Watch and Learn
Applications of Time Distributed Deep Learning to Super Smash Brothers Melee

Austin Lachance
SSBM Tag: Tinder
CPSC490: Senior Thesis Proposal
Advisor: Professor Dragomir Radev
Submitted: April 25, 2018
Abstract

Super Smash Brothers Melee (SSBM) is a classic fighting-style video game. Unlike many previously examined games like Pong, SSBM features a tremendous move set, dynamic cameras, and a real opponent whose actions are unknown to the player. In this project I will demonstrate the abilities and limitations of a deep neural network as it trains an agent to navigate this complex environment. More specifically, I utilize time-distributed deep neural networks to perform imitation learning. I show that this form of learning is able to quickly train an agent to adopt a human player’s most frequently used inputs and then integrate the model into a system that allows the agent to play in real time.

Figure 1: EVO 2016’s SSBM tournament featured 2300 participants and awarded first place over $14,000

Figure 2: Screen capture from the gameplay of SSBM players Mango and Armada as they compete in the Winner’s Finals of the Apex 2012 tournament.

(1) Introduction

Shortly after being released as a family game for the GameCube in 2001, SSBM developed a widespread competitive scene that continues to become even more extensive every day. I have been a fan of SSBM for some time both as a player and a spectator of the professional gaming scene. The game poses several interesting challenges to machine learning that will be addressed in this project. The high level goal of my work was to develop a deep neural network capable of training an agent to play SSBM at a competent or even competitive level. My initial focus was on a reinforcement learning implementation that would utilize self-play and Q-Learning in order to train in an unsupervised setting. However, after further investigation it became clear that with the available resources and projected timeline that this solution had little hope of producing favorable results. For that reason, this paper will focus on my development of an imitation learning agent. I hope to demonstrate the advantages and limitations
of time distributed convolution neural networks as a tool for video classification and real time performance in a multi-agent environment.

(2) Problem Description
As stated previously, the overall goal of the project was to create a model to imitate human gameplay on SSBM. This task, however, was comprised of several unique challenges that had to first be approached. In order to predict the controller input of the human player in such a complex environment the game was treated as a video classification problem. Specifically, my project focused on examining the data as sequences of images. By doing this the problem transitions to one of computer vision and temporal feature detection. My project thus serves as a tool capable of receiving user gameplay and developing a network specific to that players distinct play style.

(3) Data & Data Preprocessing
The process for creating the data used in this project was divided into three distinct parts: video collection, reduction of the input space, and image sequence generation. I will discuss each of these briefly, but my main focus was on the latter two.

(3.1) Video Collection & Image Creation
The data sets used in this work were derived from screen recordings of a human player. My colleague, Benjamin Rodriguez-Vars, recorded himself playing several hours of SSBM against varying levels of CPU. He then separated these video recordings into individual frame images that were saved at a rate of 15 frames per second (Note the game itself runs at 60 frames per second).

A training version of the game available online, 20XX, contains a "debug mode" allowing players to place a graphic in the upper corner of the screen that displays the user’s controller inputs in real time. The recordings were made in this mode so that a quick look at the graphic in each image would tell exactly which inputs were being pressed at that time. It was these inputs that were used as the labels for the training data.

(3.2) Reducing the Input Space
In order to reduce the complexity of the input several precautionary measures were taken with regards to in what type of gameplay was recorded and how its labels were generated. First, the character selection and stage remained constant throughout all of the recordings. The human controlled character was chosen to be Falco while the CPU was Captain Falcon. These characters were selected due to their distinct shapes and colors and as a result of strong intra-community bias in favor of watching and playing exciting characters. The stage chosen was Battlefield. This is a fairly neutral stage that remains static throughout the course of the game. In the professional SSBM community it is one of only 6 legal stages in competitive play. The stage was further simplified by adding an optional game setting that restricts the camera to a static view of the entire stage rather than the default dynamic camera. This was done in order to keep the relative sizes of the characters and stage elements static throughout gameplay keeping the focus of the data on the character positions and actions. Our hope was that these choices would limit noise and allow the network to detect the more important features.
Next, the range of inputs was limited for the purposes of this project. A GameCube Controller has 2 sticks, each with a wide range of possible inputs, as well as 10 buttons and 2 triggers. In order to reduce the total number of classes the network would have to output, the inputs were reduced to 6 angles for the first stick and 4 for the second, 4 buttons, and only 1 trigger. Further eliminating certain undesirable combinations of buttons left the total possible classes at 39. The original and reduced controllers are shown in figure 3.

(3.3) Generating Image Sequences

Once I had the set of images and its corresponding labels I grouped the images into sequences, recreating brief intervals of time within the gameplay. This was done in order to provide the model with temporal information. Rather than a single snapshot of gameplay, each sample of the data was constructed to be a series of snapshots giving a richer input and providing additional information such as character
velocities. The challenge in generating these sequences was the sheer volume of data generated. Depending on the stride, sequences heavily overlapped causing the same images to be generated several times as members of many nearby sequences. For example, choosing to have 10 frames in each sequence and a stride of 1 causes every image to be copied and stored a total of 10 times, massively increasing the size of the dataset. As a result, I avoided loading the data set in its entirety and instead generated batches of sequences on the fly whilst training. I held the number of frames in each sequence constant throughout training and while I tried different options such as 4, 6, and 10 images per sequence, I feel that this area still warrants further exploration and that the number of frames per sequence as well as the stride determining the amount of sequence overlap can be further optimized to better performance.

(4) Time Distributed Convolution Neural Network

The model I worked on was a time distributed convolutional neural network (CNN). This was selected in order to take full advantage of the data and extract temporal information from the images. The network consists of several convolutional and pooling layers acting in parallel. Each image in an image sequence is run through these layers independently. Next the outputs of each of these CNN layers were flattened and run through several fully connected layers eventually producing a softmax output with confidence scores for each of the 39 possible output classes.

I created this model due to its ability to train on multiple input images in parallel. This allowed me to provide the image sequences as input. Running each image through the CNN layers separately allowed the network to extract feature information about them without interference. Combining the outputs and running them through the fully connected layers then allowed the network to see how these features were altered between the images. This elucidates otherwise invisible temporal information that is crucial to SSBM. It is not only useful for the model to know the static positions of the two players, but also the relative velocities of the players and other temporally inferred information.

(5) Live Gameplay

For this portion of the project the game was run using the Dolphin emulator. Recent versions of Dolphin provide support for the transmission of input through named pipes. By creating a pipe and altering the configuration files of Dolphin itself I was able to write commands directly from Python to the pipe and then wait as Dolphin read from the pipe buffer and used its contents as the controller input. As the game ran I collected screenshots at a regular frame rate and kept an active queue.
of the latest *images per image sequence* number of images. The images in this queue comprised the image sequence that was fed into the time-distributed CNN. Its output was then written to the pipe, providing Dolphin with the next input command to perform in the live game. A great deal of help in getting this to function properly was received from the *libmelee* library as well as the Phillip AI.

**Results**

The results seen immediately were very promising. The accuracy of the network showed a great deal of success, while prediction values were exceptionally skewed. The output classes for the moves ‘null’, ‘left’, and ‘right a’ were consistently predicted at a significantly higher rate than other classes. This is due to the fact that the data itself was similarly skewed. While there were a large number of potential inputs to the game, many of these inputs are very situational and not used regularly. Additionally, given the speed at which images were captured it is impossible for a human player to provide inputs to the game at all time steps. As a result, ‘null’ dominated the input data and while it is a useful state, less seen inputs such as moves to help the character recover were overlooked by the network. Thus, the network was very successful in identifying the most common inputs and input sequences performed by the human actor; however these inputs were not what one might expect to see.

I attempted to mediate this issue by selecting from a pool of the 3 highest confidence output classes. However, after seeing little change I decided to raise this to the 5 highest and opted to select one of these 5 options at random a portion of the time. This helped to provide a wider range of character inputs to the game.

**Conclusion**

Ultimately, the imitation model saw little success in accurately replicating the behavior of a human player; however, it was successful in identifying and reproducing the inputs produced most frequently by the player. The model points out some underlying issues in the data that suggests that further success could be seen with a more balanced data set.

**Future Work**

While the input space was reduced a great deal, there still remain some factors that would allow the space to be further reduced and likely improve performance. The background of the chosen stage is a dynamic ‘starry sky’ that contains a variable number of stars, star locations, and a small range of background color. This was unable to be removed without affecting the necessary features including the characters and platforms. Successfully eliminating this background would reduce the input even further and place a greater emphasis on the features of interest. Additionally, exploring a wider range of hyperparameters including the fine details of the way in which the image sequences are generated certainly stands to ameliorate the model’s performance. Lastly, I would like to explore the possibility of transitioning the images into other color spaces and examining how this affects the ability of the network to detect feature and extract temporal information.

**Acknowledgements**

I would like to thank Professors Radev and Zucker for all of their help and advice. I would also like to thank Ben Vars for coming up with an amazing project idea and playing such a crucial role in its success. Finally I would like to thank Theo Epstein for introducing me to Super Smash Brothers Melee and the amazing community of players and fans surrounding it.
References


Figure 2. Heat maps showing highest points of activation representing locations in the image most strongly influencing the networks’ classification decisions. Including the user input graphic in training data causes unwanted activation. Left: Heat map on image with user input graphic. Activations are only on the graphic itself. Right: Heat map on image with graphic removed.