Artificial Intelligence Project

Abdul Rehman i21-1780
Meeran Ali i21-1743
Shaheeruddin Sheikh i21-1729
Ahsan Saqib i21-1683

DS-D

Machine Learning Street Fighter Bot

Introduction **2**
1. Project Overview **2**
1.1 Goals 2
1.2 Tools Used 2
2. How It Was Built **2**
2.1 Collecting Game Data 2
2.2 Preparing the Data 3
2.3 Training the Model 3
2.4 Making the Bot Work 3
3. System Overview **4**
3.1 Process Flow 4
3.2 Files 4
4. Results **4**
4.1 Model Accuracy 4
4.2 Problems 4
5. Ideas for Improvement **5**
5.1 Short-Term Fixes 5
5.2 Long-Term Upgrades 5
6. Conclusion **5**
7. Extra Info **5**
7.1 Features We Used 5
7.2 Model Settings 6
7.3 Data Processing Steps 6
7.4 Bot Flow 6
8. References **7**

Introduction

This report explains how we built a bot that plays *Street Fighter II Turbo* using machine learning. The bot learns from human gameplay and predicts which buttons to press based on what's happening in the game.

1. Project Overview

1.1 Goals

- Collect data from actual gameplay
- Clean and process the data
- Train a machine learning model to decide the best move
- Use the model in real time to control a bot

1.2 Tools Used

- Python 3.8
- scikit-learn (for machine learning)
- pandas (for working with data)
- BizHawk (emulator to run the game and interact with it)

---

2. How It Was Built

2.1 Collecting Game Data

We collected data like:

- Game timer
- Health of each player
- Position of players on the screen
- Player actions (jumping, crouching, etc.)
- Button inputs like A, B, X, Y, and arrow keys
- Timing of each button press

2.2 Preparing the Data

- Removed incomplete or missing data
- Converted true/false values into 0s and 1s
- Normalized the data (so all values are on the same scale)
- Added new useful features (like distance between players)
- Split the data into training (80%) and testing (20%)

2.3 Training the Model

- Used a Random Forest Classifier
- Trained the model to predict which buttons should be pressed
- Settings used:
    - 100 decision trees
    - Max depth of 10

    - Minimum 5 samples to split

2.4 Making the Bot Work

- The bot reads the game state in real time
- Processes the game state like we did with training data
- Sends the predicted button press commands to the game

---

3. System Overview

3.1 Process Flow

arduino
CopyEdit
`Collect Data → Process Data → Train Model → Use Bot`

3.2 Files

- `logs/`: Raw gameplay data
- `processed_data/`: Cleaned and ready-to-use data
- `models/`: Saved machine learning model and scalers

---

4. Results

4.1 Model Accuracy

- Accuracy on training data: 50%
- Accuracy on test data: 52%

- The bot makes decisions in less than 100 milliseconds

4.2 Problems

- Depends a lot on the quality of data
- Doesn’t recognize long combos or patterns over time
- Limited to the features we included

---

5. Ideas for Improvement

5.1 Short-Term Fixes

- Add more features
- Try better ways to select features
- Improve scaling and training methods

5.2 Long-Term Upgrades

- Use LSTM (a type of neural network that understands sequences)
- Add reinforcement learning (so the bot learns by playing)
- Mix different types of models
- Teach the bot to recognize combos and use better strategy

---

6. Conclusion

This project shows that machine learning can be used to play a complex game like *Street Fighter II Turbo*. While it works well, we can make it even better by using smarter models and more advanced techniques.

---

7. Extra Info

7.1 Features We Used

- Numbers: timer, health, positions, move IDs, distance
- Booleans (0 or 1): jumping, crouching, doing a move

7.2 Model Settings

python
CopyEdit

```
RandomForestClassifier(
    n_estimators=100,
    max_depth=10,
    min_samples_split=5,
    random_state=42
)
```

7.3 Data Processing Steps

- Load raw data
- Clean and normalize
- Split into train/test
- Scale features
- Train and save the model

7.4 Bot Flow

- Read game state
- Extract features

- Normalize
- Predict move
- Press buttons

---

8. References

- scikit-learn documentation
- Street Fighter II gameplay mechanics
- BizHawk emulator guide
- Game AI research papers