ESPECIAL

Recibido 14 Oct. 2024

Aceptado 15 Dec. 2024

ReCIBE, Año 13 No. 3, Dec.2024

Desarrollo de un agente inteligente capaz de jugar un videojuego de peleas

Development of an Intelligent Agent Capable of Playing a Fighting Video Game

Montes Perez Alan A[1*]
Torres Soto Aurora[1*]
Torres Soto M. Dolores[1*]

[1] Universidad Autónoma de Aguascalientes, Aguascalientes AGS 20131, MEX
*Correspondencia: alanroggersmp@outlook.com, aurora.torres@edu.uaa.mx ,dolores.torres@edu.uaa.mx

Abstract. Los videojuegos son una herramienta importante como campo de pruebas de algoritmos de aprendizaje por refuerzo, sin embargo, también son una buena herramienta para generar inteligencia artificial para los mismos. Esta investigación se dividió en dos fases generales, el desarrollo de un demo de videojuego de peleas llamado Brain Fighter y la creación de un modelo de aprendizaje por refuerzo basado en el algoritmo llamado Proximal Policy Optimization para entrenar un agente inteligente de modo que sea capaz de jugar satisfactoriamente Brain Fighter.

En este artículo se presenta el mejor modelo obtenido como resultado de haber experimentado con seis modelos diferentes, además se expone la metodología utilizada para llevar a cabo toda la investigación, que constituye desde la creación de Brain Fighter en Unity, la elección del algoritmo de aprendizaje por refuerzo a utilizar disponible en ML Agents, la creación del modelo, el entrenamiento y los resultados obtenidos además de las conclusiones generales del trabajo.

Keywords: Inteligencia Artificial, Videojuegos, Agente inteligente, Proximal Policy Optimization, Aprendizaje por refuerzo, Self Play, ML Agents, Unity

Abstract. Video games are an important tool as a testing ground for reinforcement learning algorithms; however, they are also a valuable resource for generating artificial intelligence within games themselves. This research was divided into two general phases: the development of a fighting video game demo called Brain Fighter and the creation of a reinforcement learning model based on the Proximal Policy Optimization algorithm to train an intelligent agent capable of playing Brain Fighter satisfactorily.

This article presents the best model obtained as a result of experimenting with six different models. It also details the methodology used to carry out the research, which includes the creation of Brain Fighter in Unity, the selection of the reinforcement learning algorithm available in ML Agents, the development of the model, its training process, the results obtained, and the general conclusions of the work.

Keywords: Artificial Intelligence, Video Games, Intelligent Agent, Proximal Policy Optimization, Reinforcement Learning, Self-Play, ML Agents, Unity

1 Introducción

Los videojuegos son una herramienta esencial como campo de pruebas para el aprendizaje por refuerzo (Tipo de aprendizaje automático (Tantawi PhD, 2023)) debido a que permiten simular características del mundo real, sin embargo, los videojuegos también se pueden beneficiar de los avances en el campo de la inteligencia artificial, pues con ayuda de esta tecnología es posible crear recursos gráficos e incluso comportamiento avanzado para los personajes no jugables.

El objetivo de este trabajo es exponer la forma como se creó y diseñó un agente inteligente que sea capaz de jugar un videojuego de peleas utilizando el aprendizaje por refuerzo y el algoritmo que lleva por nombre “Proximal Policy Optimization” (Más delante en este trabajo, se explica en que consiste este algoritmo).

En la actualidad existen varios proyectos que se han desarrollado con éxito utilizando el aprendizaje por refuerzo. Por ejemplo, Daniel Sarrià Pascual realizó una investigación con el objetivo de mejorar la calidad de los bots de Rocket League utilizando el algoritmo Proximal Policy Optimization y aprovechándose de la herramienta RLBot (Sarrià Pascual, 2022) (Plataforma dedicada al desarrollo, personalización y competición de bots en el videojuego Rocket League (Community, 2024)).

2 Marco Teórico

Para el desarrollo de este trabajo, fue esencial entender varios conceptos del campo de la inteligentica artificial, a continuación, se describe cada uno de los temas en los que se basó este trabajo de manera breve.

2.1 Aprendizaje por refuerzo

El aprendizaje por refuerzo (RL) se centra en diseñar métodos para abordar problemas de toma de decisiones secuenciales. Un agente toma decisiones seleccionando acciones basadas en la información proporcionada por su entorno. Una pieza crucial de esta información es la señal de recompensa, que sirve como retroalimentación de que tan bien o mal lo está haciendo el agente inteligente. El objetivo del agente es maximizar su recompensa acumulada a lo largo de cada episodio de entrenamiento (Landers & Doryab, 2023; Sutton & Barto, 2018).

En el aprendizaje por refuerzo, los algoritmos se dividen en dos categorías principales: basados en valor y basados en políticas. Los algoritmos basados en valor se enfocan en calcular el valor de acciones o estados para derivar una política óptima (estrategia que define qué acción debe tomar un agente en cada estado del entorno). Por otro lado, los algoritmos basados en políticas aprenden directamente una política óptima ajustando la probabilidad de elegir acciones beneficiosas.

Una red neuronal profunda en conjunto con un algoritmo de aprendizaje por refuerzo es a lo que se le conoce como aprendizaje por refuerzo profundo (DRL). La red neuronal permite que el aprendizaje por refuerzo se adapte a problemas de toma de decisiones que antes eran intratables, es decir, entornos con espacios de acción y estados de alta dimensionalidad (Biscontini, 2023).

2.2 Proximal Policy Optimization

El algoritmo Proximal Policy Optimization (PPO) pertenece a la categoría de algoritmos basados en políticas que a su vez pertenece al área del DRL e introduce una nueva noción: la proximidad (parámetro que garantiza que, al actualizar una política, no se aleje demasiado de sus valores actuales).

Este enfoque se inspira en el concepto de “recorte”, que tiene como objetivo limitar el alcance de las actualizaciones para evitar cambios abruptos. El resultado es una convergencia más estable y un mejor rendimiento del aprendizaje (Gu et al., 2022).

PPO sigue un enfoque iterativo: el agente interactúa con el entorno, recopila datos de entrenamiento, actualiza sus políticas de acuerdo con el principio de proximidad y luego repite el proceso para mejorar el rendimiento del agente con el tiempo (Gu et al., 2022).

2.3 ML Agents

El conjunto de herramientas de Unity llamado ML-Agents es un proyecto de código abierto que posibilita a investigadores y desarrolladores la creación de entornos simulados mediante el editor de Unity, permitiendo la interacción a través de una API de Python. Este conjunto de herramientas incluye el SDK de ML-Agents, que contiene todas las funciones necesarias para definir entornos en el editor de Unity.

Las características destacadas del conjunto de herramientas abarcan varios entornos de ejemplo y algoritmos de aprendizaje por refuerzo de última generación como PPO. Además, ofrece soporte para Self-Play en juegos simétricos y asimétricos, junto con la posibilidad de mejorar el desempeño de los algoritmos con el Módulo de Curiosidad Intrínseca (ICM) y la Célula de Largo Corto Plazo (LSTM) (Juliani et al., 2018).

2.4 Self Play

Self Play es una técnica utilizada en el aprendizaje por refuerzo cuando se requiere entrenar un agente inteligente que aprenda a jugar un juego de suma cero (por ejemplo, un juego de peleas 1 contra 1). Prácticamente el agente compite contra sí mismo partiendo de una política completamente aleatoria y conforme las iteraciones de entrenamiento pasan, el agente comienza a mejorar su política hasta converger en una estrategia óptima (SaturnCloud, 2023)

3 Brain Fighter

Brain Fighter (Figura 1) es un demo de videojuego de peleas desarrollado en Unity (motor gráfico). Cuenta con un personaje jugable capaz de realizar ocho golpes distintos, caminar, brincar y cubrirse. Está demo se desarrolló para tener un entorno para el modelo de RL y aunque actualmente existen algunas herramientas que nos evitan tener que desarrollar todo un entorno porque proporcionan una interfaz de comunicación entre algún juego existente y los modelos de RL, como Gym (OpenAI, 2018), no tienen la flexibilidad que se buscaba para poder experimentar (Entiéndase flexibilidad como la posibilidad de elegir de manera libre los datos que el agente inteligente recibirá en su entrenamiento).

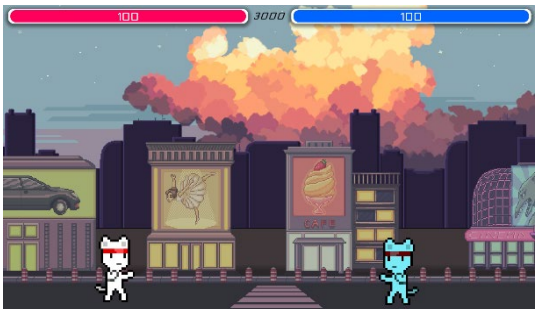

Fig. 1. Demo de Brain Fighter

4 Metodología

A modo de resumen, la Figura 2 describe las etapas que se llevaron a cabo para la realización de esta investigación. La construcción del videojuego, la selección del modelo de aprendizaje y la construcción del agente inteligente son la base de todo este trabajo porque es necesario disponer de un videojuego con posibilidad de acceso al código fuente o disponer de una forma para extraer información del juego con el fin de reunir los datos necesarios que requiere el algoritmo de aprendizaje, en este caso, se desarrolló todo el videojuego y se eligió el algoritmo PPO como algoritmo de aprendizaje por refuerzo a utilizar.

Por último, la construcción del agente inteligente implicó escoger la información que se utilizaría para entrenar, es decir, las observaciones, las señales de recompensa y las acciones posibles a realizar por el agente inteligente.

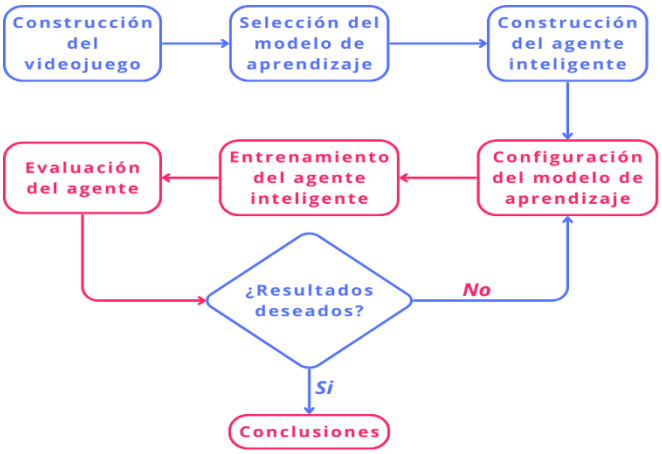

Fig. 2. Diagrama de la metodología

5 Diseño del modelo para entrenar un agente inteligente

Se diseñaron seis modelos para trabajar con ML-Agents variando los datos que el agente inteligente utilizó para entrenarse, sin embargo, solo hubo un modelo que se desempeñó correctamente en las métricas de evaluación que se presentan más delante en este trabajo y por ende los otros cinco modelos se descartaron. A continuación, se expone a detalle este modelo, el cual lleva por nombre “Modelo Delta”.

5.1 Modelo Delta

Este modelo utiliza el algoritmo PPO implementado en la herramienta ML-Agents y dio como resultado un agente inteligente que es capaz de seguir al enemigo y realizar ataques de manera satisfactoria. La tabla 1 y 2 muestra las características del modelo.

Características del modelo.

Acciones.

- No hacer nada
- Caminar
- Retroceder
- Saltar
- Golpe débil, ligero, fuerte y especial (acciones individuales)
- Patada débil, ligera, fuerte y especial (acciones individuales)
- Bloquear

Observaciones	**Posibles valores**
Estado de bloqueo del enemigo	Verdadero o Falso
Estado de daño del enemigo	Verdadero o Falso
Estado de bloqueo del agente	Verdadero o Falso
Dirección del enemigo	-1 o 1
Distancia muy cercana entre agente y enemigo	Verdadero o Falso
Vida del agente mayor a cero	Verdadero o Falso
Vida del enemigo mayor a cero	Verdadero o Falso
Resistencia del agente mayor a cero	Verdadero o Falso
Golpes encadenados	(0,1)

Tabla 1. Información (observaciones) proporcionada por el modelo

Evento que genera recompensa	Valor
Reducir distancia con el enemigo	+ 0.1
Golpear al enemigo	+ 0.075
Bloquear un ataque	+ 0.025
Incapacitar al enemigo	+ 0.075
Atacar al enemigo mientras bloquea	+ 0.025
Aumentar distancia con el enemigo	- 0.1
Recibir daño	- 0.075
Incapacitado	- 1
Ganar	+ (1 – recompensa acumulada)
Perder	- (1 – recompensa acumulada)

Tabla 2. Recompensas proporcionadas por el modelo

Ajustes adicionales.

- Enmascarado de acciones que no se pueden hacer cuando otra acción las bloquea mediante código
- Recompensa por curiosidad activada y Self Play activado
-

6 Mecanismos de evaluación para el agente inteligente

Cada vez que se entrena un agente inteligente con un modelo, se generan datos con los cuales se puede medir el rendimiento del modelo. Con esos datos se generaron gráficas del comportamiento del agente como son:

a. Gráfica de la recompensa acumulada del agente inteligente
b. Gráfica de la longitud de los episodios (Iteraciones de entrenamiento)
c. Gráfica de la pérdida de la política del agente inteligente
d. Gráfica de la pérdida de valor (Exactitud para predecir los estados futuros)
e. Gráfica de la entropía del agente inteligente
f. Gráfica de ELO del agente inteligente (Habilidad del agente inteligente)

Así mismo, el modelo se evalúa de manera visual con cuatro comportamientos que el agente debería de realizar. A continuación, se describen los cuatro comportamientos y las observaciones que pueden tener cada uno:

a. El agente inteligente se dirige hacia su enemigo: (Siempre, a veces, nunca)
b. El agente inteligente ataca cuando debe ser y no al aire: (Siempre, a veces, nunca)
c. Cantidad de golpes diferentes que el agente inteligente realiza: (1-8)
d. El agente inteligente bloquea cuando debe de ser: (Siempre, a veces, nunca)

7 Resultados

Los resultados que se obtuvieron al entrenar un agente inteligente con el modelo Delta fueron satisfactorios. Para respaldar lo anterior, se presentan las tres gráficas más importantes para la evaluación del agente inteligente, las cuales son: La gráfica de la recompensa acumulada, la gráfica de pérdida de la política y la gráfica de ELO del agente inteligente, además de los puntajes que se obtuvieron en la evaluación visual.

7.1 Resultados de la evaluación visual del agente inteligente entrenado con el modelo Delta

El modelo Delta se desempeñó excelente en tres de los cuatro comportamientos de la evaluación visual, siendo el último el único en el que el agente inteligente falló.

a. El agente inteligente se dirige hacia su enemigo: **Siempre**
b. El agente inteligente ataca cuando debe ser y no al aire: **Siempre**
c. Cantidad de golpes diferentes que el agente inteligente realiza: **6**
d. El agente inteligente bloquea cuando debe de ser: **Nunca**

7.2 Gráficas generadas del entrenamiento con el modelo Delta

En base a las figuras 3, 4 y 5 se puede concluir que el modelo Delta está bien diseñado porque el comportamiento de las figuras muestra a un agente inteligente capaz de maximizar sus recompensas y aumentar su habilidad (ELO) sin variar demasiado las decisiones que toma en cada iteración de entrenamiento (política del agente).

La figura 3 muestra una tendencia que aumenta conforme pasan las iteraciones de entrenamiento, es decir, el agente inteligente logró satisfactoriamente incrementar su recompensa recibida utilizando las acciones proporcionadas por el modelo Delta.

Fig. 3. Recompensa acumulada del agente inteligente usando el modelo Delta

La figura 4 representa la diferencia entre una política anterior y una actual. Un valor bajo representa una variación pequeña en las decisiones tomadas por el agente inteligente, es decir, el agente inteligente es consistente y estable en sus decisiones.

Fig. 4. Pérdida de la política del agente inteligente usando el modelo Delta

La figura 5 representa la habilidad del agente inteligente frente a su rival, mientras más habilidad mejor desempeño existe en comparación con el rival (esta característica solo está presente con modelos que utilizan Self Play). En este caso hay complicaciones para derrotar al rival porque el ELO no aumenta mucho respecto a su valor inicial.

Fig. 5. ELO del agente inteligente usando el modelo Delta

8 Conclusiones

Con base en los resultados obtenidos, podemos concluir que el aprendizaje por refuerzo en conjunto con el algoritmo PPO crean agentes inteligentes capaces de aprender a jugar un videojuego de peleas de manera satisfactoria.

El comportamiento del agente inteligente dependerá del diseño del modelo, es decir, que las observaciones y las recompensas otorgadas al agente inteligente juegan un papel crucial en el comportamiento que tendrá el mismo. Además, es muy importante que las recompensas estén correctamente balanceadas y que, en la medida de lo posible, las recompensas se mantengan simples y alcanzables para el agente inteligente, de otro modo, posiblemente el agente inteligente nunca descubrirá la existencia de esas recompensas.

Es posible crear agentes inteligentes que representen dificultades mayores o menores, como ya se dijo, el comportamiento de un agente inteligente depende del modelo. Con esto en mente es posible crear agentes inteligentes que sean más agresivos o defensivos o incluso una combinación de ambos.

9 Referencias

Biscontini, T. (2023). Deep reinforcement learning (deep RL). In *Salem Press Encyclopedia of Science*. Salem Press.

Community, Rlb. (2024). *RLBot*. https://rlbot.org/

Gu, Y., Cheng, Y., Chen, C. L. P., & Wang, X. (2022). Proximal Policy Optimization With Policy Feedback. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *52*(7), 4600–4610. https://doi.org/10.1109/TSMC.2021.3098451

Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., & Mattar, M. (2018). Unity: A general platform for intelligent agents. *ArXiv Preprint ArXiv:1809.02627*.

Landers, M., & Doryab, A. (2023). Deep Reinforcement Learning Verification: A Survey. *ACM Comput. Surv.*, *55*(14s). https://doi.org/10.1145/3596444

OpenAI. (2018, April 27). *Gym Beta*. Https://Openai.Com/Research/Openai-Gym-Beta.

Sarrià Pascual, D. (2022). *Development of a competitive Rocket League bot using reinforcement learning*. Universitat Politècnica de Catalunya.

SaturnCloud. (2023). *Self-Play in Reinforcement Learning*. Https://Saturncloud.Io/Glossary/Self-play-in-Reinforcement-Learning/.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tantawi PhD, R. (2023). Machine learning. In *Salem Press Encyclopedia*. Salem Press.

10 Notas Biográficas

Alan Alberto Montes Pérez, ingeniero en Computación Inteligente por la UAA, se especializa en desarrollo de videojuegos y programación avanzada. Apasionado por la IA, ha trabajado en agentes inteligentes para mejorar dinámicas de combate en juegos mediante modelos de aprendizaje por refuerzo.

Aurora Torres Soto es profesor e investigador en la Benemérita Universidad Autónoma de Aguascalientes, México., donde realizó sus estudios de Doctorado. Sus áreas de interés incluyen el aprendizaje automático, las metaheurísticas, y en general el uso de la Inteligencia Artificial aplicada.

María Dolores Torres Soto es doctora en Informática, Sistemas y de la Información con énfasis en Inteligencia Artificial de la Universidad Autónoma de Aguascalientes. Sus áreas de interés son bases de datos, aprendizaje automático, metaheurísticas y aplicación de sistemas inteligentes a la sociedad.