USING DEEP LEARNING TO PLAY PENGUIN TAG

JR CABANSAG, CABANSAG@STANFORD.EDU

INTRODUCTION

AI systems in the videogame industry are mainly created through decision trees or reinforcement learning techniques [2][3], however, I wanted to branch off of these techniques and explore the capabilities of deep learning in creating videogame AI. In this project, I used data from my gameplay and ran it through a shallow fully-connected neural network to successfully create an AI enemy that emulated my playing style.

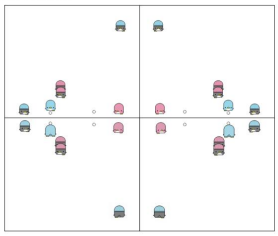
Penguin Tag is a two-player game where players play blue and pink penguins facing off against each other. Players must avoid getting tagged by the enemy penguins, which are the grey penguins wearing a beanie opposite of their color. Players can shoot snowballs at these enemies to change them to their team, and can also shoot snowballs at the enemy player to have a chance of slowing them down. The only inputs needed from the player are the 4 keys for movement (W,A,S,D, or arrow keys), as well as the G key or ALT key to shoot a snowball.

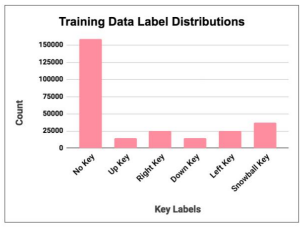
DATASET

The training data used was 2 hours of recorded gameplay, while test data was 10 minutes of recorded gameplay. Ten times a second, the game would record the game state as a feature vector, and also record my current key input at the time, to be used as the correct label for the game state. The reasoning for this was to create an AI that given a game state, would perform the same key input that I would [1].

Each feature vector contained 50 features total: each of the player penguins' x and y coordinates, speed, direction, and life counts, each of enemy penguins' x and y coordinates, directions and team, as well as up to eight snowballs' x and y coordinates and directions. Each label was a number from 0 to 5, to represent each of the 6 different key inputs (no key, up key, right key, down key, left key, and snowball key).

To increase the amount of data acquired, I performed data augmentation by flipping the game states horizontally, vertically, and both horizontally and vertically. This quadrupled the amount of data collected, and also trains the network to perform the same actions in mirrored game states.





MODELS

There were two main model architectures used. One was a shallow fully-connected network with three hidden layers, each having 50 neurons. The second model architecture was a deep fully-connected network with eight hidden layers, each with 10 neurons. Three mini-variations of networks were created, each addressing the label imbalance in the training data in different ways:

All Data: Trained on all of the data
Actionable Data: Trained only on data where the user pressed a key
Balanced Data: Trained on all the data, with costs normalized by class frequency

Training Data	No Key	Up Key	Right Key	Down Key	Left Key	Snowball Key	Total Data Accuracy	Average Accuracy	Actionable Accuracy
Shallow Network, All Data	0.999	0	0.001	0	0	0	0.577	0.1666666667	0.0002
Shallow Network, Actionable Data	0	0.174	0.423	0.164	0.419	0.705	0.189	0.3141666667	0.377
Shallow Network, Balanced Data	0.012	0.343	0.373	0.317	0.385	0.283	0.149	0.2855	0.3402
Deep Network, Balanced Data	0.02	0.196	0.176	0.179	0.217	0.333	0.203	0.1868333333	0.2202

Test Data	No Key	Up Key	Right Key	Down Key	Left Key	Snowball Key	Total Data Accuracy	Average Accuracy	Actionable Accuracy
Shallow Network, All Data	0.99	0	0.004	0.009	0	0	0.606	0.1671666667	0.0026
Shallow Network, Actionable Data	0	0.15	0.486	0.219	0.313	0.743	0.175	0.3185	0.3822
Shallow Network, Balanced Data	0.007	0.356	0.403	0.324	0.319	0.267	0.131	0.2793333333	0.3338
Deep Network, Balanced Data	0.03	0.167	0.182	0.219	0.187	0.324	0.149	0.1848333333	0.2158

PLAYING AGAINST THE AI

A webpage was created allowing us to play against each of the network models. This served as the qualitative judgment of the network's performance.

EASY MODE

Shallow Network, All Data
- outputted no key for most of the game states

Deep Network, Balanced Data
- probabilities were practically fixed, performed similar to an AI with random decisions

Shallow Network, Balanced Data
+ runs away from enemies, targets them as well
- accidentally runs into enemies while trying to attack

Shallow Network, Actionable Data
+ aggressively pelts snowballs
+ runs away from enemies and defends teammates
+ chases player

HARD MODE

FUTURE WORK

In the future, I'd like to perform a more thorough hyperparameter search for the deep network to have it fit to the data better, and perhaps create an AI that has the same frequency of no-key inputs as I do, while still being a challenging opponent. I'd also want to explore ResNets and see how an enemy trained on sequential data would perform.

REFERENCES

[1] "Learning to Fight: Deep Learning Applied to Video Games | Northwestern MSiA | Student Research", Sites.northwestern.edu, 2018. [Online]. Available: http://sites.northwestern.edu/msia/2017/09/19/learning-to-fight-deep-learning-applied-to-video-games/. [Accessed: 20- Mar- 2018].

[2] "Epic's Tim Sweeney: Deep Learning A.I. Will Open New Frontiers in Game Design", Medium, 2018. [Online]. Available: https://medium.com/@Synced/epics-tim-sweeney-deep-learning-a-i-will-open-new-frontiers-in-game-design-5682ad32454c. [Accessed: 20- Mar- 2018].

[3]"Designing Artificial Intelligence for Games (Part 1) | Intel® Software", Software.intel.com, 2018. [Online]. Available: https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-1. [Accessed: 20- Mar- 2018].