

Bots-R-Us

Iowa State University Computer Science Senior Design Team 8
Adam Riffel, Carlos Acuna, Corbin Graham, Jose Medina Mani, Nhan
Tran

About the Project

- This project was developed for Lockheed Martin Advanced Concepts Lab as part of the Iowa State University Computer Science Senior Design course (COM S 402C) in Spring 2024.
- Project began on Jan 4, 2024.
- Project ended on May 4, 2024.
- Final Product Demo: May 9, 2024.

About the Client

- Lockheed Martin Advanced Concepts Lab
- Cherry Hill, NJ
- POC: Dan Waitman, r.dan.waitman@lmco.com
- Additional Contact: John McElroy, john.r.mcelroy@lmco.com

About the Team

- Team Members:
 - Adam Riffel, Senior
 - Carlos Acuna, Senior
 - Corbin Graham, Senior
 - Jose Medina Mani, Senior
 - Nhan Tran, Senior
- Iowa State University, Ames, IA
- COM S 402C: Senior Design
- Professor: Simanta Mitra

Problem Background

Questions and reasoning for approaches we took

Research Question

- Is it possible to differentiate between a human and a bot playing a game in realtime?

Approach

1. Implement data collection from a game
2. Train and evaluate classification models on the data
3. Design a program to use our trained models and perform realtime prediction

Game Selection

- Super Mario Bros., 1985 developed by Nintendo for the NES
 - Lots of prior research
- Single-player game
- Using official ROM from the original game
- ROM plugs into decompiled emulator:
[GitHub - MitchellSternke/SuperMarioBros-C: An attempt to translate the original Super Mario Bros. for the NES to readable C/C++](#)
- Emulator is run in Kali Linux Virtual Machine
[Get Kali | Kali Linux](#)

Framework Motivation

- Generalizing one specific configuration of extracting data, data processing, and making predictions based on the data seems unfeasible.
- No Free Lunch theorem
 - Proposed by David Wolpert and William G. Macready
 - Improvement of performance in problem-solving hinges on using prior information to match procedures to problems
 - Can't just apply one procedure to any problem
- The data processing can be quite domain specific.
- Thus, we make a framework that allows the flexibility to use different data extraction, data processing, and prediction tools for a task.

Data

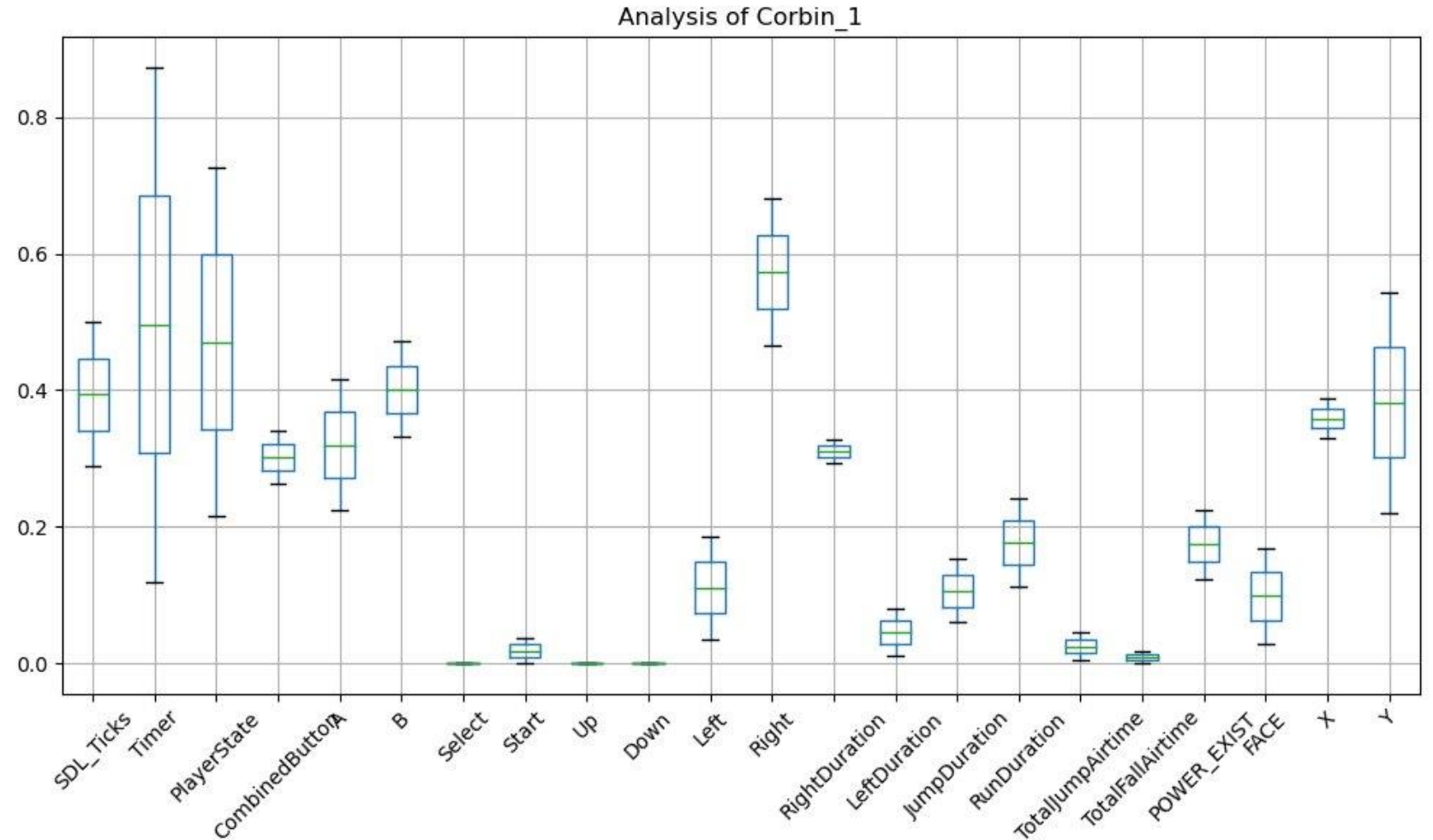
How it is collected, and evaluated

Data Collection

- Emulator is modified to output emulated memory (2KB) to STD IO Bus
 - Input player ID to label data to be logged
 - Pre-Label specific addresses known to contain important data
 - Output CSV format
- STD IO Bus is piped to a file with CSV extension
- CSV files are stored in folders differentiated by name

Data Exploration

- Significance of labeled addresses
- Every user demonstrates significant patterns across each variable with respect to time as a sequence, and as a whole



Models

Results from machine learning models applied to the data

Bots-R-Us Framework

A framework to implement our research findings and output predictions in realtime.

Memory Processing Module

A module to scrape and read in data from a given program

Implementation

- Data inputted by piping through standard in (stdin)
 - Input can be from csv file or running process
- Reads from config to identify features to be passed along
- Memory module reads from stdin to get each row of data
- Formats into whatever format needed by data processor
 - (numpy array in our implementation)
- Passes this data into a queue for the data processor.
- Once there is nothing more to read, waits for queue to empty before sending signal to shutdown

Rationale

- Original approach of bypassing the kernel ended up being too much work so we currently read from stdin by piping data into the program on the command line.
- Additionally, we added settings in the config to select features because we wanted to avoid passing huge chunks of data between processes
- The queue is to pass between processes as we want each module to be on its own thread.
- Finally, we want the program to close gracefully when there is no more data to be read. We must wait for the queue to empty before closing to close each thread properly.

Data Processing Module

A module to process and clean data

Rationale

- Raw data scraped from memory needs to be cleaned and organized in some way before being used to train models. This is especially evident as we scrape entire memory spaces.
- Feature selection on top of data cleaning/normalization to reduce the dimensions of the scraped memory space. This would manage training times on deep neural networks and even on some traditional models.
- Needs to approach data cleaning/normalization and feature selection in a generalizable way to fit any model.

Implementation

- We can set a variance threshold as a feature selector on the scraped memory. To get good results from the variance calculation, we must first spend some time buffering data from the memory scraper into memory; while this happens, nothing is output to the prediction module.
- We can set a buffer threshold, which upon reaching will perform a variance calculation across the samples and decide what features to keep and drop based on the set variance threshold.
- Filters samples from memory module to fit the feature selector, or, if targeted features are provided, acts as a passthrough from the start.

Prediction Module

A module to perform predictions on data

Prediction Module Components

- Prediction model
 - Each prediction module comes with a corresponding model that has been previously trained. This allows anyone with a pre-trained model to easily plug it in to the framework with little hassle.
- Model specific preprocessing
 - Some models require further data processing than what is done in the data processor. The user can create a sub module for preprocessing data before it is given to the prediction model.

Prediction Module Config

- The provided config comes with three parameters that can be adjusted.
- Model features
 - This is the number of features a particular prediction model has been trained on.
- Use targets
 - A toggle to utilize the targeted features.
- Targeted features
 - A list of indices that correlate with specific features one might want from their data.

Prediction Targeted Features

- Within a given domain, the user may have some pre-knowledge as to what features they may want to extract and use in their model.
- The targeted features option allows a user to only use this subset of data.
- For the Mario game, we found that a certain subset of memory addresses lined up with values that made our prediction models perform better.
 - This subset is the default list within the config.

API

A method for accessing data from the running program

Standard API

- Logging function

Main API

- Stores all of the APIs for the main class

Memory API

Data API

Prediction API

- Stores prediction

Future Work

Things that could be done to improve the framework

Data Limitations

Given our limited data set we believe that gathering more data would benefit training more accurate models. In addition, a larger variety of data representing different types of bots and a range of players in varying skill level would provide a more complete picture.

Memory Scraper

Time was spent experimenting with various ways to bypass the kernel to access a processes' memory to avoid having to modify the source code of a program. In the future it would be worth looking into a method to get a programs memory from the RAM, either through dynamic memory analysis or some form of shared memory.

Additional Models

We didn't have the time to fully implement a Markov chain.

In addition, more time to experiment fully with other models could result in a model with higher accuracies than our current models.

Varying Games

Our work was done solely on variations of Super Mario Bros, a 2D platforming game. To expand our findings, it would be beneficial to branch out and test models on a variety of game types to see if these results still hold water.

Branching out to 3D games to test our models and framework would help in proving this is applicable to a larger subset of applications. Going so far as to leave games behind and apply this to a computer's desktop would be an interesting further experiment.

Identifying Individuals

Our current progress focuses on binary classification as it only differentiates between a bot and a person, this could be expanded though to identifying individuals as well. Determining if we could identify not only the player is human, but analyzing their playstyle to determine exactly who could create many more use case scenarios for this type of work.