Imitation Learning with Vanilla and Time Distributed CNNs

Ben Rodriquez-Vars, Austin Lachance, and Dragomir Radev PhD

Department of Computer Science, Yale University

LILY Lab

Introduction

Video games are a popular space of recreation and an even more exciting place to push the limits of machine learning. A major goal for many computer scientists is to create artificial intelligence that can outperform any human. Q learning and developing complex policy gradients have proven to yield fantastic results, however in this project my focus is quickly teaching an agent how to reasonably play the game Super Smash Bros. Melee through vision. We aim to show that by feeding an agent gameplay footage that resembles a human player's optimal performance, a machine learning model can produce an agent capable of replicating the proper actions for each state. Our strategy map each frame of the game, screen-captured images of gameplay, to game-moves, which act as a label, much like an image classification problem. We develop two distinct models to accomplish this task, both of which make use of a Convolutional Neural Network (CNN) with an AlexNet architecture to process frames and output moves in real time to simulate gameplay. The first model uses a Vanilla CNN and bases its output on the previous frame, whereas the second model uses a time-distributed CNN that takes as input a sequence of previous frames.

We hope to show that this form of learning is able to quickly train an agent to adopt a human player's most frequently used inputs. We then integrate the models into a system that allows the agents to play in real time against a live opponent.

Materials and Methods

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. The game was played using a Dolphin emulator allowing us to leverage QuickTime Player to record our computer screen. The footage was then split up into images at 15 frames per second(fps), versus the 60fps it was recorded at. A special version of the game allows us to place a graphic in the upper corner of the screen that displays the user's controller inputs in real time. The labels for each image were created by checking this graphic.

In order for the model to form strong correlations in the data we reduced the input space by limiting gameplay to one distinct character matchup and a single stage. We further reduced complexity by choosing characters with unique shapes and colors, a static stage, and a fixed camera setting.

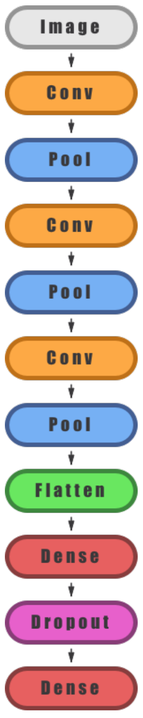

Figure1. Vanilla CNN

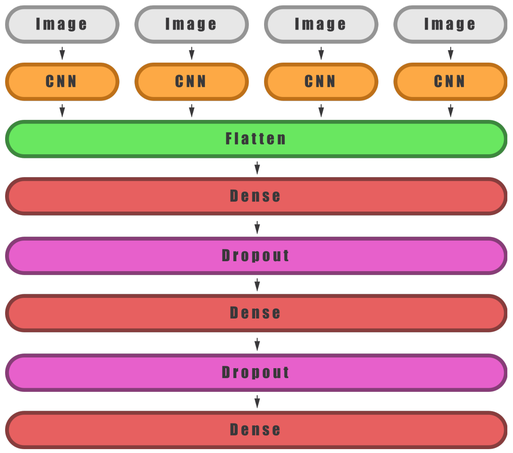

Figure 2. Time-distributed neural network design

Figure 3. Falco Sprite

Figure 4. Captain Falcon Sprite

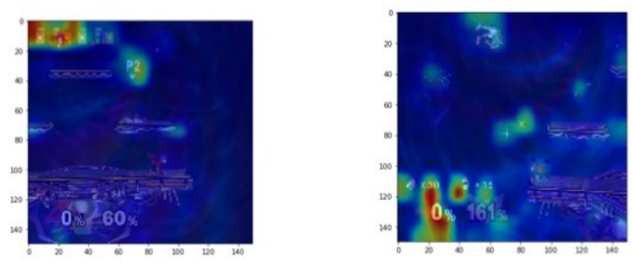

Figure 5. Heat maps showing highest points of activation representing locations in the image most strongly influencing the networks' classification decisions.
Left: Including the user input graphic in training data causes unwanted activation.
Right: Heat map on image with graphic removed solves this issue.

Fixed Camera	Image Augmentation	HUD	Loss	Accuracy
No	No	Yes	0.04	98.8%
No	No	No	2.33	47%
No	Yes	No	1.25	45%
Yes	No	No	2.06	55%
Yes	Yes	No	1.28	52%

Table 1. Results of different models based on variations of input training data

Model Descriptions

The first model utilizes a Vanilla AlexNet structure consisting of 3 convolution and pooling layers followed by a flatten layer and 3 fully connected layers. This design is shown in figure 1.

The second model implements a time distributed version of AlexNet. By taking sequences of images, representing intervals of time during the gameplay, the network aims to capture temporal information. The model sends each image in a sequence through the CNN portion of the network independently. It then collects and flattens the outputs and uses them as input to the fully connected layers that follow. This structure is shown in figure 2.

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.

Conclusion

This project showed CNN's ability to be able to help a model replicate appropriate actions when given footage of desired performance. Considering that the differences between moves were based on a character that took up such a small portion of very large image (1800x2188 pixels), that the stage background was dynamic, and that the number of output classes was so large, we were pleased with the model's accuracy. It could be improved by either altering the game to solve the above issues or by simply collecting much more data. Despite all of these obstacles, the agent was able to identify and reproduce the inputs produced most frequently by a human player in real time. This helps show the validity of making learning into an image classification problem for a lightweight and data-driven solution.

Acknowledgement

We would like to thank our advisor, Professor Dragomir Radev, for his assistance throughout the semester as well as Super Smash Bros. Melee for the inspiration and hours of fun.