SEMESTER PROJECT

ARTIFICIAL INTELLIGENCE

AI GAME BOT DEVELOPMENT

Spring'25

Submitted to:
Sir Hassan Mujataba
Sir Adil Majeed
Sir Usama Imtiaz

NATIONAL UNIVERSITY
of Computer & Emerging Sciences

GROUP DETAILS

Our group of four people collaboratively worked on developing an intelligent game bot using an MLP model. Responsibilities were equally divided among members to ensure efficient progress across all stages of the project, including data collection, model development, integration, testing, and documentation. Throughout the process, we maintained clear communication and actively supported one another in debugging, refining features, and improving overall performance. The final result reflects a unified team effort driven by shared goals and consistent collaboration.

Hamd Ul Haq (23I-0081)

- Conducted thorough testing across single-player and two-player modes.
- Helped in data collection and management of Dataset.
- Measured the bot performance and validated it.

Mohammad Haider Abbas (23I-2558)

- Coordinated extensive gameplay sessions to collect diverse data.
- Ensured coverage of all characters and scenarios, and managed dataset preprocessing for model training.
- Compiled the final project report and supporting documentation.

Ahmed Izaan (23I-0015)

- Integrated the trained MLP model into the bot.py file.
- Ensured seamless interaction with the game API and reliable action execution.
- Helped in testing the bot performance variables.

Safee Ahmed (23I-0003)

- Took charge of designing and implementing the Multilayer Perceptron (MLP) architecture.
- Predicted in-game actions that led to high accuracy in the training of the bot.
- Contributed extensively to writing and organizing the core model training code.

PROJECT DETAILS

Introduction

This report details our team's development of an AI-powered game bot for **Street Fighter II Turbo**. This document outlines our team roles, dataset creation, feature selection, rationale for choosing MLPs, model development, integration, and testing, emphasizing the work we performed to achieve a robust and generalizable bot.

Project Division

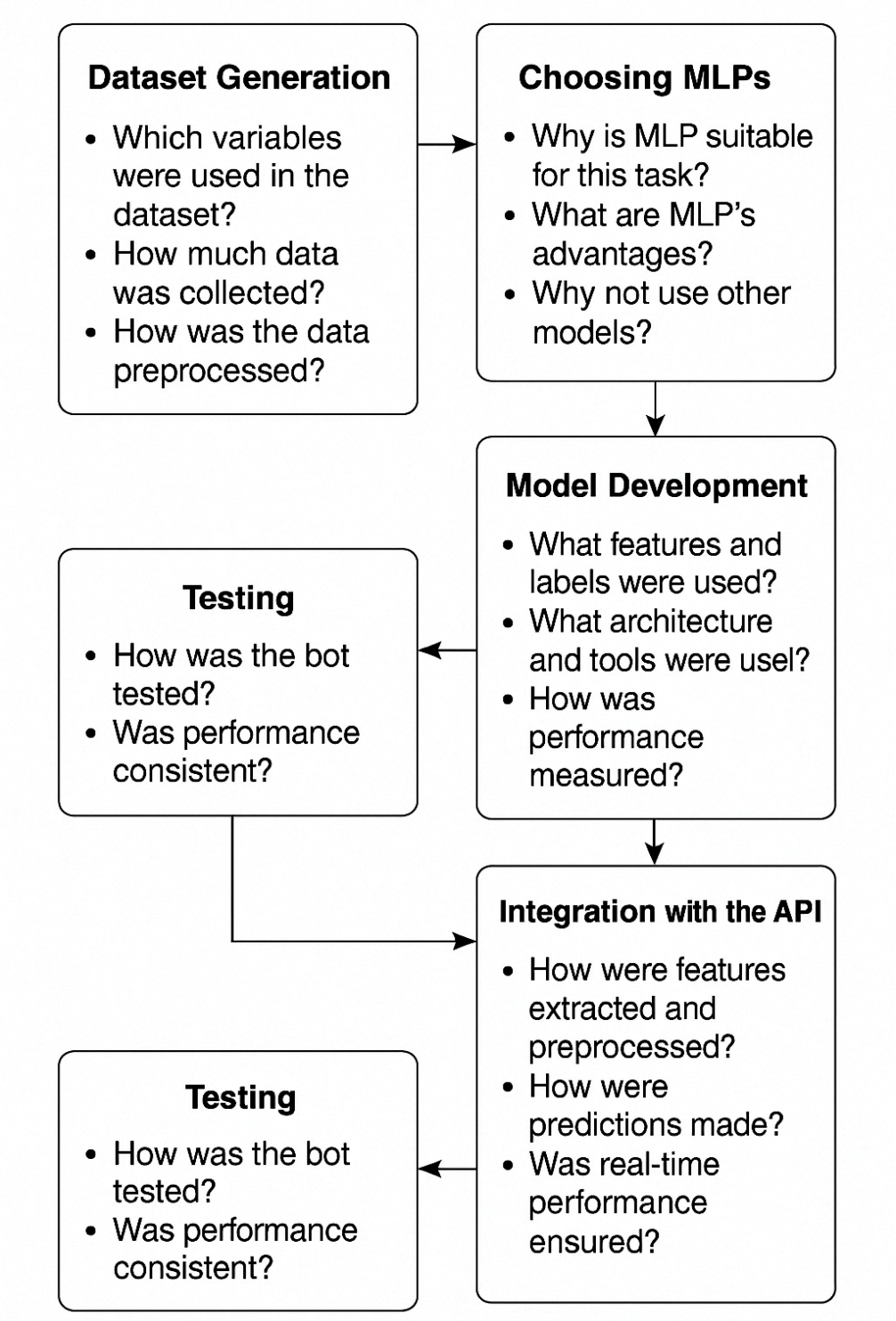
1. **Dataset Generation**
    a. Which variables were used in the dataset?
    b. How much data was collected?
    c. How was the data preprocessed?

2. **Choosing MLPs**
    a. Why is MLP suitable for this task?
    b. What are MLP's advantages?
    c. Why not use other models?

3. **Model Development**
    a. What features and labels were used?
    b. What architecture and tools were used?
    c. How was performance measured?

4. **Integration with the API**
    a. How were features extracted and preprocessed?
    b. How were predictions made?
    c. Was real-time performance ensured?

5. **Testing**
    a. How was the bot tested?
    b. Was performance consistent?


Dataset Generation
• Which variables were used in the dataset?
• How much data was collected?
• How was the data preprocessed?
Choosing MLPs
• Why is MLP suitable for this task?
• What are MLP's advantages?
• Why not use other models?
Model Development
• What features and labels were used?
• What architecture and tools were usel?
• How was performance measured?
Testing
• How was the bot tested?
• Was performance consistent?
Integration with the API
• How were features extracted and preprocessed?
• How were predictions made?
• Was real-time performance ensured?
Testing
• How was the bot tested?
• Was performance consistent?

DATASET GENERATION

To train our MLP model, we generated a large and diverse dataset, our process included:

1. **Gameplay Sessions**:
    - We played multiple rounds using all 12 playable characters to cover their unique move sets and playstyles.
    - We employed varied strategies, including aggressive attacks, defensive positioning, and frequent use of special moves, to capture different decision making scenarios.
    - Our particular preference was Dhalsim. His unique ability to attack from a distance proved highly effective, allowing us to inflict significant damage while maintaining a safer, more defensive position."
    - Gameplay sessions spanned approximately 48 hours, ensuring comprehensive data collection.

2. **Data Recording**:
    - We recorded the game stats in CSV using API's GameState and Command objects; the dataset included 35 variables:
        - **Game Context**: timer (round time, 0–100 seconds).
        - **Player 1 Attributes**: p1_id (character ID), p1_health (0–100), p1_x (x-coordinate), p1_y (y-coordinate), p1_jumping (boolean), p1_crouching (boolean), p1_in_move (boolean), p1_move_id (current move ID).
        - **Player 2 Attributes**: p2_id, p2_health, p2_x, p2_y, p2_jumping, p2_crouching, p2_in_move, p2_move_id (opponent data).
        - **Relative Features**: p1_rel_x (relative x-distance between players), p1_rel_y (relative y-distance), p1_facing_opponent (boolean), p1_health_diff (p1_health - p2_health)
        - **Action Outputs**: 12 boolean button states (act_B, act_Y, act_X, act_A, act_L, act_R, act_Up, act_Down, act_Left, act_Right, act_Select, act_Start) for SNES gamepad inputs.
        - **Outcome**: win (boolean indicating if Player 1 won the round).
    - These variables captured spatial relationships, player conditions, and strategic context critical for making an accurate game bot.

DATASET GENERATION (2)

3. **Data Volume**:
    - We collected approximately **300,000 state-action pairs**, providing a robust dataset to train the MLP model.
    - Playing so many different character matchups really helped us understand which characters have an edge over others. This knowledge then influenced how we chose our moves and changed our strategies depending on who we were fighting.
    - The large volume ensured coverage of rare scenarios, such as specific character matchups, low-health situations, or complex move combinations.

4. **Preprocessing**:
    - We cleaned the dataset to remove incomplete entries caused by emulator glitches.
    - Numerical features (timer, p1_health, p1_x, p1_y, p2_health, p2_x, p2_y, p1_rel_x, p1_rel_y, p1_health_diff) were normalized to [0, 1] using min-max scaling.
    - Categorical features (p1_id, p2_id, p1_move_id, p2_move_id) were one-hot encoded, expanding the feature set to ~50 columns.
    - Boolean features (e.g., p1_jumping, act_Up) were kept as 0/1 values.
    - The preprocessed dataset was structured based on a sample dataset provided, ensuring consistent column names and formats.

DATASET IMAGE

```
p1_x,p1_y,p1_health,p1_is_jumping,p1_is_crouching,p1_in_move,p1_move_id,p2_x,p2_y,p2_health,p2_is_jumping,p2_is_crouching,p2_in_move,p2
205,192,176,0,0,0,0,307,192,176,0,0,0,0,146,0,0,0,0,0,0,0,0,0,0,0,0
205,192,176,0,0,0,0,307,192,176,0,0,0,0,146,0,0,0,1,0,0,0,0,0,0,0,0
206,192,176,0,0,0,0,307,192,176,0,0,0,0,146,0,1,0,1,0,0,0,0,0,0,0,0
206,192,176,0,0,0,0,307,192,176,0,0,0,0,146,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,0,0,307,192,176,0,0,0,0,146,0,1,0,0,0,0,0,0,0,0,0,0
206,192,176,0,1,0,0,307,192,176,0,0,0,0,146,0,1,1,0,0,0,0,0,0,0,0,0
206,192,176,0,1,0,0,307,192,176,0,0,0,0,146,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,1,0,0,307,192,176,0,0,0,0,146,0,0,1,0,0,0,0,1,0,0,0,0
206,192,176,0,1,1,1,307,192,176,0,0,0,0,146,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,146,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,146,0,0,0,1,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,146,0,1,0,1,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,146,0,0,0,0,0,0,0,0,0,0,0,0
227,192,176,0,0,1,1,304,192,176,0,0,0,0,146,0,1,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,146,0,1,1,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,146,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,146,0,0,1,0,0,0,0,1,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,146,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,146,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,1,0,0,1,0,1,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,0,0,0,0,0,0,0,0,0,0,0
227,192,176,0,0,1,1,304,192,176,0,0,0,0,145,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,0,1,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,1,1,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,1,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,1,0,1,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,0,0,1,0,0,0,1,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,1,0,0,1,0,1,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,0,0,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,0,1,0,0,0,0,0,0,0,0,0
206,192,176,0,0,1,1,307,192,176,0,0,0,0,145,0,1,1,0,0,0,0,0,0,0,0,0
```

```
63701 139,7,120,98,192,0,0,0,0,10,120,178,192,0,0,0,0,-80,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63702 139,7,120,98,192,0,0,0,0,10,120,178,192,0,0,0,0,-80,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63703 139,7,120,96,192,0,0,0,0,10,120,178,192,0,0,0,0,-82,0,1,0,0,0,0,0,0,0,0,0,1,0,1
63704 139,7,120,96,192,0,0,0,0,10,120,178,192,0,0,0,0,-82,0,1,0,0,0,0,0,0,0,0,0,1,0,1
63705 139,7,120,94,192,0,0,0,0,10,120,178,192,0,0,0,0,-84,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63706 139,7,120,94,192,0,0,0,0,10,120,178,192,0,0,0,0,-84,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63707 139,7,120,96,192,0,0,0,0,10,120,178,192,0,0,0,0,-82,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63708 139,7,120,96,192,0,0,0,0,10,120,178,192,0,0,0,0,-82,0,1,0,0,1,0,0,0,0,0,0,0,0,1
63709 139,7,120,98,192,0,0,0,0,10,120,180,192,0,0,1,0,-82,0,1,0,0,0,1,0,0,0,0,0,1,0,1
63710 139,7,120,98,192,0,0,0,0,10,120,180,192,0,0,1,0,-82,0,1,0,0,1,1,0,0,0,0,0,0,1,1
63711 139,7,120,100,192,0,0,0,0,10,120,182,192,0,0,0,0,-82,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63712 139,7,120,100,192,0,0,0,0,10,120,182,192,0,0,0,0,-82,0,1,0,0,0,0,1,0,0,0,0,0,1,0,1
63713 138,7,120,98,192,0,0,0,0,10,120,184,192,0,0,1,0,-86,0,1,0,0,1,0,1,0,0,0,0,0,0,0,1
63714 138,7,120,98,192,0,1,0,0,10,120,184,192,0,0,0,0,-86,0,1,0,0,0,0,0,0,0,0,1,0,0,0,1
63715 138,7,120,98,192,0,0,1,132101,10,120,184,192,0,0,0,0,-86,0,1,0,0,0,0,1,0,0,0,0,0,0,1
63716 138,7,120,98,192,0,0,1,0,10,120,184,192,0,1,0,0,-86,0,1,0,1,1,1,0,0,0,0,0,0,0,0,1
63717 138,7,120,98,192,0,0,0,0,10,120,184,192,0,0,0,0,-86,0,1,0,0,1,0,0,0,0,0,0,0,0,1
63718 138,7,120,96,192,0,0,0,0,10,120,184,192,0,0,0,0,-88,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63719 138,7,120,96,192,0,0,0,0,10,120,184,192,0,0,0,0,-88,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63720 138,7,120,96,192,0,0,0,0,10,120,184,192,0,0,0,0,-88,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63721 138,7,120,94,192,0,0,0,0,10,120,182,192,0,0,0,0,-88,0,1,0,1,1,0,0,0,0,0,0,0,0,1
63722 138,7,120,94,192,0,0,0,0,10,120,182,192,0,1,0,0,-88,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63723 138,7,120,94,192,0,0,1,132101,10,120,180,192,0,0,0,0,-86,0,1,0,0,0,1,0,0,0,0,0,0,0,1
63724 138,7,120,94,192,0,0,0,0,10,120,180,192,0,0,0,0,-86,0,1,0,0,0,1,0,0,0,0,0,0,0,1
63725 138,7,120,94,192,0,0,0,0,10,120,180,192,0,0,1,132101,-86,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63726 138,7,120,94,192,0,0,0,0,10,120,180,192,0,0,0,0,-86,0,1,0,1,0,1,0,0,0,0,0,0,1,0,1
63727 138,7,120,94,192,0,0,0,0,10,120,178,192,0,0,1,263170,-84,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63728 138,7,120,94,192,0,0,0,0,10,120,178,192,0,0,0,0,-84,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63729 138,7,120,94,192,0,0,0,0,10,120,178,192,0,0,0,0,-84,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1
63730 138,7,120,94,192,0,0,0,0,10,120,178,192,0,0,0,0,-84,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63731 138,7,120,94,192,0,0,0,0,10,120,178,192,0,0,0,0,-84,0,1,0,0,1,0,0,0,0,0,0,0,0,1
63732 138,7,120,94,192,0,0,0,0,10,120,178,192,0,0,0,0,-84,0,1,0,1,0,0,0,0,0,0,0,0,0,1
63733 137,7,120,94,192,0,0,0,0,10,120,180,192,0,0,1,132101,-86,0,1,0,0,0,0,0,0,0,0,0,0,0,1
63734 137,7,120,94,162,1,0,1,263170,10,120,178,192,0,0,0,0,-84,-30,1,0,1,0,0,0,0,0,1,0,0,0,0,0,1
63735 137,7,120,94,192,0,0,0,0,10,120,178,192,0,0,0,0,-84,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1
63736 137,7,120,94,192,0,0,0,0,10,120,176,192,0,0,0,0,-82,0,1,0,0,0,0,0,0,0,0,0,0,1,0,1
63737 137,7,120,94,192,0,0,0,0,10,120,176,192,0,0,0,0,-82,0,1,0,0,0,0,0,0,0,0,0,0,0,0,1
63738 137,7,120,94,192,0,0,0,0,10,120,178,192,0,0,0,0,-84,0,1,0,0,0,0,0,0,0,0,0,1,0,0,1
```

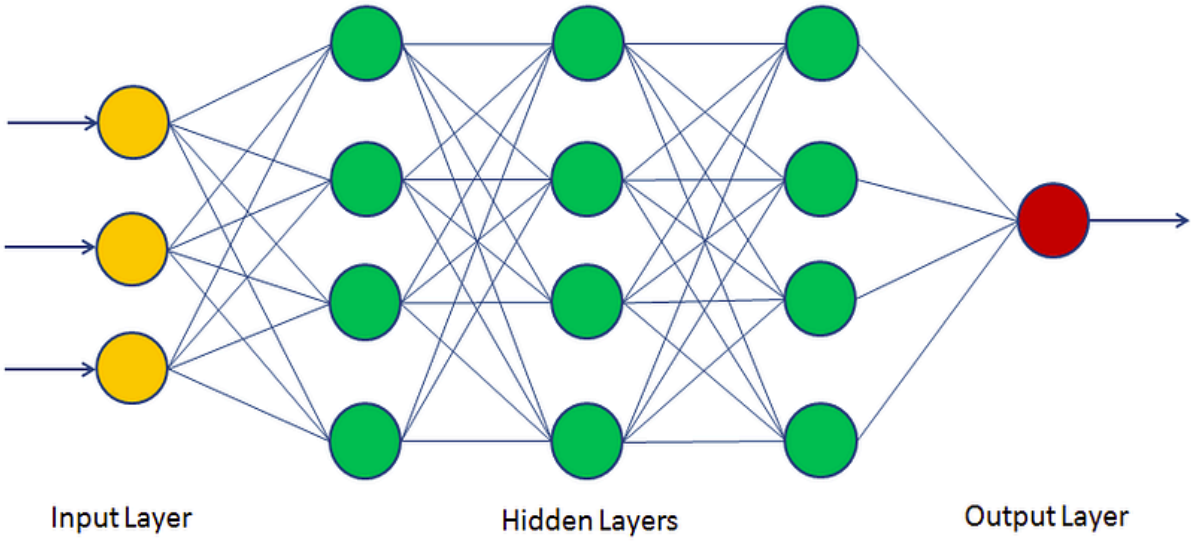
```
11369 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11370 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11371 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11372 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11373 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11374 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11375 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11376 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11377 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11378 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11379 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11380 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11381 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11382 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11383 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11384 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11385 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11386 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11387 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11388 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11389 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11390 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11391 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11392 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11393 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11394 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11395 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11396 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11397 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11398 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11399 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11400 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11401 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11402 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11403 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11404 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11405 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
11406 81,0,84,204,192,0,0,0,0,7,255,399,192,0,0,0,0,-195,0,1,-171,0,0,0,0,0,0,0,0,0,0,0
```

CHOOSING MLPS

We selected a Multi-Layer Perceptron (MLP) due to the following reasoning:

- **Game Characteristics:**
  - The task of predicting 12 boolean button states is a multi-label classification problem, which MLPs handle effectively due to their ability to model non-linear relationships.
  - The game does not require sequential decision-making over long time horizons (unlike reinforcement learning tasks), as each frame's action depends primarily on the current state. This made MLPs, which process inputs independently, a natural fit.
- **MLP Advantages**:
  - **Not Too Advanced**: They aren't complex making them feasible for our project's scope and the emulator's real-time requirements. They avoided the complexity of models like CNNs, which are better suited for image-based inputs rather than our structured numerical data.
  - **Not Too Basic**: Compared to simpler models like logistic regression, MLPs can capture complex interactions between features (e.g., how p1_rel_x and p1_health_diff influence attack decisions). Their hidden layers allow modeling of non-linear patterns, such as when to execute a special move based on opponent proximity and health.
  - **Simplicity of Implementation**: This allowed us to focus on feature engineering and data collection rather than grappling with framework complexities.
  - **Flexibility**: MLPs can handle the multi-label output (12 buttons) by using sigmoid activation in the output layer, accommodating the game's need for simultaneous button presses (e.g., act_Up and act_X for a jump kick).
- **Alignment with Project Goals**:
  - The project required a generic bot capable of playing any character, and MLPs excel at generalizing across diverse inputs when trained on a large dataset like ours.
  - The structured nature of our 300,000-sample dataset, with clear feature-action pairs, aligned well with MLP's supervised learning paradigm, avoiding the need for unsupervised or reinforcement learning approaches (explicitly prohibited).

MLPS



I have included an illustrative image of an **MLP** with only **3** input neurons for simplicity. If I were to visualize the full model with all **35+** input variables (such as timer, player positions, health values, actions, etc.), the diagram would be too large and detailed to fit meaningfully within a single image.

MODEL DEVELOPMENT

The development process included:

1. **Input Features**:
   - The model used all 35 variables, including player attributes, opponent data, relative features, and game context.
   - Relative features (p1_rel_x, p1_rel_y, p1_facing_opponent, p1_health_diff) were particularly important for strategic decisions, such as whether to approach or retreat.
2. **Output Labels**:
   - The model predicted the 12 button states (act_B, act_Y, etc.) as a multi-label classification task, allowing simultaneous button predictions.
3. **Architecture**:
   - The MLP had:
     - An input layer matches the feature count.
     - Three hidden layers with 256, 128, and 64 neurons, using ReLU activation to model complex patterns.
     - An output layer with 12 neurons and sigmoid activation, producing probabilities for each button (thresholded at 0.5).
   - We experimented with layer sizes and settled on this configuration for optimal performance without overfitting.
4. **Training**:
   - The dataset was split into 80% training (240,000 samples) and 20% testing (60,000 samples).
   - We trained the model using the Adam optimizer (learning rate 0.001) for 150 epochs, with early stopping (patience of 10 epochs) to prevent overfitting.
   - Binary cross-entropy loss was used for the multi-label task, with class weights to address imbalances (e.g., act_Start was rarely pressed).
   - Training was performed on a standard laptop, taking ~2 hours due to the dataset size and model complexity.
5. **Evaluation**:
   - The model achieved an average accuracy of **70%** across all button predictions on the test set, with precision and recall above 80% for most buttons.
   - We analyzed performance by character and matchup, confirming consistent results across all 12 characters.
   - The winning variable showed the bot won ~70% of rounds against CPU opponents, indicating effective decision-making.

INTEGRATION WITH THE API

We integrated the trained MLP into the bot.py file by updating the fight function, which processes the GameState object and returns a Command object. The process involved:

1. **Feature Extraction:**
    - The function extracted the 35 variables from GameState, computing relative features (p1_rel_x, p1_rel_y, p1_health_diff, p1_facing_opponent) dynamically.
    - Features were preprocessed to match the training data (normalization, one-hot encoding).
2. **Prediction:**
    - The preprocessed features were fed into the MLP to predict the 12 button probabilities.
    - Probabilities were thresholded at 0.5 to set binary button values.
3. **Command Creation:**
    - The predicted button states were assigned to a Command object's Buttons instance, returned to the game for the next frame.
4. **Implementation Details:**
    - We used joblib to save and load the trained model, enabling efficient integration.
    - The integration was tested to ensure real-time performance within the emulator's frame rate.

TESTING

We conducted extensive testing to validate the bot's performance:

- **Single-Player Mode**: Tested against CPU opponents with all 12 characters, ensuring the bot adapted to different opponent styles.
- **Edge Cases**: Evaluated performance in low-health scenarios, cornered positions, and during special move executions.
- **Diversity**: The diverse dataset (from multiple players and strategies) ensured the bot handled varied opponent behaviors, such as aggressive or defensive play.

IMAGES


12 fps
1P 2500 HCK 50000 2P 1100
KO
RYU
38
KEN


11 fps
1P 1500 HCK 50000 2P 600
KO
RYU
46
KEN


171 fps
1P 3000 HCK 50000 2P 1600
KO
RYU
33
KEN
RYU WINS

CONCLUSION

Our team successfully developed an AI bot for **Street Fighter II Turbo** using a **Multi-Layer Perceptron** model trained on a **300,000-sample** dataset collected from diverse gameplay across all 12 characters. The **35-variable** feature set, including spatial, contextual, and relative features, enabled the MLP to achieve a **70% accuracy** in predicting button inputs, resulting in effective gameplay against CPU opponents. Choosing MLPs for their **simplicity** and ability to model the game's non-linear decision space ensured a **robust, generalizable bot**. This project showcased our ability to apply machine learning to a complex, real-time task, producing a bot that performs reliably across varied characters and scenarios, and deepened our understanding of AI-driven game development.

THANK YOU!

NATIONAL UNIVERSITY
of Computer & Emerging Sciences