Machine Learning for N-Dimensional Spatial Reasoning Tasks on the Web

Blake Moody, Jiehyun Kim, Sanghyuk Kim, Daniel Haehn

UMass Boston
UNIVERSITY OF MASSACHUSETTS BOSTON MACHINE PSYCHOLOGY

Abstract

Spatial reasoning is key to solving complex tasks in dynamic, high-dimensional spaces. Yet, current models are resource-intensive and rely heavily on human input. We introduce Snake-ML, a web-based simulation tool that provides an efficient, intuitive platform for developing spatial navigation strategies. It uses unsupervised learning and integrates computer vision for target localization and boundary detection, improving stability and performance. Quantitative analysis shows a **4.58x training speedup**, highlighting Snake-ML's potential for applications in robotics, autonomy, and AI.

Introduction

We introduce our tool Snake-ML, a web-based tool for simulating the training of models to navigate higher-dimensional spaces using various unsupervised learning methods. Snake-ML addresses the dependency issues mentioned by utilizing well-established web standards like Web GL and ECMAScript, allowing it to run entirely on the browser. This allows large fleets of diverse computing devices to train models independently of a centralized server. Snake-ML is modeled after the classic arcade game 'Snake,' while being generalize to n-dimensions.

Methodology

To compare performance, we measured the time for a single update game simulation on both the edge and cloud frameworks. For benchmarking, we used an Asus Zephyrus G15 with an AMD Ryzen 9 5900HS (8-core, 3.3 GHz) CPU, 16GB DDR4 of memory, and an NVIDIA RTX 3070 (8GB GDDR6). Four hundred samples were collected, covering 20 different batch sizes with 20 trials per batch size.

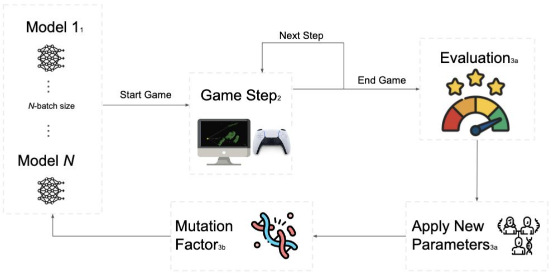


Figure 1: The schematic overview of the internal training loop of Snake-ML. Implemented using Genetic Learning Algorithm (GA). 1) First, the population of snakes is initialized with random weights. 2) Then, the games are all run in parallel and do not conclude until all of the games are over. 3a) Evaluate the fitness scores of each snake candidate and pass the weights of the best snakes onto the next generation of snakes. 3b) Apply a mutation factor to encourage behavior exploration. 4) Repeat.

WIP Demo:
snake-ml.moody.mx
<- Check out our Repository!

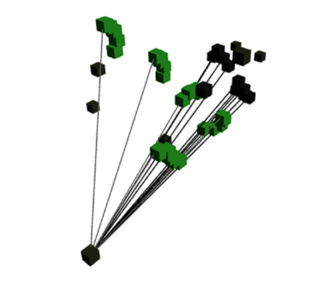


Figure 3: A batch of snakes (green) being trained in the browser. The black lines depict each snake's displacement vector to its targeted food (brown).

Table 2: Edge vs Cloud training speed benchmark across batch sizes.

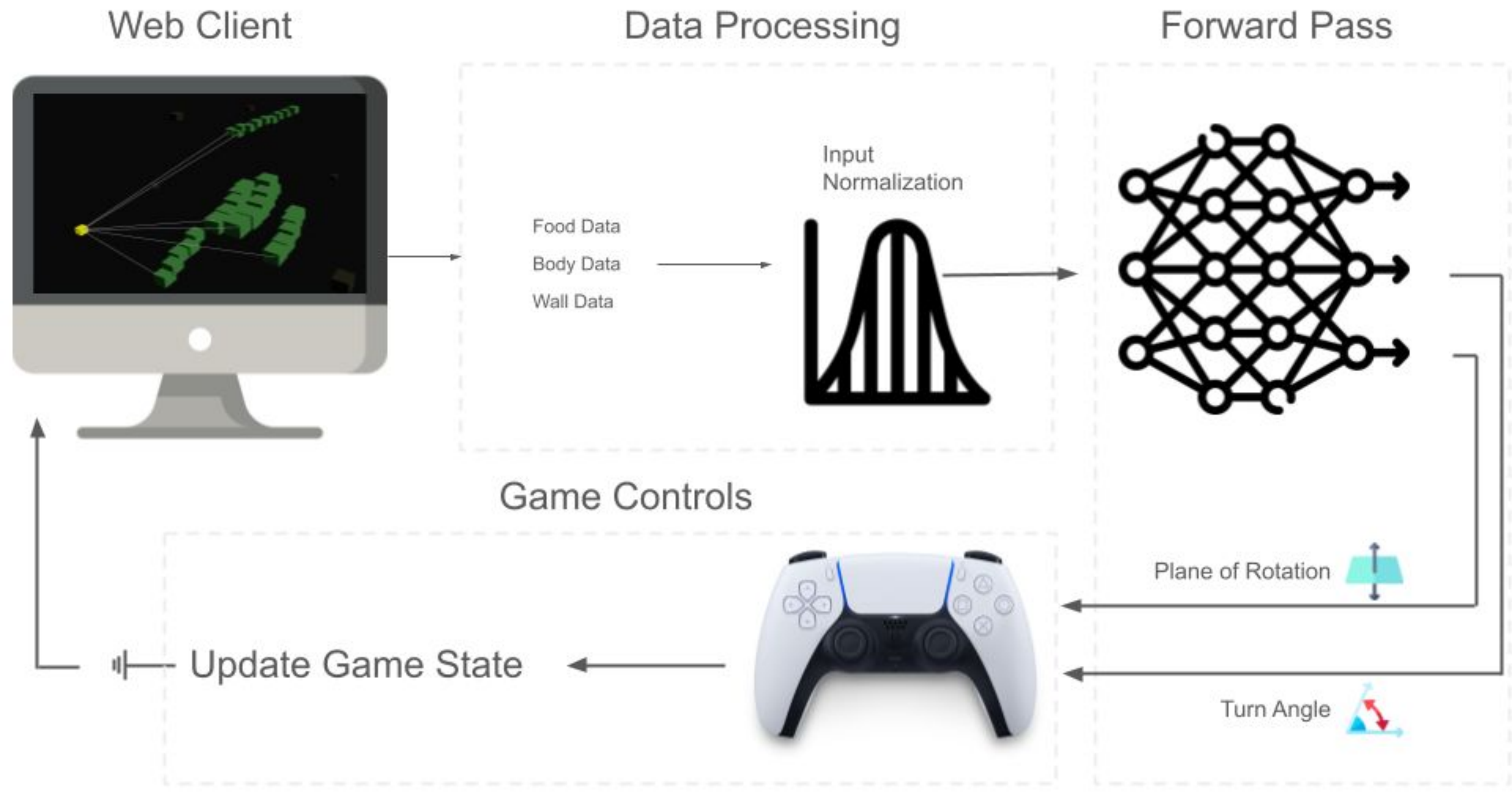
No. of Snakes	Cloud (ms)	Edge (ms)	Edge Speedup
1	1	18	0.08×
100	143	45	3.19×
200	287	74	3.87×
300	430	107	4.02×
400	575	137	4.19×
500	722	169	4.28×
600	871	203	4.29×
700	1016	244	4.17×
800	1159	278	4.17×
900	1300	318	4.09×
1000	1447	335	4.32×
1100	1600	362	4.42×
1200	1737	402	4.32×
1300	1878	425	4.42×
1400	2025	474	4.27×
1500	2174	497	4.37×
1600	2336	510	4.58×
1700	2461	538	4.58×
1800	2606	583	4.47×
1900	2766	617	4.48×
Average			**4.03**×
Max			**4.58**×

Results

As shown in Table, the difference in the average update times between the edge and cloud frameworks increases as the batch size increases. Comparing both cases, we found that the edge framework achieves, on average, 4.03x speedup over the cloud framework and up to 4.58x overall. For the sake of this comparison, the cloud framework was executed on the same device as the client that renders the training, so we assumed that network latency was negligible. However, in practical applications, network latency is expected to be a more significant factor in cloud computing than edge computing.

Conclusion

This paper introduces a new web-based framework for training and simulating models to play the snake game in n-dimensions. We implement a genetic learning algorithm to train the model without requiring computationally expensive gradient calculation. Additionally, we perform heavy data augmentation as a pre-processing step before feeding the augmented data into our equivariant neural network. This approach leads to a size reduction in the overall parameter size of the neural network and notable hardware performance gains.


Web Client
Data Processing
Forward Pass
Input
Normalization
Food Data
Body Data
Wall Data
Game Controls
Plane of Rotation
Update Game State
Turn Angle