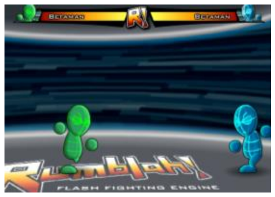
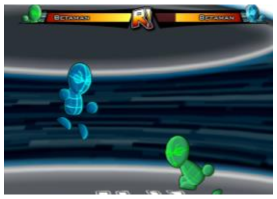
Learning to Fight: Deep Learning Applied to Video Games

Artificial Intelligence (AI) is commonly used in video games to control non-human "computer" players. The AI used in video games often relies only on simple logic and heuristics, so "computer" players do not always have human-like playstyles. This is unfortunate because games that feature human-like AI could be very popular amongst both recreational and competitive gamers. AI trained to play like specific humans could be particularly appealing. Imagine the headlines to market such games: "Play Grand Theft Auto cooperatively with AI trained by celebrity Snoop Dogg" or "Fight against AI trained by World Champion Player Mew2King".

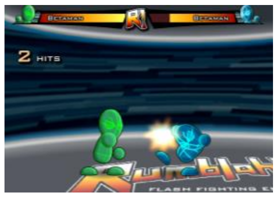
Inspired by the work of Stanford PhD students Chen and Yi [1], MSiA students Dustin Fontaine, Dylan Fontaine, Annie Didier, and Michael Cho set out to use Deep Learning to create a human-like video game AI. Their approach relied on image-classification instead of reinforcement learning which is more commonly used with video games. The team started by recording a player (Dustin) playing the game "Rumblah: Flash Fighting Engine", a simple fighting game where two players move around the screen and hit each other until one gets knocked out. The team recorded approximately 1 hour of gameplay, capturing 10 screenshots per second and labelling screenshots according to which key Dustin was pressing while the screenshot was taken. The four possible labels for screenshots were "Move Left", "Move Right", "Punch", and "Do Nothing" (if Dustin was not pressing any keys). Some example screenshots are shown below. Dustin was controlling the green character.



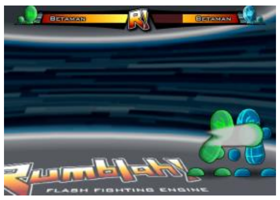
"Move Right"



"Move Left"



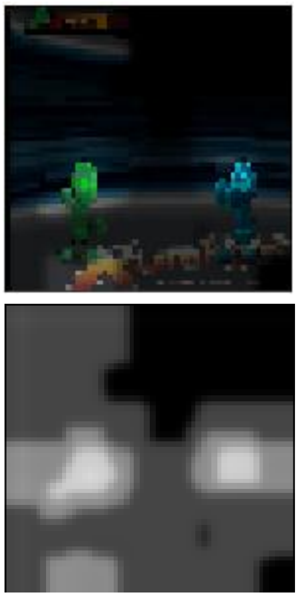
"Punch"

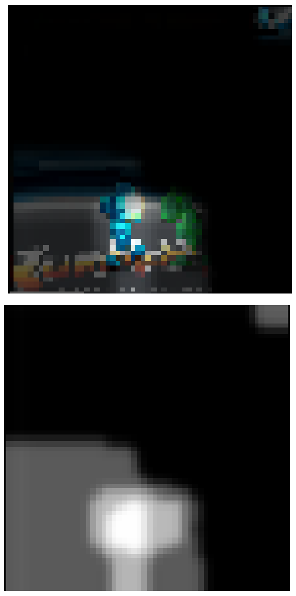


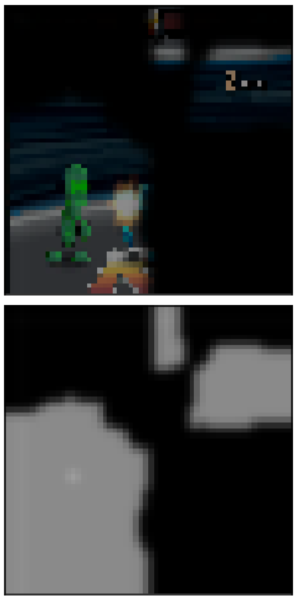
"Do Nothing"

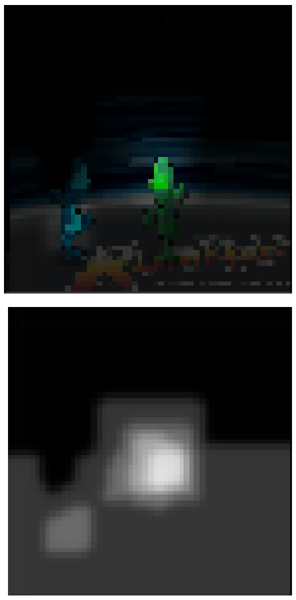
The next step towards creating the AI involved fitting a neural network to the collected screenshots. If the team could build a model that could accurately label new screenshots according to what key Dustin would press in that scenario, it could be used to control an AI that plays the game like Dustin. They started by fitting a sequential neural network to screenshots and then fitting a more robust convolutional neural network. The convolutional model performed better than the sequential model, achieving nearly 60% accuracy and 80% top-2 class accuracy. This may seem low, but it is not bad when you consider the randomness of human behavior and the fact that Dustin may does not always press the same keys every time in a given situation.

In order to understand why the neural network made the classifications that it did and what parts of each screenshot the model was focusing on, the team used a visualization technique involving heatmaps. The white pixels in each heatmap were given large weight by the model when it was making its classification and the black pixels were given little (or no) weight. See the example heatmaps below. It is interesting to see that the models focused on the same parts of the screen that a human does when playing. In the screenshot classified as "Move Right", the focus is on the head of each player. This makes sense, because a player considers the relative positions of their character and the opponent when deciding to move left or right – if the opponent is out of attack range then the player will move towards them. In the screenshot classified as "Punch", the focus is again on the heads of both players and also the flash from a previous punch. Since the opponent is within the green character's attack range, the model makes the correct decision to throw a punch. Most of the time the model focused on the heads of each player, but sometimes other information was considered. In the screenshot classified as "Do Nothing" you can see that the combo meter at the top-right of the screen was given some weight by the model.

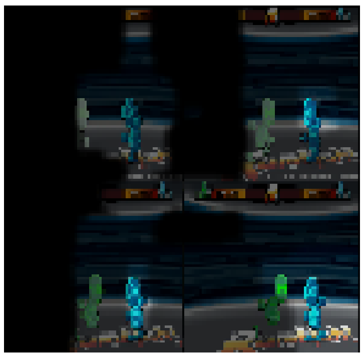

"Move Right"

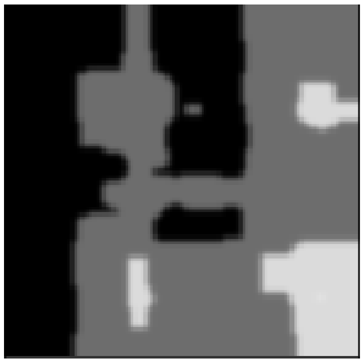

"Punch"


"Do Nothing"


"Move Left"

A human player considers more information than just the current screenshot when deciding which button to press, so shouldn't a model trying to imitate human behavior do the same? After fitting the sequential and convolutional models to single images, the team decided to fit the same models to "filmstrips" of four consecutive images. "Filmstrips" capture information from the last few moments of gameplay instead of just the current moment. This technique was successful and the filmstrip models performed slightly better than the original models.

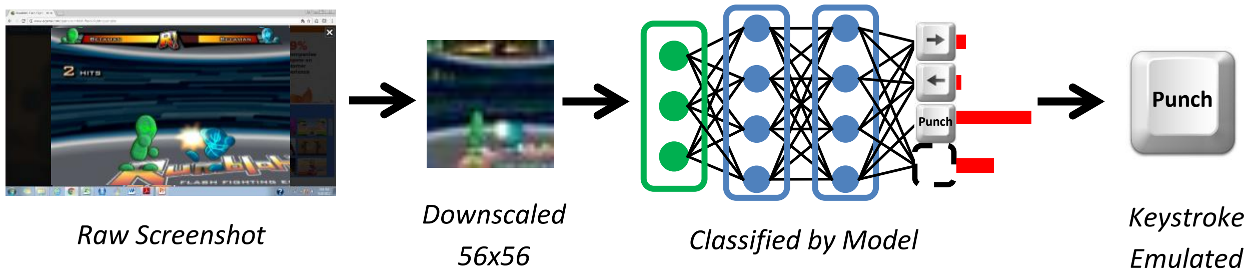




The heatmaps produced by the filmstrip models make a lot of sense. More weight was given to pixels on the screenshot representing the current moment (bottom-right) and less weight was given to pixels on the screenshots taken prior to the current moment.

Finally, after fitting various neural networks to recordings of Dustin's gameplay, the team needed to operationalize their models to control an AI in the game. The AI would be controlled by a script that records screenshots of live gameplay, scales the screenshots to the appropriate size, plugs them into a neural network, then takes the output from the model (which key to press) and emulates keystrokes on a keyboard. The diagram below represents this process.

The team was worried that a script of this sort would not be fast enough to keep up with live gameplay. However, speed turned out to not be an issue. The Python script was able to execute in real time on a standard personal laptop processing screenshots and emulating keystrokes in a fraction of a second.



A video of the team's AI in action is shown below. The blue player is controlled by a human and the green player is controlled by the team's AI. You can see that the AI plays well, chasing the blue player around the screen and punching him repeatedly when he is within attack range. Although the AI does not perfectly copy Dustin's playstyle, this demo serves as a good proof-of-concept for using image classification models to train video game AI. More training data, further tuning, and providing more information as inputs to the model (such as the time series of previous keys pressed) could make the model more accurate. The same framework could be applied to any action game that does not involve complex strategy, so it would be interesting to see how the model performs in other settings.

Embedded Video: https://www.youtube.com/watch?v=hZuv2RaDGE8

[1] https://arxiv.org/pdf/1702.05663.pdf