DESDE 2013
https://repository.uaeh.edu.mx/revistas/index.php/icbi/issue/archive
Pädi Boletín Científico de Ciencias Básicas e Ingenierías del ICBI

ISSN: 2007-6363

Publicación Semestral Pädi Vol. 10 No. Especial 4 (2022) 7-17

# Neuroevolución de redes neuronales híbridas en un agente robótico (NRNH-AR)
# Neuroevolution of hybrid neural networks in robotic agent (NRNH-AR)

*C. Vásquez-Jalpa, M. Nakano-Miyatake, H. Perez-Meana*[a]

[a] *Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Culhuacán, Instituto Politécnico Nacional, 04440, Ciudad de México, México.*

**Resumen**

Se ha desarrollado un Agente Robótico capaz de aprender del entorno dinámico por el que navega, el cual tiene como objetivo hallar un objeto específico. Para el crecimiento de su aprendizaje, se ha creado la Neuroevolución de Redes Neuronales Híbridas en un Agente Robótico (NRNH-AR) que combina redes como la CNN para entender el entorno y ANN para realizar acciones, esto se complementa con el Aprendizaje por Refuerzo Profundo y la Política de Gradiente. Sin embargo, para que el algoritmo tenga éxito prácticamente en un robot físico, se han considerado además dos bloques: el Hardware y la mecánica involucrada, pues se entrenará de forma on-line para evitar problemas de latencia y limitación de ancho de banda. Con esta investigación, se ha demostrado que con la NRNH-AR es posible implementar el Aprendizaje por Refuerzo Profundo dentro de un Robot, optimizando en tiempo y costo computacional teniendo un aprendizaje evolutivo.

*Palabras Clave:* Edge Computing, Aprendizaje por Refuerzo Profundo, Neuroevolución, DDPG, Política de Gradiente

**Abstract**

A Robotic Agent capable of learning from the dynamic environment through which it navigates has been developed, which aims to find a specific object. For the growth of their learning, the Neuroevolution of Hybrid Neural Networks in a Robotic Agent (NRNH-AR) has been created that combines networks such as CNN to understand the environment and ANN to perform actions, this is complemented by Deep Deterministic Policy Gradient (DDPG) composed by Deep Reinforcement Learning and Policy Gradient. However, for the algorithm to be successful practically in a physical robot, two blocks have also been considered: The Hardware and the mechanics involved, as it will be trained online to avoid latency problems and bandwidth limitation. With this research, it has been shown that with NRNH-AR it is possible to implement Deep Reinforcement Learning within a Robot, performing edge computing, in which there is not latency problem, optimizing time and computational cost through an evolutionary learning.

*Keywords:* Edge Computing, Deep Reinforcement Learning, Neuroevolution, DDPG, Policy Gradient

## 1. Introducción

Toda la información que generan los modelos de inteligencia artificial se almacena y procesa en grandes centros de datos y aunque esta tecnología representa un gran avance, no está exenta de ciertos problemas, uno de los cuales es la latencia y limitación de ancho de banda (Satyanarayanan, 2017), es decir, el tiempo que tarda el servidor en recibir la señal generada por el dispositivo o el usuario. El "*edge computing*" soluciona este problema porque es un nuevo paradigma para proporcionar servicios en la nube informática capacidades en el borde de la red cerca de los usuarios móviles (Tao *et al.*, 2022).

Por lo tanto, optar por un sistema autónomo, donde todo el cómputo de aprendizaje automático se realiza en el propio dispositivo, es más efectivo que tener una conexión a la nube, por lo que sectores como la conducción autónoma confían en este modelo de sistema informático para realizar todos los cálculos del vehículo y detectar de forma fiable carreteras, peatones, semáforos, otros vehículos, entre otros.

En este caso, para abordar este problema, se propuso un enfoque de programación de carga de trabajo que desarrolla el Aprendizaje por Refuerzo Profundo (ARP) con el objetivo de equilibrar la carga de trabajo, reducir el tiempo de servicio y la tasa de tareas fallidas (Tao *et al.*, 2022) mientras tanto, adoptamos un proceso de aprendizaje donde a priori requiere

*Autor para la correspondencia: H. Perez-Meana
**Correo electrónico**: vasquezjalpacarlos@hotmail.com (Carlos Alberto Vázquez-Jalpa), mnakano@ipn.mx (Mariko Nakano-Miyatake), hmperezm@ipn.mx (Héctor Pérez-Meana).

**Historial del manuscrito:** recibido el 23/04/2022, última versión-revisada recibida el 10/08/2022, aceptado el 19/08/2022, publicado el 05/10/2022. **DOI:** https://doi.org/10.29057/icbi.v10iEspecial4.9070

que el algoritmo sea entrenado en entornos de simulación o bien pasar por aprendizajes supervisados.

La navegación es un problema fundamental de los robots móviles, por lo que el Aprendizaje por refuerzo profundo ha recibido una atención significativa debido a su fuerte capacidad de representación y aprendizaje de experiencia (Zhu y Zhang, 2021). Para mejorar el tiempo de entrenamiento de los agentes robóticos en el ARP, sin dañar su estructura física, se irá cambiando el tipo de entrenamiento conforme se van aumentando la base de datos hecha por el mismo agente, es decir, se iniciará con el Aprendizaje Supervisado (AS), ya que necesita ayuda el agente para empezar a comprender el entorno físico A, teniendo la cantidad mínima de datos para entender el entorno A, se pasará al Aprendizaje No Supervisado (ANS).

El proyecto va enfocado a optimizar el tiempo de entrenamiento de un agente robótico físico implementando ARP en su interior el algoritmo Twin Delay 3 (TD3), propuesto por Fujimoto et al. (2018), con la finalidad de hallar el objeto que se le indique encontrar. El método propuesto mejora el clásico Gradiente de Política Determinística Profunda (DDPG) (Chen *et al.*, 2019).

## 2. Sistema implementado

Para el buen desempeño de la Neuroevolución de Redes Neuronales Híbridas en un Agente Robótico (NRNH-AR) es necesario equilibrar tres bloques importantes: el Hardware, el Software y la Mecánica involucrada, estos se ubican en el interior del agente para su procesamiento de "Edge computing".

### *2.1. Características del Hardware*

En varias investigaciones no se considera un diseño real para la implementación de sus algoritmos que a priori han sido simulados, o bien se cuenta con un buen agente robótico, pero no toman en cuenta señales de sensores como entrada a las redes neuronales.

En la figura 1 se muestran los componentes usados como Hardware para llevar a cabo el NRNH-AR.

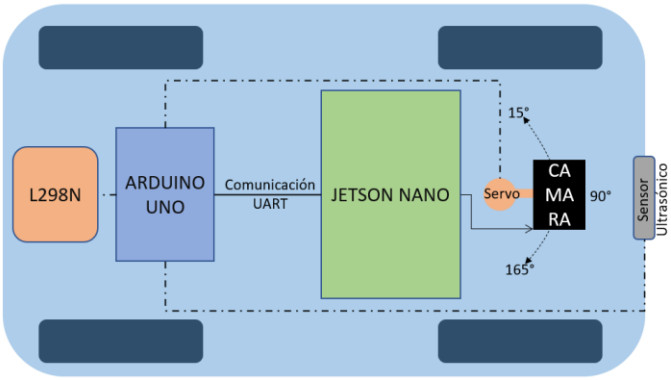


Fig. 1. Configuración y conexión de componentes electrónicos y del Hardware del agente robótico.

Como se observa en la figura 1, cuenta con un Jetson Nano y un Arduino los cuales están conectados de forma serial comunicándose por el transmisor-receptor asíncrono universal o mejor conocido por protocolo UART, ya que este presenta resultados experimentales del funcionamiento del sistema donde la transmisión de datos es estable durante un largo período de tiempo (Grigoryeva *et al.*, 2021). A su vez, es común utilizar una velocidad de 9600 baudios/segundo, sin embargo, Jetson Nano sólo permite 115200 baudios/segundo (Stateczny *et al.*, 2022), (Ma y Alborati,2019), (Wróbel *et al.*, 2020).

La salida de voltaje del GPIO del Jetson Nano son de 3.3v, el cual es insuficiente para mover los motores y servo motores, por lo que se usa Arduino cuya salida es de 5v, en este segundo se ubican el módulo L298N y un servomotor, cuyo funcionamiento son el desplazamiento del agente y colocar la cámara en 15°, 90°, 165° para capturar las características del entorno respectivamente. A su vez, tiene como entrada un sensor ultrasónico posicionado a 60°, este sensor funcionará para que el agente no colisione y, por ende, no dañe su funcionamiento. En la sección 2.3 *Características mecánicas* se explica más sobre la posición del sensor ultrasónico.

Por otro lado, en el Jetson se conecta una cámara CSI, cuyas características se muestran en la sección 4 *Características de los componentes*, para recopilar datos del entorno por el que navega y que, dentro del Software, ubicado en el Jetson, se transforman junto con la información del sensor ultrasónico para que sean entradas del sistema de redes neuronales.

### *2.2. Software implementado*

A menudo, en la programación de robots fuera de línea (off-line), el entorno robótico es idealizado (el virtual) no refleja con precisión el real. En esta situación, estamos en presencia de un entorno parcialmente desconocido (PUE) (Lindeberg, 2 012). Por lo tanto, los sistemas robóticos deben tener cierto grado de autonomía para superar esta situación, especialmente cuando existe contacto, además se ha descubierto que los robots móviles en línea (on-line) son más efectivos que los fuera de línea (off-line) para navegar en entornos desconocidos y dinámicos (Lillywhite *et al*., 2013), (Vasques Jalpa *et al.*, 2021), (Placed y Castellanos, 2020).

El algoritmo desarrollado es una optimización del Twin Delay 3 (TD3) o mejor conocido como el DDPG (Deep Deterministic Policy Gradients) para el Aprendizaje por Refuerzo Profundo para el desempeño de la Neuroevolución de Redes Neuronales Híbridas en un Agente Robótico (NRNH-AR) para que funcione se han considerado los elementos que a continuación se explican.

#### *2.2.1 Software implementado*

Como se busca optimizar el Aprendizaje por Refuerzo Profundo en un agente robótico físico. Se ha dividido el entrenamiento en tres partes: El aprendizaje Supervisado, el Aprendizaje No Supervisado y el Aprendizaje por Refuerzo Profundo.

En el Aprendizaje Supervisado, se busca que el agente adquiera experiencia de forma rápida al indicarle que acciones debe realizar tomando en cuenta la imagen de entrada del entorno y la señal del sensor ultrasónico.

En el Aprendizaje No Supervisado, se sustituirá las decisiones de quien supervisaba en el aprendizaje anterior por

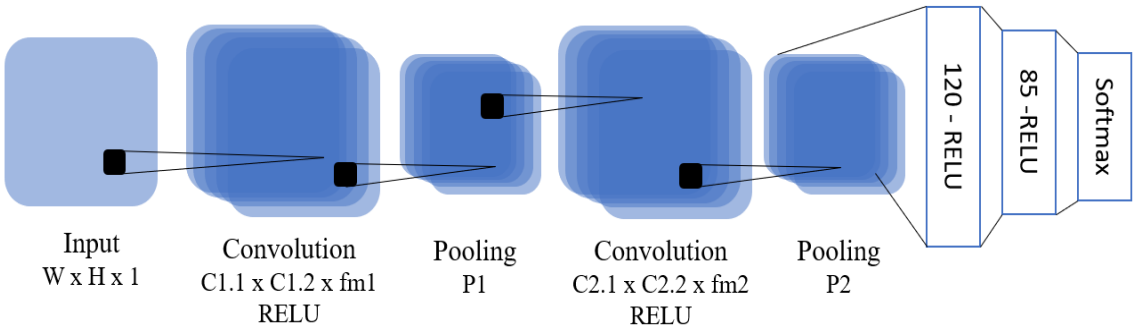


Fig. 3 Configuración CNN óptima para el proyecto, donde el número de salidas se modificará dependiendo la cantidad de casos nuevos que se encuentren. En un principio serán 5 clases las cuales son consecuencia de los casos que se tengan (010, 011, 110, 111 y el objeto a hallar).

un sistema de redes neuronales en paralelo, calculando 2 acciones cuya entrada es la misma imagen y comparándolas entre sí.

En cuanto al Aprendizaje por Refuerzo Profundo, el agente tomará como base lo aprendido anteriormente, aquí se agregan los castigos y premios, además del tiempo que tarda en lograr su objetivo, que en este caso es hallar un objeto específico. Cabe mencionar que para la clasificación de las imágenes en la base de datos se cuenta con una CNN la cual se presenta en el apartado 2.2.2.

Como se puede ver en la figura 2, el proceso de aprendizaje iniciará tomando en cuenta 3 variables: el estado *S*, el número de iteración *i* y el número de época *ep*, es decir, S=0, *i*=1 y *ep*=0.

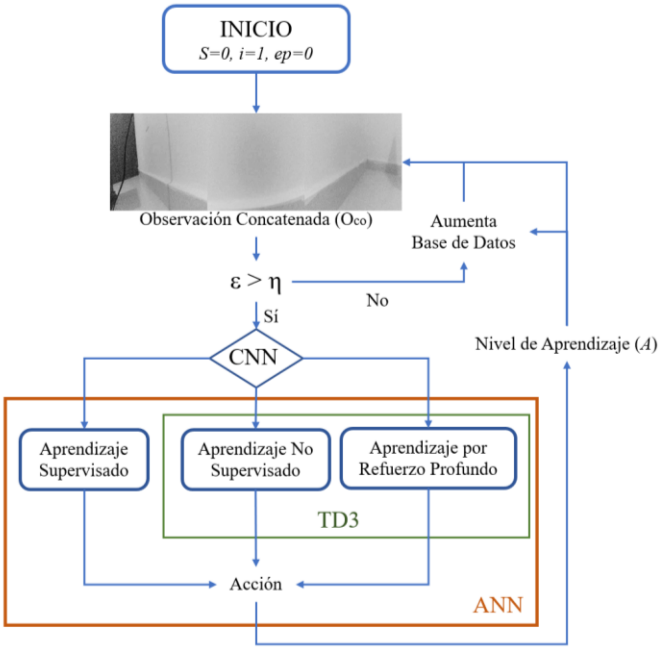


*Fig. 2 Diagrama del proceso de aprendizaje del agente robótico.*

Luego, el agente avanzará tomando fotografías del entorno generando la base de datos representando la no colisión. El agente se detendrá sólo si el sensor manda señal activa en el estado S, toma 3 fotografías en los grados 15°, 90° y 165°, los concatena y dicha imagen toma nombre de *observación concatenada* (*Oco*).

Se tiene que si $\varepsilon$ es mayor que η indica que tiene imágenes suficientes para entrenar la CNN, de lo contrario se van agregando a la base de datos. $\eta$ es la cantidad mínima de información del entorno para su entrenamiento. El valor de η depende de la cantidad de épocas e iteraciones que se deseen realizar, entre mayor sea mejor exactitud tendrá la CNN.

El valor η es el resultado del producto del número de imágenes a analizar con las iteraciones, por lo que si $\varepsilon$>η puede realizar el aprendizaje correspondiente, actualizando los pesos de las redes neuronales del sistema completo, minimizando la función de pérdida $J(\varphi)$ con base a su gradiente $\nabla_{\varphi} J(\varphi)$ (Placed y Castellanos, 2020).

La sección del aprendizaje se modificará conforme aumenta el número de épocas, pasando por el Aprendizaje Supervisado (AS), Aprendizaje No Supervisado (ANS), llegando al Aprendizaje por Refuerzo Profundo (ARP) en ese orden.

### *2.2.2 Red Neuronal Convolucional*

La implementación del algoritmo TD3 en el agente, consta de 3 etapas: etapa de exploración, etapa de entrenamiento y la etapa de inferencia.

En la etapa de exploración, el agente robótico no sólo recopilará datos, sino también será capaz de clasificar lo que observa mientras va entrenando las redes de aprendizaje, para ello se crearon redes neuronales convolucionales (CNN) para tareas de clasificación de imágenes (Faisal *et al*., 2013).

De acuerdo con el tipo de uso de la CNN se tienen los parámetros a considerar para generar la que mejor convenga (Raja y Pugazhenthi, 2012), (Marjovi y Marques, 2011), (Li *et al*., 2016), (Lowe, 2004).

En este proyecto, se propone obtener los parámetros de forma *off-line* al simular los datos en CoppeliaSim y con los resultados del tiempo y exactitud de la CNN obtener la configuración optima del agente, dicha configuración es la que se muestra en la figura 3.

En la figura 3, la CNN tiene 4 principales casos, donde 010 representa que tiene un obstáculo enfrente, teniendo opción de ir por la derecha o por la izquierda; 011 cuya única salida es del lado izquierdo; 110 aquí la única salida vista es por el lado derecho; 111 representa que no hay camino por ningún lugar observable pudiendo retroceder o dar la vuelta.

En la tabla 1, se muestra cómo es el desempeño de la configuración seleccionada comparada con otras configuraciones realizadas. Las variables W, H (ancho, alto de la imagen de entrada), C1.1, C1.2 (dimensión del kernel del

primer filtro de convolución), fm1 (número de mapa de características creadas), P1 (el tamaño del primer maxpooling), C2.1, C2.2 (dimensión del kernel del segundo filtro de convolución), fm2 (número de mapa de carácterísticas creadas por el filtro) y P2 (el tamaño del segundo maxpooling) , se pusieron a prueba para obtener la mejor configuración. W y H son las imágenes de entrada las cuales se ha mencionado que tienen una resolución de 60×60 las cuales se pondrán, sin embargo, se colocará también las de la versión original que es 150×150 para la comparación. Lo primero que se modificó fueron los MaxPooling.

La mejor configuración de MaxPooling es la marcada en negro, porque la relación tiempo-exactitud es la más elevada. Luego de ello se modificaron los filtros de convolución, cuyos valores se presentan en la tabla 2.

Tabla 1. Modificación de MaxPooling

| WxHx1 | P1 | P2 | Tiempo (M:SS) | Exactitud |
|---|---|---|---|---|
| 150x150x1 | 2/2 | 2/2 | 3:12 | 0.97 |
| 60x60x1 | 2/2 | 2/2 | 0:36 | 0.95 |
| **60x60x1** | **3/3** | **2/2** | **0:31** | **0.95** |
| 60x60x1 | 5/5 | 3/3 | 0:29 | 0.91 |
| 60x60x1 | 5/5 | 5/5 | 0:30 | 0.90 |
| 60x60x1 | 4/4 | 5/5 | 0:29 | 0.88 |

Tabla 2. Modificación de Filtros convolucionales.

| fm1 | C1.1xC1.2 | fm2 | C2.1xC2.2 | Tiempo | Exactitud |
|---|---|---|---|---|---|
| 32 | 6x6 | 64 | 5x5 | 0:38 | 0.94 |
| | 5x5 | | 7x7 | 0:39 | 0.77 |
| | 12x12 | | 4x4 | 0:44 | 0.86 |
| 18 | 6x6 | 32 | 5x5 | 0:31 | 0.91 |
| | 6x6 | | 4x4 | 0:31 | 0.91 |
| 18 | 7x7 | 27 | 9x9 | 0:38 | 0.92 |
| | 6x6 | | 5x5 | 0:31 | 0.91 |
| 10 | 6x6 | 20 | 5x5 | 0:30 | 0.92 |
| | 6x6 | | 7x7 | 0:30 | 0.87 |
| **20** | **6x6** | **20** | **4x4** | **0:31** | **0.95** |
| | 6x6 | | 5x5 | 0:32 | 0.92 |
| 15 | 6x6 | 15 | 5x5 | 0:31 | 0.93 |

Cada una de las configuraciones se entrena para obtener los pesos y el modelo de la red.

Los pesos y modelos se ponen a prueba con otras imágenes, tomando como base las cantidades de imágenes que aciertan se calcula el porcentaje de efectividad, y una vez más, se evalúa cual configuración es mejor con la relación tiempo-exactitud de esta forma se hallaron los valores de la configuración de la CNN, dichos valores son: C1.1×C1.2=6×6, fm1=20, P1=3/3, C2.1×C2.2=4×4, fm2=20 y P2=2/2.

### *2.2.3 Etapa de Aprendizaje Supervizado*

En este aprendizaje se aplica el Deep Q-Learning con mejora continua, es decir cuenta con una red actor y una red crítica, donde la primera realiza la acción y la segunda evalúa la calidad de las acciones realizadas, respectivamente. El actor, tiene como entrada una neurona que es la imagen concatenada, luego hay dos capas ocultas, la primera con 250 neuronas y la segunda con 200; como salida es una neurona cuyo valor máximo es de 4, esta con función sigmoidal. Por otro lado, la red crítica tiene como entrada 2 neuronas, una para la observación y otra para la acción predicha o real, con los mismos datos de capas ocultas, pero la salida es una neurona con función sigmoidal.

La acción de inicio de esta etapa está calculada por (1). La cual indica la acción predicha $a_p$ (la cual puede ser: adelante, vuelta a la izquierda, vuelta a la derecha, atrás o estar detenido) que es el resultado de la política de los pesos de conexión $\mu$ de la observación concatenada $O_{co}$.

$$a_p = \pi\mu(O_{co}) \tag{1}$$

El siguiente paso es calcular la calidad de la acción predicha en la iteración $i$, $Q_{pi}$, tomando como base la política de los pesos de conexión $\sigma$ con la Observación concatenada y con la acción predicha, cuya representación está dada por (2).

$$Q_{pi}(O_{co}, a_p) = \pi\sigma(O_{co}, a_p) \tag{2}$$

Luego, el agente robótico procede a realizar la acción real $a_r$ (adelante, vuelta a la izquierda, vuelta a la derecha, atrás, detenido), cuyo valor de calidad será calculado con (3) para compararla con la predicha. La política de los pesos de conexión de esta red se representa con $\sigma'$.

$$Q_{ri}(O_{co}, a_r) = \pi\sigma'(O_{co}, a_r) \tag{3}$$

En caso de que se cumpla la igualdad, es decir que $Q_{pi}(O_{co}, a_p) = Q_{ri}(O_{co}, a_r)$, los datos de $O_{co}$ será guardado en la base de datos (S, $O_{co}$), en caso contrario se deben modificar los pesos de conexión de la red que predice la acción en la dirección que se incrementan los valores Q, es decir, modifica los pesos de conexión cada vez que la calidad aumenta; dicho valor es representado por (4).

$$\nabla_\mu J(\mu) = \sum \nabla a Q_i(O_{co}, a_r)\,|_{a=\pi\mu(O_{co})} \nabla_\mu \pi_\mu(O_{co}) \tag{4}$$

Este proceso se realizará iteración por iteración. Obteniendo el valor de los parámetros de la red, se inicia el ciclo. (Lv *et al*., 2019)

La red que evolucionará a través de los 3 aprendizajes será la que predice la acción dependiendo de lo observado en el entorno en el que se encuentra navegando.

En el proceso de aprendizaje del AS, el sistema de redes toma como entrada la observación concatenada y da como resultado la calidad del proceso realizado, con el cual, posteriormente se obtendrá el valor del aprendizaje. El valor del aprendizaje se explicará más adelante.

### *2.2.4 Etapa de Aprendizaje No Supervizado*

A diferencia que el aprendizaje anterior donde se utilizan conceptos del DDPG, esta aplica el Twin Delay 3 (TD3), que es la optimización del Deep Q-Learning, colocando un actor y dos críticos gemelos. La configuración de las redes es similar a la presentada en el apartado anterior.

En esta etapa se diseñan redes gemelas, 2 redes actores, una modelo y otra target y 4 redes críticas, dos modelos y dos targets (Fujimoto *et al*., 2018), cuyo funcionamiento es en paralelo, en primera instancia, tomando como entrada la observación (resultado de la concatenación de las 3 imágenes tomadas por la cámara), se obtienen (5) y (6).

$$a_p = \pi\mu(O_{co}) \quad (5)$$
$$a_r = \pi\mu'(O_{co}) \quad (6)$$

donde $\mu\ y\ \mu'$ son pesos de conexión de las redes gemelas actores.

Después, la red analiza si la acción real fue adecuada usando el sensor ultrasónico, el valor repercute en el aprendizaje. Al igual que en el aprendizaje anterior, se calculan las calidades: predicha y real. Tomando en cuenta la observación concatenada y la acción, el cálculo de estas se muestra con (7) y (8).

$$Q_{pi}(O_{co}, a_p) = \pi\phi(O_{co}, a_p) \quad (7)$$
$$Q_{ri}(O_{co}, a_r) = \pi\phi'(O_{co}, a_r) \quad (8)$$

donde $\phi\ y\ \phi'$ son pesos de conexión de las redes gemelas criticas.

Luego se obtiene el error cuadrático medio, cuyo valor debe ser menor a la iteración anterior, calculado con (9). En caso de que el error cuadrático sea mayor que el anterior, se deben modificar los parámetros de la red con (4), de lo contrario seguirá guardando información en la base de datos.

$$Error = MSE\_Loss\left(Q_{pi}(O_{co}, a_r), Q_{ri}\right) \quad (9)$$

Es importante mencionar que, en la etapa, de AS, evalúa iteración tras iteración porque apenas está aprendiendo el agente, mientras que en ANS toma un bloque de 5 iteraciones para calcular el porcentaje de similitudes entre el estado real y el estado predicho. En el proceso de aprendizaje del ANS, el sistema de redes toma como entrada la observación concatenada, ejecuta redes gemelas para obtener ambas acciones y da como resultado la calidad del proceso realizado, con el cual, posteriormente se obtendrá el valor del aprendizaje.

*2.2.4 Etapa de Aprendizaje por Refuerzo Profundo*

La configuración de las redes es igual a la del apartado anterior, ya que usa el mismo TD3. Aquí conserva los gemelos de la etapa anterior y se implementan términos como el descuento $\tau$, la recompensa $r$ y las calidades obtenidas por el "Deep Q-Learning", mostradas en (10) y (11), cuya base de su desarrollo se presenta en (İrem *et al.*, 2020).

$$Q_{pi}(O_{co}, a_p) = \pi\phi(O_{co}, a_p) \quad (10)$$
$$Q_{ri}(O_{co}, a_r) = \pi\phi'(O_{co}, a_r) \quad (11)$$

En el ARP, la recompensa depende del descuento y de la calidad, esto se muestra en (12).

$$r = (1 - \tau)Q \quad (12)$$

donde $Q$ de la calidad obtenida, es decir, por (10) y por (11).

Para mejorar el aprendizaje se colocará al agente en un entorno diferente donde este debe encontrar nuevamente un objeto especificado por medio de castigos y premios.

Al igual que en el AS, se calcula el error cuadrático. El error obtenido, se recalcula tomando en cuenta el descuento, quedando como (13).

$$Error = Error + (1 - \tau) * Error \quad (13)$$

A diferencia del Aprendizaje Supervisado (AS) y del No Supervisado (ANS), el ARP toma 2 iteraciones para calcular el porcentaje de similitudes entre el estado real y el estado predicho, utilizando como base el algoritmo de doble retardo TD3 (S. Fujimoto et al. 2018).

En el proceso de aprendizaje del ARP, el sistema de redes toma como entrada la observación concatenada, ejecuta redes gemelas para obtener ambas acciones y se descuenta el factor $\tau$ para calcular el valor del aprendizaje que explica posteriormente.

*2.3 Características Mecánicas*

Una de las clasificaciones de los vehículos móviles terrestres es donde los dividen de acuerdo con la configuración de locomoción los cuales son la configuración diferencial y la configuración Ackerman. La mayoría de los desarrolladores de vehículos robóticos terrestres tienen la configuración diferencial, independientemente del diseño de la llanta (Dong *et al.*, 2021), (Lepej *et al.*, 2015), (Gon-Woo, 2014), (Cos-Cholula y Hernandez-Mendez, 2020), (Wang y Shi, 2019), sin embargo, como se busca tener un mayor acercamiento a los grados de libertad de un automóvil convencional se utilizará la configuración Ackerman, mostrada en la figura 4.

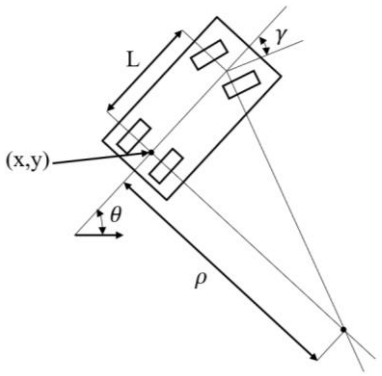


Fig. 4 Configuración Ackerman.

En la figura 4 se tiene como punto de referencia del espacio x,y, punto que llamaremos *posición del vehículo*, que a su vez tiene un punto de orientación con respecto a un marco de referencia inercial θ. *L* es la distancia entre las llantas delanteras y traseras (135mm), este valor es importante ya que indica que tan grande es el arco de giro del vehículo, con valor ρ como la distancia que hay entre el robot y el centro instantáneo de rotación que se crea. γ es el ángulo de dirección de los neumáticos delanteros, cuyo valor va de -30° a 30° donde 0° corresponde a ir adelante.

La posición del robot en el espacio está dada por el vector $p$ *de* (14).

$$p = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \quad (14)$$

Para calcular las variables de posición del agente, se utilizan (15), (16) y (17) (Oscar, 2021), (Me_k_tronico, 2020).

$$\dot{x} = u_s \cos\theta \quad (15)$$

$$\dot{y} = u_s \operatorname{sen}\theta \quad (16)$$

$$\dot{\theta} = \frac{u_s}{L} \tan u_\gamma \quad (17)$$

Donde $u_s$es la velocidad de avance y $u_\gamma$la velocidad de giro de dirección de ruedas.

En la figura 5, se muestra un plano del agente, donde se ubica en la parte frontal un sensor ultrasónico a 60 grados con respecto al suelo, este diseño se debe a que, si el entorno cuenta con objetos por debajo de este rango, el sensor no se activará, por lo que el agente no los considerará como obstáculos y seguirá adelante, de lo contrario el sensor se activará considerando que en la parte frontal hay un obstáculo que se debe evitar. Esta función es útil ya que no siempre estará el agente en un mismo nivel de piso, ya que habría rampas u otros objetos posibles de pasar.

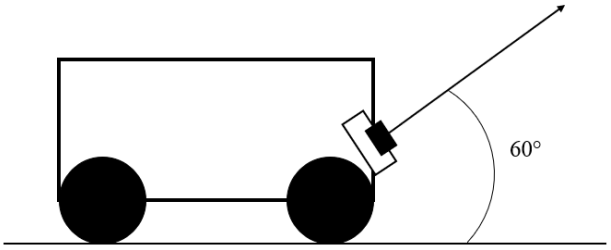


Fig. 5 Configuración a 60° del sensor ultrasónico en el Agente robótico.

Las dimensiones del agente son 235×132×180mm, longitud, ancho y altura respectivamente, con un peso de 840g.

Mientras van pasando las iteraciones el sensor se va usando menos, indicando que el NRNH-AR va evolucionando, sin embargo, como el entorno puede ser dinámico, es decir, se modifica a través del tiempo, por lo que el sensor ayuda a que el agente robótico no colisione.

## 3. Características de los componentes

Las características de los elementos del Agente Robótico se encuentran en la tabla 3.

Tabla 3. Características de los elementos del Agente Robótico.

| Hardware | Especificaciones Técnicas |
|---|---|
| JETSON NANO | GPU 128-core Maxwell<br>CPU Quad-core<br>ARM A57 a 1.43GHz<br>RAM: 4GB 64bits<br>8MP |
| Cámara CSI | Chip de Módulo Sony IMX219<br>Resolución de 3280 x 2464<br>Tamaño 1/4 pulg |
| Motores | Tensión nominal 3-50v<br>14425-23000 rpm<br>Eficiencia 71.61% |
| | Corriente 0.11-0.486 A |
| Servomotor | Torque 2.5kg/cm<br>Voltaje 4.8-6v<br>Velocidad 0.1s<br>Dimensión 32x32x12mm |
| Batería | 10000 mAh<br>67.6 x15.9x142.8mm<br>Tres puertos de salida 5V/2A<br>Peso 255g |
| Arduino UNO | Microcontrolador: ATmega328<br>Voltaje de operación: 5V<br>Voltaje de entrada recomendado: 7-12V<br>Pines digitales: 14 (de los cuales 6 son salidas PWM)<br>Pines de entrada análogos 6<br>Corriente por pines: 40mA<br>Corriente para pines 3.3V; 50mA<br>Memoria flash 32KB (0.5kB usados para bootloader<br>Velocidad del reloj: 16MHz. |
| Sensor ultrasónico HC-SR04 | Salida digital<br>Rango de distancia: 2cm a 400cm<br>Resolución: 3cm<br>Frecuencia central: 40kHz<br>Duración aproximada de pulso recibido sin obstáculos 38ms<br>Voltaje de Alimentación: 5V DC<br>Corriente en reposo: < 2mA<br>Corriente de operación: 15A |

## 4. Entrenamiento

El algoritmo se implementó en el agente robótico diseñado con configuración Ackerman y se tomará en cuenta el Nivel de Aprendizaje (NA, explicada en la sección 5 Resultados experimentales) el cual mostrará la tasa de éxito del NRNH-AR, el tiempo que demora en realizar un ciclo, este valor nos mostrará el paso al siguiente estado, el valor de la cantidad de la base de datos (ε) y el valor condicional de la CNN para poder entrenar (η). El entrenamiento se dividió en 2 partes, ejecutar el algoritmo en un entorno estático y luego ejecutarlo en un entorno dinámico.

### *4.1. Entornos estáticos*

En la primera parte, el agente debe hallar un objeto que se encuentra en los entornos físicos, representados por la figura 6.

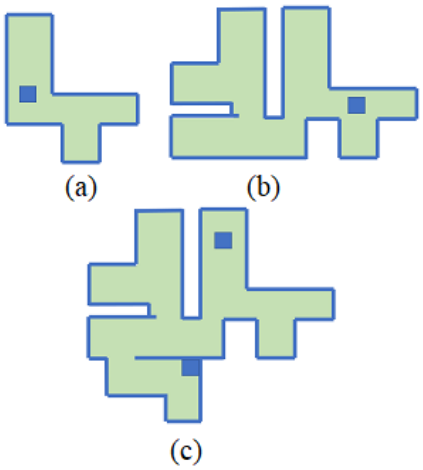


Fig. 6 Configuración de los entornos (a), (b) y (c) basados en los entornos físicos las líneas y los cuadrados azules son obstáculos.

La configuración del entorno mostrado en la figura 6A, se llevarán a cabo las etapas de Aprendizaje Supervisado y No Supervisado para que el agente robótico adquiera suficiente experiencia y conocimiento, por otro lado, en los entornos B y C se ejecuta la etapa de Aprendizaje por Refuerzo Profundo.

En la figura 7 se representa el trayecto seguido en el entorno C desde el punto de inicio hasta que encuentra el objeto.

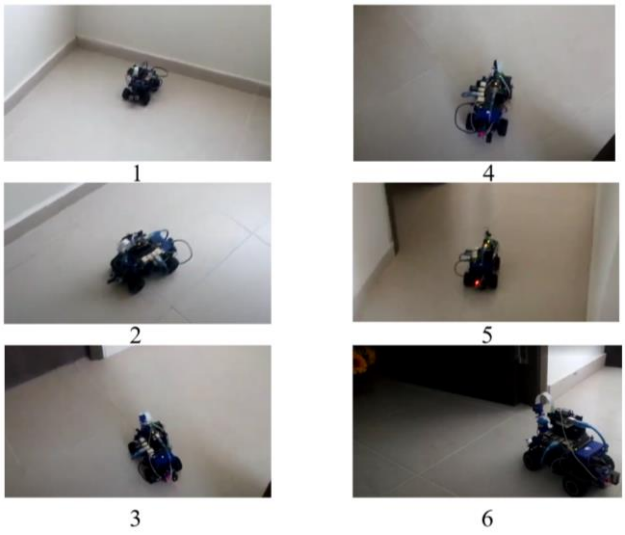


Fig. 7 Proceso de desplazamiento del agente en un entorno estático. 1) Posición de inicio, 2) Selección se acción ir adelante, 3) Selección de acción detenerse, 4) Selección de acción giro a la derecha, 5) Selección de acción ir adelante, 6) Selección de acción giro a la izquierda, donde encuentra el objeto.

### *4.2. Entorno dinámico*

En la segunda parte, la meta por lograr del Agente Robótico (AR) es hallar un cubo de Rubik en un entorno mostrado en la figura 8. El entorno es dinámico, ya que se mueve la posición de los bloques.

Fig. 8 Entorno donde se realizan las pruebas del algoritmo NRNHN-AR

En la figura 9 se representa cómo se entrenó en el entorno dinámico, mostrando cómo selecciona la acción dependiendo de lo que visualiza en el entorno cambiado.

## 5. Resultados Experimentales

Para medir cuan eficiente se convierte el algoritmo NRNH-AR, se coloca una variable llamada Aprendizaje, que va en un rango de 0 a 1, donde 0 representa una red nada eficiente y 1 un sistema de redes eficiente, para el cual se utilizan las siguientes ecuaciones de la tabla 4 para calcularla dependiendo del tipo de aprendizaje en el que se encuentren.

### *5.1. Resultados del entorno estático*

Se realizaron 817 épocas en total, cuyo número de iteraciones depende de la cantidad de tiempo que tarde el agente desde su punto de origen hasta en encontrar el objeto.

En la figura 10 se muestra el comportamiento de este aprendizaje a lo largo de las épocas. Se observa en la figura 10 que en las etapas de Aprendizaje Supervisado (AS) y por Refuerzo Profundo (ARP), las curvas de evolución se muestran inestable, al contrario del Aprendizaje No Supervisado (ANS) que muestra estabilidad en la evolución, dichas caídas se deben por colisionar, tener un error de predicción y, en el caso del ARP, el tiempo que demora en cumplir su objetivo.

Es importante destacar que hay caídas de NA en el cambio de aprendizajes (de AS a ANS, y de ANS a ARP). Sin embargo, la mayoría de los entornos son dinámicos por lo que el agente se puso a prueba en un entorno nuevo y dinámico, dejando la base de datos generada en este apartado.

### *5.2. Resultados del entorno dinámico*

Se realizaron 300 épocas en total, cuyo número de iteraciones de cada época depende de la cantidad de tiempo que tarde el agente desde su punto de origen hasta en encontrar el objeto.

En la figura 11 se muestra el comportamiento del nivel de aprendizaje a lo largo de los tres tipos de aprendizaje (NRNH-AR) con respecto a los valores de ε y η. El valor η depende de la configuración de la CNN la cual se calcula al multiplicar el número de épocas de la CNN por el número de pasos de la CNN, como se muestra en (18).

$$\eta = número\ de\ épocas\ de\ CNN * número\ de\ pasos\ de\ CNN \quad (18)$$

Entiéndase por épocas como el número de veces que se van a iterar el conjunto de datos durante el entrenamiento y los pasos corresponde al número de veces que se va a procesar la información en cada una de las épocas. Cada época tendrá n pasos.

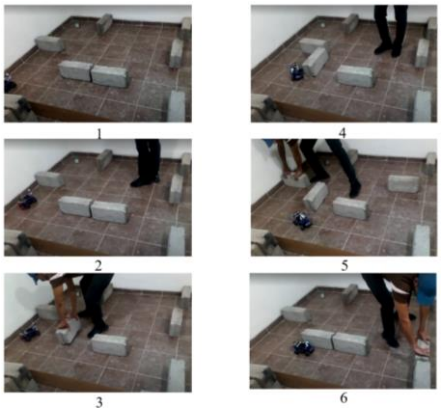


Fig. 9 Proceso de desplazamiento del agente en un entorno dinámico. 1) Posición de inicio, 2) Selección de acción ir adelante, 3) Selección de acción girar a la derecha y se modifica entorno, 4) Selección de acción del entorno modificado giro a la derecha, 5) Selección de acción ir adelante, 6) Selección de acción giro a la izquierda y se modifica el entorno.

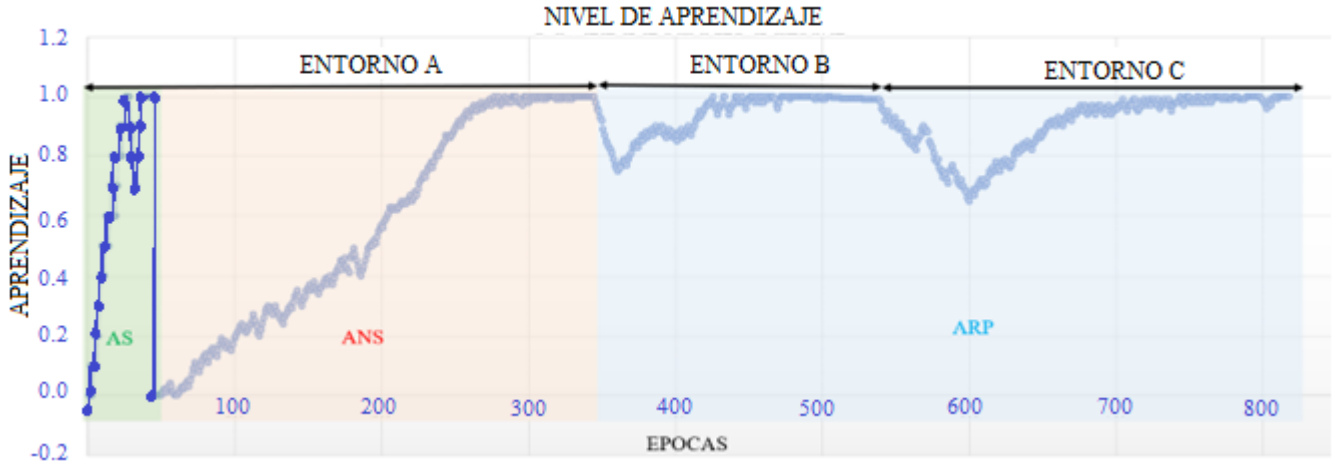


Fig. 10 Nivel de aprendizaje en un entorno estático

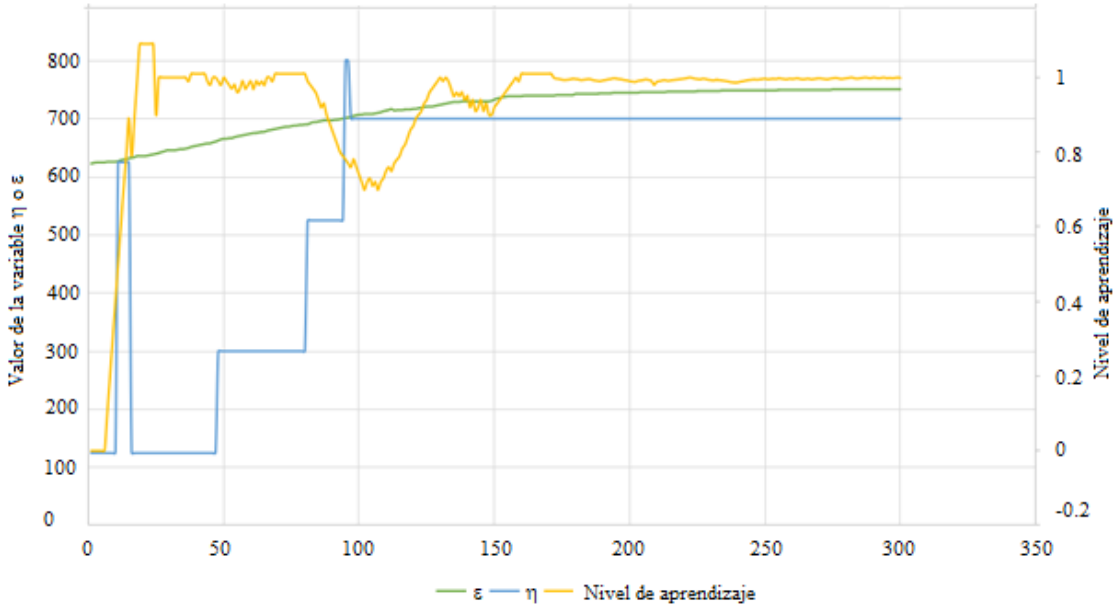


Fig. 11 Entorno donde realizan pruebas del algoritmo NRNH-AR

Tabla 4. Cálculo de la variable aprendizaje en los distintos métodos.

| | |
|---|---|
| AS | $Si\ a_p = a_r$<br>$\therefore \boldsymbol{A = +0.1}$<br>$Si\ a_p \neq a_r$<br>$\therefore \boldsymbol{A = -0.1}$<br><br>$A: Aprendizaje$<br>$a_p: acción\ predicha;$<br>$a_r: acción\ real$ |
| ANS | $\boldsymbol{A = 1 - Error}$<br><br>donde *Error* está dado por (9) |
| ARP | $\boldsymbol{A = 1 - Error}$<br><br>donde *Error* está dado por (13) |

En la figura 12 se identifica hasta alrededor de la época 100, la condición ε > η no se cumple, lo cual perjudica al sistema completo. Tener la cantidad mínima en la base de datos es importante para que el Nivel de Aprendizaje (NA) evolucione. En la época 0 se tiene las siguientes condiciones: ε = 623; η = 125; NA = 0.

El valor de ε es la cantidad de fotografías que se tienen en la base de datos, los cuales corresponden a los entornos estáticos, es decir, 623 fotografías. Se decide colocar el nivel de aprendizaje igual a 0 ya que es un entorno totalmente nuevo.

Regresando a la figura 11, en la época 300 se obtuvieron los siguientes datos: ε = 752; η = 700; NA = 99.9. El valor de ε aumenta con el paso de las épocas, sin embargo, con el paso del entrenamiento cada vez guarda menos información, es decir, para entender este entorno, bastaron 129 imágenes de este (752-623). Esto se debe a que entre más parecido sea el camino por el que pasa, menos imágenes necesitará para entender el entorno. Una razón por la que el NA disminuye se debe a la relación de ε > η, sin embargo, depende también de la exactitud con la que predice lo observado y procesado por la CNN como se muestra en la figura 12.

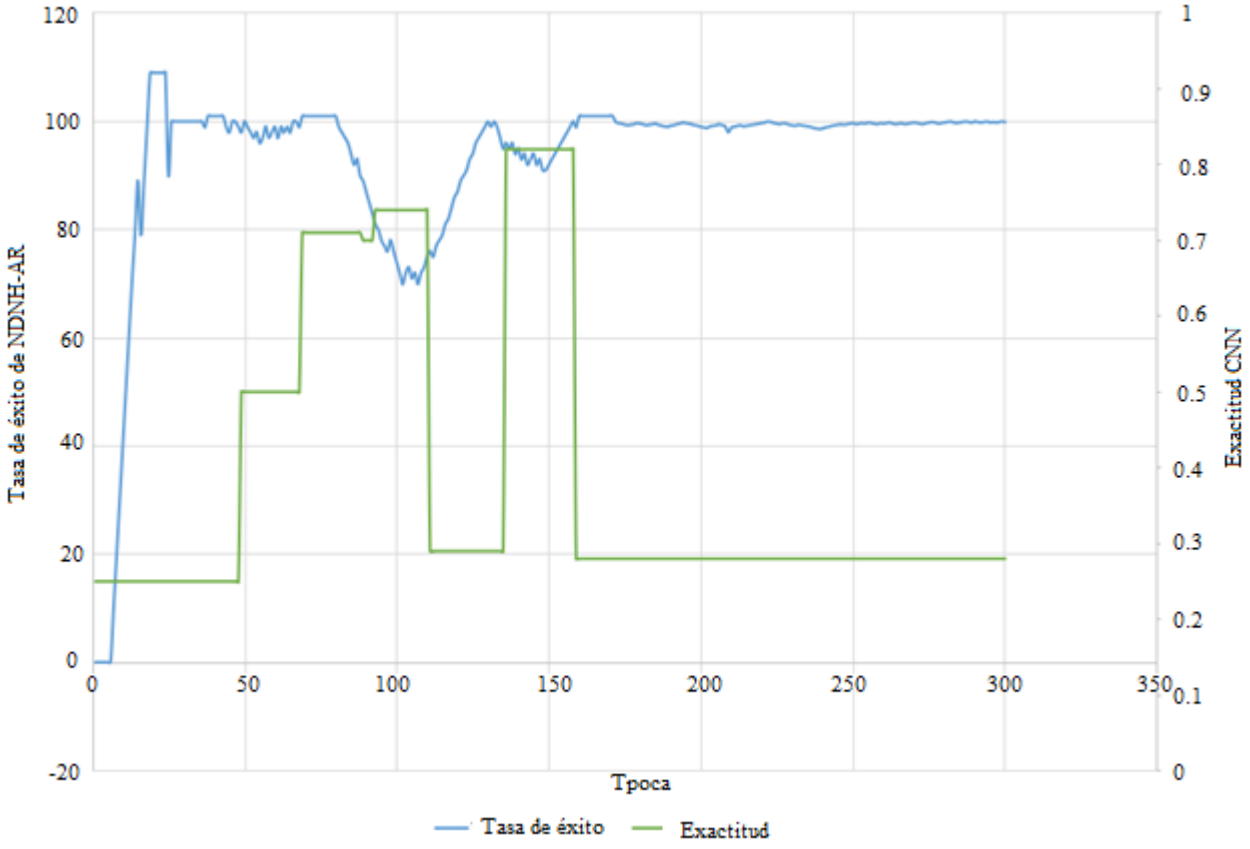


Fig.12 Comparación gráfica entre la Tasa de éxito de NRNH-AR con la exactitud que tiene la CNN

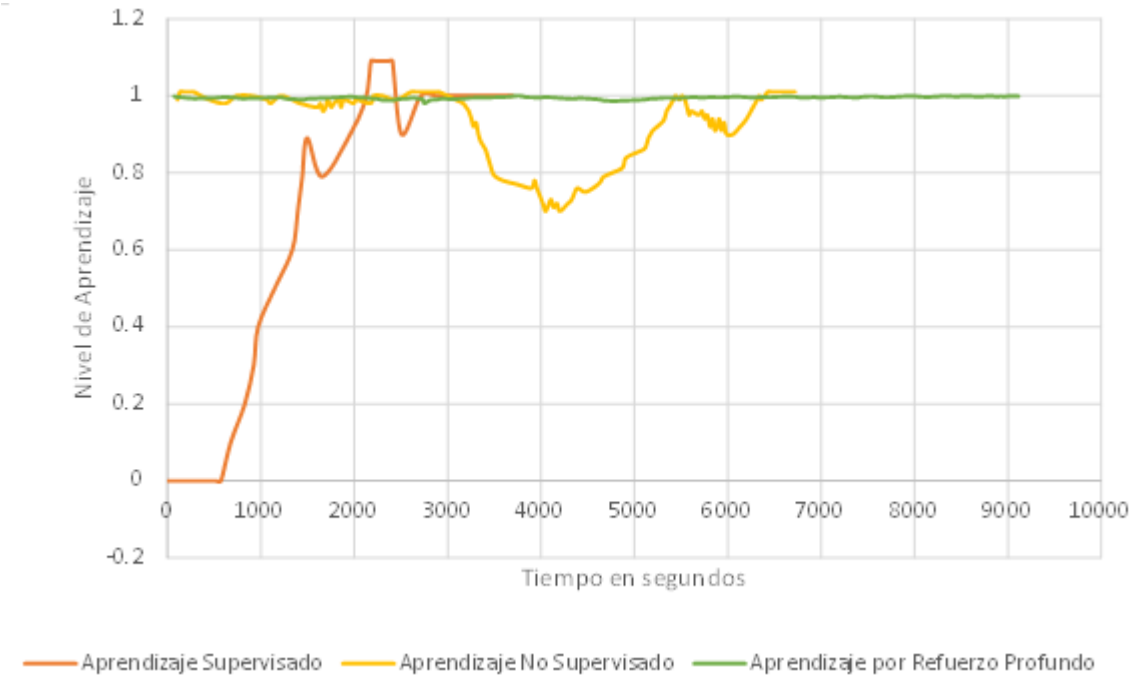


Fig. 13 Comparación de la estabilidad de las diferentes etapas de la NRNH-AR

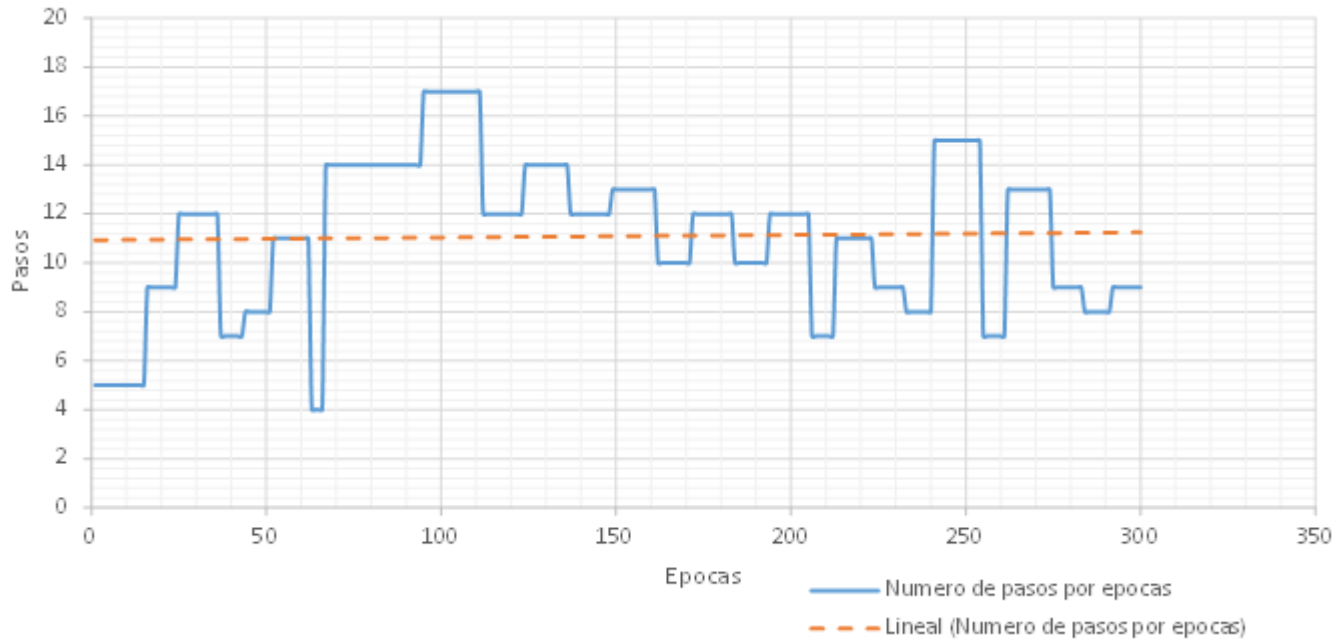


Fig. 14 relación de pasos que tiene cada época de entrenamiento de la NRNH-AR

Se observa la relación de que entre mayor sea la exactitud de la red CNN menor será el desempeño de la NRNH-AR. Las épocas donde se muestra el cambio de valor de *Exactitud CNN* son cuando se entrenaba con la nueva base de datos. La exactitud de la CNN debe ser menor que 0.8, el ideal para nuestro modelo, para que la curva de aprendizaje tenga estabilidad, la exactitud de CNN fue de 0.28. La condición para cambiar de tipo de aprendizaje es que tenga NA=1 en 10 épocas seguidas, ya que se considera que no es posible para el sistema de redes aprender más, sin embargo, en el ARP no fue posible obtener dicha continuidad ya que sus valores oscilaban entre 0.997 a 1. La leve elevación de tasa de éxito corresponde al 1% por arriba del 100%, al considerar que se trata de un modelo de entorno dinámico y físico, se toma como válido una tolerancia del +-1% como nivel de error que puede tener la NRNH-AR al predecir las acciones con respecto a las imágenes de entrada. Este 1% de tolerancia se ha colocado en el programa.

En la figura 13 se compara la estabilidad que tiene cada etapa de la NRNH-AR con respecto del tiempo de entrenamiento. Como era de esperarse, en el Aprendizaje Supervisado no es estable, (Fujimoto *et al*., 2018) explican que el modelo DDPG (sistema de una red actor y una red crítica) no cuenta con buena exactitud de predicción, hay variabilidad. Por otra parte, la red de ANS tiene una caída, como se explicó anteriormente se debe al valor de exactitud de la CNN y de la condición $\varepsilon > \eta$. Por último, la red del ARP es la que mejor estabilidad tiene al conservarse en un rango menor de valores altos de NA.

La NRNH-AR no tiene un número constante de pasos en cada época, en la figura 14 se muestra la cantidad de pasos que tiene cada época, como resultado, el número de pasos promedio del entrenamiento en un entorno dinámico es de 11 pasos, como se mencionó anteriormente, los pasos corresponden al número de veces que se va a procesar la información en cada una de las épocas.

Al momento de entrenar, la vida útil de la batería del módulo L298N era de aproximadamente de 15min, lo que causaba que el entrenamiento tardara un tanto más de tiempo, este es un aspecto importante, ya que el desplazamiento no es el mismo cuando se tiene la batería al 100% que al 60% o cualquier otro porcentaje, a pesar de que es posible controlar la velocidad de giro, el rango de operación es de 3-5v, con una potencia de 0.51w. Eso genera que las redes se modifiquen aún si ya habían obtenido pesos correctos, ya que se va adaptando a la carga que tiene la batería.

Otro aspecto es el suelo, ya que en ocasiones derrapaban las llantas traseras ocasionando que no se desplace el agente. Los resultados y parte del proceso de las pruebas físicas pueden verse en **https://www.youtube.com/watch?v=SUtYk2Ee1du**.

## 6. Conclusiones

Usar Jetson Nano es una opción para entrenamientos de aprendizaje reforzado, no reforzado y por refuerzo profundo, sin embargo, no tiene la capacidad computacional óptima para realizar un fluido entrenamiento de redes NRNH-AR.

El desempeño del algoritmo NRNH-AR depende de la condición $\varepsilon > \eta$ y de la exactitud que tiene la CNN implementada, entre mayor sea la exactitud de CNN, el Nivel de Aprendizaje del NRNH-AR decrecerá, como se mencionó anteriormente.

Entre muy variada sean las imágenes de entrada en el agente, más tiempo tardará en estabilizarse la red en un rango de entre NA=0.997 a NA=1. En nuestro caso específico, fueron necesarias 129 imágenes tomadas para estabilizarse en el entorno dinámico y 623 en el entorno estático, esto se debe a que el entorno es muy similar tanto el piso por el que navega, como los obstáculos.

La cantidad promedio de pasos que hay en cada época es de 11 al evaluar el algoritmo en un robot físico.

El algoritmo NRNH-AR funciona en el agente robótico, sin embargo, para obtener mejores resultados se modificará el bloque mecánico y el hardware considerando la funcionalidad en tamaño, peso y diseño. Además, para tener un

entrenamiento continuo y más rápido se debe contar con un sistema de energía duradera.

También como trabajo futuro se implementará el NRNH-AR en diferentes tipos de robots para comprobar cuan viable el algoritmo.

## Referencias


Chen, W., Zhou, S., Pan, Z., Zheng, H., Liu, Y (2019). Mapless Collaborative Navigation for a Multi-Robot System Based on the Deep Reinforcement Learning. Volume26 Issue 5 doi: 10.3390/app9204198

Diaz-Arango, G., Vazquez-Leal, H., Hernandez-Martinez, L., Jimenez-Fernandez, V. M., Heredia-Jimenez, A., Ambrosio, R. C., Huerta-Chua, J., De Cos-Cholula, H., Hernandez-Mendez, S. (2020). Multiple-Target Homotopic Quasi-Complete Path Planning Method for Mobile Robot Using a Piecewise Linear Approach. MDPIST ALBAN-ANLAGE 66, Ch-4052 Basel, Switzerland. Volume 20, Issue 11, Article Number 3265, doi: 10.3390/s20113265

Dong Gi, G., Kyon Mo, Y., Min Ro, P., Jehun, H., Jaewan, K., Joonwoo, L., Kap Ho, S. (2021). Marker-Based Method for Recognition of Camera Position for Mobile Robots. Sensors 2021, 21, 1077. Doi: 10.3390/s21041077

Faisal, M., Hedjar, R. Sulaiman, M. A. (2013). Fuzzy Logic Navigation and Obstacle Avoidance by a Mobile Robot in an Unknown Dynamic Environment. International Journal of Advanced Robotic Systems, College of Computer and Information Sciences, King Saud University, Saudi Arabia. doi: 10.5772/54427

Fujimoto, S., Hoof, H., Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. Proceedings of the 35 th International Conference on Machine Learning, Stockholm, Sweden PMLR 80.

Gon-Woo, K (2014). Expanded Guide Circle-based Obstacle Avoidance for the Remotely Operated Mobile Robot. Springer singapore pte ltd#04-01 cencon i, 1 tannery rd, Singapore 347719, SINGAPORE. Volume 9, Issue 3, Page 1034-1042, doi: 10.5370/JEET.2014.9.3.1034

Grigoryeva, S., Alimkhanova, A., Batalova, M. (2021). Research of indoor temperature data transmission using visible light communication technology. IMET 2020 Journal of Physics: Conference Series. 1843 012004 doi: 10.1088/1742-6596/1843/1/012004

İrem, M., Alper, K. T., Beyda, T. Ahmet, B. T., Oğuz, Y. (2020). FUHAR: A transformable wheel-legged hybrid mobile robot. Elsevierradarweg 29, 1043 nx amsterdam, netherlands. Volume 133, Article Number 103627. Doi: 10.1016/j.robot.2020.103627

Lepej, P., Maurer, J., Uran, S. (2015). Dynamic Arc Fitting Path Follower for Skid-steered Mobile Robots. Sage publications INC2455 Teller rd, thousand Oaks, CA 91320. Volume 12, Article Number 139, doi: 10.5772/61199

Li, N., Zhao, X., Yang, Y., Zou, X. (2016). Objects Classification by Learning-Based Visual Saliency Model and Convolutional Neural Network. Hindawi ltdadam house, 3rd flr, 1 fitzroy sq, London w1t 5hf, England. Volume 2016, Article Number 7942501. Doi: 10.1155/2016/7942501

Lillywhite, K., Lee, D., Tippetts, B., Archibald, J. (2013). A feature construction method for general object recognition. Pattern Recognition, vol. 46, no. 12, pp. 3300–3314, doi: 10.1016/j.patcog.2013.06.002.

Lindeberg, T. (2012), Scale invariant feature transform. Scholarpedia, vol. 7, no. 5, pp. 2012–2021. Doi: 10.4249/scholarpedia.10491

Lowe, D.G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60, 91–11 doi: 10.1023/B:VISI.0000029664.99615.94

Lv, P., Wang, X., Cheng, Y., Duan, Z. (2019). Stochastic Double Deep Q-Network. National Natural Science Foundation of China, 61772532. Doi: 10.1109/ACCESS.2019.2922706

Ma, L., Alborati, M. (2019). Wireless Inter-Vehicle Communication among VEX Robots. Proceedings of the 2019 IEEE 11th International Conference on Engineering Education, ICEED 2019 8994949, pp. 78-83 doi: 10.1109/ICEED47294.2019.8994949.

Marjovi, A., Marques, L. (2011). Multi-robot olfactory search in structured environments. Robotics and Autonomous Systems. 59867-881. doi: 10.1016/j.robot.2011.07.010

Me_k_tronico. (2020). Cinemática de robot móvil tipo triciclo. [Archivo de video]. https://www.youtube.com/watch?v=3xKEOaKttos

Oscar Ramos (2021). Cinemática de Robots Móviles (parte 1/2). [Archivo de video]. https://www.youtube.com/watch?v=2Asd4RH3Gmw

Placed, J., Castellanos, J. (2020). A Deep Reinforcement Learning Approach for Active SLAM. Applied science,10, 8386. doi:10.3390/app10238386

Raja, P., Pugazhenthi, S. (2012). On-line path planning for mobile robots in dynamic environments. Neural network world. Volume 22, Issue 1, Page 67-83. Doi: 10.14311/NNW.2012.22.005

Satyanarayanan, M. (2017) The Emergence of Edge Computing. Computer 50(1):30–39. Doi: 10.1109/MC.2017.9

Stateczny, A., Gierlowski, K., Hoeft, M (2022). Wireless Local Area Network Technologies as Communication Solutions for Unmanned Surface Vehicles. Sensors 2022, 22, 655. Doi: 10.3390/s22020655

Tao Z., Jian W., Jilin Z., Congfeng J. (2022). Deep Reinforcement Learning-Based Workload Scheduling for Edge Computing. Journal of Cloud Computing: Advances, Systems and Applications, doi: 10.1186/s13677-021-00276-0

Vasquez-Jalpa, C., Nakano-Miyatake, M., Escamilla Hernandez, E. (2021). A deep reinforcement learning algorithm based on modified Twin delay DDPG method for robotic applications. 21st International Conference on Control, Automation and Systems (ICCAS), pp. 743-748, doi: 10.23919/ICCAS52745.2021.9649882.

Wang, H., Shi, J. (2019). Design and Modeling of a Novel Transformable Land/Air Robot. Hindawi ltdadam house, 3rd flr, 1 fitzroy sq, London w1t 5hf, England. Volume 2019, Article Number 2064131, doi: 10.1155/2019/2064131

Wróbel, K., Karwatowski, M., Wielgosz, M., Pietroń M., Wiatr, K. (2020). Compressing sentiment analysis cnn models for efficient hardware processing. Computer Science, 21(1). Doi: 10.7494/csci.2020.21.1.3375

Zhu, K, Zhang, T (2021). Deep Reinforcement Learning Based Mobile Robot Navigation: A Review. Volume26 Issue 5 Page 674-691. Doi: 10.26599/TST.2021.9010012