VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

Estrategias de inteligencia artificial para jugar videojuegos

Artificial Intelligence Approaches to Playing Video Games

Daniel Yahir Andrade Herrera[1], Jorge Antonio Barroso Solórzano[1], Alfredo Medina García[1], Dora Luz Almanza Ojeda[1].
[1]Departamento de Ingeniería Electrónica, División de Ingenierías del Campus Irapuato-Salamanca, Sede Salamanca, Universidad de Guanajuato.
dy.andradeherrera@ugto.mx, ja.barrososolorzano@ugto.mx, a.medina.garcia@ugto.mx, dora.almanza@ugto.mx

Resumen

Los videojuegos son un área de entretenimiento que ha permanecido en el interés del público de todas las edades, generando ganancias multimillonarias a nivel mundial. A este éxito se suma el interés en el área de las ciencias de la computación por desarrollar un sistema artificial que también pueda jugar y superar las metas de los videojuegos, con el fin de generar mejores agentes inteligentes. En este contexto, este proyecto se centra en estudiar estrategias de inteligencia artificial que resuelvan videojuegos para plataformas de 8 bits, con diferentes objetivos y enfoques de control de agentes. Se inicia con este tipo de videojuegos porque la configuración y el conocimiento del entorno de trabajo que simulan requieren un bajo costo computacional y permiten concentrar la mayoría de los recursos de cómputo en la propuesta de modelos avanzados de inteligencia artificial. La metodología empleada consiste en entrenar el algoritmo *Proximal Policy Optimization* (PPO) para analizar el aprendizaje del agente a lo largo de distintos niveles del videojuego Galaxian, un clásico de la consola Atari 2600. Las distintas versiones entrenadas se comparan para seleccionar la mejor a través de métricas de recompensa media, de tasa de supervivencia y de un análisis de comportamiento en casos críticos (evasión de ataques). Al comparar con modelos o soluciones iniciales, encontramos que la integración de una lectura directa de la memoria RAM del emulador permite alcanzar un desempeño superior y mitigar comportamientos subóptimos, lo que valida nuestra metodología como una aproximación robusta para el diseño de agentes inteligentes en videojuegos de estrategia.

Palabras clave: agente inteligente; videojuegos; PPO; Galaxian.

Introducción

El desarrollo de agentes autónomos capaces de interactuar y resolver entornos virtuales complejos es uno de los pilares fundamentales de la investigación contemporánea en inteligencia artificial. Desde las publicaciones pioneras de DeepMind (Mnih *et al*., 2013), los videojuegos clásicos de la consola Atari 2600 se han consolidado como el estándar inicial para evaluar algoritmos de Aprendizaje por Refuerzo Profundo (Deep RL). Estos entornos ofrecen un equilibrio ideal: presentan una alta dimensionalidad visual y reglas lógicas estrictas, mientras mantienen un costo computacional manejable. En particular, los videojuegos de estrategia y de disparos espaciales, como Galaxian, plantean retos no triviales para los modelos de aprendizaje, tales como la necesidad de una evasión precisa, la escasez de recompensas inmediatas y el riesgo de desarrollar comportamientos conservadores (pacifismo táctico) o de explotar vulnerabilidades del motor gráfico. En este artículo se propone la implementación y mejora progresiva de un agente basado en el algoritmo Proximal Policy Optimization (PPO) (Schulman *et al*., 2017), diseñado para dominar el entorno del videojuego de Atari Galaxian en su versión de 8 bits. Se documenta la evolución desde un modelo puramente visual hasta una arquitectura híbrida que integra la lectura directa de la memoria RAM del sistema para estabilizar el aprendizaje. Lo anterior permite un análisis detallado de diferentes métricas para validar el rendimiento y los desafíos de la convergencia de la red neuronal, así como de la entrada de datos útiles.

VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

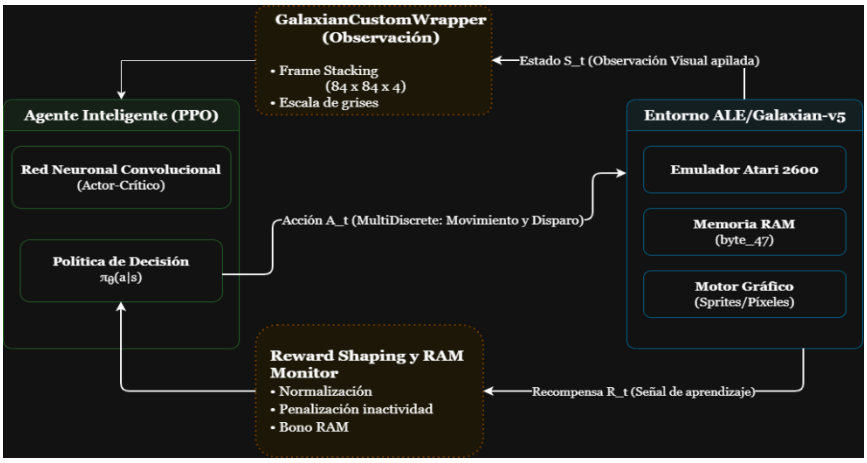


Figura 1. Arquitectura basada en el algoritmo PPO. Se ilustra el ciclo de interacción entre el agente inteligente y el entorno ALE/Galaxian-v5.

Metodología

La propuesta para programar un agente inteligente que resuelva el videojuego Galaxian-v5 se muestra en la Figura 1. En el diagrama de bloques se muestra cómo se define el agente PPO, basado en una red neuronal convolucional y en una política de decisión, que permitirá evaluar las mejores opciones del agente durante la toma de decisiones. Enseguida se define cómo son las observaciones del entorno que permiten determinar la acción a seguir por el agente: en este caso, movimiento o disparo. El sistema de recompensa permite brindar retroalimentación sobre las acciones que se van considerando. La última parte presenta el entorno ALE en el que es posible evaluar el agente propuesto.

Plataforma de simulación de videojuegos

El desarrollo y la evaluación del agente inteligente se llevaron a cabo en el entorno de simulación Arcade Learning Environment (ALE) (Bellemare *et al*., 2013), integrado mediante la interfaz de Gymnasium (Towers *et al*., 2023). Se seleccionó el entorno de 8 bits ALE/Galaxian-v5. Esta plataforma emula la consola Atari 2600, abstrae el comportamiento del sistema en un proceso de decisión de Markov (MDP) en el que el agente interactúa mediante secuencias continuas de observaciones, acciones y recompensas. Formalmente, el entorno se define mediante una tupla $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, donde:

- $\mathcal{S}$ es el espacio de estados. Debido a que el agente no percibe el estado interno completo del emulador a partir de un único fotograma, se define una historia de interacciones $s_t = x_1, a_1, x_2, \ldots, a_{t-1}, x_t$, lo que transforma el entorno en un MDP finito sobre secuencias temporales.
- $\mathcal{A}$ representa el espacio de acciones legales del videojuego.
- $\mathcal{P}(s' \mid s, a)$ es la función de transición de probabilidad transicional del emulador.
- $\mathcal{R}(s, a)$ denota la función de recompensa escalar.
- $\gamma \in [0,1)$ es el factor de descuento para retornos futuros.

Con el objetivo de incrementar la robustez y la capacidad de generalización del modelo ante escenarios de alta complejidad dinámica, se aplicó una estrategia basada en *Curriculum Learning*. Para ello, el entorno virtual se configuró explícitamente en su variante de dificultad avanzada, estableciendo el parámetro mode=3. Esta modificación altera la disposición inicial y los patrones de ataque de las oleadas de naves enemigas respecto del modo por defecto, lo que obliga al modelo a converger hacia políticas estables de control y supervivencia. La Figura 2 muestra una imagen del entorno del videojuego Galaxy, donde podemos ver al agente en la parte inferior, con forma de nave espacial, y sus objetivos de alcance, situados en la parte superior, con símbolos que representan naves espaciales de diferentes colores.



Figura 2. Entorno de simulación Arcade Learning Environment (ALE) ejecutando el videojuego Galaxian (Atari 2600).

Modelo de Inteligencia Artificial (PPO)

El núcleo del aprendizaje del agente se fundamenta en el algoritmo de Optimización de Política Próxima (Proximal Policy Optimization, PPO) implementado mediante la biblioteca Stable-Baselines3 (Raffin *et al*., 2021), un método de gradiente de política *On-Policy*. A diferencia de las aproximaciones basadas en funciones de valor tradicionales, como DQN (Mnih *et al*., 2013), PPO optimiza directamente una política parametrizada $\pi_\theta(a \mid s)$. Con el fin de evitar actualizaciones destructivas en la política que degraden irreversiblemente el desempeño del modelo bajo funciones de pérdida estándar, PPO maximiza un objetivo sustituto recortado (*clipped surrogate objective*) definido por la ecuación 1:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right] \qquad (1)$$

donde el radio de probabilidad de la acción elegida bajo los parámetros actuales y anteriores se denota por:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \qquad (2)$$

La función $\hat{A}_t$ representa la estimación de la ventaja en el paso de tiempo t, encargada de cuantificar si la acción tomada produjo un desempeño superior al promedio esperado en ese estado. El operador clip confina el radio de probabilidad dentro del intervalo $[1-\epsilon, 1+\epsilon]$ mediante un hiperparámetro de tolerancia $\epsilon = 0.25$. Este mecanismo suprime los incentivos para modificar la política cuando el gradiente sugiere un salto excesivamente grande, estabilizando matemáticamente la convergencia de los pesos.

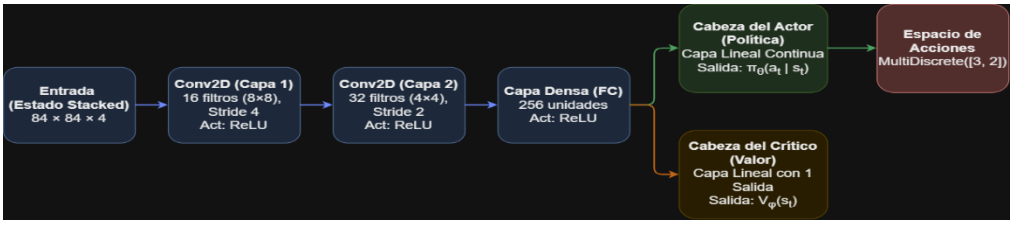


Figura 3. Arquitectura de la red neuronal convolucional (CnnPolicy) implementada para el algoritmo Actor-Crítico.

La arquitectura del aproximador de funciones, mostrada en la Figura 3, emplea una política basada en Redes Neuronales Convolucionales (*CnnPolicy*). Esta topología extrae, de forma jerárquica, características espaciales del mapa de píxeles mediante filtros locales recurrentes, mapeando los estímulos visuales directamente a una distribución de probabilidad sobre las acciones (Actor) y a una estimación del valor del estado (Crítico).

VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

Diseño del Espacio de Estados, Acciones y Recompensas (Wrapper Personalizado)

Las señales nativas del emulador de Atari presentan retos de alta dimensionalidad visual y una escasez de retroalimentación. Con el fin de mitigar estos problemas y viabilizar el entrenamiento, se diseñó e implementó un componente de envoltura personalizado (GalaxianCustomWrapper) estructurado según los siguientes criterios de ingeniería de características. Primero, las capturas de pantalla originales de la consola (210×160 píxeles en formato RGB con paleta de 128 colores) implican un costo computacional elevado. La figura 3 ilustra el proceso de acomodo del tensor y el procesamiento de cada imagen en el entorno de Atari. Cada imagen en bruto x_t se somete a una función de transformación $\phi(x_t)$ basada en el paradigma establecido por DeepMind (Mnih *et al*., 2013):

$$\phi: \mathbb{R}^{210\times160\times3} \rightarrow \mathbb{R}^{84\times84} \quad (3)$$

Consistente en la conversión a escala de grises mediante luminancia ponderada y un posterior redimensionamiento a una matriz simétrica de 84×84 píxeles mediante interpolación de área. Para dotar al modelo de noción cinemática, se aplica un esquema de apilamiento temporal (*Frame Stacking*) a los últimos $k = 4$ fotogramas, ordenados cronológicamente. La observación final entregada a la red neuronal en el instante t se define formalmente como:

$$S_t = [\phi(x_{t-3}), \phi(x_{t-2}), \phi(x_{t-1}), \phi(x_t)] \in \mathbb{R}^{84\times84\times4} \quad (4)$$

Este tensor multidimensional almacena la información necesaria para inferir variables físicas ocultas en imágenes estáticas, como el vector de velocidad y la trayectoria de los proyectiles.

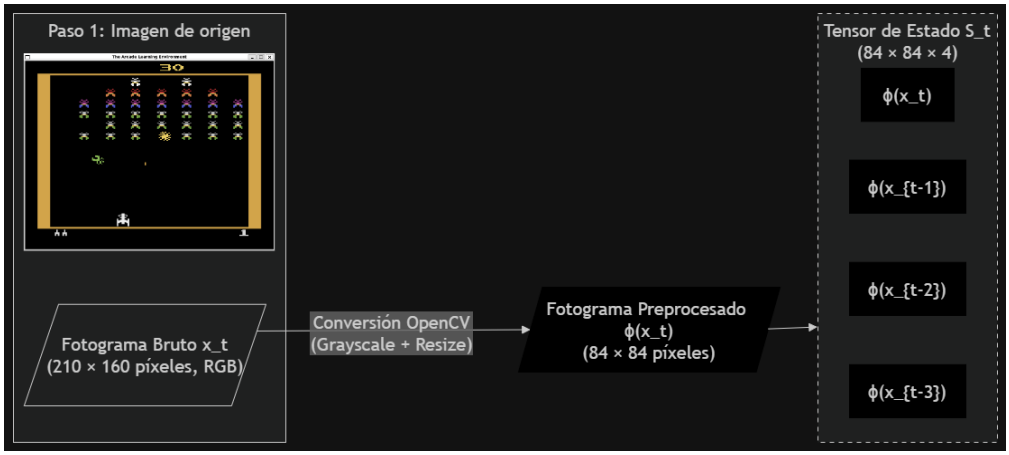


Figura 4. Pipeline de preprocesamiento visual y apilamiento temporal (Frame Stacking). Cada fotograma bruto de 210×160 píxeles es convertido a escala de grises y redimensionado a 84×84.

Segundo, el espacio de acciones nativo de ALE restringe la exploración al indexar comandos combinados unidimensionales. Con el propósito de agilizar la convergencia, las salidas del modelo se reestructuraron en un espacio compuesto por variables discretas, indexado mediante la especificación *MultiDiscrete([3, 2])*. Esta parametrización desacopla formalmente la dinámica del agente en dos dimensiones independientes: el eje de movimiento horizontal ($a_t^m \in \{0$:Estático, 1:Izquierda, 2:Derecha$\}$) y el eje del sistema de armas ($a_t^f \in \{0$:No disparar, 1:Disparar$\}$). El wrapper intercepta en cada paso el vector de acción conjunta $A_t = (a_t^m, a_t^f)$ y realiza la traducción lógica a las señales binarias nativas requeridas por la biblioteca de emulación de Atari. Tercero, la ingeniería de recompensas o Reward Shaping. El sistema de puntuación bruta de Galaxian presenta problemas de escalamiento y de retraso temporal que inducen inestabilidad en los gradientes del estimador crítico. Se formuló una función de recompensa intrínseca densa R_t expresada matemáticamente como una combinación lineal de incentivos y penalizaciones:

$$R_t = R_{score} + R_{align} - P_{inact} \quad (5)$$

VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

Cada componente fue diseñado bajo los siguientes fundamentos. La normalización de la puntuación (R_{score}) base entregada por el entorno, es escalada de forma lineal mediante la operación $R_{score} = \frac{r_{base}}{50.0}$, limitando la magnitud de las derivadas del error durante la retropropagación. Además se considera el Incentivo de Alineación Espacial (R_{align}) en el cual, si el agente ejecuta una acción de disparo ($a_t^f = 1$), se calcula la distancia absoluta en el eje horizontal entre la coordenada central del agente (x_{player}) y la coordenada media ponderada del centro de masa visual de la flota enemiga ($\bar{x}_{enemies}$). El incentivo se define de forma matemática como:

$$R_{align} = \frac{\alpha}{1.0 + |x_{player} - \bar{x}_{enemies}|} \qquad (6)$$

donde α es un factor de ponderación física que estimula al agente a posicionarse debajo del vector de ataque de los enemigos antes de abrir fuego.

Otras consideraciones importantes son las penalizaciones por inactividad (P_{inact}). Para mitigar conductas ultraconservadoras en los extremos de la pantalla, se incorporaron dos temporizadores de estado pasivo. Si el agente permanece en estado estático durante más de $\tau_{mov} = 30$ fotogramas, o entra en un periodo de paz sin destrucción de enemigos superior a $\tau_{kill} = 120$ fotogramas, se añade una penalización constante negativa en P_{inact} para penalizar el estancamiento conductual. En lo anterior se incluye la mitigación de vulnerabilidades visuales mediante RAM. En fases experimentales previas, el uso de aproximaciones de visión artificial para identificar transiciones de nivel indujo un fallo de explotación conductual (*exploit*). El modelo descubrió que, al suicidarse voluntariamente, el desvanecimiento transitorio de los sprites en pantalla emulaba las condiciones de un nivel limpio y permitía obtener fraudulentamente la recompensa de fin de nivel. Para erradicar de raíz esta inconsistencia conductual, el wrapper personalizado intercepta directamente el vector de 128 bytes que compone la memoria RAM del emulador de Atari. A través del monitoreo matemático estricto del **byte_47** (registro de hardware que actúa como el contador de nivel físico interno de la consola), se calcula la función de transición de nivel exacta:

$$\Delta Level = RAM[47]_t - RAM[47]_{t-1} \qquad (7)$$

Si $\Delta Level = 1$, se inyecta de forma determinista un súper bono de $+50.0$ puntos al agente libre del proceso de recorte de gradientes generales. La lectura determinista de la memoria RAM elimina el ruido visual provocado por los fotogramas y asegura que el agente desarrolle una política orientada estrictamente al avance legítimo de la partida. Por último, se muestra la configuración de los hiperparámetros del experimento, que resume las decisiones del diseño del agente inteligente.

***Tabla 1**. Configuración de hiperparámetros del experimento.*

Hiperparámetro	configuración	Comportamiento	Justificación
Algoritmo	PPO (CnnPolicy)	Actualización directa sobre los parámetros de la política y aproximación de la función de valor mediante extracciones visuales jerárquicas.	Optimización de política On-Policy basada en convoluciones para aprendizaje visual profundo.
Paralelismo (NUM_ENVS)	6 entornos (SubprocVecEnv)	Ejecución simultánea de 6 instancias del emulador en procesos aislados de CPU.	Recolección síncrona/asíncrona de trayectorias distribuidas en múltiples hilos de CPU.
Tasa de aprendizaje	Decreciente Lineal (5e-5 a 0)	Reducción progresiva del tamaño del paso de gradiente a medida que	Garantiza un ajuste fino y sutil sobre los pesos de la red, evitando el olvido catastrófico.

		avanza el contador global de pasos.	
Horizonte Temporal (n_steps)	2048 pasos	Cantidad de interacciones consecutivas recopiladas por entorno antes de calcular la pérdida.	Longitud de la trayectoria recolectada por entorno antes de ejecutar la actualización.
Tamaño de Lote (batch_size)	256 muestras	Tamaño de las particiones de memoria empleadas para actualizar los pesos de la red en cada iteración del optimizador.	Subconjunto extraído del buffer para el cálculo del gradiente en cada época.
Épocas de Optimización	4 épocas	Número de pasadas completas del algoritmo de optimización sobre el buffer de experiencia acumulado.	Iteraciones de gradiente descendente programadas sobre el buffer de datos actual.
Coeficiente de Entropía	0.01	Penalización añadida a la pérdida que promueve la aleatoriedad de las acciones si es alta.	Controla la exploración. Un valor bajo fuerza al agente a explotar la política exitosa encontrada.
Factor de Descuento (γ)	0.996	Factor exponencial de reducción aplicado sobre las recompensas futuras lejanas.	Factor de descuento a largo plazo. Prioriza la supervivencia extendida en el tiempo
Rango de Recorte (clip)	0.25	Umbral límite permitido para las desviaciones entre la política nueva y la política vieja.	Factor de contención que limita los cambios bruscos permitidos en la actualización de la política.

Para asegurar el correcto flujo de las experiencias recopiladas en los 6 entornos paralelos bajo la arquitectura PPO, el procesamiento lógico sigue la secuencia formalizada en el diseño del algoritmo base de DeepMind. La Figura 5 muestra, en pseudocódigo, el flujo de trabajo realizado paso a paso.

Resultados

El desempeño del agente se evaluó mediante métricas de entrenamiento extraídas de TensorBoard y se ejecutaron 10 episodios de prueba con el mejor modelo guardado en el paso de entrenamiento de 8 millones.

Progresión del Entrenamiento

El entrenamiento se ejecutó en etapas evolutivas a partir de un modelo base, acumulando más de 16 millones de pasos (*timesteps*). Como se aprecia en la Figura 6, el agente base (V10, línea verde) mostró una curva de aprendizaje estable, pero se estancó en su desempeño al alcanzar los 8 millones de pasos, con un promedio de 80 puntos. Al intentar forzar un comportamiento distinto al alterar severamente las penalizaciones (V11, línea azul), el rendimiento decayó drásticamente, lo que sugiere que el agente se confundió con las señales de castigo. No obstante, al implementar la validación de nivel mediante la lectura de RAM (V12, línea roja), el modelo exhibió un salto inmediato y contundente en la recompensa media, superando la barrera de los 110 puntos y manteniendo un desempeño consistentemente superior al de sus predecesores antes de que su entrenamiento fuera detenido a los 12 millones de pasos.

VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

Algoritmo 1: Entrenamiento Actor-Crítico (PPO) con Wrapper Personalizado

Inicializar parámetros de la red del Actor-Crítico θ con pesos aleatorios.
Configurar entorno Gymnasium `ALE/Galaxian-v5` en dificultad avanzada (`mode=3`).
Establecer buffer de trayectorias con horizonte temporal $T = 2048$ pasos.
for episodio $= 1$ hasta M **do**
    Recibir fotograma bruto inicial x_1 desde el emulador.
    Ejecutar preprocesamiento visual: grises y redimensión $\phi(x_1) \rightarrow 84 \times 84$.
    Inicializar estado apilado $S_1 = [\phi(x_1), \phi(x_1), \phi(x_1), \phi(x_1)]$.
    Leer registro inicial de nivel físico de la consola: $L_{\text{prev}} = \text{RAM}[47]$.
    for $t = 1$ hasta T **do**
        Evaluar política $\pi_\theta(A_t|S_t)$ y seleccionar acción conjunta $A_t = (m_t, f_t)$.
        Ejecutar acción A_t en `GalaxianCustomWrapper`:
            – Mapear A_t a comandos primitivos binarios de Atari.
            – Ejecutar el paso en el emulador; obtener x_{t+1} y $\text{score}_{\text{base}}$.
        Calcular Ingeniería de Recompensas Intrínsecas:
            – Normalizar recompensa por destrucción: $R_{\text{score}} = \text{score}_{\text{base}}/50.0$
            – Calcular bono geométrico R_{align} basado en centro de masa enemigo.
            – Evaluar penalizaciones P_{inact} si hay inactividad táctica o pacifismo.
        Mitigar vulnerabilidad visual interceptando la telemetría del sistema:
            $L_{\text{curr}} = \text{RAM}[47]$
        if $L_{\text{curr}} > L_{\text{prev}}$ **then**
            $R_{\text{RAM}} = +50.0$ ▷ Super bono determinista por cambio de nivel
            $L_{\text{prev}} = L_{\text{curr}}$
        else
            $R_{\text{RAM}} = 0.0$
        end if
        Calcular recompensa neta: $R_t = R_{\text{score}} + R_{\text{align}} + R_{\text{RAM}} - P_{\text{inact}}$.
        Procesar nueva observación $S_{t+1} = [\phi(x_{t-2}), \phi(x_{t-1}), \phi(x_t), \phi(x_{t+1})]$.
        Almacenar transición (S_t, A_t, R_t, S_{t+1}) en el buffer de trayectorias.
        $S_t = S_{t+1}$
    end for
    Estimar ventajas A_t empleando el Crítico sobre las trayectorias acumuladas.
    for época $= 1$ hasta 4 **do**
        Muestrear minibatches de tamaño 256 del buffer.
        Actualizar pesos θ maximizando la pérdida recortada $L^{\text{CLIP}}(\theta)$ con clip $= 0.25$.
        Decrementar la tasa de aprendizaje de forma lineal hacia 0.
    end for
end for

Figura 5. Pseudocódigo del experimento.

VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

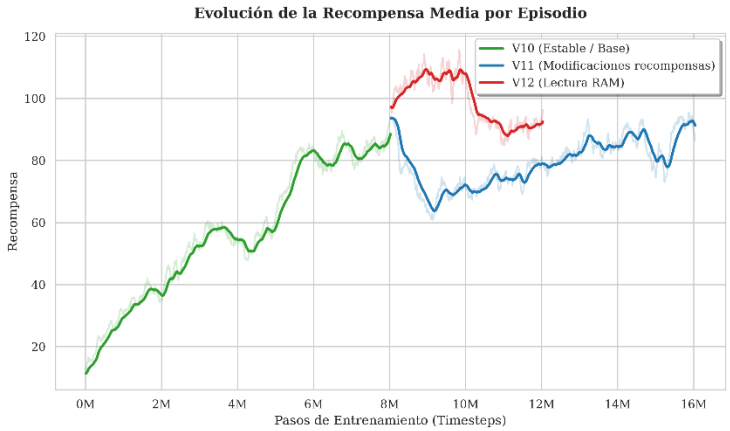


Figura 6. Comparativa de la recompensa media por episodio (supervivencia en fotogramas) a lo largo de 16 millones de pasos de entrenamiento.

Evaluación del Modelo Óptimo

Se aisló el punto de control óptimo, previo al colapso de la política, para una evaluación determinista. Los resultados promedio frente a versiones anteriores (o sin lectura de RAM) se presentan en la Tabla 1.

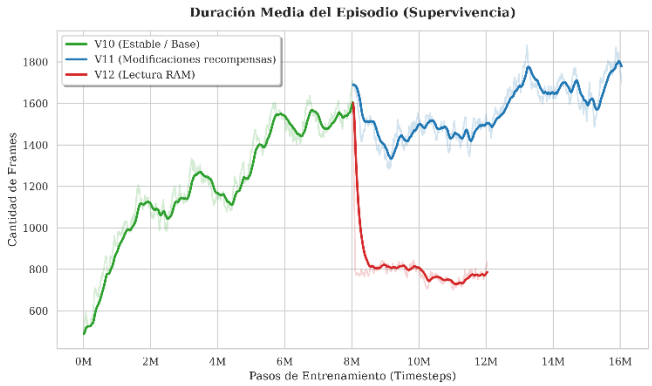


Figura 7. Comparativa de la duración media del episodio (supervivencia en fotogramas) a lo largo de 16 millones de pasos de entrenamiento.

El análisis de supervivencia observado de la Figura 7, revela la principal diferencia de comportamiento entre las estrategias. La versión con penalizaciones alteradas (V11) aprendió a prolongar artificialmente la partida, sobreviviendo durante casi 1800 fotogramas, pero obteniendo bajas recompensas; desarrolló un comportamiento "cobarde" o de pura evasión. En contraste, la versión óptima con lectura de RAM (V12) redujo drásticamente la duración del episodio a un promedio de 800 fotogramas. Esto demuestra de forma cuantitativa que el agente de la V12 es altamente eficiente y agresivo: no retrasa el juego escondiéndose en los bordes de la pantalla, sino que elimina rápidamente la amenaza enemiga para forzar el cambio de nivel y obtener su bonificación

VOLUMEN 48
XXXI Verano De la Ciencia
ISSN 2395-9797
www.jovenesenlaciencia.ugto.mx

Durante estas pruebas, se observó cualitativamente que el agente desarrolló comportamientos emergentes de evasión, esquivando ráfagas enemigas sin contar con reglas explícitas para ello, producto del apilamiento espacial-temporal (Frame Stacking) y la penalización por pérdida de vidas.

Discusión de resultados

La implementación del GalaxianCustomWrapper con acceso directo al byte 47 de la memoria RAM del emulador resolvió el problema de abuso de bugs o exploits observado en iteraciones tempranas. Al otorgar una recompensa escalar masiva de +50.0 exclusivamente al transicionar al siguiente nivel, el algoritmo PPO logró alinear la optimización de la política con el objetivo real del juego: sobrevivir y limpiar la oleada.

Sin embargo, el análisis profundo de las métricas internas de la red neuronal justifica por qué el entrenamiento de la versión óptima (V12) se detuvo alrededor de los 12 millones de pasos, lo que evidencia una convergencia prematura. Como se muestra en la Figura 8 (derecha), a partir de los 10 millones de pasos se observó una anomalía en la actualización de la red, evidenciada por un pico masivo en la Divergencia KL. Esto significa que el optimizador realizó un ajuste de pesos demasiado agresivo, lo que desestabilizó brevemente el modelo. Como consecuencia directa de este ajuste (Figura 8, izquierda), la métrica entropy_loss convergió abruptamente a cero. En el aprendizaje por refuerzo, cuando la entropía se agota, el agente pierde toda su capacidad de exploración y se convierte en un sistema rígidamente determinista. Matemáticamente, la red neuronal dejó de experimentar nuevas acciones y se estancó en la política descubierta hasta ese momento (*estancamiento exploratorio determinista*). Este fenómeno sugiere que los hiperparámetros, en particular la tasa de decaimiento del aprendizaje (*learning rate schedule*) o el coeficiente de entropía, no fueron suficientes para mantener la curiosidad del agente en escenarios de alta densidad de proyectiles.

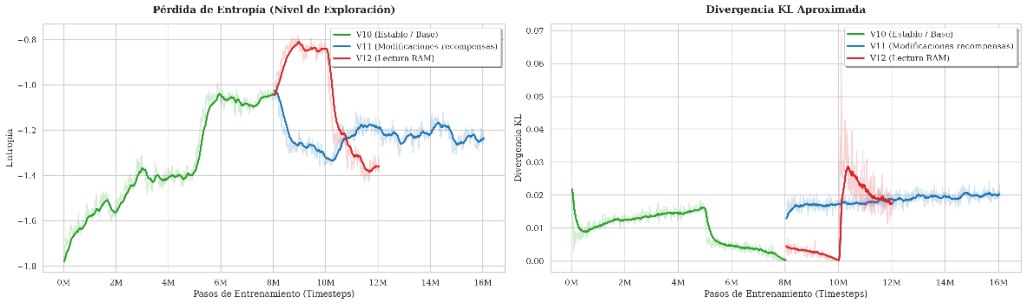


Figura 8. Anomalías detectadas durante la convergencia del modelo óptimo (V12) a los 10 millones de pasos.

Conclusiones

En este trabajo se demostró la viabilidad de entrenar un agente de Aprendizaje por Refuerzo Profundo (PPO) capaz de superar múltiples niveles en el entorno altamente dinámico de *Galaxian*. El uso de técnicas de ingeniería de recompensas híbridas —combinando procesamiento visual (CNN y Frame Stacking) con la lectura directa de la memoria RAM para detectar cambios globales de estado— demostró ser una solución robusta para evitar que el agente explote fallas lógicas del entorno.
El modelo resultante es capaz de operar con un desempeño cercano a los 4000 puntos, exhibiendo comportamientos complejos de evasión y precisión de disparo, cumpliendo así con los objetivos planteados para este proyecto de investigación. Para futuras iteraciones, se proponen las siguientes líneas de mejora:

- Ajuste de Hiperparámetros (Hyperparameter Tuning): Implementar herramientas como *Optuna* para encontrar el decaimiento óptimo del learning_rate y prevenir el colapso de la política observado en la etapa final del entrenamiento.
- Curriculum Learning Avanzado: Iniciar el entrenamiento del agente directamente en niveles superiores (manipulando la RAM) para exponer a la red neuronal a escenarios de alta dificultad desde fases tempranas.

- Exploración de otros algoritmos: Comparar el rendimiento de PPO frente a algoritmos de tipo *Off-Policy* continuos adaptados o variaciones como SAC (Soft Actor-Critic), para evaluar mejoras en la eficiencia de las muestras y la estabilidad a largo plazo.

Referencias

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The Arcade Learning Environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47, 253-279. https://doi.org/10.1613/jair.3912

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-Baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268), 1-8.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, S., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., ... & Younis, O. G. (2023). Gymnasium. *Zenodo*. https://doi.org/10.5281/zenodo.8127025