Player Experience Extraction from Gameplay Video

Zijin Luo, Matthew Guzdial, Nicholas Liao, and Mark Riedl

School of Interactive Computing
Georgia Institute of Technology
Atlanta, GA 30332 USA
{zijinluo, mguzdial3, nliao7}@gatech.edu, riedl@cc.gatech.edu

Abstract

The ability to extract the sequence of game events for a given player's play-through has traditionally required access to the game's engine or source code. This serves as a barrier to researchers, developers, and hobbyists who might otherwise benefit from these game logs. In this paper we present two approaches to derive game logs from game video via convolutional neural networks and transfer learning. We evaluate the approaches in a Super Mario Bros. clone, *Mega Man* and *Skyrim*. Our results demonstrate our approach outperforms random forest and other transfer baselines.

Introduction

Analyzing gameplay is a core concern in the video game industry. Game developers can evaluate players experience to help improve game quality. Narrators can present better game commentary by incorporating detailed information. Tournament organizers can utilize the information extracted from gameplay to enhance the viewing experience. Researchers can analyze and compare gameplay to better understand game development and design. However, analyzing gameplay requires access to high quality representations of player experience. Game logs are sequences of player actions and events generated by the game engine and are the standard for representing player experience. However, access to game logs is restricted by technical issues and privacy concerns. Typically only the game's developers can access game logs. Beyond access, game logs are cumbersome given that they require patching the game for any update or change. Further, they cannot be applied outside of controlled instances, for example they cannot help in analyzing the gameplay footage of streamers or be used to understand player behavior for modded or player-created content.

Machine Learning provides one potential solution for this problem. In particular, there exists a large area of work concerned with action recognition for real world video (Soomro, Zamir, and Shah 2012), which one could imagine applying to the problem of extracting representations of player experience from gameplay video. However, such approaches require large amounts of paired training data of video labeled with activities, which restricts this approach to those with the access or resources to create a training dataset.

In this paper, we present two approaches using convolutional neural networks (CNNs) and transfer learning to learn models that convert from gameplay video to logs. Our first model uses a CNN to predict the logs from corresponding frames in the video using a paired dataset. This model learns a set of features to track automatically, cutting back on developer authoring burden, but still requires a large dataset. The second approach consists of two parts: an existing video frame to activity model and a transfer algorithm to adapt the model to recognize and report actions in a new game domain.

The remainder of this paper is organized as follows: first, we cover related work in approximating logs of game events or activities. Second, we overview the naïve CNN approach, trained on a dataset of game video and associated game events, and an evaluation of this system. Third, we cover the transfer learning approach and evaluate it compared to other transfer learning methods. We end with discussion, future work and conclusions. Our primary contributions are the application and evaluation of deep neural networks and transfer learning to the problem of player experience extraction from gameplay video. As a secondary contribution we make available a public dataset of player actions for the game *Skyrim*. To the best of our knowledge this represents the first attempt to apply machine learning to the problem of deriving measures of player experience from video.

Related Work

Player modeling (Yannakakis et al. 2013), the study of computationally modeling the players of games, represents a related field to this research given that it requires representations of player experience. However, most player modeling approaches begin with the assumption that one has access to logs of game events. Towards this end prior player modeling research into Super Mario has relied on the Infinite Mario engine (Togelius, Karakovskiy, and Baumgarten 2010) or recreated unique Mario clones (Liao, Guzdial, and Riedl 2017) to collect game logs. Alternatively, player modeling research has required a partnership or public sharing of game logs by game companies (Drachen, Canossa, and Yannakakis 2009; Sabik and Bhattacharya 2015). In the worst case in terms of researcher effort, player modeling has required parsing gameplay videos by hand to extract events (Hsieh and Sun 2008). Camilleri et al. (2017) look to build

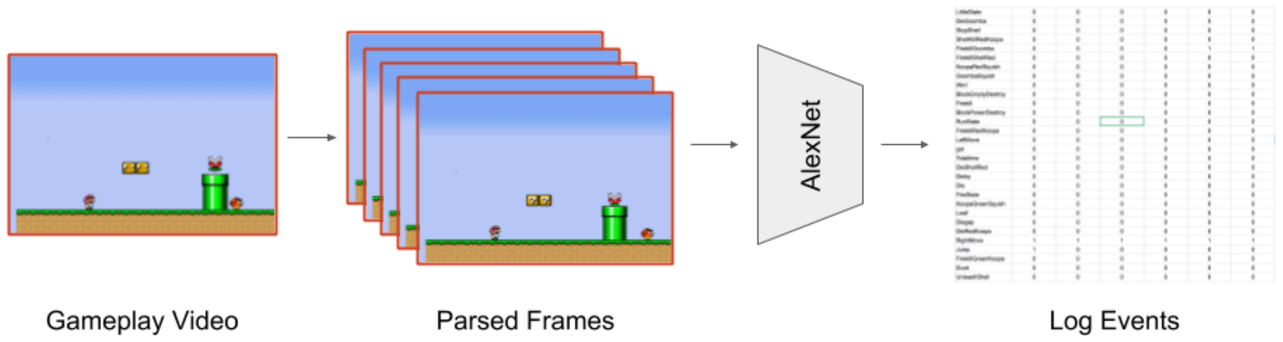

Figure 1: The mapping of gameplay video to its log events by using the naïve approach.

a general model of player affect across different games, but this still requires access to the unique event logging system of each game.

There exist prior approaches to extract design knowledge via machine learning for procedural content generation (Summerville et al. 2017) that makes use of gameplay video as a data source (Guzdial and Riedl 2016). Summerville et al. (2016) extracted player paths from gameplay video, but do not extract any non-movement related events. Guzdial et al. (2017) learn game rules from gameplay video, which as a secondary effect outputs sequences of game events. Both approaches make use of OpenCV (Bradski and Kaehler 2000) to transform from raw pixels of individual frames of a video into a list of game entities and their positions on screen. However, this process requires that users provide a definition of all possible game entities to the system, which must be redefined for every new game.

There exists a large set of applications of activity recognition for real world games and sports (Zhu et al. 2006). However, this is largely enabled by the existence of large datasets of real world human activity (Soomro, Zamir, and Shah 2012). Naïvely, one might think that differences in game aesthetics would indicate a need for individual datasets of this size for each game to apply these methods. However, we demonstrate that one can create high-quality activity recognition models for realistic games with only a small corpus.

To the best of our knowledge this work represents the only general approach for training automatic models for deriving player experience from gameplay video. Jacob et al. (2014) represents the most related example of prior work. The researchers made use of a game-dependent image recognition method to collect logs of actions from gameplay video of *Super Mario World*. The authors do not report results of this process, which would take substantial re-authoring to adapt to another game. Bao et al (2017) present an approach utilizing OpenCV to extract the location of graphical user interface windows. As previously discussed, applying OpenCV to parse video requires defining all possible entities that might be in the video, which makes generality challenging. Fulda et al. (2018) presented an approach to derive player activity labels from gameplay video of *Skyrim* using off the shelf computer vision methods to get captions of current frames and then applied natural language processing methods to classify the activity. However, they presented inconclusive results.

Paired Approach

In this section we present what one might consider the standard (or naïve) solution to the problem of learning a mapping of game video frames to events with convolutional neural networks (CNNs). Namely, collecting a dataset of the frames of gameplay video paired with the active game events for each frame.

Our process is as follows. First, we collect some number of example gameplay videos. Second, we parse this gameplay video into individual frames. For the purposes of this work, we extracted 12 frames per second, as we wanted to minimize the frames that represented the same in-game events. We arrived at 12 frames per second empirically, but our approach is general to any FPS. Third, we pair these frames with the labels for active game events happening in each frame and train an AlexNet neural network (Krizhevsky, Sutskever, and Hinton 2012) to map between video frames and the game event labels. We visualize this process in Figure 1. We chose to make use of AlexNet as it is a well-known CNN architecture with high performance on low quality image classification problems with a large number of classes. AlexNet has been described in more detail in prior publications, but it has five CNN layers and three fully connected layers. We chose to use *Double Cross Entropy* as the loss function, and *Adam* as the optimizer.

To apply AlexNet to this class we alter the size of the fully connected layer to be of size E, where E indicates the maximum number of in-game events. AlexNet's final fully connected layer makes use of ReLU activation, which means each output varies from -1.0 to 1.0. For predicting what final events occurred in a given frame we make use of a threshold of 0.5, marking each appropriate event as active if its index in the vector exceeds this value. We settled on 0.5 after examining the final performance of the trained model before comparing it against the test set. This final layer then becomes a classification over a multi-hot vector of all game events. For example if the only first event were active in a frame we would represent that as $\langle 1, 0, 0...0\rangle$, if only the second event were active in a frame we would represent that

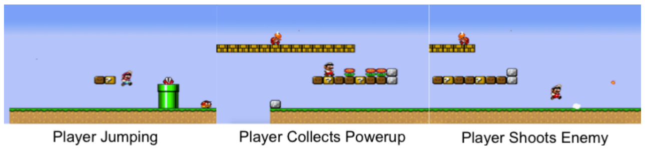

Figure 2: Examples of three frames in Gwario and the corresponding event occurring.

as $\langle 0, 1, 0...0 \rangle$, and so on. It is possible and even likely that many events will co-occur in the same frame. We make use of PyTorch (Paszke et al. 2017) for training our version of AlexNet.

Paired Data Approach Evaluation

In this section we evaluate the application of our paired data approach for learning a mapping from gameplay video frames to game events. To evaluate this approach we make use of a game for which we have access to the underlying game logging system, a games with a purpose (GWAP) clone of Super Mario Bros. called *Gwario* (Siu, Guzdial, and Riedl 2017). This logging system contains thirty possible event types such as an enemy death, a player collecting a coin, etc. We give examples with three frames with one of these event occurrences in Figure 2. Note that these events capture both the player's behavior and non-player actions in the game.

Given that we focus on reducing the authoring burden on those who could benefit from these models we only make use of two instances of gameplay, using two gameplay videos and their associated game logs as our data for this evaluation. This represents a total of 3500 instances of frames paired with the events currently occurring in said frame. Due to the nature of Gwario, which required players to make classification decisions for images of purchasable goods, the majority of game frames had no events. This means that a naïve system that guessed no event occurred for each frame could achieve an accuracy of 88%. For frames with multiple events we calculated partial accuracy for that frame as $1 - x/n$, where x is the number of incorrect event guesses and n is the total number of event guesses. We make use of a five fold cross-validation, which translates to roughly a minute of test video for each video, given our choice of frames per second.

We compare our approach to a random forest given its prior uses as a baseline for similar prior work (Guzdial, Sturtevant, and Li 2016). We make use of the SciPy Random Forest implementation (Jones, Oliphant, and Peterson 2014), which is a 10 tree random forest. To encourage generality we limit the depth to 100 layers. We further make use of a random baseline, which randomly predicts a random number of events (up to the maximum number of co-occurring events) for each frame.

We summarize the results in terms of average test accuracy in Table 1 across all five folds. Across all folds our approach outperforms the random forest baseline by an average of roughly 10%, only exhibiting about 5% test error. Given that there were only thirty possible events, this indicates that on average AlexNet got one to two events incorrect for each frame. Comparatively, the random forest classifier had a test error of roughly 15%, meaning 4-5 incorrect events per frame on average. Guessing no event at all outperforms the random forest baseline, which may run counter to one's expectations. This issue will be general to other games in which players are not constantly performing actions. This points to the difficulty of the problem and better situates the significance of AlexNet.

Table 1: Comparison of the average test accuracy of a 5 fold cross-validation between our approach and three baselines.

AlexNet	Random Forest	Random	No-Event
94.01±0.68	85.91±0.67	0.97±0.83	88.0±0.0

Transfer Approach

Our paired data approach has three major issues limiting its practicality. First, it requires a large, well-labeled dataset. Second, it takes a large amount of training and computation power to train a new model. Third, it does not demonstrate a sufficient level of performance to replace within-engine logging systems, still exhibiting 5% test error. In this section we demonstrate a second approach that uses transfer learning to address these limitations.

Transfer learning typically takes the form of training a neural network architecture on some existing, large, and well-labeled dataset such as ImageNet (Deng et al. 2009). The final layer of this neural network is then retrained to adapt the model to a novel domain, freezing the other layers to leverage the high quality features learned during the initial training. Transfer learning tends to require less training time on the new domain than training from scratch.

A standard transfer learning approach may not suffice. Consider the case of trying to train a frame-to-events model for Gwario by transfering a model from another, more general dataset such as ImageNet. ImageNet has 1000 label classes, requiring a neural network with a final layer of size 1000. To transfer the model to Gwario, which has 30 event types, a standard transfer learning approach would replace the network's final layer with one of length 30 and retrain just the final, fully-connected layer on available Gwario log data while holding all other weights constant. However, we would not anticipate the same features from ImageNet, composed of images of real world objects and animals to work well with the pixel graphics of Gwario.

To address the issue of mismatching features we use a specific transfer learning method called *student-teacher learning*, sometimes called *knowledge distillation* (Wong and Gales 2016; Furlanello et al. 2018). The student-teacher method replicates one network's weights into an independent network with a distinct and, typically, smaller architecture. One only trains the "teacher" model once, which can then be used as the basis for an arbitrary number of "student" networks, which require a comparatively smaller dataset and training time. We visualize this approach in Figure 3. The intuition behind this approach is that you are giving the net-

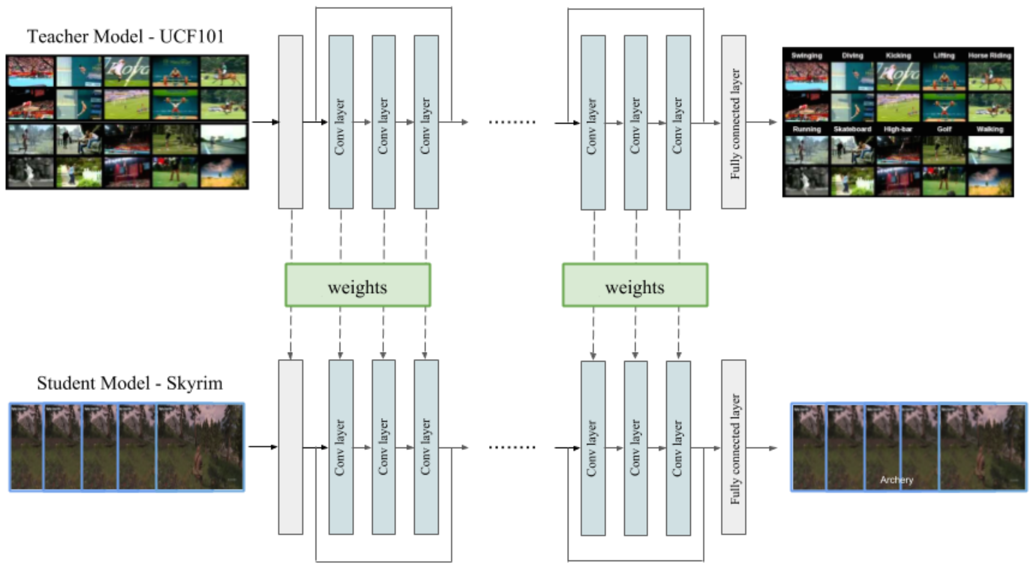

Figure 3: The mechanism of student-teacher method: creating a student model from the teacher model by copying the weights from it and retrain on the new dataset

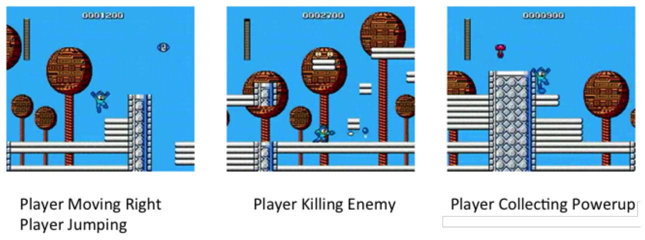

Figure 4: Examples of frames tagged with actions from the *Mega Man* dataset.

work a starting point in a related domain (e.g. *Super Mario Bros.*) and then training it as usual to adjust it to a related domain (e.g. *Megaman*). The difference between this and more standard transfer learning method is that the entirety of the student network is retrained after transfer, allowing the system to learn new features. Notably student-teacher approaches typically involve going from a larger to a smaller network, but this is not a requirement.

This approach helps to overcome the limitations of the naïve supervised approach to log generation. First, it requires a much smaller dataset for the target game. Second, it requires a much shorter training time to converge. Third, student-teacher networks typically demonstrate higher performance than training on the same dataset naïvely (Furlanello et al. 2018).

Transfer Evaluation: Mega Man

We investigate the application of this transfer approach to a pixel-based game that parallels the naïve supervised data approach. We took a single gameplay video of a single level of *Mega Man* from the Nintendo Entertainment System and

Table 2: Comparison of the average accuracy of our student-teacher transfer learning approach for Mega Man to three baselines.

Student-Teacher	adaptation	Random	No-Event
80.92±0.06	80.09±0.05	15.50±0.02	73.78±0.0

hand-coded it with five of the thirty event types from our paired data approach evaluation. These were *player moving right*, *player moving left*, *player jumping*, *player shoots enemy*, and *player collects powerup*. We visualize four instances of these events across three frames in Figure 4. We then ran an evaluation with a 80-20 train-test split, using the AlexNet model trained on Gwario from the naïve supervised data approach evaluation as the teacher and a new AlexNet model as the student network.

We evaluate against two baselines. For the first we compare against domain adaptation (Tommasi and Caputo 2013), in which backpropagration was used to train the same architecture on a combined dataset of the five shared classes in Gwario and *Mega Man* videos. This technique is much slower but represents a naïve approach. For the second baseline we make use of the same random guessing baseline from the prior evaluation. Our *Mega Man* video was downloaded from YouTube and represents expert play. Thus, there were few frames without action, but always guessing no event could still achieve an accuracy of roughly 74%.

We summarize the results in Table 2. The student teacher performs the best, but only roughly 0.8% better than the domain adaptation approach. This is a small improvement over the paired data approach, but this may be due to the fact that Gwario was not similar enough in aesthetic to *Mega Man* to

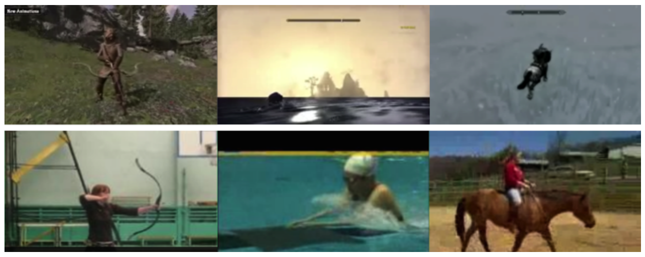
Figure 5: Comparison between three Skyrim actions (top) and UCF-101 actions (bottom). From left to right archery, breaststroke, and horse riding.

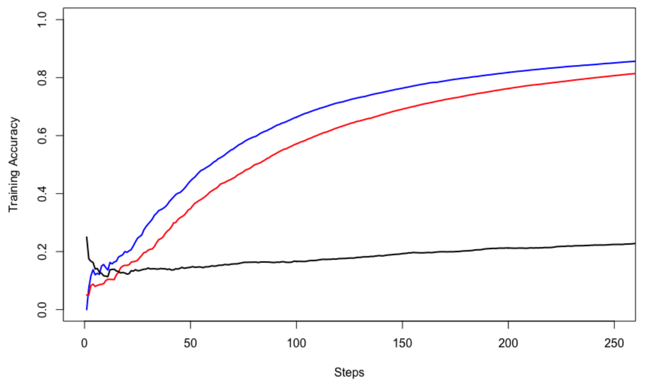

Figure 6: The training accuracy over the first 250 epochs for the student-teacher approach (blue), ImageNet transfer baseline (red), and domain adaptation baseline (black).

fully harness the potential of the student-teacher approach. However, the student-teacher approach only took a single epoch to converge, making it far more time efficient than the paired data approach and domain adaptation baseline.

Transfer Evaluation: Skyrim

In the previous section we demonstrate the difficulty in transferring a model from one sprite-based game to another sprite-based game. That makes sense when logs are available for both games. In this section we show that our transfer learning approach can be used to generate logs for a relatively photorealistic game—*Skyrim*—given a real world video dataset. We trained a teacher network on the UCF-101 dataset (Soomro, Zamir, and Shah 2012), which is a comprehensive video dataset of 101 human behaviors. We made this choice given that it allows our teacher model to obtain an excellent basis for real human action recognition. For our teacher and student neural network architecture we make use of the resnet-152 model (He et al. 2016), given that it is popular and has known high-quality performance. We lack the space for a full description of resnet-152, but note that it is a two-stream 152-layer CNN (Simonyan and Zisserman 2014a), meaning it takes both raw pixel images and the difference between each pair of frames as input. While this is a very large neural network, it is less complex than the similarly popular VGG net (Simonyan and Zisserman

Table 3: A comparison of our student-teacher transfer learning approach compared to an ImageNet baseline and a naïve domain adaptation approach.

	UCF-101	ImageNet	adaptation
50-50	**99.92**±**0.08**	99.87±0.10	74.78±18.94
66-33	**99.99**±**0.02**	99.94±0.05	83.91±13.04
83-17	**100.00**±**0.02**	99.96±0.06	90.40±13.04

2014b). Further, due to its popularity it is possible to find pre-trained resnet-152 models, further reducing the computation burden.

For our transfer domain, we use the game, *Skyrim*, a popular 3rd person perspective role-playing adventure game in which players can engage in a wide variety of activities. We further chose it as our domain for the evaluation given that there exists prior research that could benefit from a more accurate activity model of *Skyrim* (Fulda, Murdoch, and Wingate 2018).

We compiled ten, five second gameplay clips of ten player activities in *Skyrim* from YouTube, which we make publicly available.[1] These activities are: archery, breaststroke, crossbow, dance, dodge, fly, horse riding, run, skydiving, and waving weapon. These activities are also ten of the one-hundred and one activity labels present in UCF-101, which in theory should improve the transfer learning performance. We visualize three of these activities in Figure 5 for both UCF-101 and *Skyrim*. Notably, the *Skyrim* frames include graphical user interface elements, and include some modded or player-created elements.

The procedure to apply the student-teacher method was straightforward. We first trained our resnet-152 teacher model on UCF-101, which obtains a testing accuracy of 81% on the held-out UCF-101 test batch. We then prepared another resnet-152 as the student model. As shown in the Figure 3, we copied the weights from the teacher model to the student model and re-sized the last fully connected layer to match the ten labels in our new dataset for the target game. Then, we trained the student model on the new dataset.

We make use of two baselines. For the first we compare against domain adaptation, in which backpropagration was used to train the same architecture on a combined dataset of the ten shared classes in UCF-101 and our Skyrim videos. As a second baseline we compare against the same architecture first trained on ImageNet and then applied transfer learning to retrain the final layer to classify for our ten cases, which represents a typical transfer learning approach. We ran 50-50, 66-33, and 83-17 train-test splits across our Skyrim video dataset. The last split is atypical—one might instead see an 80-20 split—but it fit more evenly given the size of our dataset.

We compare the average results with standard deviation for all three splits in Table 3. Given there were only ten possible activities and no overlap a random baseline would average around 10% test accuracy, which all approaches outper-

[1]https://github.com/IvoryCandy/Skyrim-Human-Actions

formed. Our approach outperformed both other approaches consistently, performing near-perfectly. ImageNet transfer has similarly high accuracy, but distribution of accuracies across all test data was greater.

Figure 6 represents the average training accuracy for every training step for all three approaches across all splits. The teacher-student network is in blue, the ImageNet baseline is in red, and the domain adaptation approach using backpropagation is in black. This demonstrates that the student-teacher approach converged faster than the ImageNet baseline. However, both approaches eventually converged to nearly 1.0 training accuracy.

Our approach for deriving player activities for realistic games shows excellent performance, even only given five training videos for each of the target activities. The ImageNet-based transfer approach also performed well, which does not benefit from the similarity in real human and humanoid 3D game characters. We take this as a positive sign, that there exists a large amount of high quality datasets one could pull from for this video-to-log task. It also demonstrates the ability for CNNs to adapt between real world features and games with realistic aesthetics quickly.

Discussion

We present two approaches in this paper for going from video to a representation of player experience. The best version of our approach in terms of performance was our student-teacher approach to *Skyrim* with UCF-101 as the teacher training set. This demonstrates the importance of large, high quality datasets. Compare this to the Gwario evaluation and Mega Man evaluations, which drew on only two videos as their dataset or teacher network training set. We anticipate that the datasets one has access to will have a major impact in the success of these approaches. In particular, we anticipate the higher performance of *Skyrim* was due to the fact that general, real world features translated to photo-realistic 3rd person games better than sprite-based games where game art is highly stylized.

We acknowledge that such deep neural network approaches to function approximation for gameplay video to player experience will never be as accurate as access to a game engine logging system of game events. However, the results of this paper indicate that it can be a viable alternative in cases in which one has the resources to tag gameplay video as in the first approach or an existing trained model and some smaller tagged dataset as in the second approach. However, we do not advocate for the use of either approach in parsing single videos of a specific player experience, given that there will be some amount of error or noise. Instead we anticipate applying this approach when a researcher, hobbyist, or developer wishes to parse a large amount of gameplay video to extract overall trends, or other cases where the error rate's impact has less impact.

We note that training an initial teacher model can be computationally intensive. We found that our resnet-152 teacher model for both UCF-101 and ImageNet took several days to initially converge, even training on high-powered GPUs. We anticipate researchers will have more success finding pre-trained models and using these as a teacher model, given that training the student network can take as little as a single epoch. However, this is still a limitation on the current work given our desire to make parsing gameplay video to extract game events and player activities more accessible.

Future Work

Our results prove that we can develop video action recognition models for some video games. However, our transfer learning approach relies on the existence of an existing, high-quality dataset or a pre-trained model on such a dataset. This is a limitation, especially when it comes to visual aesthetics. The majority of high-quality datasets in existence for image recognition are composed of realistic images, which limits the potential application of these datasets to games with a pixel art, cartoon-like, or generally unrealistic visual aesthetics, given that we anticipate very different features. Even with transfer learning, there is still a need for powerful computation to train the initial model. For future work we intend to explore the possibility of stylistically altering these realistic datasets to particular game aesthetics.

We anticipate even with the performance of our current techniques that there are many possible applications. For example, we imagine applying these approaches as a means of generating features for game human subject studies in which one does not have access to the underlying game engine as in (Fulda, Murdoch, and Wingate 2018). Alternatively, we anticipate that such a system could be applied to extract player experience measures for applications into player modeling, for example determining categories of player types for new games or genres (Yannakakis et al. 2013). Further, given the speed of a neural network at test time, we anticipate the ability to apply this system to applications of live game commentary or shout-casting, either human or artificially intelligent (Dodge et al. 2018).

Conclusions

In this paper we present two approaches for learning a model that converts from gameplay video to representations of player experience. Our first approach represents a standard application of convolutional neural networks to this problem, training on a dataset of gameplay video frames paired with game events. Our second approach to derives a model through transfer learning on small sets of tagged videos. We evaluate these models for a Super Mario clone, *Mega man* and *Skyrim*, and present a corpus we developed for the latter. Our results demonstrate that both approaches stand as reasonable approximations of true game logs. We hope that researchers may apply these methods to further all areas of games research.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. IIS-1525967. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

Bao, L.; Li, J.; Xing, Z.; Wang, X.; Xia, X.; and Zhou, B. 2017. Extracting and analyzing time-series hci data from screen-captured task videos. *Empirical Software Engineering* 22(1):134–174.

Bradski, G., and Kaehler, A. 2000. Opencv. *Dr. Dobbs journal of software tools* 3.

Camilleri, E.; Yannakakis, G. N.; and Liapis, A. 2017. Towards general models of player affect. In *Affective Computing and Intelligent Interaction (ACII), 2017 Seventh International Conference on*, 333–339. IEEE.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 248–255. IEEE.

Dodge, J.; Penney, S.; Hilderbrand, C.; Anderson, A.; and Burnett, M. 2018. How the experts do it: Assessing and explaining agent behaviors in real-time strategy games. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 562. ACM.

Drachen, A.; Canossa, A.; and Yannakakis, G. N. 2009. Player modeling using self-organization in tomb raider: Underworld. In *Computational Intelligence and Games, 2009. CIG 2009. IEEE Symposium on*, 1–8. IEEE.

Fulda, N.; Murdoch, B.; and Wingate, D. 2018. Threat, explore, barter, puzzle: A semantically-informed algorithm for extracting interaction modes. In *Proceedings of the 1st Knowledge Extraction from Games Workshop*. AAAI.

Furlanello, T.; Lipton, Z. C.; Tschannen, M.; Itti, L.; and Anandkumar, A. 2018. Born again neural networks. *arXiv preprint arXiv:1805.04770*.

Guzdial, M., and Riedl, M. 2016. Game level generation from gameplay videos. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Guzdial, M.; Li, B.; and Riedl, M. O. 2017. Game engine learning from video. In *26th International Joint Conference on Artificial Intelligence*.

Guzdial, M.; Sturtevant, N.; and Li, B. 2016. Deep static and dynamic level analysis: A study on infinite mario. In *Experimental AI in Games Workshop*, volume 3.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hsieh, J.-L., and Sun, C.-T. 2008. Building a player strategy model by analyzing replays of real-time strategy games. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, 3106–3111. IEEE.

Jacob, L. B.; Kohwalter, T. C.; Machado, A. F.; Clua, E. W.; and de Oliveira, D. 2014. A non-intrusive approach for 2d platform game design analysis based on provenance data extracted from game streaming. In *Computer Games and Digital Entertainment (SBGAMES), 2014 Brazilian Symposium on*, 41–50. IEEE.

Jones, E.; Oliphant, T.; and Peterson, P. 2014. {SciPy}: open source scientific tools for {Python}.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Liao, N.; Guzdial, M.; and Riedl, M. 2017. Deep convolutional player modeling on log and level data. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 41. ACM.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Sabik, A., and Bhattacharya, R. 2015. *Data-driven Recommendation Systems for Multiplayer Online Battle Arenas*. Ph.D. Dissertation, Masters thesis, Johns Hopkins University.

Simonyan, K., and Zisserman, A. 2014a. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, 568–576.

Simonyan, K., and Zisserman, A. 2014b. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Siu, K.; Guzdial, M.; and Riedl, M. O. 2017. Evaluating singleplayer and multiplayer in human computation games. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, 34. ACM.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Summerville, A.; Guzdial, M.; Mateas, M.; and Riedl, M. O. 2016. Learning player tailored content from observation: Platformer level generation from video traces using lstms. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Summerville, A.; Snodgrass, S.; Guzdial, M.; Holmgård, C.; Hoover, A. K.; Isaksen, A.; Nealen, A.; and Togelius, J. 2017. Procedural content generation via machine learning (pcgml). *arXiv preprint arXiv:1702.00539*.

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

Tommasi, T., and Caputo, B. 2013. Frustratingly easy nbnn domain adaptation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, 897–904. IEEE.

Wong, J. H., and Gales, M. J. 2016. Sequence student-teacher training of deep neural networks.

Yannakakis, G. N.; Spronck, P.; Loiacono, D.; and André, E. 2013. Player modeling. In *Dagstuhl Follow-Ups*, volume 6. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Zhu, G.; Xu, C.; Huang, Q.; Gao, W.; and Xing, L. 2006. Player action recognition in broadcast tennis video with applications to semantic analysis of sports game. In *Proceedings of the 14th ACM international conference on Multimedia*, 431–440. ACM.