Player Modeling using Gameplay Videos

Abhijeet Krishnan
North Carolina State University
akrish13@ncsu.edu

Abstract

Player modeling allows us to create rich, personalized experiences for players of digital games. Most work on the subject requires access to game states in the form of player logs, or direct game engine access. Prior work has been done to identify player actions from gameplay videos. We present a technique to build a player model using gameplay videos from the video game *Dishonored*. The model is built by training an LSTM on the gameplay videos. We describe the architecture of the LSTM and the results obtained.

Introduction

Player modeling is the study of computational models of players in games (Yannakakis et al. 2013). It is a computational representation of the player which can be used to predict player experience. Prior work on player modeling has required access to gameplay logs, often obtained from the game itself (Drachen, Canossa, and Yannakakis 2009) or obtained via player surveys (Pedersen, Togelius, and Yannakakis 2010). This data is not easy to obtain. Gameplay logs require access to the game engine, and since most games with a large player bases are commercial in nature [1], their code is closed-source and cannot be used for data gathering unless published by their owners. Player surveys are labor-intensive to collect.

Gameplay videos represent an rich source of data for modeling players. The popularity of livestreaming on platforms like Twitch [2] and YouTube Gaming [3] allow us to leverage a large number of videos on many different games for player modeling. Prior work by Luo et. al. (Luo et al. 2018) focused on utilizing gameplay videos to identify player actions in a game. We use gameplay videos from the video game *Dishonored* to model players using an overlay model (Polson and Richardson 1988). We retrain the Inception v3 ConvNet (Szegedy et al. 2015) on images generated from the gameplay videos. We pass the features of this model to an LSTM (long short term memory) which can classify the sequential frames.

Copyright © 2019, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1] https://www.githyp.com/?type=steam-player-counts&sort=alltime

[2] https://www.twitch.tv/

[3] https://www.youtube.com/gaming/

Background

Dishonored

Dishonored is a 2012 stealth action-adventure video game developed by Arkane Studios and published by Bethesda Softworks (Wikipedia contributors 2019). A central mechanic in Dishonored is the concept of ”chaos”. Players can choose to play a level stealthily; without alerting any enemies and without violence - a low chaos playstyle. Players can also choose to engage most enemies and kill them - a high chaos playstyle. The choice of playstyle affects minor story beats throughout the game and also impacts the ending seen by the player.

Inception v3 ConvNet

The Inception v3 ConvNet is a convolutional neural network (CNN) developed by Google which is trained on the ImageNet (Deng et al. 2009) data set. It achieved 1st runner-up position in ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2015. It consists of only about 42 layers and requires much fewer parameters than similarly-performing architectures like VGGNet (Simonyan and Zisserman 2014). Since it is a powerful model, we can retrain it on our data set to obtain features which can be passed to another model, thus reducing the number of parameters we need to train.

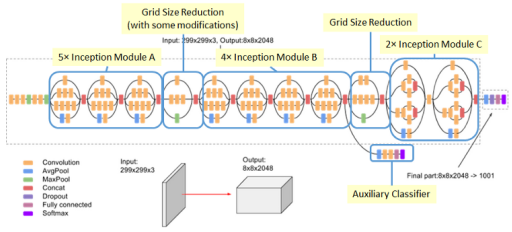

Figure 1: Inception v3 ConvNet Architecture

LSTM

LSTMs are a type of recurrent neural network (RNN) which operate on sequential data (Andrej Karpathy 2015). They allow us to classify sequences of data. A video is essentially

a sequence of frames, and hence we can use an LSTM to classify them.

Approach

Objective

We wanted to get a classifier which would be able to correctly classify gameplay videos of *Dishonored* into two classes based on whether it was a high chaos or low chaos gameplay style.

Data collection

We utilized data from two walkthrough playlists made by the YouTube channel WikiGameGuide [4]. We utilized the first five parts of both playlists. The videos were split into train/test in a 3:2 ratio. They were downsampled to 320p and then split into individual frames.

Retraining Inception v3

We follow the approach presented in (TensorFlow). We save the softmax and pool layers for each frame to disk for use with the LSTM. We convert the individual features for each frame into a sequence of frames to train the LSTM with. We empirically choose a sequence length of 40 frames per sequence.

Training the LSTM

The architecture of the network is a single LSTM layer with 256 nodes. This is followed by a dropout of 0.2 to help prevent over-fitting and a fully-connected softmax layer to generate our predictions (Matt Harvey 2016). The total number of trainable parameters was $34,612,738$. The model was trained for 100 epochs with a batch size of 1. Training time took approximately 10 minutes on the Google Colab platform with a GPU runtime.

The code for our implementation can be found at the link [5]

Results

Discussion

Challenges

Small size of data set This data set is too small to achieve meaningful results on. A more extensive data collection effort would need to be made to obtain a more robust model. This also contributes to the low accuracy of the model and high variance in loss during training. Spikes in the accuracy are almost certainly due to overfitting the data.

Improvements and Future Work

The model would significantly benefit from more data. Future work will include using newer models like Inception v4 and more powerful LSTM networks.

[4] https://www.youtube.com/channel/UCCiKcMwWJUSIS_WVpycqOPg

[5] https://colab.research.google.com/drive/1URqYIjs-8sQ17O3sDF-JghrUkwzcv5lg

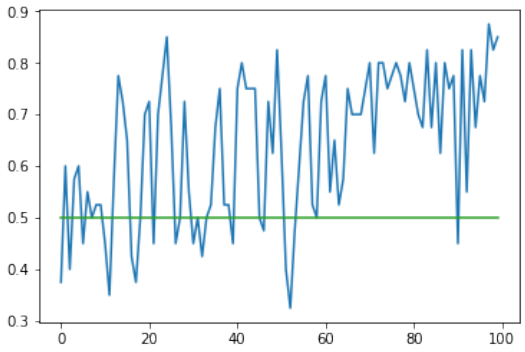

Figure 2: Training graph (0.5 baseline)

References

Andrej Karpathy. 2015. The unreasonable effectiveness of recurrent neural networks. [Online; accessed 23-April-2019].

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

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

Luo, Z.; Guzdial, M.; Liao, N.; and Riedl, M. 2018. Player experience extraction from gameplay video. In *Fourteenth Artificial Intelligence and Interactive Digital Entertainment Conference*.

Matt Harvey. 2016. Continuous video classification with tensorflow, inception and recurrent nets.

Pedersen, C.; Togelius, J.; and Yannakakis, G. N. 2010. Modeling player experience for content creation. *IEEE Transactions on Computational Intelligence and AI in Games* 2(1):54–67.

Polson, M. C., and Richardson, J. J., eds. 1988. *Foundations of Intelligent Tutoring Systems*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc.

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

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2015. Rethinking the inception architecture for computer vision. *CoRR* abs/1512.00567.

TensorFlow. How to retrain an image classifier for new categories. [Online; accessed 23-April-2019].

Wikipedia contributors. 2019. Dishonored — Wikipedia, the free encyclopedia. [Online; accessed 23-April-2019].

Yannakakis, G. N.; Spronck, P.; Loiacono, D.; and André, E. 2013. Player modeling. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.