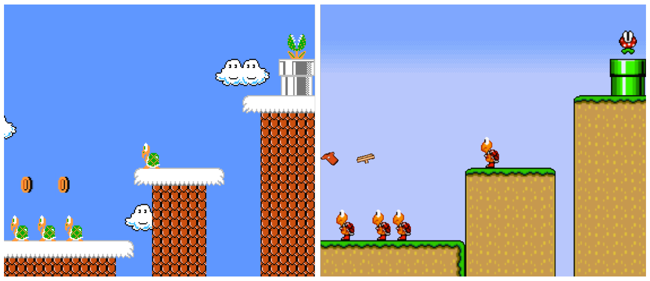
Figure 6: Maximally activated visualizations of four of the eight filters for fun.



eight trained filters for our CNN trained on the challenge (Figure 4), frustration (Figure 5), and fun datasets (Figure 6).

Figure 4 presents visualizations of four of the eight trained filters for the challenge labels. Figure 4(a) demonstrates two sequence

Figure 7: Comparison of Gwario to Super Mario Bros.: Lost Levels, on which it is based.



pairs where one game log includes the player restarting the level where the other game log section has the same player standing still as "large mario". The other visualized filters largely involve comparisons of progress. For example, Figure 4(d) compares a sequence where the player jumps forwards versus a sequence where the player is standing still.

Figure 5 presents visualizations of four of the eight trained filters for the frustration labels. In this case two of the visualized filters demonstrate effects of the normalization described in our system overview. Figure 5(b) includes two level restarting events back to back in one of the two features. Given that the normalization process "squishes" the game event logs, this means in the original playthrough the player died twice in a row in quick enough succession that the normalization process removed the intervening events. Figure 5(c) instead demonstrates multiple, contradictory events co-occurring, which is caused by the "squishing" from normalization. Thus in the top sequence the player is jumping back and forth as Mario, while in the bottom sequence the player goes from "fire mario" takes damage and becomes "large mario", before taking more damage and attempting to get away as "small mario".

Figure 6 contains the four visualizations of the maximally activated filters for the fun labels. We bring special attention to 6(c), which compares a player activating a ?-block versus a player squishing an enemy then launching into a ?-block as a particularly illustrative example of an intuitive, comparatively more fun moment.

We do not empirically validate the maximally activated filters, but note that in general they tend to match our intuition. Frustration and fun are highly impacted by how quickly the player dies, and all three are dependent on relative rates of progress.

6 GWARIO EVALUATION

For our second evaluation we applied out system to *Gwario*, a game with a purpose (or GWAP), adaption of the Japanese sequel to *Super Mario Bros.*, *Super Mario Bros.: The Lost Levels*. GWAPs are games that outsource work in the form of a game. In this instance, the player may take the same actions, but in addition to finding the end of a level, the player attempts to collect items that answer a human computation question. We give an example of the transformation of a level from *Super Mario Bros.: Lost Levels* to *Gwario* in Figure 7. Note that the coins have been replaced with items the player collects to answer a human computation question, which gives coins a greater importance over the default game. We wish to see whether this technique could be generalized across games, at least to similar games.

Table 3: Mean and median accuracy across 10 folds for the Gwario dataset.

	Challenge		Frustration		Fun	
	mean	mdn	mean	mdn	mean	mdn
Level	52.7%	59.1%	43.6%	45.5%	49.1%	50.0%
Log	75.5%	77.3%	78.2%	77.3%	66.4%	68.2%
Hybrid	**78.2%**	**81.8%**	**80.9%**	**81.8%**	**76.4%**	**77.3%**

Table 4: Mean and median R^2 across 10 folds for the Gwario dataset.

	Challenge		Frustration		Fun	
	mean	mdn	mean	mdn	mean	mdn
Level	-0.674	-0.670	-0.942	-1.021	-0.821	-0.700
Log	0.1323	0.245	0.225	0.224	-0.115	0.134
Hybrid	**0.364**	**0.448**	**0.457**	**0.553**	**0.241**	**0.354**

6.1 Gwario Evaluation Setup

We draw on the dataset from the study conducted in Siu et al in [14]. Players were asked to play two levels from a pool of four adapted from *Super Mario Bros.: The Lost Levels*, and rank them based on challenge, fun, and frustration. We note that 58 players took part in the study, resulting in 116 data points. As in the previous evaluation we split the dataset into 10 cross-folds.

One difference between *Gwario* and *Super Mario Bros.* was that the two had different events, which meant slightly different matrices. Notably, *Gwario* includes piranha plants, an enemy type hidden in pipes (with two associated events, player deaths to piranha plants and piranha plants killing the player), and *Infinite Mario* included cannons that shot unique "bullet bill" enemies (with two associated events). The levels of *Gwario*, based on the sequel to Super Mario Bros., were also significantly longer and more challenging than the levels from the first study. In addition, while the *Gwario* levels had been adapted from *Super Mario Bros.: The Lost Levels*, they were still designed by a human expert, as opposed to the majority of the levels from the first study, where were automatically generated.

6.2 Gwario Results

We report results over the three categories in tables 3 and 4. In the tables we use "Log" to indicate our system, "Level" to indicate the Guzdial et al. system reliant on level structure [9], and "Hybrid" to indicate the combined architecture of the two prior systems. As in the first evaluation we report the coefficient of determination, a measure of predictability of the label, based on the input data. We use the Wilcox test to evaluate statistical significance in output between pairs of systems.

For both the challenge and frustration categories we find that both our log and hybrid system perform very similarly as can be seen in both the accuracy and R^2 values. In addition, we are unable to reject the null hypothesis that the two systems output comes

Figure 8: Maximally activated visualizations of four of the eight filters for challenge.

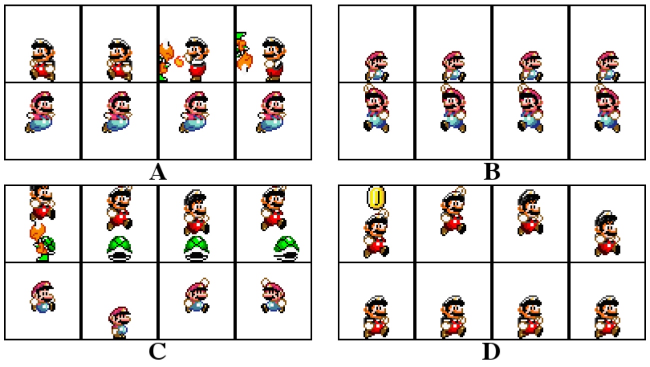
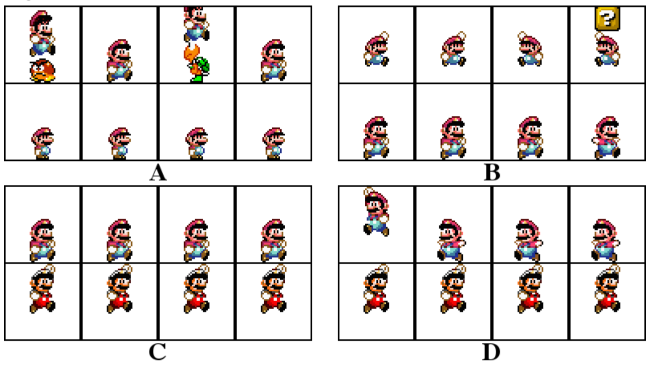


Figure 9: Maximally activated visualizations of four of the eight filters for frustration.

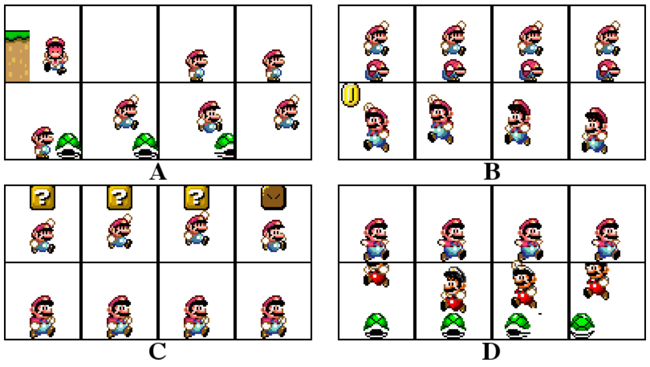


from the same distribution (unable to say they aren't significantly different). However both the hybrid and log systems output is significantly more accurate than the level systems predictions ($p < 0.05$).

Fun differs from challenge and frustration in that the log and hybrid predictions differed significantly. That is that the hybrid system output was significantly more accurate than both other systems and the log system's output was significantly more accurate than the level system. This is reflected in both the accuracy and R^2 values, and from the relatively similar performance of the log-system, seems to arise more from comparative difficulty in predicting Fun for our log system and therefore for the hybrid system as well.

We find overall that these results suggest that the player logs were much more predictive than the level information for the professionally designed *Gwario* levels, perhaps due to greater variation and size. However, given the success of the hybrid approach, this indicates that both the log and level systems contributed to the predictive power, suggesting they again represented complimentary approaches.

Figure 10: Maximally activated visualizations of four of the eight filters for fun.



6.3 Gwario Learned Actions

In this section, we visualize the patches of comparative event sequences that maximally activate the learned filters our "log" CNN trained to predict. We visualize four of the eight trained filters for our CNN trained on the challenge (Figure 8), frustration (Figure 9 and fun categories (Figure 10).

Figure 8 represents the four sequence pairs that maximally activated four of the eight trained filters of our CNN. Overall the sequence pairs match our intuition, with two focusing on defeating enemies as opposed to making progress (Figures 8a and 8c), and one focused on making progress versus not (Figure 8b). We note in particular Figure 8d, which highlights the player collecting a coin versus making forward progress. This is important for the system to learn given the added role coins play in *Gwario*.

Figure 9 covers the maximally activating sequence pairs filters for the frustration category. All of the sequence pairs seem to focus on one sequence making significantly more progress than the other. Figures 9a and 9b both include one sequence of the player making great progress while in the other sequence the player barely moves, if at all. Figure 9d compares forward progress against the player moving backwards, and Figure 9c involves different qualities of forward progress.

Figure 10 covers the fun category filters. Overall the sequence pairs seem focused on collecting things, and "unleashing" the shells of green koopas, an enemy type. Unleashing the shells fits our intuition of a fun moment as the *Gwario* levels tend to have long rows of enemies and knocking a shell into them will defeat them all without the player having to put in additional effort.

We find the visualizations of the sequence pairs that maximally activated filters for *Gwario* to match our intuition as designers. This makes sense given the relative performance of the log system in this evaluation in comparison to the first.

7 GENERALIZATION EVALUATION

We have thus far demonstrated that our systems can perform reasonably accurately across two distinct, but similar games. However, each of these games required completely retraining our system.

Table 5: Average accuracies from training on on the entirety of one dataset and predicting on the other.

	Train Mario; Test Gwario			Train Gwario; Test Mario		
	Challenge	Frustration	Fun	Challenge	Frustration	Fun
Level	49.12%	**51.75%**	**49.12%**	51.78%	47.55%	49.11%
Log	44.74%	50.00%	46.49%	**53.11%**	**48.67%**	**53.33%**
Hybrid	**59.65%**	48.25%	46.49%	51.11%	47.55%	52.89%

Table 6: Mean and median accuracies across 10 folds with a dataset made of equal halves of the two datasets.

	Challenge		Frustration		Fun	
	mean	mdn	mean	mdn	mean	mdn
Level	63.6%	63.6%	61.4%	59.1%	55.0%	54.6%
Log	62.3%	59.1%	63.6%	65.9%	**61.8%**	**59.1%**
Hybrid	**71.4%**	**70.5%**	**71.8%**	**72.7%**	58.6%	56.8%

Ideally we would be able to create a more general game player modeling system, at least across games of the same genre, in order to cut back on training time and present player experience predictions for games without training data. In this section we present the results of an evaluation to address the issue of generalizability. We note that the only major alteration of our system to this evaluation was the inclusion of the full set of events from both games for our game matrix.

Given that two games that we ran our individual evaluations on, *Super Mario Bros.* and *Gwario*, represent very similar games (one a variation on the sequel of the other), we first demonstrate that the system still recognizes them as two different games. In particular, we ran a simple evaluation where we trained on the entirety of one game's dataset and tested on the other for the three categories of player experience that the two games shared. We present these results in Table 5. All of the average accuracies are close to the naive baseline of 50% for the task, suggesting that a system trained on one game made predictions no better than random. Thus we can conclude with some certainty that these two games differ to some extent, suggesting that a generalizability evaluation can be run.

For our generalizability evaluation we composed a dataset made of half *Gwario* and half *Super Mario Bros.* data, selected randomly from each dataset. We then split this new "mixed" dataset into ten folds, with each fold containing data half from each game. We then ran a similar evaluation as above, and report the average and median accuracies in Table 6. We note that while there is a drop of about 10% from the average accuracies we saw when each system was trained on each individual game, the accuracies are above the random baseline of 50%. This suggests that it is possible to apply these techniques more generally across different, if similar, games. If we could determine a universal set of event log types, it may be result in higher accuracies. We discuss this in our Future Work.

8 DISCUSSION

We present results of our approach applied to two separate games. We find that our log approach, especially in combination with a prior level-focused approach, performs with high accuracy on predicting various categories of user experience. In this section we identify the most important takeaways. Notably we discuss implications in terms of our sequence modeling log CNN and the potential of our hybrid approach.

We find that our game-log based CNN can aid in the prediction of player experience, and given the visualizations of maximally activating sequence pairs, appears to learn intuitive design knowledge. We note that its performance seemed to reflect designer expectations across different tasks. For example, it performed less well when predicting how well "designed" a player thought a level was, a term that brings to mind notions of level architecture. Further, it has the potential to aid designers in revealing new facets of player experience. For example, it's relatively superior performance at the task of prediction player "creativity" tags, suggests that players might reflect more on the events they experienced than level architecture when asked how "creative" a level was.

Our hybrid approach represents a technique that, without any designer-defined super-features, can reach accuracies of roughly 80%. While there is clearly more to be done (roughly 20% more), we contend that this represents a successful first attempt at a game-log based system to automatically predict player experience.

9 FUTURE WORK

We propose analyzing player actions for modeling player experience based on sequences of frames. These sequences appear to encode high-level actions in response to challenges in the level, which are then correlated with player preference. Since level structures were represented as an image and player movements were a series of frame data, we believe it would be feasible to pass in raw video playthroughs.

Since we used the native logging systems for *Infinite Mario Bros.* and *Gwario*, when we examined the mixed models, there was not a perfect one to one mapping of logs types. *Gwario* distinguished between red and green koopas whereas *Infinite Mario Bros.* grouped them together. The attack patterns of green koopa shells and red koopa shells are different, so when the model trained on *Mario* and tested on *Gwario*, it would not distinguish between the two. If we passed in directly from video, the network may distinguish between these if needed.

Another limitation of our technique is that it requires designer-authored logging systems to record player movements. By directly analyzing video, it may be possible for a neural network to determine its own set of logs and then log playthroughs. The question of whether our techniques can generalize across genres is another direction for research. It would be interesting to analyze how a player utilizes terrain in turn-based strategy games, where level structure also plays an important role in player actions. Since terrain design is so vital to this genre, it would be worthwhile to see if the CNN could generate new designs. From there we could look at patterns between platformers and turn-based strategy games. We would also like to examine a level generator that takes input from the preference learners. In a controlled test, we would like to verify that the CNN-LOG positively influences a level generator towards designs that the player enjoys. Since this technique captures block arrangements as well as how the player reacts to them, it may be able to design new arrangements to elicit specific reactions from the player.

10 CONCLUSIONS

We present a novel approach to player modeling that shows high accuracy with a low designer authoring burden. We test our approach over two distinct games, a clone of *Super Mario Bros.* and *Gwario*, a human computation version of the sequel to Mario. We demonstrate high accuracy over these two games, and demonstrate the ability to build a more general model that accounts for variations between the two. Our work relies on a convolutional neural net (CNN) to read in a matrix of player actions called a game log and make predictions on relative player experience. We present A novel technique that learns what elements of player experience to track without access to the game engine and can potentially afford more developers access to player modeling by lowering design burden.

REFERENCES

[1] Mohamed Abou-Zleikha and Noor Shaker. 2015. Evolving random forest for preference learning. In *European Conference on the Applications of Evolutionary Computation*. Springer, 318–330.

[2] Mike Ambinder. 2011. Biofeedback in gameplay: How valve measures physiology to enhance gaming experience. In *game developers conference*, Vol. 2011.

[3] Sander Bakkes, Shimon Whiteson, Guangliang Li, George Viorel Vişniuc, Efstathios Charitos, Norbert Heijne, and Arjen Swellengrebel. 2014. Challenge balancing for personalised game spaces. In *Games Media Entertainment (GEM), 2014 IEEE*. IEEE, 1–8.

[4] Glen Berseth, M Brandon Haworth, Mubbasir Kapadia, and Petros Faloutsos. 2014. Characterizing and optimizing game level difficulty. In *Proceedings of the Seventh International Conference on Motion in Games*. ACM, 153–160.

[5] Changxing Ding and Dacheng Tao. 2015. Robust face recognition via multimodal deep face representation. *IEEE Transactions on Multimedia* 17, 11 (2015), 2049–2058.

[6] Anders Drachen, Alessandro Canossa, and Georgios N Yannakakis. 2009. Player modeling using self-organization in Tomb Raider: Underworld. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*. IEEE, 1–8.

[7] Matthew Guzdial and Mark Riedl. 2016. Game Level Generation from Gameplay Videos. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

[8] Matthew Guzdial and Mark Riedl. 2016. Learning to Blend Computer Game Levels. In *Seventh International Conference on Computational Creativity*.

[9] Matthew Guzdial, Nathan Sturtevant, and Boyang Li. 2016. Deep static and dynamic level analysis: A study on Infinite Mario. In *Experimental AI in Games Workshop*, Vol. 3.

[10] Jesper Juul. 2010. *A casual revolution: Reinventing video games and their players*. MIT press.

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3431–3440.

[12] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, and others. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[13] Noor Shaker, Mohammad Shaker, and Mohamed Abou-Zleikha. 2015. Towards generic models of player experience. In *Proceedings, the Eleventh Aaai Conference on Artificial Intelligence and Interactive Digital Entertainment (aiide-15)*. AAAI Press.

[14] Kristin Siu, Matthew Guzdial, and Mark Riedl. 2017. Evaluating Single-player and Multi-player in Human Computation Games. *arXiv preprint arXiv:1703.00818* (2017).

[15] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[16] Adam Summerville, Matthew Guzdial, Michael Mateas, and Mark O Riedl. 2016. Learning player tailored content from observation: Platformer level generation from video traces using LSTMs. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

[17] Christian Thurau, Christian Bauckhage, and Gerhard Sagerer. 2003. Combining Self Organizing Maps and Multilayer Perceptrons to Learn Bot-Behaviour for a Commercial Game.. In *GAME-ON*. Citeseer, 119.

[18] Julian Togelius, Renzo De Nardi, and Simon M Lucas. 2006. Making racing fun through player modeling and track evolution. (2006).

[19] Julian Togelius, Sergey Karakovskiy, and Robin Baumgarten. 2010. The 2009 mario ai competition. In *IEEE Congress on Evolutionary Computation*. IEEE, 1–8.

[20] Georgios N Yannakakis and John Hallam. 2011. Ranking vs. preference: a comparative study of self-reporting. In *International Conference on Affective Computing and Intelligent Interaction*. Springer, 437–446.

[21] Georgios N Yannakakis, Pieter Spronck, Daniele Loiacono, and Elisabeth André. 2013. Player modeling. In *Dagstuhl Follow-Ups*, Vol. 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[22] Georgios N Yannakakis and Julian Togelius. 2011. Experience-driven procedural content generation. *IEEE Transactions on Affective Computing* 2, 3 (2011), 147–161.