Enhancing Player Enjoyment with a Two-Tier DRL and LLM-Based Agent System for Fighting Games

Shouren Wang, Zehua Jiang, Fernando Sliva, Sam Earle, Julian Togelius

INTRODUCTION

- Most existing agent studies on ighting games focus only on the win rate, and largely overlook player enjoyment.
- Conventional methods employing MCTS or DDQN face practical challenges due to computational costs and an excessive focus on win rates.

- This study aims to maximize the enjoyment players experience during matches.

INTRODUCTION

Proposed method: the two-tier agent (TTA) system

- The first layer houses a DRL-based agent for gameplay, while the second layer houses an LLM that selects an opponent from the first layer.
- A model architecture designed for TTA, a modular reward function composed of multiple reward terms, and a hybrid self-play learning method.

PRELIMINARIES

Environment
Street Fighter II: Champion Edition (12 balanced characters with special moves) on OpenAI Gym Retro; special moves require precise, temporally-extended input sequences.

Task formulation
Task formulation modeled as an MDP. The Retro emulator is deterministic, so PvE (vs. built-in CAPCOM AI) stays deterministic, while self-play (vs. past DRL agents) makes the environment stochastic.

Deep RL
The policy network is trained with PPO (implemented via stable-baselines3).

LLM & hyper-agent
In-Context Learning lets an LLM (DeepSeek-R1) handle tasks from prompts alone; a hyper-agent selects the best sub-agent rather than acting directly.

PRELIMINARIES

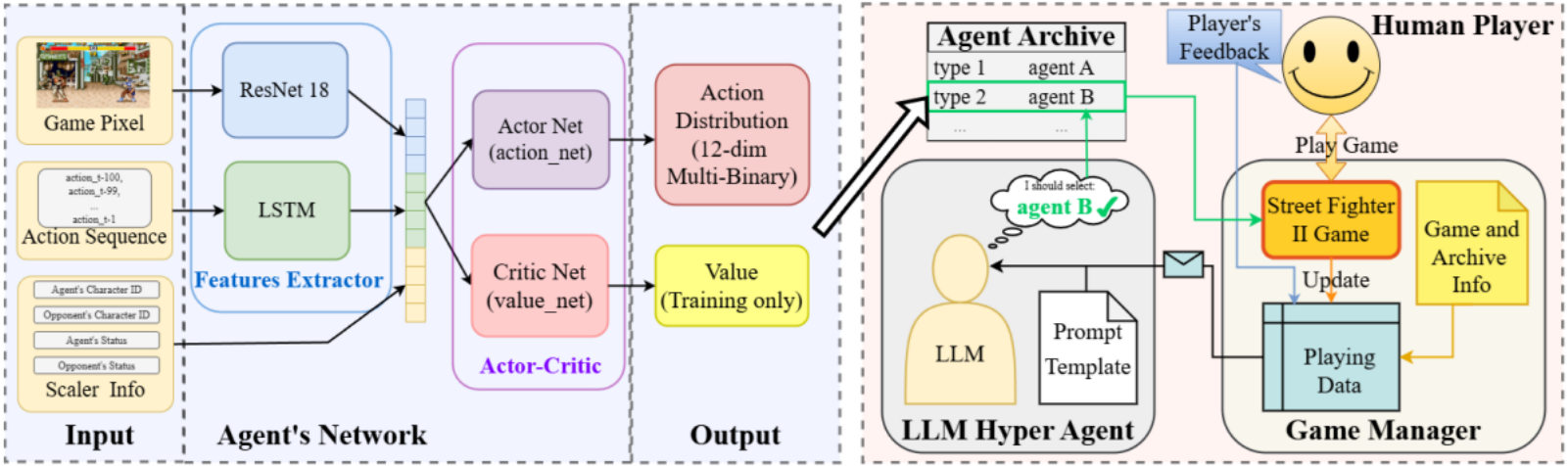
Four factors hypothesized to drive player enjoyment in competitive matches

Opponent skill level
Close, balanced matches against evenly matched opponents are the most enjoyable.

Advanced techniques
Players enjoy opponents who use advanced skills; losing to a skill-less opponent feels unfulfilling, a situation often called "smurfing."

Opponent diversity
Facing varied play styles is more enjoyable than dealing with repeated, various tactics.

Matchmaking
Selecting suitable opponents from player data ties these factors together and significantly boosts enjoyment.


Game Pixel
Action Sequence
Agent's Character ID
Opponent's Character ID
Agent's Status
Opponent's Status
Scaler Info
Input
ResNet 18
LSTM
Features Extractor
Agent's Network
Actor Net (action_net)
Critic Net (value_net)
Actor-Critic
Action Distribution (12-dim Multi-Binary)
Value (Training only)
Output
(a)
Agent Archive
type 1 agent A
type 2 agent B
I should select:
agent B
LLM
Prompt Template
LLM Hyper Agent
Player's Feedback
Human Player
Play Game
Street Fighter II Game
Update
Game and Archive Info
Playing Data
Game Manager
(b)

METHOD

Network architecture of the DRL agent

Inputs

game pixels, scalar in-game states, and the past n action steps.

Feature extractor

- ResNet18 (CNN) for visual features. The CNN extractor processes game pixels, extracting visual features and producing a feature vector.
- LSTM (RNN) for temporal action-sequence features. Extract temporal features from the action history of the past n steps.
- Scalar in-game state information is also concatenated into the vector.

Actor-Critic

The actor (MLP) outputs a 12-dim multi-binary action distribution; the critic estimates the value function (training only).

METHOD

Modularized reward function and hybrid training

Reward terms

- Raw HP/damage, special-move, projectile, distance, in-air, time, and action costs, each computed separately and summed per step.

Two inputs

- An info dictionary (detects actions/states).
- A customizable reward-terms dictionary (per-component coefficients).

Diverse agent types

Reward configs yield projectile, special-move, defensive, air, coward, newbie, and key-spamming agents.

Hybrid training

PvE-only overfits into a fixed macro and self-play-only collapses to kick-spamming, so the two are mixed (30% PvE / 70% self-play).

METHOD

Hyper-agent tier: selecting each player's next opponent

Agent archive
Agent archive stores the DRL agents grouped by play style.

Game Manager (GM)
Maintains the agent archive, records player data, ask players for feedback, and interacts with the LLMHA to select the next opponent for the player.

LLM Hyper-Agent (LLMHA)
Integrates the information from the GM with a prompt template and uses a LLM to infer the next opponent.

EXPERIMENTS

Experimental setup

DRL agent

- ResNet-18 used without modification
- For the PPO algorithm, value function coefficient = 1, entropy coefficient = 0.01, and learning rate linearly decayed from 2.5×10^{-4} to 2.5×10^{-6}.

Hybrid training

30% PvE / 70% self-play, a 50% character-flip rate, and 12 parallel environments.

Hyper-agent

DeepSeek-R1-Distill-Llama-8B run locally with 8-bit quantization on an RTX 3080 laptop GPU (16GB VRAM).

TABLE I
NETWORK PARAMETER

LSTM (RNN)		Actor Network		value Network	
Parameters	**Values**	**Parameters**	**Values**	**Parameters**	**Values**
Input dim	12	Layer1 dim	512	Layer1 dim	512
Hidden dim	128	Layer2 dim	256	Layer2 dim	256
Num layers	2	Layer3 dim	128	Layer3 dim	128
Batch first	True	Layer4 dim	128	Layer4 dim	128
Dropout	0.1	Output dim	12	Output dim	1
		Activation	Tanh	Activation	Tanh

The CNN module is default ResNet18 and not listed here.

TABLE II
REWARD TERMS

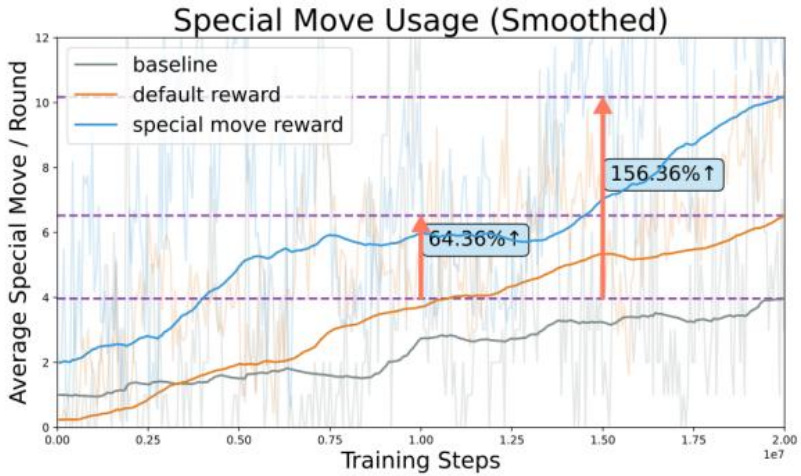
Parameters	Description
Reward scale	Scale the reward by multiplying it by this term
Raw reward coefficient	Scale the HP-based reward by multiplying it by this term
Special move bonus	Scale the HP-based reward caused by special moves by multiplying it by this term
Projectile bonus	Scale the HP-based reward caused by projectiles by multiplying it by this term
Distance bonus	Scale the HP-based reward according to distance between agent and opponent by multiplying it by this term
Special move reward	Reward for triggering a special move
Projectile reward	Reward for triggering a projectile
Distance reward	Positive/Negative value for reward keeping a long/close distance from the opponent
In air reward	Reward for being in air
Time reward	Positive/Negative value for reward take more/less time for each round
Cost coefficient	Scale the cost (penalty) by multiplying it by this term
special move cost	Cost for triggering a special move
Regular attack cost	Cost for triggering a regular attack
Jump cost	Cost for triggering a jump
Vulnerable frame cost	Cost for being in a vulnerable frame, i.e. frames that agent cannot control the character

The description of modules in the modularized reward function.

EXPERIMENTS

Mastery of Advanced Techniques

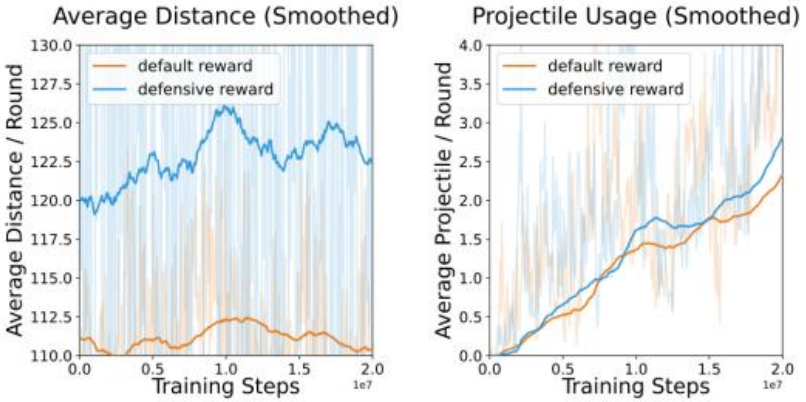
We compared the agent trained using the proposed architecture and modular rewards against the baseline (CNN+MLP).



EXPERIMENTS

Distinct game-playing styles

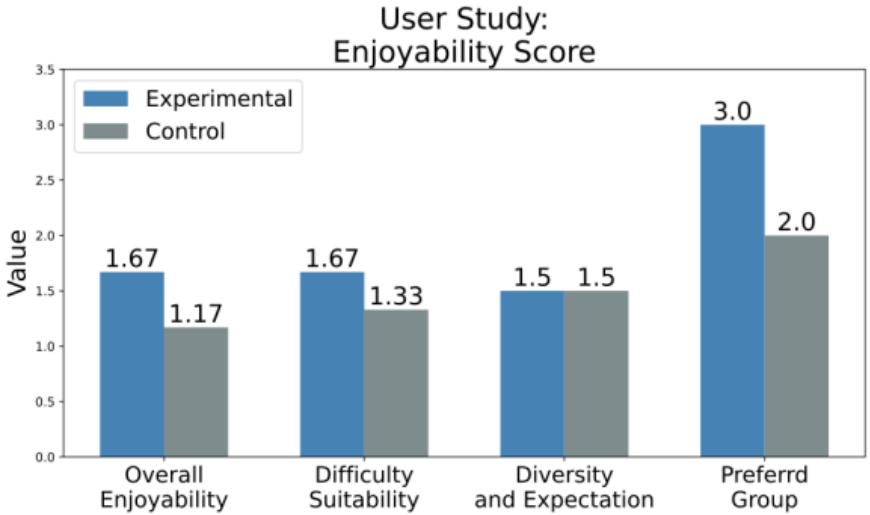
For example, the defensive type showed a tendency to maintain a longer distance from the opponent and use projectiles a little more than the default agent, confirming that the reward function can induce style learning.



EXPERIMENTS

User Study

- They ran a small user study with six participants.
- The experimental group faced opponents chosen by the LLMHA, and the control group faced randomly chosen ones, five matches each.
- Single-choice items scored 0–3; aligned with the PENS framework (competence, autonomy).



DISCUSSION

Limitations and open challenges

Ranged opponents
Strong in close-range melee, but weak against projectile/zoning play, since special-move recovery frames get punished.

Imprecise skills
Special moves are learned only approximately (regular attacks rise too); stronger LSTM or transformer attention could improve timing.

High-level tactics
Spacing, link combos, and hit-confirm aren't learned yet — possibly due to limited training, low utility for non-human agents, or architectural limits.

Action space
The 12-dim multi-binary action space may be too simplistic; HRL or learned action representations could help.

CONCLUSION

- TTA has the ability to generate diverse and skilled DRL agents.
- An LLM hyper-agent dynamically selects suitable opponents from gameplay data and feedback.
- Special-move usage +156.36% and player enjoyment +42.73%, confirming the system's effectiveness.

Future work

Stronger long-range adaptation, transformer-based temporal precision, and RLHF instruction tuning for the hyper-agent.

Thank you for listening