Experimental AI in Games: Papers from the AIIDE Workshop
AAAI Technical Report WS-16-22

Deep Static and Dynamic Level Analysis: A Study on Infinite Mario

Matthew Guzdial[1,3], **Nathan Sturtevant**[2], and **Boyang Li**[3]
[1]School of Interactive Computing, Georgia Institute of Technology
[2]Computer Science Department, University of Denver
[3]Disney Research
mguzdial@gatech.edu, sturtevant@cs.du.edu, albert.li@disneyresearch.com

Abstract

Automatic analysis of game levels can provide assistance to game designers and procedural content generation. We introduce a static-dynamic scale to categorize level analysis strategies, which captures the extent that the analysis depends on player simulation. Due to its ability to automatically learn intermediate representations for the task, a convolutional neural network (CNN) provides a general tool for both types of analysis. In this paper, we explore the use of CNN to analyze 1,437 Infinite Mario levels. We further propose a deep reinforcement learning technique for dynamic analysis, which allows the simulated player to pay a penalty to reduce error in its control. We empirically demonstrate the effectiveness of our techniques and complementarity of dynamic and static analysis.

Introduction

For many modern games, well-designed levels are at the core of a fun experience and player retainment. As games can contain hundreds of levels[1], evaluating all game levels with user studies can become very expensive. Therefore, the ability to automatically evaluate a game level along multiple design criteria, such as difficulty (Pedersen, Togelius, and Yannakakis 2009a), enjoyment (Sweetser and Wyeth 2005a) and aesthetics (Hunicke, LeBlanc, and Zubek 2004), becomes a useful tool for game level designers with limited resources. The past decade has witnessed the quick proliferation of research on this topic (e.g., Berseth et al. 2014; Tremblay et al. 2014).

We can categorize techniques for automatic game level evaluation as a scale between *static analysis* and *dynamic analysis* based on their reliance on simulated players. Static analysis encompasses evaluation of level structure without simulating gameplay. But this is not to say static analysis is incapable of modeling gameplay or game mechanics. For example, a regression analysis can reveal correlation between platform shapes and difficulty, which can be attributed to game mechanics unknown to static analysis. In comparison, dynamic analysis relies on simulation to determine how a player might act in a level. Many techniques fall between these two extremes, such as those that rely on static features that encode likely player behavior (e.g. counting gaps to approximate jumps (Pedersen, Togelius, and Yannakakis 2009b)) or those that partially simulate interaction with a level (e.g. only movement in a first person shooter level (Shi and Crawfis 2013)).

Designing effective features for static analysis often requires knowledge of game mechanics. For example, for Super Mario Bros., it may be useful to differentiate between small gaps that can be run over and large gaps that cannot. To minimize manual coding and acquire important patterns from data, we turn to convolutional neural networks (CNNs) (Lang and Hinton 1988; LeCun et al. 1989), which are capable of learning multi-level feature representations. In this paper, we explore the use of CNNs for analyzing the difficulty, enjoyment, and aesthetics of Infinite Mario Bros. levels. Experiments show the CNN outperforms traditional methods and the learned representations capture our intuition about game difficulty.

Dynamic analysis has access to potential player interactions with a level, providing additional information for analytical purposes. Our analysis shows that a simple A* algorithm produces information that improves our CNN prediction, when paired with more traditional static metrics. Nevertheless, an additional challenge arises in the construction of artificial agents that play like humans. To address this issue, we propose a deep reinforcement learning agent that simulates two major aspects of human players. First, the agent faces imprecision in its control, which forces it to favor a safe path over a fast but risky path. Second, the agent has a "focus" mechanism, which allows it to pay a penalty to reduce control imprecision. The amount of focus then becomes a surrogate for tension in the game. We leave the evaluation of this agent for future work.

Related Work

The most common approaches for static game level analysis are *game design patterns* (Bjork and Holopainen 2004) and *computational metrics* (Smith, Whitehead, and Mateas 2010; Horn et al. 2014). A game design pattern refers to a high-level, descriptive "solution" to a common game design problem, which can be used for evaluation and generation of content such as game levels (Hullett and Whitehead 2010; Liapis, Yannakakis, and Togelius 2013). Com-

[1]In May 2016, *Candy Crush Soda Saga* contained 885 levels.

putational metrics refer to low-level, technical metrics designed to capture level characteristics of procedurally generated levels, historically focused on platformer game levels.

Dahlskog and Togelius (2014) identified a set of commonly occurring patterns from the original Super Mario Bros. (e.g. a group of three enemies in a row), and used these patterns to evaluate generated levels. To some extent this work is similar to our own analysis of level aesthetics, however they make no attempt to simulate player experience or explicitly evaluate levels in terms of fun or difficulty.

Computational metrics have traditionally been applied to platformer game levels, making much of the work in the field relevant to our own (Smith, Whitehead, and Mateas 2010; Horn et al. 2014). Canossa and Smith (2015) present the most complete list of current computational metrics for platform levels derived from design theory and novice designer intuition, intended to evaluate difficulty and aesthetics (e.g. the density and frequency of enemy clusters). These metrics are intended to inform human or artificial designers, and do not tend to reflect the ratings of human players (Marino, Reis, and Lelis 2015).

Outside of both the traditional computational metrics defined by Smith *et al.* (2010) and game design patterns exists a variety of work on deriving strategies for static analysis of game levels. Such work tends to focus upon a single subjective feature, such as difficulty (Shi and Crawfis 2013; Tremblay et al. 2014) or aesthetics (Cook, Colton, and Pease 2012; Cook and Smith 2015; Tremblay and Verbrugge 2015). In this paper we focus on three objectives: difficulty, fun, and visual aesthetics, but our approach is sufficiently general to many other subjective features due to the ability of representation learning.

Dynamic analysis encompasses strategies for the automatic evaluation of levels based on some simulation of player experience. Many of the modern approaches in dynamic analysis take inspiration from the notion of game "flow" (Sweetser and Wyeth 2005b), a notion that a player's growing skill at a game must be met with equivalent challenge to maximize enjoyment. The majority of work in dynamic analysis tends to focus on enjoyment (Togelius, De Nardi, and Lucas 2007; Iida, Takeshita, and Yoshimura 2003) or a combination of difficulty and enjoyment (Togelius and Schmidhuber 2008; Cook, Colton, and Gow 2012; Bauer, Cooper, and Popovic 2013). Cook, Colton, and Gow (2012) and Bauer, Cooper, and Popovic (2013) both use simulated playouts in order to determine if a platformer level matches a target difficulty, based on the reachability and risks of level sections.

Our work explores the combination of static and dynamic analysis in order to successfully predict the measures of enjoyment, challenge, and aesthetics on a per-level basis. The field of *player modeling* focuses on deriving models of player behavior or preference in order to evaluate features of a level on a per-player basis with static and dynamic analyses (Yannakakis et al. 2013). Notably, there exists player modeling work that shares the domain of Super Mario Bros., where we have turned for inspiration for some of our high-level features (Pedersen, Togelius, and Yannakakis 2009b; Shaker, Yannakakis, and Togelius 2010). However, due to its nature, player modeling work requires knowledge of an individual player's experience with a level for its dynamic analysis, data which is difficult and time-consuming to collect. Instead, our work draws on simulated players to stand in for a theoretical "optimal" player.

There exists a set of prior work in the Super Mario Bros. domain focused on modeling elements of levels with neural networks. Summerville and Mateas (2016) make use of a simulated player and neural network architecture, but with a focus on level generation. While their learned model of level design encodes an intrinsic evaluative metric, it is based on evaluating the best level components to add during level generation rather than high-level characteristics such as player enjoyment. Jain et al. (2016) make use of an autoencoder to in part look at automatically identifying the "style" of a level, which is similar to our prediction of aesthetics.

Static Analysis with Convolutional Networks

In this section, we predict the difficulty, enjoyment, and aesthetics ratings of Infinite Mario Bros. levels directly from level maps using a convolutional neural network. As we expect, the CNN outperforms a traditional baseline and is capable of extracting useful features for prediction. Further, we complement the static analysis by CNN with features extracted from an A* algorithm's search history, which represent a shallow dynamic analysis. The combination yields substantial performance improvements, suggesting complementarity of the two strategies.

Convolutional Neural Networks

Convolutional neural networks have gained massive popularity recently for processing visual information due to their capability to learn multi-level representations that are superior to hand-crafted features (Razavian et al. 2014). A CNN typically contains convolution layers, pooling layers, and fully connected layers. A convolutional layer contains multiple filters, which are used to scan the input image from left to right and from top to bottom. At each position of the filter, corresponding values in the filter and the image are multiplied and the sum of products is returned. More formally, let a filter be a n-by-m matrix F. For an input matrix Φ^c of size $k \times l$, we have

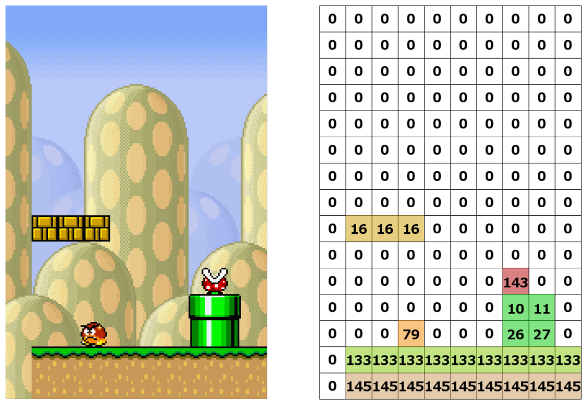
$$a_{p,q} = \sum_{1 \le i \le n, 1 \le j \le m} F_{i,j} \Phi^c_{p+i,q+j} \tag{1}$$

where p, q slide the filter across the input matrix and i, j iterate over positions within the filter and corresponding positions of the input matrix. The input then goes through an activation function (e.g., a sigmoid) to create the output matrix O^c:

$$O^c_{p,q} = f\left(a_{p,q}\right) \tag{2}$$

A max-pooling layer returns the maximum element within a certain patch of the input matrix. Let Φ^{max} and O^{max} denote the input and output matrices respectively, for an n-by-n max-pooling, we have

$$O^{max}_{p,q} = \max_{1 \le i \le n', 1 \le j \le n'} \Phi^{max}_{np+i,nq+j} \tag{3}$$



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	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	16	16	16	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	143	0	0
0	0	0	0	0	0	0	10	11	0
0	0	0	79	0	0	0	26	27	0
0	133	133	133	133	133	133	133	133	133
0	145	145	145	145	145	145	145	145	145

Figure 1: Each sprite of an IMB level (left) is represented by an integer in the grid (right). For example, a goomba is represented by 79.

The height and width of the output matrix are $1/n$ of the input matrix.

A fully connected layer can be expressed as a matrix multiplication and an activation function. The input to the fully connected layer is reshaped into a vector ϕ of size d. The output vector o of size d' is computed as

$$o = g(W\phi) \tag{4}$$

where the d-by-d' matrix W are the weights of this layer and $g(\cdot)$ is another activation function. In order to reduce overfitting, we make use of the dropout technique (Srivastava et al. 2014), which randomly disables some connections in the network for each pass. Training of a CNN is typically performed with gradient descent and backpropagation.

Experiments

We perform a series of experiments to explore static analysis with convolutional neural networks and its complementarity with dynamic analysis.

Data Our experiments rely on a dataset created by Reis et al. (2015), which is composed of 1,437 generated Infinite Mario Bros. (IMB) levels. Each level is tagged by human volunteers on a nine-point Likert scale for difficulty, enjoyment, and visual aesthetics. This is the largest dataset on Mario-like games that we are aware of. We perform regression to predict the reported enjoyment, aesthetics and difficulty for each level.

We note two limitations with this data set. First, 547 levels (38% of all levels) were rated by only a single volunteer, and the remainder are rated by anywhere from two to eight individuals (we take the median value in this case). Second, the levels are about a fifth as long as the Super Mario Bros. levels, with an average size of 40 in-game "tiles". Despite these limitations, the size of this dataset makes it the most appropriate choice for training of deep neural network models as an initial exploratory study.

As input to CNNs, an IMB level is represented as a grid of integers, equivalent to the in-game tiles that the levels are constructed from. Each level component is represented by a unique integer. Figure 1 shows this representation for a vertical slice of level.

Table 1: Results of regression for difficulty, aesthetics, and enjoyment from three CNN variants.

Measure	Method	Errors Mean	Errors Median	R^2
Difficulty	CNN-Map	1.22	0.96	0.39
	CNN-A*	1.28	1.05	0.35
	CNN-T	1.3	1.08	0.33
	CNN-All	0.92	0.72	0.64
Aesthetics	CNN	1.13	0.9	0.09
	CNN-A*	1.15	0.9	0.05
	CNN-T	1.18	0.92	0.04
	CNN-All	1.15	0.89	0.07
Enjoyment	CNN	1.09	0.9	0.16
	CNN-A*	1.11	0.91	0.14
	CNN-T	1.15	0.95	0.09
	CNN-All	1.04	0.85	0.22

Table 2: Results of two random forest variations for difficulty, aesthetics, and enjoyment.

Measure	Method	Errors Mean	Errors Median	R^2
Difficulty	RF	1.11	1.0	0.42
	RF-MAP	1.01	1.0	0.57
Aesthetics	RF	1.41	1.0	-0.46
	RF-MAP	1.15	1.0	-0.06
Enjoyment	RF	1.21	1.0	-0.16
	RF-MAP	1.06	1.0	0.05

Setup Our basic neural network (shown in Figure 2) contains three convolutional layers with 8, 16, and 32 filters respectively. Each convolutional layer is followed by a max-pooling layer with a 2×2 field. At the end of the network, we have one fully connected layer with an input dimension of 448 and an output dimension of 1. The total number of parameters in the network is 928. The rectified linear activation function is used for convolutional layers and a linear activation function for the fully connected layer. Our dropout ratio is 0.5. The only input to the basic network is the IMB level map, so it represents a pure form of static analysis. We denote this network as CNN-Map.

To complement static analysis, we extract 4 features from an A* agent run through each level and feed them directly to the fully connected layer without dropout. We call this network CNN-A*. The A* player searches for the fastest path through a level and solves each level without dying. From its search history, we extract the first feature: the number of states expanded divided by the width of the level (1). Although the agent does not die when playing the level, during the search it may visit a state where Mario dies. We count

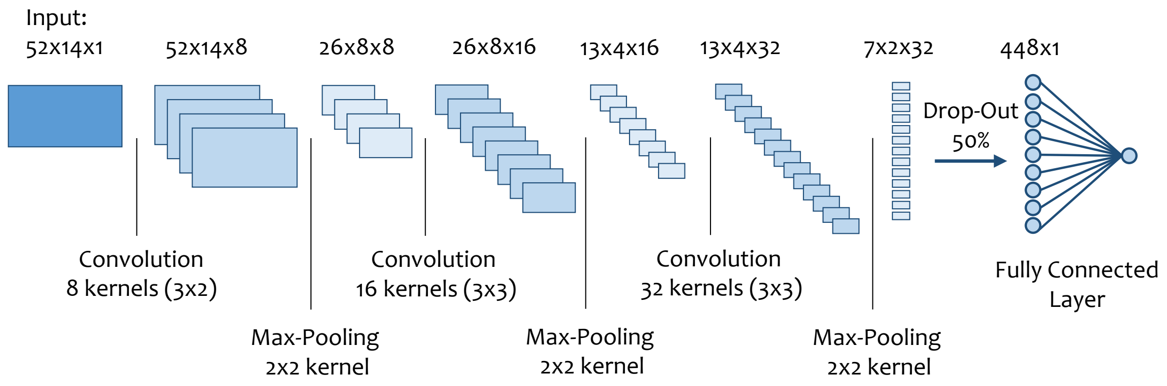


Figure 2: A diagram of our convolutional neural network structure.

these deaths and differentiate between the times of falling in a gap (2) and the number of death caused by enemies (3). Finally, we count the number of enemies killed during playing the game (4). Stomping on an enemy usually slows Mario down, so A* tends to avoid it. Therefore, if A* has to kill an enemy, it could indicate that the enemy is in a position that is difficult to avoid. In sum, these 4 features capture, to an extent, the difficulty of the level from simulated play traces.

In addition, we create a third network (CNN-T) that takes in several level features that are commonly used in the literature: the number of enemies, gaps, power-ups, cannons, and blocks (Pedersen, Togelius, and Yannakakis 2009b; Shaker, Yannakakis, and Togelius 2010). These features are fed as input to the fully connected layer. This represents a traditional "static" approach to level analysis, without requiring any simulated play traces.

The fourth network (CNN-All) takes in all features, including the level map and those used by CNN-A* and CNN-T. All networks use the sum of squared error as the loss function. Training was performed with AdaGrad (Duchi and Singer 2011) and L2 regularization over 1200 epochs.

Lastly, for comparison with a traditional regression technique, we build two random forest regressors. One makes use of only the manually designed A* and level features (denoted as RF), while the other makes use of the manually designed features and the level map (RF-MAP).

Results Table 1 shows the results from the CNNs and Table 2 shows the results from the random forests. We report averages over 20 random splits with the training set containing 80% of levels and the testing set containing 20%. In addition to mean and median absolute errors, we also report the coefficient of determination or R^2, which is a measure of how much variation in the data can be explained by regression. For a number of ground truth values y_i, their mean $\bar{y}$ and corresponding predictions $\hat{y}_i$, R^2 is defined as

$$R^2 \equiv 1 - \frac{SS_{reg}}{SS_{tot}} \equiv 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2} \tag{5}$$

R^2 can be considered as a measure of regression quality. The ground truth mean $\bar{y}$, computed from the test set, provides an uninformed baseline. A large R^2 indicates the regression works significantly better than predicting $\bar{y}$. A negative R^2 indicates that the regression underperforms in comparison to the mean of the ground truth values.

Our best neural networks outperform the RF baselines by 7 percentage points on difficulty, 15 percentage points on aesthetics and 17 percentage points on enjoyment. Interestingly, the A* features and the traditional level features by themselves do not improve performance. However, when combined, these manually designed features improve performance on difficulty by as much as 25 percentage points.

The random forests perform well on difficulty but poorly on aesthetics and enjoyment. This is not unexpected as our manual features are primarily designed for difficulty. Contrasting to our baseline, the results suggest CNN makes more effective use of level map data than random forest.

Discussion Across all conditions, the median errors are better than mean error, suggesting the existence of outlier levels which are difficult to predict. For example, with difficulty, CNN-All's prediction is off by five points or more in two percent of test examples, always under-predicting the difficulty of the levels. Given that 547 levels are rated by a single individual, this is somewhat expected, as individuals may have different understanding of the three criteria.

We observe that CNNs tend to have lower median errors than random forests, but comparable mean errors. This suggests the CNNs pick up useful patterns that appear in most levels, and are more robust under the presence of noisy labels. In many user studies, it is difficult to guarantee every participant understand evaluation criteria in the same way. Thus, robustness under noise is quite valuable.

We can explain the most data variation in difficulty (64%), followed by enjoyment (22%) and aesthetics (only 9%). This is partially due to the fact that aesthetic and enjoyment ratings have smaller variance than difficulty (See Table 3). That is, aesthetic and enjoyment ratings have smaller SS_{tot}. Another possibility is that definitions of aesthetics and enjoyment are fuzzier than difficulty, causing volunteers to rate the levels with different definitions in mind, creating obstacles for prediction. To verify this hypothesis we compute Spearman's rank correlation coefficient on the ratings of the 582 levels rated by two individuals (shown in Table 3). We

Table 3: Characteristics of the three objectives, including variance and Spearman's rank correlation coefficient *rho*. The correlation is computed on 582 levels with two ratings.

Objective	Variance	*rho*
Difficulty	4.07	0.46
Aesthetics	2.33	0.17
Enjoyment	2.31	0.26

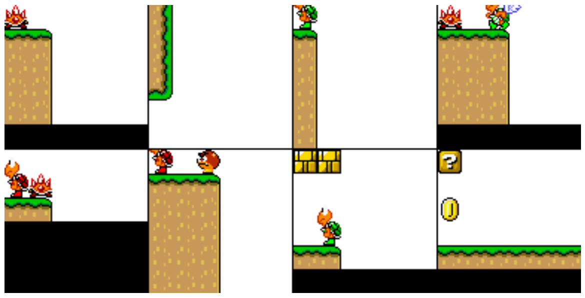


Figure 3: The eight level patches that maximally activate the first eight filters of the second layer of the CNN.

find similar trends between the correlation of ratings and the CNN-All R^2 values, with difficulty having the highest correlation between ratings and aesthetics having the lowest.

The relationships between different sets of features are worth discussing. The A* features and traditional level features by themselves do not improve performance. This may be taken as contrary to our hypothesis that dynamic and static analyses complement each other. However, combining these features yields significant performance gains. Delving deeper into the performance of CNN-All, we found the increase arose from combining two traditional level features, namely the number of enemies and number of gaps, with the A* features. A CNN using the map and these 6 features achieves a mean error of 0.93 and a median error of 0.74. Intuitively, knowing number of deaths due to enemies and gaps is only helpful after knowing how many enemies and gaps exist in a level. This interesting discovery suggests that dynamic features can indeed facilitate static analysis, but designing correct features remains a critical issue.

Qualitative Evaluation To evaluate CNN's ability to extract useful features from raw map levels, we visualize the level patches that maximally activate the learned filters for CNN-All trained to predict difficulty. Figure 3 demonstrates that many filters encode relationships between gaps and enemies. Of particular interest are the last two on the second row. The last encodes "question" blocks, which may contain power-ups and reduce difficulty. The second last shows a koopa under a row of coin blocks. As a koopa occupies two tiles, the blocks prevent Mario from jumping and stomping on the koopa, representing a challenge to the player. These filters suggest CNN is capable of identifying useful representations that predict the difficulty objective.

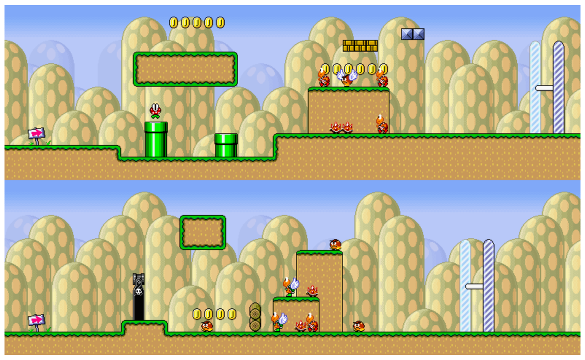


Figure 4: Two examples of levels that are rated as the highest difficulty, but are predicted by our system to be moderate.

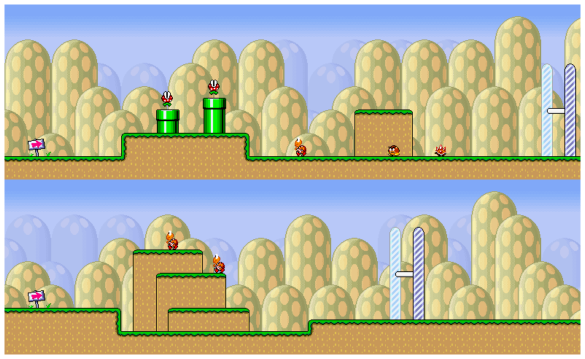


Figure 5: Two examples of levels that are rated near lowest difficulty, but are predicted by our system to be moderate.

We further present examples of levels where our prediction of difficulty failed in Figure 4 and Figure 5. It is generally not straightforward to explain why the network fails on individual cases, but we hypothesize that it is partially due to the A* agent playing too well at the difficult levels. An AI player behaving more similarly to humans may provide some remedy.

Dynamic Analysis with Deep Reinforcement Learning

Dynamic analysis considers the dynamics of the gameplay and actual play traces, so it can provide insights not available in static analysis. However, an obstacle of automatic dynamic analysis lies in the disparity between a human player and a simulated computational player. In this section, we propose a deep reinforcement learning agent that plays like human players with adjustable skill levels.

We argue that the most common challenge faced by a player in a Mario-type game, such as platform games and infinite runners, is not finding the correct strategy. Rather, the challenge lies in controlling the character precisely by pressing the right button at the right time; lack of precision often leads to harm or death. This type of control-dominant game is in contrast to strategy-dominant games like chess

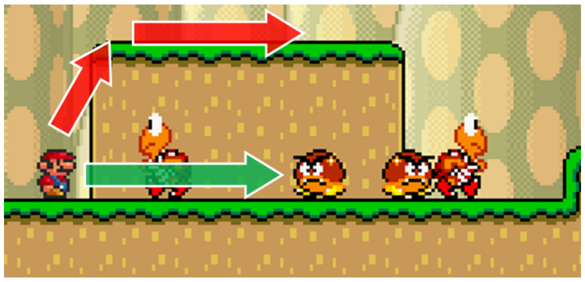


Figure 6: Two possible paths in an Infinite Mario level. An A* player prefers the lower path (green arrow) whereas we expect a human player to choose the upper path (red arrows) most of the time.

or poker. A computer player equipped with perfect control can master a control-dominant game easily. A simple A* algorithm is known to excel at Super Mario Bros (Togelius, Karakovskiy, and Baumgarten 2010).

However, in order to model attributes of a game level as perceived by human players, we need the simulated player to play like a human. Figure 6 illustrates the difference between a human player and an A* player. An optimal A* player, in order to minimize the time spent in a level, takes the lower path and evades or stomps on every enemy, since jumping onto the platform would slow it down. In contrast, we expect a human player who understands his or her own imperfect control to jump on the platform to evade all enemies. We can see a similar scenario in the top level in Figure 5, which suggests that this disparity impacts difficulty prediction.

We propose to model such behavior by giving a reinforcement learning (RL) agent imperfect controls, which are captured by the agent's actions having stochastic effects. In order to cope with stochasticity and maximize reward, the agent would be forced to choose a safe path over a risky one. We formally model a control-dominant game as a Markov Decision Process (MDP). An MDP contains a sequence of time steps $0, 1, 2, \ldots$. At time t, the agent is situated at state s_t and has a set of available actions $A(s_t)$. When the agent executes an action $a_t \in A(s_t)$, the next state s_{t+1} is randomly drawn from a probability distribution $P(s_{t+1}|s_t, a_t)$. The agent then receives a deterministic reward $r_{t+1} = R(s_t, a_t, s_{t+1})$. In a finite horizon setting, the agents aims to maximize its reward $\sum_{k=0}^{T} \gamma^k r_{t+k+1}$ where γ is a discount factor between 0 and 1, and T is the horizon. We seek a solution to an MDP as a deterministic policy $\pi(s) = a$ that maps a state to an action.

We identify timing as the main source of imprecise control faced by human players; we consider pressing the wrong button (e.g., pressing jump when intending for fire/speed) to be rare. This is modeled by the stochastic transition function $P(s_{t+1}|s_t, a_t)$. For example, when the jump action is executed in state s, if the player is on the ground, there is a non-zero probability that s' reflects the state resulted from the player hitting jump a little earlier or a little later. Similar imprecision is modeled for the release of buttons.

Furthermore, we propose a technique for modeling tension in the reinforcement learning framework. This technique is motivated by how humans deal with stressful situations (Rice 1999). When coping with an immediate crisis, the human body releases epinephrine and glucocorticoids, as well as increases oxygen and glucose supply in the blood, which temporarily improves performance. However, sustained stress can damage the body and lead to multiple health risks. The subjective feeling of tension is directly related to high levels of epinephrine.

As a computational analogy, we introduce a "focus" mechanism. The RL agent can choose a focus level when executing an action. A high focus reduces the errors in the control but also creates a negative reward, as captured by the reward function $R(s_t, a_t, s_{t+1})$. This negative reward is better than death but worse than spending a few extra seconds in the level. To maximize its reward, the agent must elevate its focus only when facing a situation that requires precise control and cannot be easily evaded. As a result, we expect the focus level throughout a game to provide a surrogate for a tension curve, a representation of the tension a human player feels over time as they play through a game. In practice, the focus mechanism can can be implemented as having multiple sets of actions, one set for each focus level.

Following Mnih et al. (2013), we propose to use a convolutional neural network to learn a state-action value $Q(s, a)$ using off-policy temporal difference learning and ϵ-greedy exploration. Temporal difference learning bootstraps the learning by adjusting the network's estimate of $Q(s, a)$ based on other estimates.

Conclusions

In this paper, we offer a categorization of game level analysis as static and dynamic analysis. We identify some challenges faced by the two classes of techniques: the design of useful features and creating AI players similar to humans. As a powerful function approximator, a convolutional neural network provides some answers to both challenges. Its use in reinforcement learning can help to create an AI player that imitates humans players with imprecise control.

In our experiments, we utilize a CNN for the purpose of predicting the difficulty, enjoyment, and aesthetics for an Infinite Mario Bros. level. Our network outperforms a strong baseline in the form of a random forest and extracts useful intermediate features automatically. Further, we show features extracted from play traces complement this type of static analysis. Taken together, we present a novel and successful approach to predicting human ratings of gameplay experiences in platformer game levels.

Acknowledgement

The authors thank Jim McCann for valuable discussion and Levi Lelis and Julian Hernandez for providing updated data from the Infinite Mario Bros. user study. This paper contains images generated by the Infinite Mario Bros. game.

References

Bauer, A. W.; Cooper, S.; and Popovic, Z. 2013. Automated redesign of local playspace properties. In *The 8th International Conference on the Foundations of Digital Games*, 190–197.

Berseth, G.; Haworth, M. B.; Kapadia, M.; and Faloutsos, P. 2014. Characterizing and optimizing game level difficulty. In *The 7th International Conference on Motion in Games*, 153–160.

Bjork, S., and Holopainen, J. 2004. *Patterns in game design*. Game Development Series. Charles River Media.

Canossa, A., and Smith, G. 2015. Towards a procedural evaluation technique: Metrics for level design. In *The 10th International Conference on the Foundations of Digital Games*.

Cook, M., and Smith, G. 2015. Formalizing non-formalism: Breaking the rules of automated game design. In *The 10th International Conference on the Foundations of Digital Games*.

Cook, M.; Colton, S.; and Gow, J. 2012. Initial results from co-operative co-evolution for automated platformer design. In *European Conference on the Applications of Evolutionary Computation*, 194–203. Springer.

Cook, M.; Colton, S.; and Pease, A. 2012. Aesthetic considerations for automated platformer design. In *The 9th Artificial Intelligence and Interactive Digital Entertainment*.

Dahlskog, S., and Togelius, J. 2014. A multi-level level generator. In *The IEEE Conference on Computational Intelligence and Games*, 1–8.

Duchi, E. H. J., and Singer, Y. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*.

Horn, B.; Dahlskog, S.; Shaker, N.; Smith, G.; and Togelius, J. 2014. A comparative evaluation of procedural level generators in the mario ai framework. In *The 9th International Conference on the Foundations of Digital Games*.

Hullett, K., and Whitehead, J. 2010. Design patterns in fps levels. In *The 5th International Conference on the Foundations of Digital Games*, 78–85.

Hunicke, R.; LeBlanc, M.; and Zubek, R. 2004. MDA: A formal approach to game design and game research. In *Challenges in Game AI Workshop at the 19th National Conference on Artificial Intelligence*.

Iida, H.; Takeshita, N.; and Yoshimura, J. 2003. A metric for entertainment of boardgames: its implication for evolution of chess variants. In *Entertainment Computing*. Springer. 65–72.

Jain, R.; Isaksen, A.; Holmgård, C.; and Togelius, J. 2016. Autoencoders for level generation and style identification. In *The 2nd Computational Creativity and Games Workshop*.

Lang, K., and Hinton, G. 1988. A time delay neural network architecture for speech recognition. Technical Report CMUCS-88-152.

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R.; Hubbard, W.; and Jackel, L. 1989. Backpropagation applied to handwritten zip code recognition. volume 1, 541–551.

Liapis, A.; Yannakakis, G. N.; and Togelius, J. 2013. Towards a generic method of evaluating game levels. In *The 9th Artificial Intelligence and Interactive Digital Entertainment Conference*.

Marino, J. R.; Reis, W. M.; and Lelis, L. H. 2015. An empirical evaluation of evaluation metrics of procedurally generated mario levels. In *The 11th Artificial Intelligence and Interactive Digital Entertainment Conference*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing Atari with deep reinforcement learning. Technical report, Deepmind Technologies. arXiv:1312.5602.

Pedersen, C.; Togelius, J.; and Yannakakis, G. N. 2009a. Modeling player experience in super mario bros. In *The IEEE Symposium on Computational Intelligence and Games*.

Pedersen, C.; Togelius, J.; and Yannakakis, G. N. 2009b. Modeling player experience in super mario bros. In *IEEE Symposium on Computational Intelligence and Games*, 132–139.

Razavian, A. S.; Azizpour, H.; Sullivan, J.; and Carlsson, S. 2014. Cnn features off-the-shelf: An astounding baseline for recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Reis, W. M. P.; Lelis, L. H. S.; and Gal, Y. 2015. Human computation for procedural content generation in platform games. In *The IEEE Conference of Computational Intelligence and Games*, 99–106.

Rice, P. L. 1999. *Stress and Health*. Brooks/Cole-Wadsworth.

Shaker, N.; Yannakakis, G. N.; and Togelius, J. 2010. Towards automatic personalized content generation for platform games. In *The 6th AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*.

Shi, Y., and Crawfis, R. 2013. Optimal cover placement against static enemy positions. In *The 8th International Conference on the Foundations of Digital Games*, 109–116.

Smith, G.; Whitehead, J.; and Mateas, M. 2010. Tanagra: A mixed-initiative level design tool. In *The 5th International Conference on the Foundations of Digital Games*, 209–216.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. *The Journal of Machine Learning Research* 15(1):1929–1958.

Summerville, A., and Mateas, M. 2016. Super mario as a string: Platformer level generation via lstms. In *The 1st International Conference of DiGRA and FDG*.

Sweetser, P., and Wyeth, P. 2005a. Gameflow: A model for evaluating player enjoyment in games. *ACM Computers in Entertainment* 3(3).

Sweetser, P., and Wyeth, P. 2005b. Gameflow: a model for evaluating player enjoyment in games. *Computers in Entertainment (CIE)* 3(3):3–3.

Togelius, J., and Schmidhuber, J. 2008. An experiment in automatic game design. In *The IEEE Conference on Computational Intelligence and Games*, 111–118.

Togelius, J.; De Nardi, R.; and Lucas, S. M. 2007. Towards automatic personalised content creation for racing games. In *The IEEE Symposium on Computational Intelligence and Games*, 252–259. IEEE.

Togelius, J.; Karakovskiy, S.; and Baumgarten, R. 2010. The 2009 mario ai competition. In *IEEE Congress on Evolutionary Computation*, 1–8.

Tremblay, J., and Verbrugge, C. 2015. An algorithmic approach to decorative content placement. In *The 11th Artificial Intelligence and Interactive Digital Entertainment Conference*.

Tremblay, J.; Torres, P. A.; Brasil, B.; and Verbrugge, C. 2014. Measuring risk in stealth games. In *The 9th International Conference on Foundations of Digital Games*.

Yannakakis, G. N.; Spronck, P.; Loiacono, D.; and André, E. 2013. Player modeling. *Dagstuhl Follow-Ups* 6.