



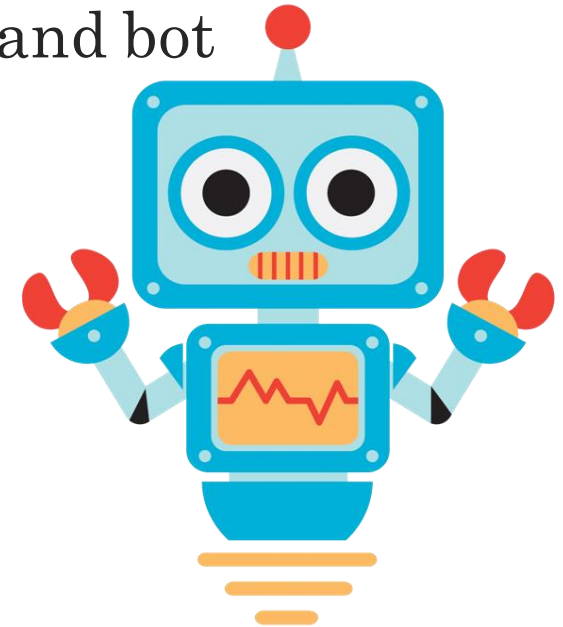
Bots-R-Us

Team 8 Final Presentation

Adam Riffel, Carlos Acuna, Corbin Graham, Jose Medina Mani, Nhan Tran

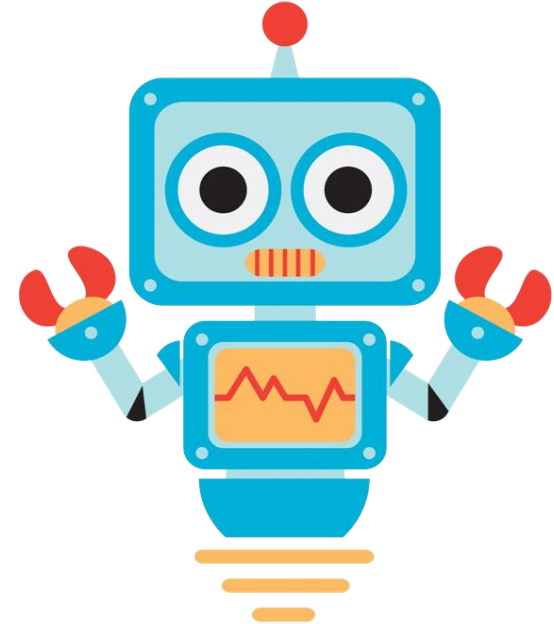
Problem Description

- Primary Research Question
 - Is it possible to differentiate between a human and bot playing a game?
- Secondary Research Questions
 - Can we do this from a naive approach?
 - Can we identify a human or bot in real time?
 - Can our solution be generalized to multiple platforms?



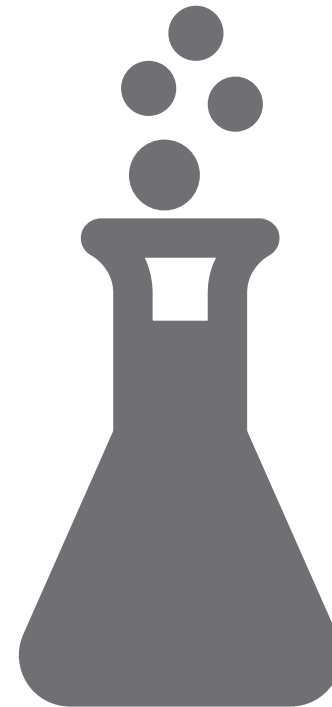
Approach

- Proved it is possible to differentiate
- Designed framework to address each approach
 - Naïve approach & Calculated Approach
 - Online & Offline Prediction



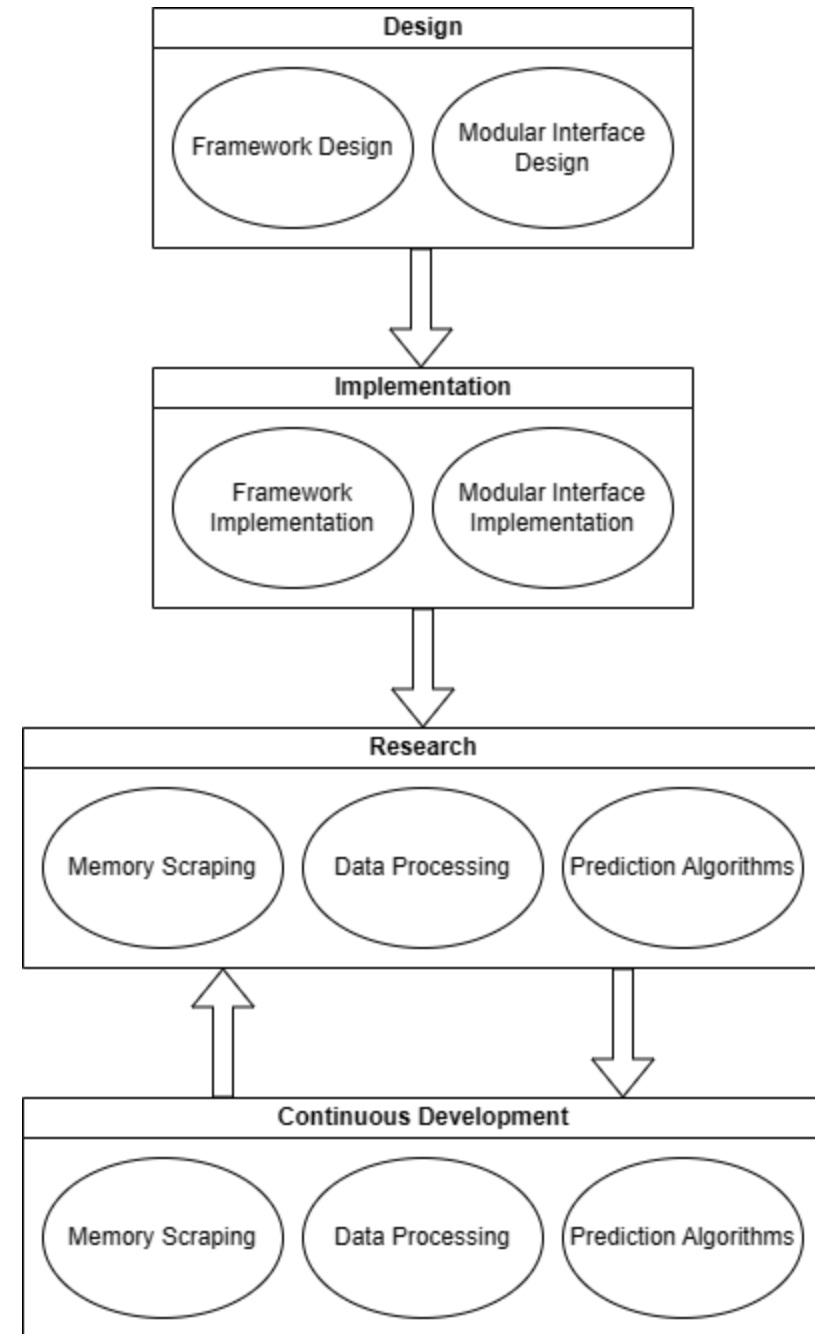
Our Lifecycle

1. Initial Research
2. Initial Design & Implementation
3. Additional R&D



Business Implementation Plan

1. Initial Design
2. Initial Implementation
3. Continuous Research
4. Continuous Development



Research Findings

Memory Scraping

- Kernel implements methods of abstraction
- Explored function hooking
- Explored system call intercepts
- Explored subprocesses
- Result: Additional Research Required

Data Exploration

- Explored relevant features
- Explored dimension reduction
- Explored superscoring
- Result: Normalization & Feature Targeting

Algorithmic Prediction

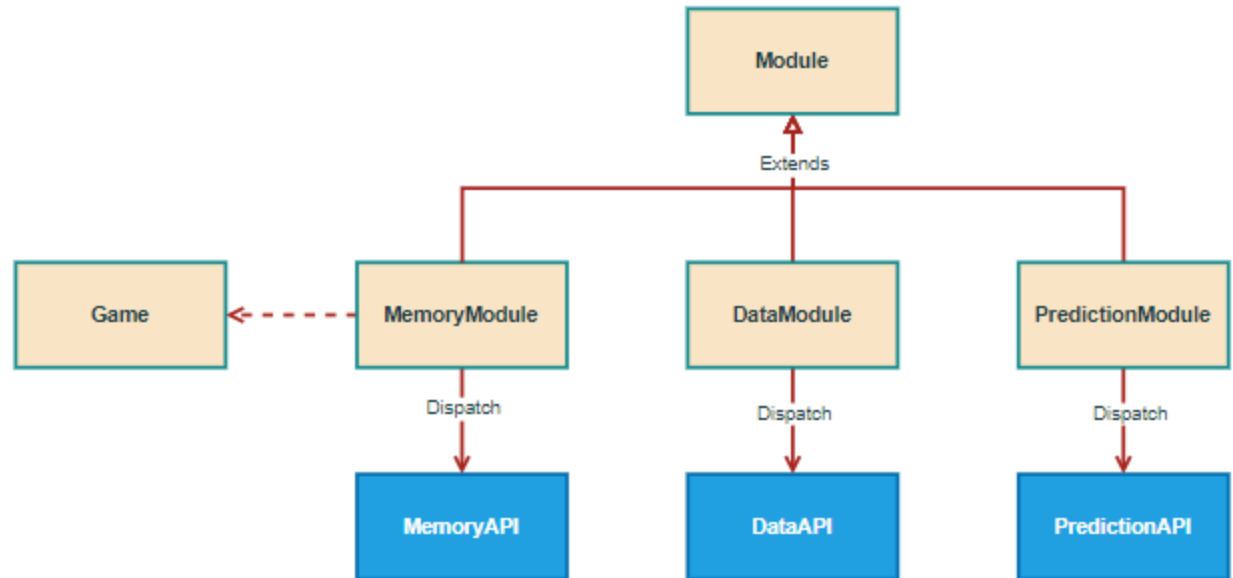
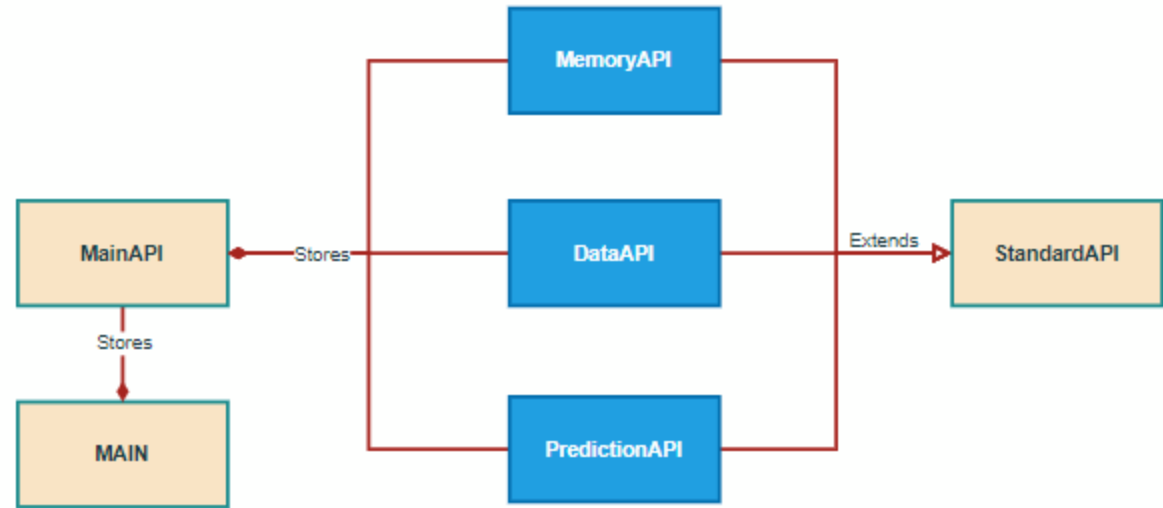
- Explored various machine learning approaches
- Explored score-based algorithms
- Result: LSTM

Framework Design

Design Rationale

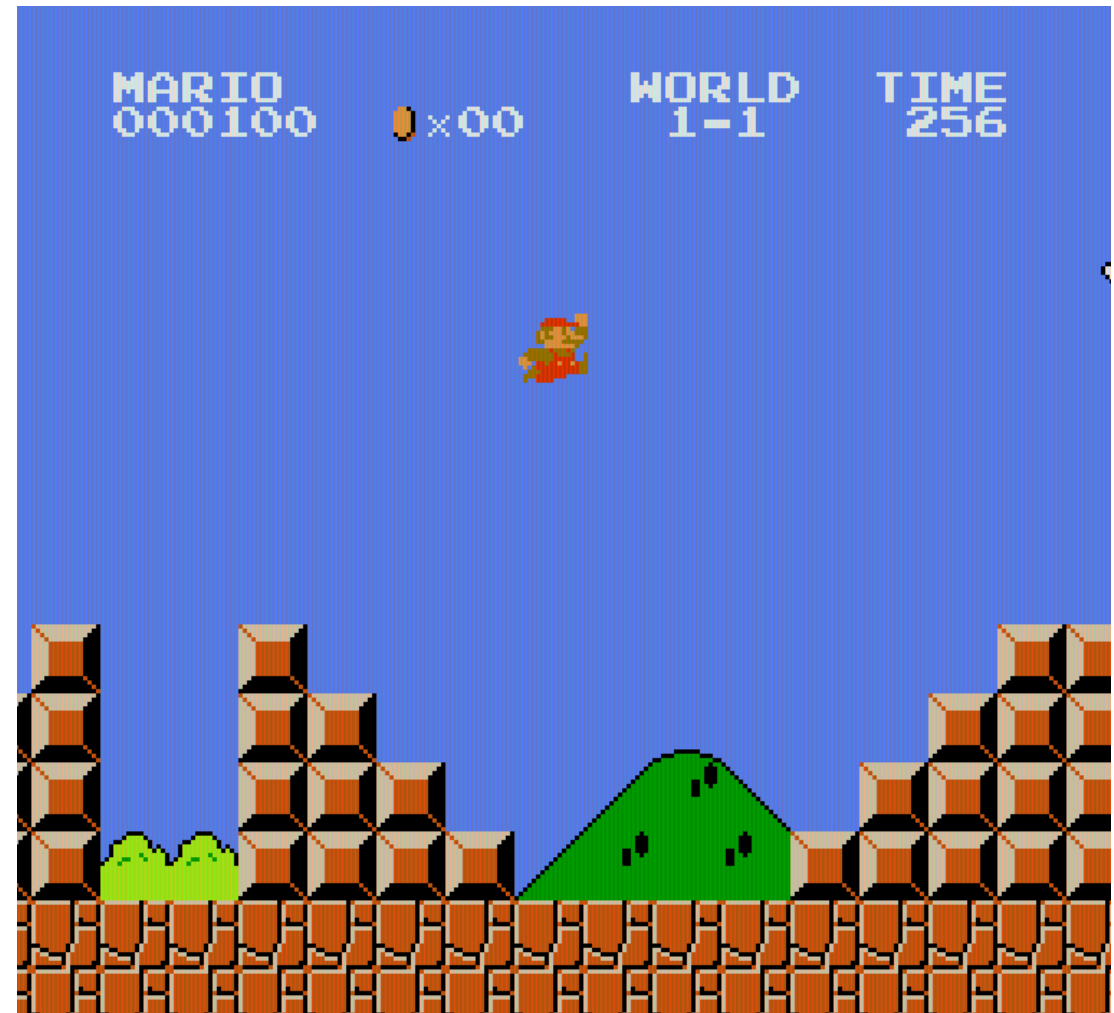
- Modular design supports multiple approaches
- Swappable interfaces:
 - Allows for client to create and use modules of their own
- API:
 - Allows for client to access the inner-workings
- Multiprocessing supports subprocess-threading

Simplified Architecture



Modules

- Memory Module
 - Data collection
- Data Module
 - Data processing
- Prediction Module
 - Data prediction



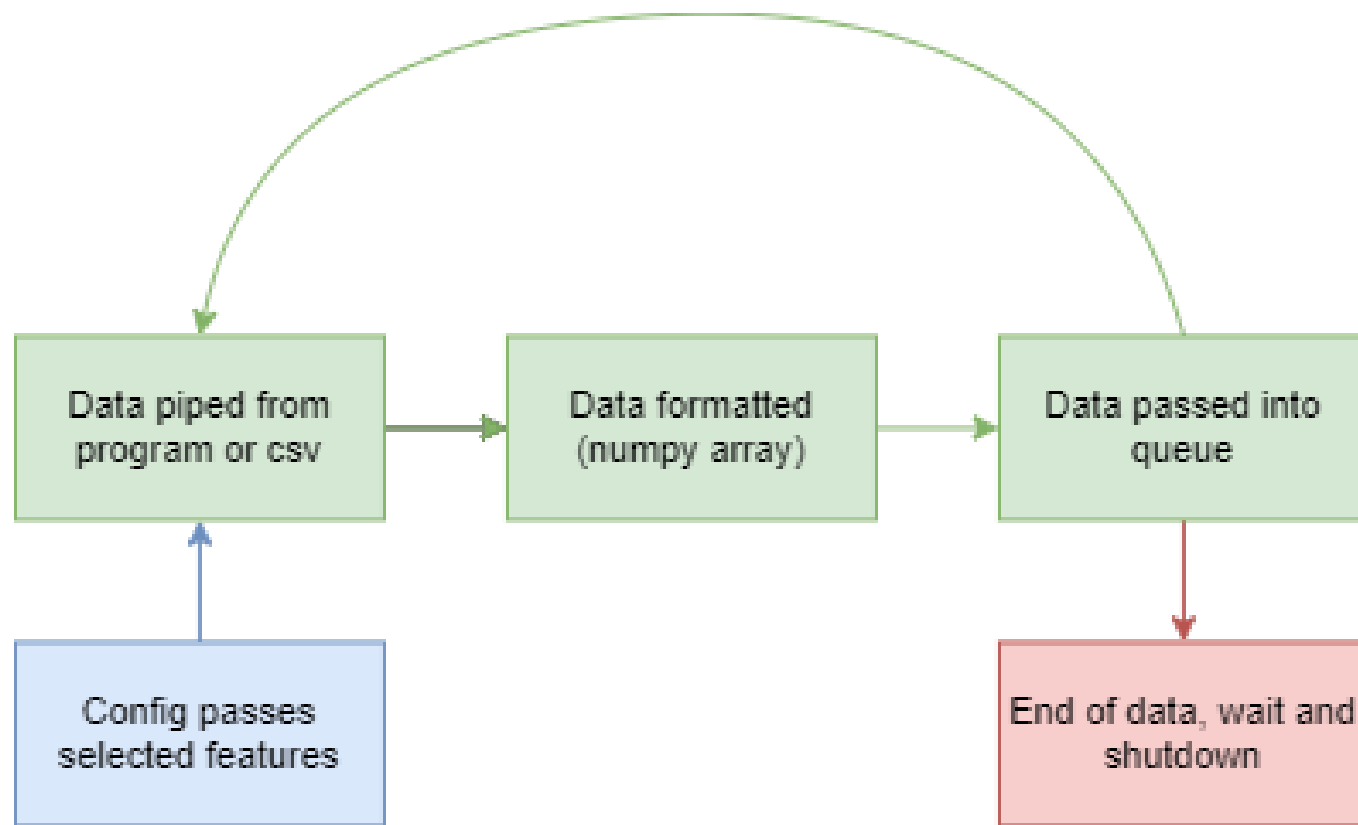
API

- Standard API
 - Implements logging
- Main API
 - Hosts modular APIs
- Memory API
- Data API
- Prediction API
 - Hosts model API
 - Access prediction

Memory Module

Design & Implementation

- Initialize feature selection
- Input data through standard in (stdin)
- Read from stdin to get each row of data
- Format data for data processor
- Pass data into queue for data processor
- At end of data, wait and shutdown



Rationale

- Bypassing the kernel too much work, read from stdin
- Config settings select features avoid unnecessary data passing
- Queue for multiprocessing
- Data runs out and queue empties to close

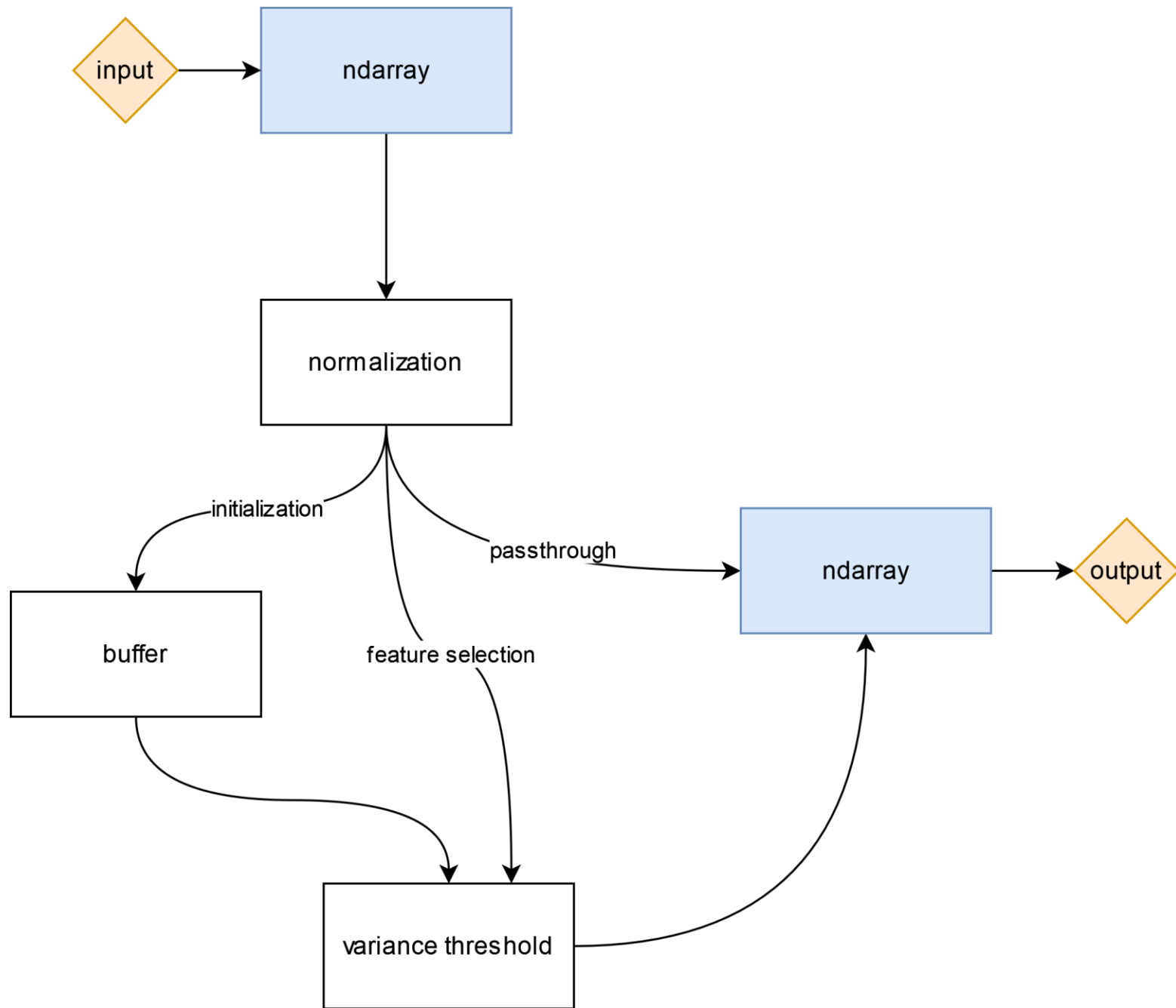
Data Processing Module

Rationale

- Data cleaning and normalization
- Feature selection
- Generalize use across different models

Design & Implementation

- Takes unprocessed input from Memory Module and produces a standard formatting for Prediction Module
- Feature selection for live training, or passthrough for pretrained models with targeted features
- Variance threshold as the default plugin



➤ Challenges

- Account for variable input from Memory Module
- Produce standardized output for Prediction Module
- Generalizable feature selection technique across all models

Prediction Module

Live Training vs. Pretrained Models

- Not all models support live training easily.
- Pretrained models require less start up time

Design

- Components
 - Prediction model
 - ONNX format
 - Model specific preprocessing
- Targeted features
 - Utilize domain expert knowledge

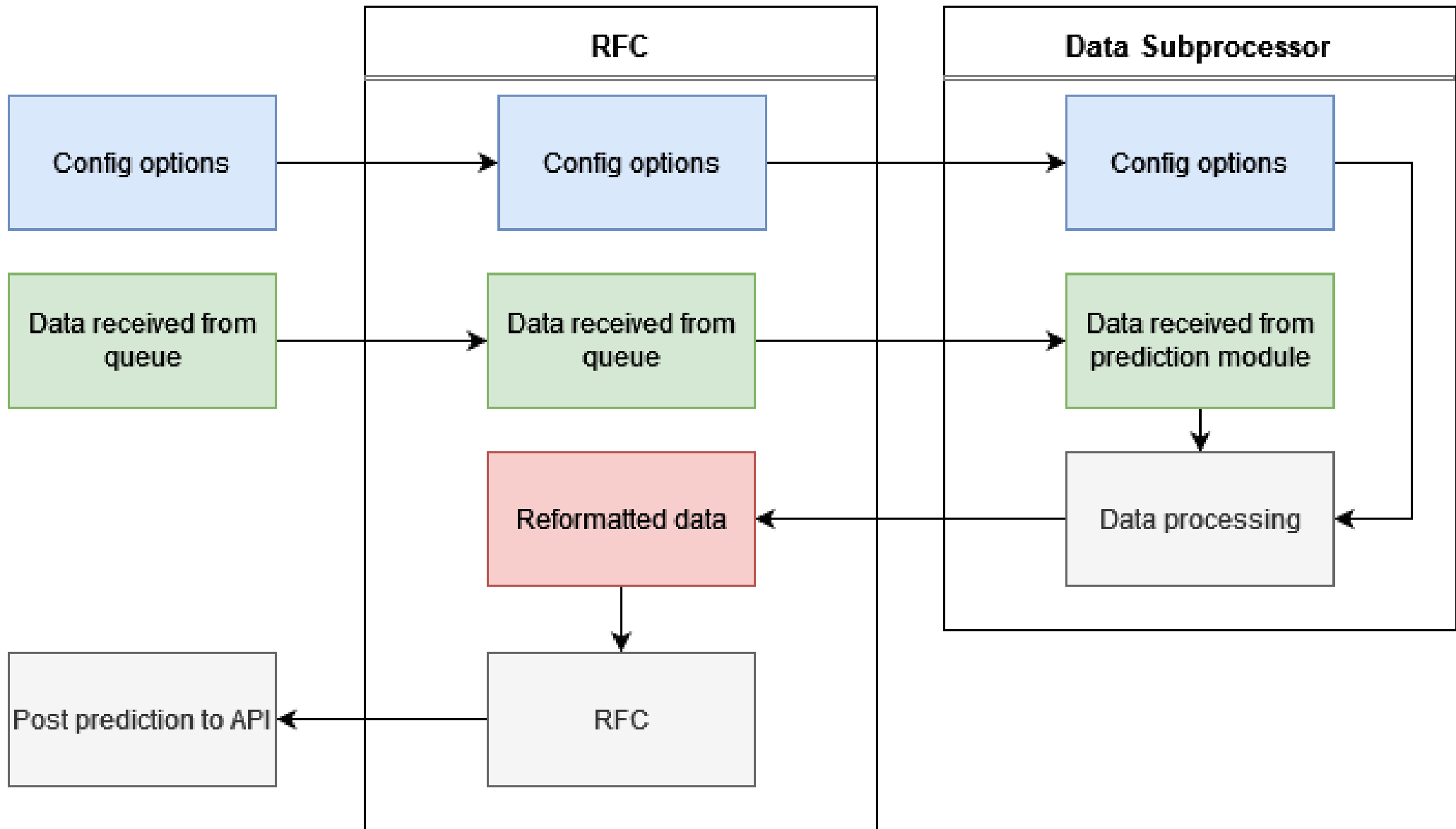


ONNX

Implementation

Random Forest Classifier

- Preliminary Research
 - Helps identify feature importance
- Model specific preprocessing
 - Data shape fitting



Challenges

Random Forest Classifier

- Inaccurate pretrained model
- Sequential data
- Real-time training is difficult

Overview

LSTM Autoencoder

- Unsupervised Learning
- Sequence to Sequence
- Encoder encodes input sequence
- Decoder reconstructs input sequence
- Anomaly Detection
- Reconstruction Loss

Features

LSTM Autoencoder

FRAME, X, Y, UP, DOWN, LEFT, RIGHT, SPEED, ONGROUND

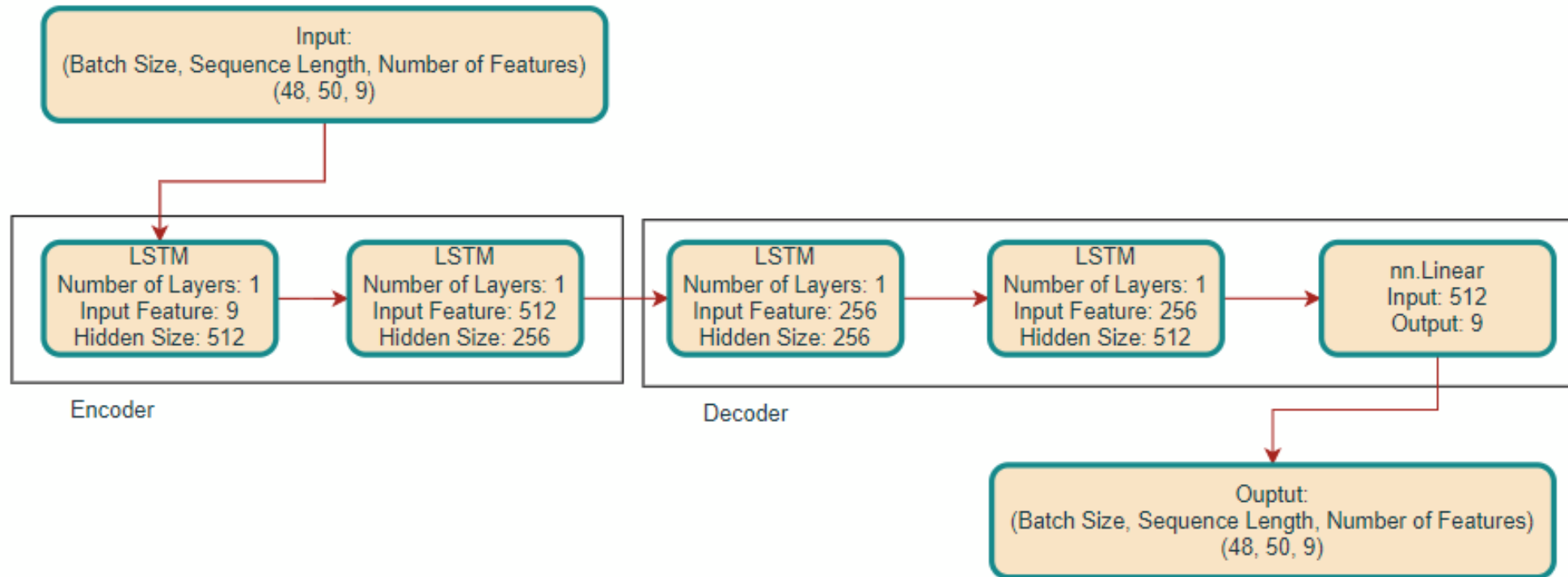
SEQUENCE LENGTH = 50

Implementation

LSTM Autoencoder

Human data is varied
Train on bot data

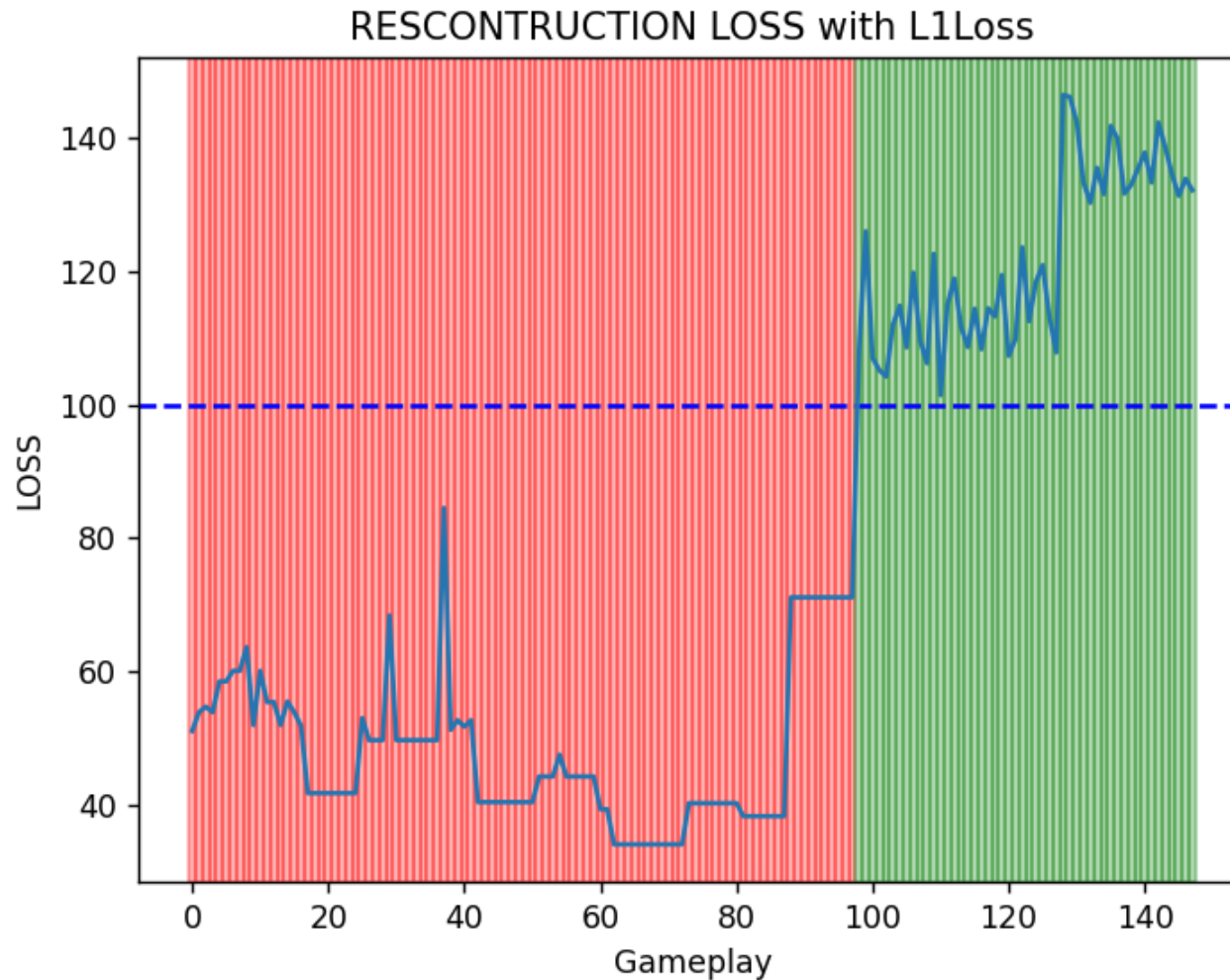
Implementation LSTM Autoencoder



Loss: Mean Average Error (MAE)

Result

LSTM Autoencoder



Framework Integration

LSTM Autoencoder

- List of **sequence predictions** over the gameplay
- Threshold = 100
- For each sequence:
 - Reconstruction Loss below threshold: Bot, 1
 - Reconstruction Loss above threshold: Human, 0
- Final prediction: the average of predictions

Challenges

LSTM Autoencoder

- Limited variations of bots
- Data overfitting
- Long training time
- Environmental data

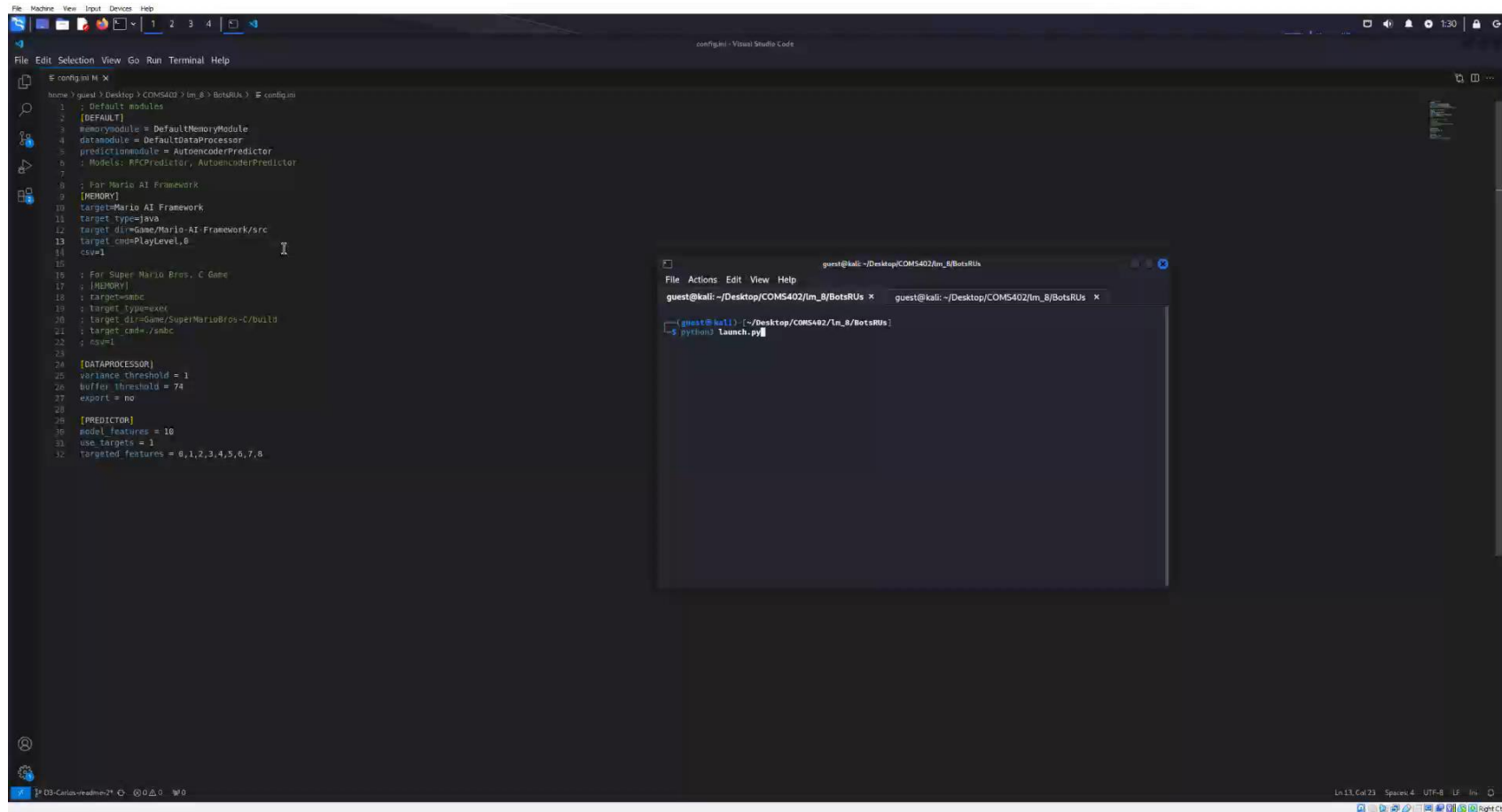
In the future:

- Autoencoder for a feature
- Segment length starting from beginning

Teamwork Contributions

Module	Framework Design	Memory Module	Data Module	Prediction Module
Team Members	Corbin Graham	Adam Riffel	Jose Medina Mani	Carlos Acuna, Nhan Tran
Research Topic	Memory Scraping	Prediction	Data Exploration	
Team Members	Corbin Graham, Carlos Acuna, Jose Medina Mani	Carlos Acuna, Corbin Graham, Jose Medina Mani, Nhan Tran	Adam Riffel, Corbin Graham	

Demo (without GUI)



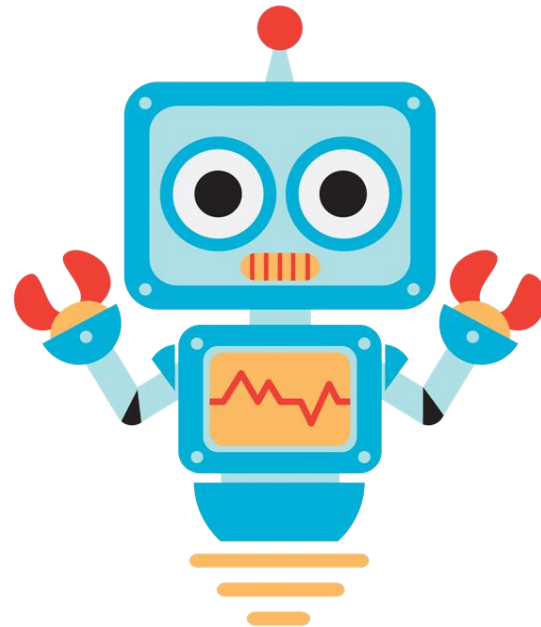
The image shows a screenshot of the Visual Studio Code editor. The main editor area displays a configuration file with the following content:

```
1 : default modules
2 : [DEFAULT]
3 : memorymodule = DefaultMemoryModule
4 : datamodule = DefaultDataProcessor
5 : predictionmodule = AutoencoderPredictor
6 : Models: RFCPredictor, AutoencoderPredictor
7
8 : For Mario AI Framework
9 : [MEMORY]
10 : target=Mario AI Framework
11 : target_type=java
12 : target_dir=Game/Mario AI Framework/src
13 : target_cmd=PlayLevel_0
14 : csv=1
15
16 : For Super Mario Bros. C Game
17 : [MEMORY]
18 : target=smabc
19 : target_type=cxx
20 : target_dir=Game/SuperMarioBros-C/build
21 : target_cmd=./smbc
22 : csv=1
23
24 : [DATAPROCESSOR]
25 : variance_threshold = 1
26 : buffer_threshold = 74
27 : export = no
28
29 : [PREDICTOR]
30 : model_features = 10
31 : use_targets = 1
32 : targeted_features = 6,1,2,3,4,5,6,7,8
```

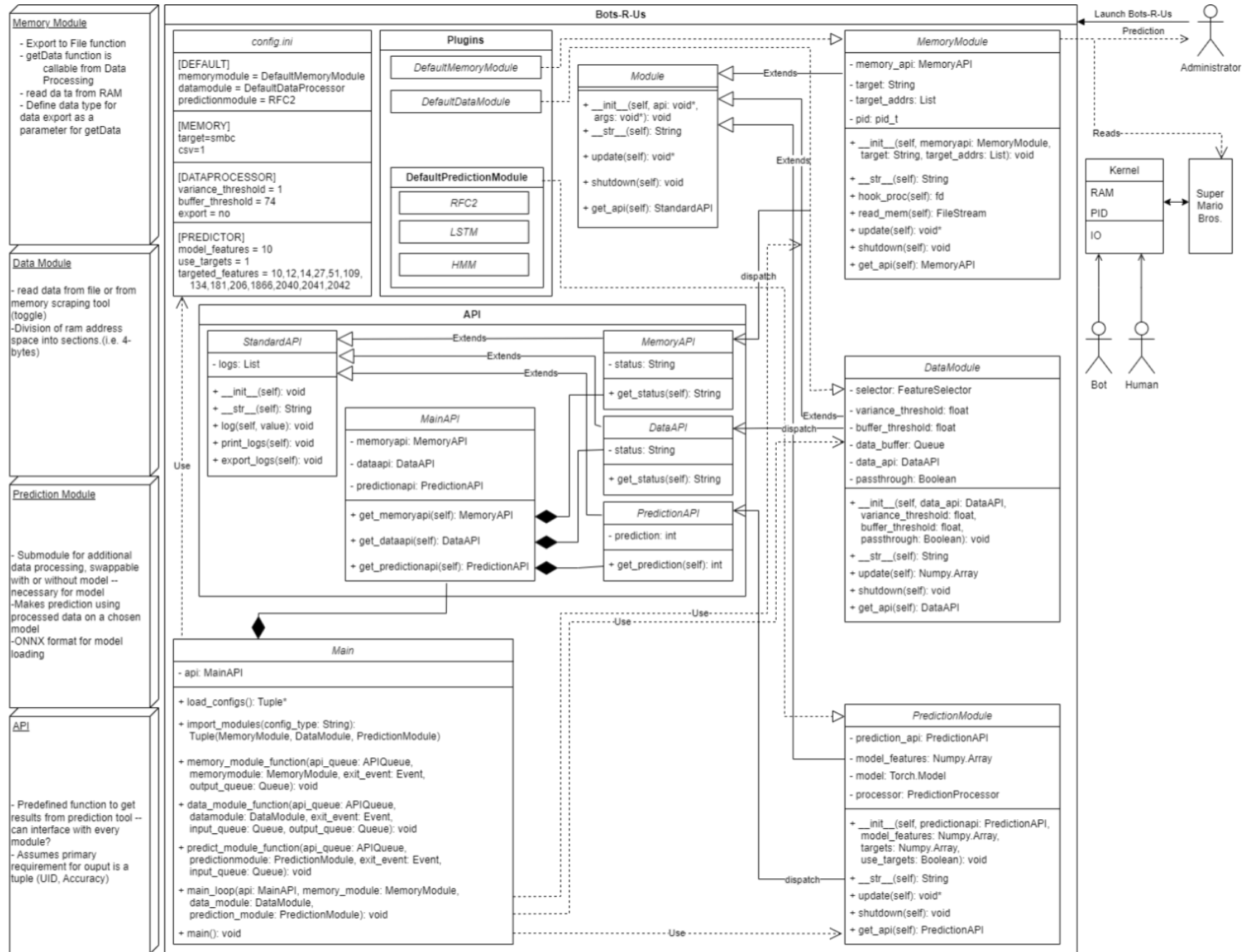
An integrated terminal window is open in the foreground, showing the following command and output:

```
guest@kali: ~/Desktop/COMS402/In_8/BotsRUs
- $ python3 launch.py
```

Questions?



The Architecture



LSTM Autoencoder prediction code

```
# SEQUENCE PREDICTION
if (reconstruction_loss < threshold):
    predictions.append(1) # BOT
else:
    predictions.append(0) # HUMAN

# AVERAGE PREDICTION OVER GAMEPLAY SO FAR
average_prediction = sum(predictions) / total_number_of_predictions
```