Using Digital Image Processing Techniques to Train a Neural Network

Andrew L. Hanshaw, Joel Q. Quanbeck, Genzo Namikawa, Kwame T. Ampofo

Abstract—This project aims to train a neural network to play a simple video game using purely visual input. This project will use digital image processing to identify various visual aspects of the game.

Index Terms—Digital image processing, High performance computing, Neural networks.

I. Introduction

NEURAL networks can be trained to perform many different tasks. Unfortunately, this training often requires direct access to internal variables of the program the neural network is being trained with, which are often unable to be monitored directly unless the original source code is available. This project aims to train a neural network to perform a task using visual cues only, by using digital image processing techniques to analyze images and extract useful information.

The program the neural network will be trained on is the dinosaur game built into Google Chrome, Fig. I. The simplicity of this game, both in controls and in visuals, should allow for relatively easy training of the neural network and identification of the visual elements of the game. The SDSU "Roaring Thunder" computing cluster [1] will be used to run the game and train the neural network. Parallelizing this task is necessary to quickly create and simulate each generation so the network can evolve as quickly as possible. Using this cluster will allow for each node of a given generation to be run on different cores, speeding up the process of generation significantly.

Previously, this game has been used to train neural networks, Fig. I, [2]. However, these previous attempts recreated the game to provide a neural network with important variables such as game speed and distance from obstacles. This project aims to extrapolate these variables by visually identifying and tracking the obstacles and background before feeding them back into the neural network. This type of gameplay emulates how a human would play the game, where the player makes inputs based on the feedback they receive from the visuals of the game.

Based on the knowledge gained on using image processing to feed neural networks, further research could be done to train neural networks on visual data from other sources. Some examples include advanced robotic vision and weather anomaly detection. These examples are much more computationally intensive and require more advanced
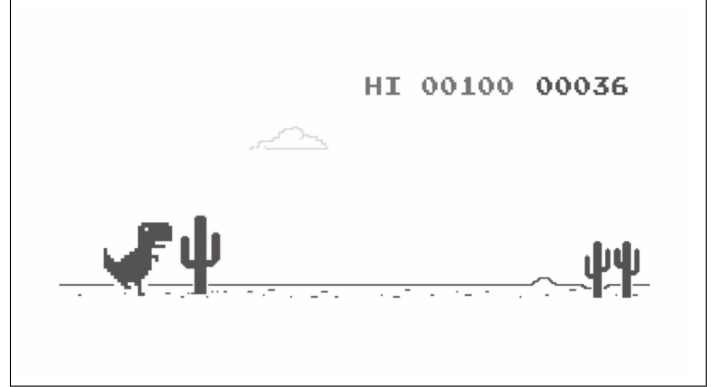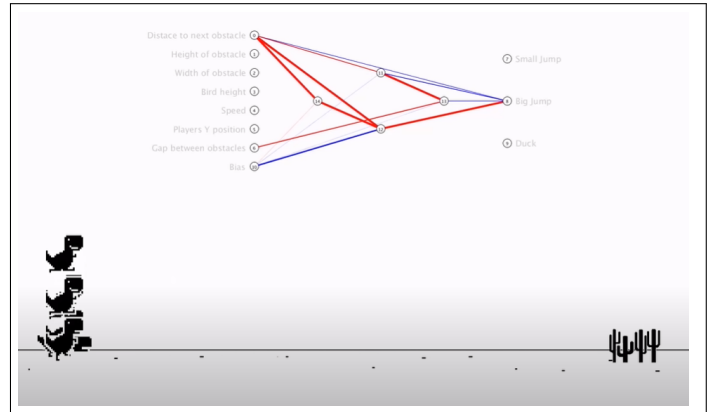

Fig. 1. The Google Chrome dinosaur game.

Fig. 2. A neural network training on a copy of the Chrome dinosaur game.

computer vision algorithms to solve their respective problems. Further research into feeding a neural network based on a virtual recording of a software program without internal variable access could allow for AI manipulation of more complex video games or automated crawling of webpages from the perspective of a user.

In Section II the necessary background information used when creating this project is described. Subsections include software overview, project challenges, and high performance computing. Conclusions for this paper are presented in Section III.

II. Background Information

A. Software Overview

For this project, Tensorflow was preliminarily selected to create the neural network. It's ubiquitous use in the field
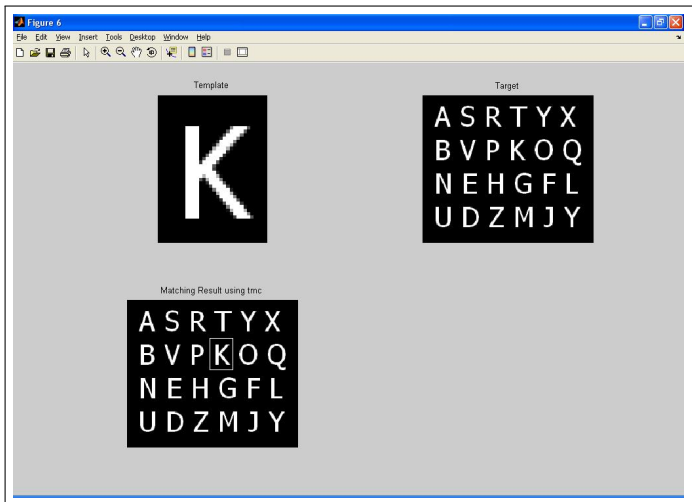
Fig. 3. A template matching function in MATLAB

Fig. 4. The spritesheet used in the Chrome dinosaur game

makes for easy support for any issues that arise during the development of the project. MATLAB was selected to run the image processing algorithms on the collected images due to the project team's familiarity with the software. Since the dinosaur game is built into the Chrome browser, which is based on the open-source Chromium browser, the source code for the dinosaur game is available to the public. As such, the sprites used in the game, Fig. II-A can be easily extracted and used in the image processing software algorithm to classify which sprites are on the screen at a given time [3]. The most robust and fastest method of classifying these images will likely be to run a template matching function on them, Fig. II-A, [4]. The Roaring Thunder computing cluster uses the Slurm scheduler to schedule jobs for each node, so familiarity with that software is necessary to ensure the program is correctly distributed on the computing nodes. OpenMP will be used to parallelize the program, as it is being taught to the project team by Dr. Tim Hansen of South Dakota State University. The project will be primarily coded in C or C++ as the method of parallel processing being taught, OpenMP, is only available in the above languages and Fortran. Other supplementary languages may be used, although most lower-level optimizations and features will be written in the aforementioned C or C++.

B. Project Challenges

One immediate challenge discovered when researching this project was how to run the game on the computing cluster. As the cluster computer is a Unix system, running a fully-functional Chromium browser would be difficult, if not impossible. Furthermore, even running the source code for the dinosaur game may be difficult, as it's written in JavaScript. Finally, as the game runs in real-time at greater than thirty frames per second, analyzing each frame may take too long.

One potential solution for this problem is to devise a method to temporarily pause the game while the image processing algorithm runs, before advancing to the next frame. This should allow for the game to be played at the pace of the computer. Another potential solution to the problem would be to only process a reduced number of frames per second. This way, we can abstain from the allure of needing to change the original code by restricting the amount of data that is processed at once. The problem created by using this method would be a reduced reaction window for the AI.

C. High Performance Computing

Another important aspect to this project is the parallelization of the finished program. Neural networks are inherently multithreaded, which allows for them to scale well when provided more resources. However, this scalability does not come without cost. Some overhead work must be done to ensure the program is parallelized efficiently. Familiarity with general program paralellization methods learned in EE592/CS592 (High Performance Computing) is required to ensure the program is properly parallelized. Familiarty with the OpenMP syntax will allow for these parallelization methods to be utilized properly.

III. Conclusions

Overall, performing this project will expose those working on it to many new software libraries and programming methods, both in terms of creating the program to play the dinosaur game and use it train a neural network, but also in parallelizing the program to run on the computing cluster. The challenges expected to be faced when interfacing with the different software should prove to be an exercise in becoming familiar with reference material of each software. The expected outcome of the project will create an entertaining and impressive show of AI mastery over simple video games thanks to their easily scalable nature.

Acknowledgments

References

[1] SDSU Information Technology. Research Computing. Accessed: Sep. 4, 2020. [Online]. Available: https://www.sdstate.edu/information-technology/research-computing

[2] Code Bullet. AI learns to play Google Chrome Dinosaur Game. Accessed: Sep. 4, 2020. [Online]. Available: https://www.youtube.com

[3] wayou. T-Rex Runner Github. Accessed: Sep. 4, 2020. [Online]. Available: https://github.com/wayou/t-rex-runner

[4] Alaa Eleyan. Template Matching. Accessed: Sep. 4, 2020. [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/20061-template-matching