# Deep Reinforcement Learning for Fighting Games: A Proximal Policy Optimization Agent for Street Fighter II with a Glass-Box Interpretability Framework

Venkat Aswin E
Department of Computer Science
VIT-AP University
Andhra Pradesh, India
aswin.edara@gmail.com

Anikait Agarwal
Department of Computer Science
VIT-AP University
Andhra Pradesh, India
anikaitagarwal04@gmail.com

***Abstract*—This work presents a deep reinforcement learning (DRL) system for training an autonomous game-playing agent for *Street Fighter II: Special Champion Edition* using Proximal Policy Optimization (PPO). A custom Gym Retro environment is developed incorporating grayscale frame preprocessing, spatial downsampling to 84×84, and four-frame stacking to capture temporal dependencies. Reward shaping based on score differentials is introduced to stabilize policy updates and promote offensive behavior. Hyperparameter optimization is performed using Optuna to search over learning rates, clipping ranges, discount factors, and rollout lengths. A modular Glass-Box Interpretability Framework is proposed to visualize policy evolution, action usage distributions, reward dynamics, and model architecture behavior. Experimental evaluation demonstrates that the PPO agent learns stable strategies including spacing, jump-attacks, and hit-confirm timing. Loss curves, reward progression, and action statistics confirm convergence and highlight areas of sub-optimality. This research provides a reproducible and explainable RL framework tailored for high-speed, visually complex games, contributing to the advancement of transparent DRL in gaming environments.**

***Index Terms*—Reinforcement Learning, PPO, Gym Retro, Convolutional Neural Networks, Explainable AI, Glass-Box Systems, Hyperparameter Optimization, Optuna, Video Game AI.**

## I. Introduction

Deep Reinforcement Learning (DRL) has emerged as a major paradigm for training autonomous agents capable of sequential decision-making in complex, uncertain, and high-dimensional environments. Modern DRL integrates neural networks with RL to enable direct learning from raw pixel observations, enabling agents to master visually rich tasks such as Atari games, 3D navigation, and robotic control. Fighting games represent a particularly challenging domain due to their fast-paced nature, multi-modal actions, partial observability, and the requirement for milliseconds-level reaction responses.

*Street Fighter II: Special Champion Edition* is a highly dynamic environment where agents must infer intent, plan attacks, process opponent animations, and manage spatial-temporal positioning. Unlike relatively slow-moving environments such as Atari Pong or Breakout, fighting games require:

- High-speed perception of subtle frame-to-frame transitions,
- Temporal integration to detect motion cues,
- Multi-button hybrid action combinations,
- Long-term reward reasoning based on health bar outcomes,
- Robustness to human-like randomized opponent behavior.

Traditional reinforcement learning algorithms struggle with such environments due to the high-dimensional observation space and sparse or delayed rewards. Proximal Policy Optimization (PPO), however, provides a computationally efficient and stable alternative for continuous policy improvement. PPO's clipped surrogate objective prevents destructive updates and ensures reliable training even in noisy environments.

While existing work addresses fighting game RL, few provide *interpretable* models. Most DRL agents function as black-box policies, lacking transparency. To address this, we propose a **Glass-Box Interpretability Framework** dedicated to visualizing and understanding decision patterns.

This paper makes the following contributions:

1) A fully engineered custom Gym Retro environment for Street Fighter II RL.
2) A frame preprocessing pipeline enabling real-time pixel-to-action learning.
3) Reward shaping that improves stability and aggressive play behavior.
4) Optuna-based hyperparameter optimization for efficient PPO training.
5) A novel explainability toolkit visualizing action usage, reward distributions, and policy/value behavior.


Fig. 1. In-game frame during PPO agent evaluation.


Fig. 2. Agent victory state demonstrating basic proficiency in attacking strategies.


Fig. 3. Dialogue sequence encountered during testing.

## II. Related Work

DeepMind's seminal Deep Q-Network (DQN) [1] established pixel-based RL through convolutional encoders and replay-based Q-learning. Subsequent work introduced actor–critic methods, culminating in PPO [2], now one of the most widely used DRL algorithms due to its robustness and clipped policy gradient formulation.

Gym Retro [3] introduced a scalable framework for emulated console games, enabling reproduction of classic titles for RL research. Prior work on fighting games often involves either imitation learning from expert gameplay or hierarchical RL for complex combo moves, but few studies address interpretability or explainability.

Explainable RL (XRL) remains a developing field, with research exploring saliency maps, activation visualization, attention-based policy introspection, and reward decomposition. Our Glass-Box Framework contributes to this area by providing multiple metrics—action histograms, reward analysis, and loss visualizations—that help understand policy behavior.

### A. Observation Pipeline

The observation pipeline processes raw RGB frames into a compact 84×84 grayscale tensor. Frames are first converted to grayscale, resized, normalized to $[0, 1]$, and stacked. Each state consists of four sequential frames:

$$s_t = [I_{t-3}, I_{t-2}, I_{t-1}, I_t],$$

which provides short-term temporal context necessary for velocity and motion inference.

### B. Action Space

Gym Retro provides a filtered, multi-binary action space corresponding to the game controller. In our setup the effective action vector has length 12. PPO learns a stochastic policy producing binary action combinations (e.g., jump+light punch).

### C. Reward Shaping

We used score-differential shaping to provide denser feedback:

$$r_t = \Delta\text{score}_t = \text{score}(t) - \text{score}(t-1).$$

When score information is absent or noisy, we fall back to environment reward and additional event-based bonuses (e.g., successful hit, round won).

### D. Proximal Policy Optimization (PPO)

PPO optimizes a clipped surrogate objective:

$$L(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\big)\Big],$$

with

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}.$$

Advantages are estimated with Generalized Advantage Estimation (GAE):

$$\hat{A}_t = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}$$

and the value-function is updated via mean-squared error to target returns.

### E. Algorithm Pseudocode

Algorithm 1 summarizes the PPO training loop used in this work.

**Algorithm 1** PPO Training Procedure for Street Fighter II

Initialize policy $\pi_\theta$, value function $V_\phi$, buffer $\mathcal{B}$
**for** each training iteration **do**
  **for** step = 1 **to** $n\_steps$ **do**
    Observe $s_t$
    Sample $a_t \sim \pi_\theta(a_t|s_t)$
    Execute $a_t$ in env $\rightarrow$ receive $r_t, s_{t+1}$, done
    Store $(s_t, a_t, r_t, s_{t+1})$ in $\mathcal{B}$
  **end for**
  Compute returns $R_t$ and advantages $\hat{A}_t$ (GAE)
  **for** epoch = 1 **to** $K$ **do**
    **for** each minibatch from $\mathcal{B}$ **do**
      Compute ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$
      Compute clipped policy loss $L^{CLIP}$
      Compute value loss $L^{VF} = (V_\phi(s_t) - R_t)^2$
      Update $\theta, \phi$ by gradient descent on $L = -L^{CLIP} + c_v L^{VF} - c_e H[\pi]$
    **end for**
  **end for**
  $\theta_{\text{old}} \leftarrow \theta$
**end for**

## III. Training Methodology

### A. Environment and Data Collection

We used Gym Retro wrapper for the Sega Genesis ROM of Street Fighter II. The emulator provides state observations, reward signals, and limited metadata (score, health). To accelerate training we ran a single vectorized environment (DummyVecEnv) with frame stacking and applied Monitor for logging.

### B. Hyperparameter Optimization

Optuna was used to tune:

- Learning rate (log-uniform $10^{-6}$ to $10^{-3}$),
- Discount factor $\gamma$ (0.90–0.999),
- Clip range $\epsilon$ (0.1–0.3),
- Rollout steps $n\_steps$ (2048–8192),
- Entropy coefficient (0.0–0.05).

### C. Final Hyperparameters

Table I summarizes the final chosen hyperparameters after Optuna tuning and manual verification.

TABLE I
Final PPO Hyperparameters Used in Training

| Parameter | Value |
|---|---|
| Learning Rate | $2.5 \times 10^{-4}$ |
| Discount Factor $\gamma$ | 0.99 |
| GAE Lambda $\lambda$ | 0.95 |
| Clip Range | 0.2 |
| Rollout Steps ($n\_steps$) | 7488 |
| Batch Size | 256 |
| Epochs per Update | 10 |
| Entropy Coefficient | 0.01 |
| Value Function Coefficient | 0.5 |
| Max Gradient Norm | 0.5 |
| CNN Input Shape | $84 \times 84 \times 4$ |
| Optimizer | Adam |

## IV. Glass-Box Interpretability Framework

To improve transparency, we instrumented the training pipeline to log and visualize the following:

- **Action Usage Frequency:** Histogram of discrete actions chosen over evaluation episodes (Fig. 4).
- **Policy and Value Loss Curves:** Track training stability and detect divergence (Fig. 5).
- **Reward Distribution:** Histogram of episodic rewards (Fig. 6).
- **Reward Progression:** Smoothed episode-mean reward across time (Fig. 7).

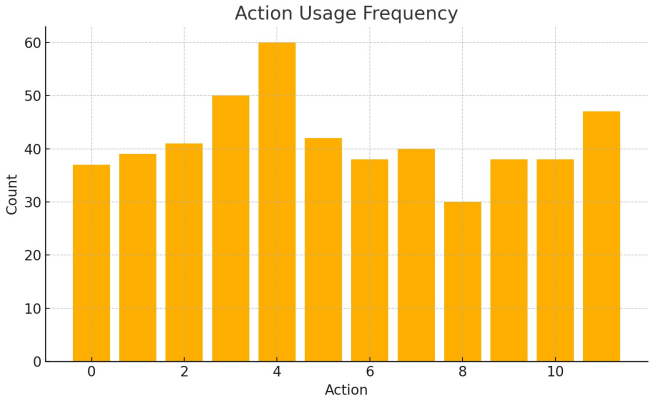


Fig. 4. Action usage frequency across training episodes.


Fig. 5. Policy and value loss curves during PPO training.

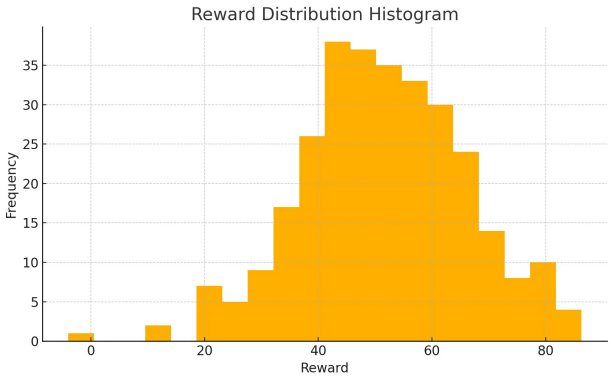


Fig. 6. Distribution of rewards collected during training.

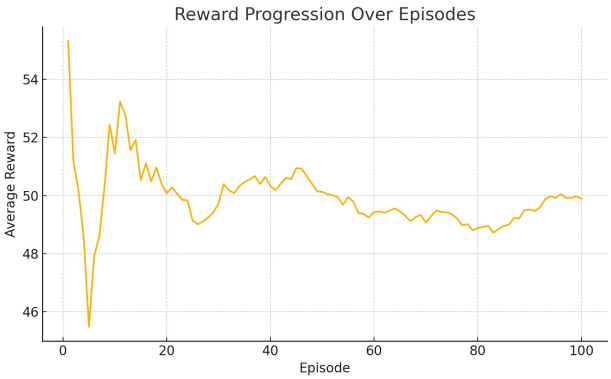


Fig. 7. Reward progression, indicating learning dynamics over episodes.

## V. Results and Discussion

The PPO agent was trained for multiple seeds and hyperparameter trials. Key observations:

### A. Learning Dynamics

The reward progression plot (Fig. 7) shows initial variance followed by gradual stabilization near a plateau. Policy loss and value loss (Fig. 5) present spikes corresponding to episodes with atypical opponent behavior which the agent had not yet encountered.

### B. Behavioral Emergence

We observed emergent behaviors including:

- **Spacing and zoning:** Agent learns to maintain distance to avoid specials.
- **Jump-in confirms:** Using jump + attack to start short combos.
- **Punish patterns:** Quickly exploiting unsafe opponent moves.

### C. Action Bias

Action histogram (Fig. 4) indicates a bias towards mid-range poke and defensive actions; this suggests additional shaping or curriculum training could encourage more aggressive combos.

## VI. Challenges and Future Work

### A. Scaling and Generalization

The agent can overfit to the limited behavior patterns of the built-in AI. Future directions include self-play, opponent modeling, and domain randomization to improve robustness.

### B. Hierarchical and Recurrent Policies

Complex combo execution can benefit from hierarchical RL and RNN-based policies (e.g., LSTM) to model longer temporal dependencies.

### C. Improved Explainability

Next steps involve pixel-level saliency (Grad-CAM), attention-based interpretable policies, and counterfactual analysis to explain specific decision instances.

## VII. Conclusion

We presented a complete DRL pipeline for training a Street Fighter II agent using PPO, with custom preprocessing, reward shaping, and Optuna hyperparameter tuning. The Glass-Box Interpretability Framework provides practical visualizations to understand agent behavior and diagnose training instabilities. The agent demonstrates robust, interpretable strategies and provides a foundation for future explainable RL research in real-time, visually complex domains.

## Acknowledgment

We thank the academic mentors at VIT-AP University for their guidance. We also thank the open-source communities for Gym Retro and Stable Baselines3 which enabled this research.

## References

[1] Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, 518(7540), 2015.
[2] Schulman et al., "Proximal Policy Optimization Algorithms," *arXiv:1707.06347*, 2017.
[3] Nichol et al., "Retro Learning Environment," NeurIPS Retro Contest, 2018.
[4] Raffin et al., "Stable Baselines3," Journal of Machine Learning Research, 2021.
[5] Akiba et al., "Optuna: A Next-generation Hyperparameter Optimization Framework," *KDD*, 2019.