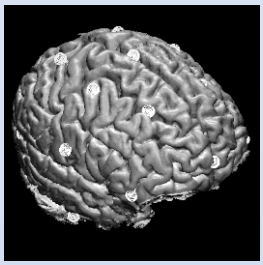
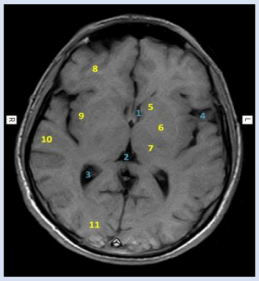
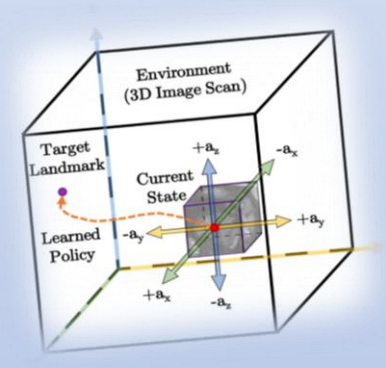
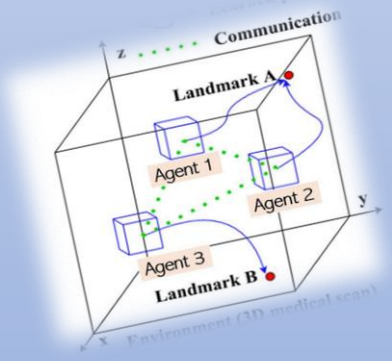




El trabajo versa sobre el Aprendizaje Por Refuerzo Profundo MultiAgente, aplicado a la identificación de puntos de referencia anatómicos en imagenes cerebrales obtenidas mediante RMI y Ecografías.Estudio comparativo entre difernetes arquitecturas y configuraciones de los diferentes agentes inteligentes.





Communicative Reinforcement LearningAgents for Landmark Detection in Brain Images

Multi-agent Deep Reinforcement Learningfor Anatomical Landmark Detection

Guy Leroy y Dr. Amir Alanasary

José Javier Gutierrez Gil

Contenido

1. Introducción

El presente trabajo se realiza como **proyecto** de la asignatura de **Aprendizaje Maquina** del **Grado de Ciencia de Datos**. Dicha asignatura plantea el **estudio** de diferentes **técnicas** y **modelos** de **Aprendizaje Automático** en diferentes áreas. Donde la observación, adquisición y el análisis de los datos, junto con dichos modelos y algoritmos de aprendizaje, ofrece una rama de la inteligencia artificial blanda(narrow/weak AI) que ha **permitido la evolución de la IA y una nueva era de la IA a todos los niveles económicos/social** . Para la realización de esta tarea **se ha optado** por el estudio de **una de las ramas del Machine Learning** estudiada en clase como es el **Aprendizaje por Refuerzo**, **RL** por sus siglas en inglés (**Reinforcement Learning). En concreto**, el artículo sugerido plantea la **utilización de una técnica de aprendizaje por refuerzo profundo multiagente** . Utiliza la potencia de las **redes neuronales** para el **aprendizaje de los parámetros de la función** Q **o** la **política** de **múltiples agentes que colaboran** en el **aprendizaje**, permitiendo que **aprendan simultáneamente** a medida que cada agente **busca una política óptima** para **alcanzar el objetivo** (ya sea un **objetivo individual o colectivo**). Se **apuesta por unir** la **potencia** del RL (**Reinforcement Learning**) con la potencia del DL (**Deep Learning**) para **detectar de forma automática puntos de referencia anatómicos**. **Además**, **utiliza la filosofía** que contribuyo el nacimiento de la IA como son los **Agentes Inteligentes** (Wooldridge, 1999), como **sistema perceptivo autónomo capaza de aprender e interactuar de forma autónoma con su entorno**. Notar que el **RL** es un **subconjunto** de los **Agentes Inteligentes**, en tanto cuanto es un **Agente capaz de aprender de sus acciones** y del **entorno** en el que se encuentra, tomando sus propias decisión (política). **Confluyen** por tanto **otras filosofía** de actuación cercanas a la **teoría de juegos**, **modelos teóricos/prácticos** de la acción social. Aspectos referentes a la **comunicación**, **cooperación** o **resolución de conflictos** presentes en los **sistemas Multiagentes (MAS)** (Shoham, 2007) donde **varios agentes inteligentes** se **comunican** para **aprender de forma simultánea** con ayuda de la **información de todos los agentes presentes** (McArthur, 2007). Una forma de **resolución de problemas** de forma **distribuida** o **paralela** de los sistemas inteligentes (DAI Distributed Artificial **Intelligence**) (Dorri, 2018). **En este problema** para construir el **Agente Inteligente** se toma como base un **algoritmo** muy conocido en la literatura de **RL como es el DQN** (Osband, 2016), (Anschel, 2017) y **sobre él construye una arquitectura multiagente (Multi-Agent Competitive Reinforcement Learning**) (Lazaridou, 2020).

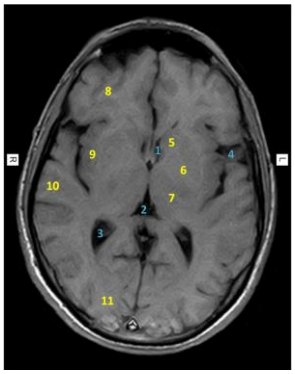


Fig1: Puntos referencia anatómicos cerebro https://radiologia2cero.com/como-interpretar-rm-de-cerebro/

La **identificación de estos puntos de referencia anatómicos** es un **paso previo y vital para el análisis y estudio de las imágenes médicas**. Dichos puntos de referencia anatómicos **son marcadores universales esenciales que permiten la identificación y ubicación exacta de diferentes regiones anatómicas** dentro de la región musculoesquelética de estudio. La identificación de estos marcadores **permite realizar la reconstrucción precisa de la región de interés**, realizar **estudios morfológicos**, **evaluar malformaciones**, plantear la **creación de planos de corte exactos** para identificar regiones de interés. Son especialmente **utilizados en las neurocirugías** y **Microquirúrgica** en 3D de la superficie Cerebral.

Por tanto, nos planteamos una inmersión en el mundo del **procesamiento de imágenes médicas para el apoyo al diagnóstico y técnicas quirúrgicas**. Un tema ampliamente estudiado en diferentes investigaciones a lo largo de los últimos 30 años. Y que no han podido ser extrapoladas fuera del mundo de la investigación como se hubiera deseado dado su alto coste computacional y dificultad para la identificación y localización precisa de las diferentes regiones presentes en las imágenes tomadas. **Es con los avances en el hardware y el surgimiento**

de las nuevas técnicas de Machine Learning y Deep Learning cuando ha resurgido un alto grado de interés de dichas técnicas médicas. Ya no sólo como apoyo en la visualización interna y el diagnóstico por imagen de las afecciones del paciente, sino **como herramienta principal en la toma de decisiones del médico**. Lo vemos en innumerables clínicas dentales donde la reconstrucción maxilofacial 3D es un paso previo para posibles cirugía o para presentar al cliente el aspecto final. Se utiliza ampliamente en microcirugías, en diagnósticos automáticos previa supervisión de expertos en diferentes afecciones musculoesqueléticas. Las mismas técnicas son utilizadas por los médicos en **prácticas para identificar las regiones internas y saber localizar en 3D todos los, órganos, tejidos y partes blandas** por las que van a tener que al intervención.

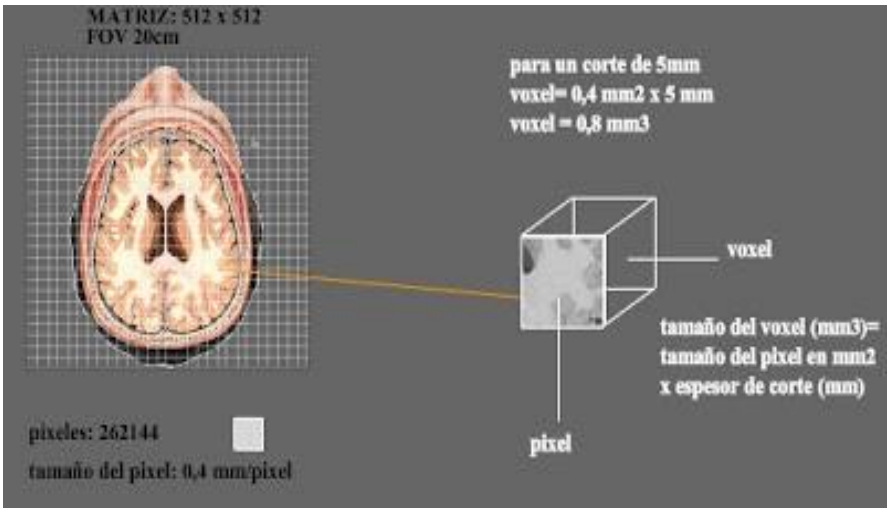


Fig 2:TAC http://wwwimagenologia607.blogspot.com/2011/03/tomografia.html)

Notar que **el articulo** escogido toma como **base** la reconstrucción 3D de la **imagen**, y **sobre dicha imagen** se plantea el **uso** de las técnicas de **RL** y **Deep Learning** para la **detención de puntos de referencia anató**micos. Por tanto, debemos saber que la estructura de datos para la reconstrucción de las imágenes médicas 3D es el Voxel. Un **voxel** es la **representación 3D de cada pixel de la imagen**, cuyo valor "se le suele asignar la densidad media de dicho volumen"[1] es el **valor medio del pixel o region de interés (ROI**). Por tanto, debemos tener claro en todo momento que **nos movemos por un sistema de referencia de la imagen**, extrapolando dicho sistema de referencia al **sistema de referencia de su estructura de voxels donde tenemos "mapeada" o representada la imagen**.

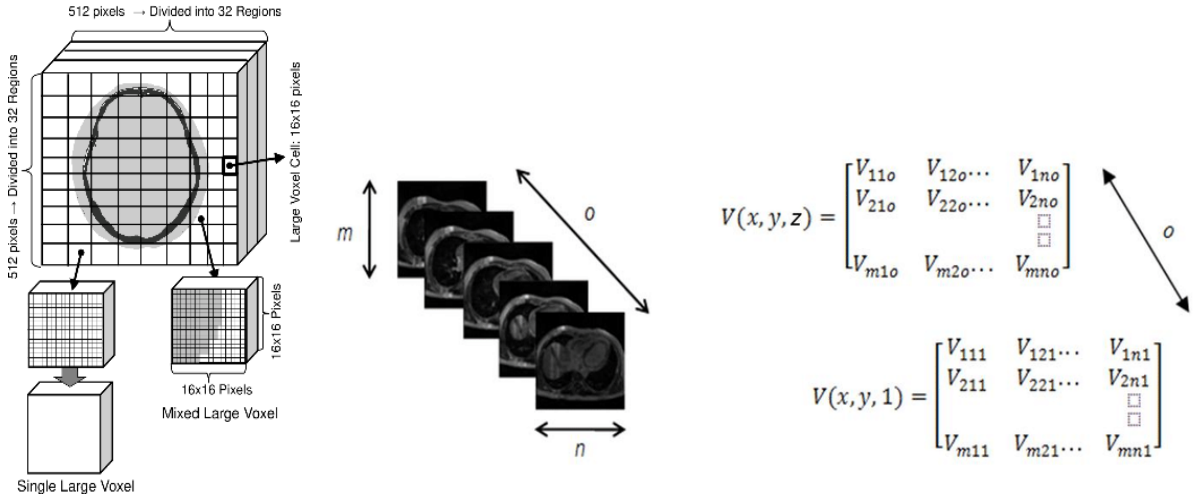


Fig3: RM - Gavidia, Giovana & Martín-Landrove, Miguel & Cerrolaza, Miguel & Soudah, Eduardo. (2011)

[1] Realmente esto no es así, sino que se deben realizar diferentes operaciones para obtener el valor de color y poder segmentar la imagen. En el caso más sencillo de un algoritmo de reconstrucción en volumen y umbralización básica con trazado de rayos puedes realizar una interpolaciones triliterales, entre los diferentes valores de cada vértice del cubo. Y el valor de cada vértice tiene una pequeña contribución de los cubos adyacentes a él.

2. Motivación

Una principal motivación es la realización de una breve investigación con el objetivo de **identificar una de las técnicas de ML (Machine Learning) estudiadas en la parte de teoría bajo el espectro de un problema real**. **Describir dicha técnica** y **explicar cómo la han utilizado** para **la detección automática de puntos de referencia anatómicos**.

Por otro lado, vemos que la **utilización de RL** bajo el modelo **de múltiples agentes comunicándose para compartir información mediante aprendizaje profundo** es un **tema de actualidad** y que está **aportando innumerables investigaciones** en todo el ámbito del ML y DL. Este hecho pude ser debido a que el uso del aprendizaje profundo como aproximador de funciones no lineales, puede favorecer a los algoritmos de aprendizaje por refuerzo a la hora de fijar sus parámetros y así obtener una política óptima del agente. **Un motivación intrínseca al suso de redes neuronales** es el hecho de **identificar como resuelven los problemas de sobreajuste** asociado a dichas técnicas. Además de **identificar que técnicas son las que se están utilizando** para conseguir la **comunicación entre diferentes agentes**. Y como todas estas técnicas se utilizan en el procesamiento de imágenes médicas.

3. Alcance

En primera instancia, se hace un **repaso teórico general y breve de los modelos de aprendizaje por refuerzo centrándonos en la familia Q-Learning y su derivado DQN** que es el que utiliza como base el artículo. Posteriormente **se expone las modificaciones realizadas al algoritmo DQN** y como se **resuelven los problemas de sobreajuste** de **las redes neuronales**. Se identifica **relevancia de trabajar con múltiples agentes** y las posibles configuraciones. Para, por último, **describir los modelos utilizados en el estudio de investigación** (C-MARL, collab-DQN y Single Agent DQN), enfatizando en **cómo implementan la comunicación entre los diferentes agentes** , tema de interés de la investigación. Después, se **pretende describir la experiencia de ejecución del código y comentarios sobre su implementación**. Para acabar con **algunas sugerencias sobre el artículo y su implementación.**

4. Estructura

Teniendo en cuenta la motivación y objetivo del presente trabajo. El **documento se divide en** diferentes **partes donde se expone la investigación desarrollado por Guy Leroy y Dr. Amir Alansary** en el departamento de computación del Imperial College London[2] cuyo trabajo se expuso en el artículo "***Communicative Reinforcement Learning Agents for Landmark Detection in Brain Image[3]s***" publicado en 2020 y **citado en nueve ocasiones en otras investigaciones**[4] similares. De esta forma, la estructura del documento queda:

I. Introducción, Motivación y estructura del trabajo.
II. Un aparte principal donde se resume el articulo y la investigación realizada por Guy Leroy.
    a. Métodos y Modelo: Se identifica el modelo de RL utilizado en la investigación y se describen los dos bancos de datos establecidos para el estudio.
    b. Resultados y conclusión de la investigación: donde se detallan los resultados y

[2] https://gml16.github.io/projects/mastersthesis.pdf
[3] https://arxiv.org/abs/2008.08055
[4] https://scholar.google.es/scholar?cites=12255804045704558317&as_sdt=2005&sciodt=0,5&hl=es

conclusiones del mismo

III. Reflexiones: Se reflexiona sobre la experiencia del estudio y ejecución del código asociado al mismo. Identificando sus partes.

IV. Sugerencias. Se indican unas posibles variaciones a tener en cuenta para posibles futuras investigaciones. Así como pequeñísimas modificaciones en el código de la investigación.

5. Objetivo del Trabajo de estudio

En el artículo escogido se propone la **utilización de un sistema de aprendizaje por refuerzo profundo (DRL) con múltiples agentes cooperando** (**C-MARL**) (J. Lin, 2020) para **detectar automáticamente puntos de referencia anatómicos en imágenes médicas 3D**. Por lo general, **la identificación de dichos puntos** de referencia se realiza de **forma manual** sobre las imágenes obtenidas. Lo que suele producir **una variabilidad importante entre diferentes expertos** a la hora de **localizar y fijar con exactitud dichos puntos**. Lo que puede derivar en **errores importantes a la hora de localizar ciertos órganos o regiones internas**, identificar **anomalías cuando realmente no las hay** o generar problemas en el guiado y seguimiento de una microcirugía, etc. Además, es necesario **mantener expertos en dominios específicos para cada tipo de imagen y región de estudio**.

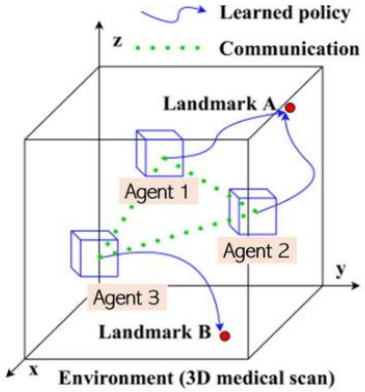


Fig4: Puntos referencia anatómicos. Leroy, G., Rueckert, D., & Alansary, A. (2020).

En concreto, su objetivo es la de **comparar dicha arquitectura (C-MARL), con distintos números de agentes** que cooperan en el aprendizaje **frente a otros modelos de aprendizaje con agentes individuales**. Y también el de **mantener distintas configuraciones** (cada uno busca un punto de referencia o todos los agentes buscan los mismos puntos de referencia).

Notar que **los modelos a comparar se evalúan sobre dos conjuntos de datos distintos**: Unas imágenes obtenidas por Resonancia Magnética (MR) de adultos y otras imágenes cerebrales de ecografías fetales. El **artículo de investigación asegura que en esta ocasión no es necesario tener un banco de imágenes elevado** ya que al trabajar con **múltiples agentes que procesan diferentes regiones de la imagen**, con esto ya es **suficiente para el entrenamiento de los modelos**. Notar que en **ningún momento se introduce la imagen completa al modelos** sino las **regiones de interés por donde van circulando los agentes**.

Una de las **hipótesis de partida** de la investigación es que **los puntos de referencia anatómicos para cada región no son aleatorios**, sino que **se mantienen en la misma ubicación en todos los pacientes**. Y, por tanto, la **detección automática mediante diferentes técnicas es posible y mejora la identificación manual**.

6. Modelos y Técnicas

6.1. Aprendizaje por Refuerzo

Podríamos determinar el **aprendizaje por refuerzo** como **una** de las tres patas del **aprendizaje automático**, junto al aprendizaje supervisado y el no supervisado. En el aprendizaje automático un **agente aprende a tomar la acción correcta (a_t) que le lleve de un estado (s_t) a otro (s_{t+1}) en base a una recompensa dada (R_t).** Por tanto, no estamos **prediciendo** un valor ni una clasificación sino un **conjunto de acciones (política π)** que nos lleve de un estado inicial aun **estado final (episodio)** con una **máxima recompensa**, se puede decir que **aprendemos del futuro**, de la **experiencia acumulada** dada la recompensa por el **estado actual** o la **acción-estado** que se **decide tomar**. En estos modelos, por tano, podemos **considerar cinco elementos** identificativos:

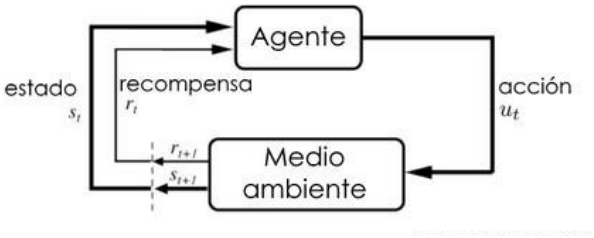


Fig5: Esquema general RL.

- **La política** (π) que define el comportameinto del agete.
- **El entorno o estado** (S_t) en el cual se encuentra el agente y atendiendo a él realiza una u otra acción dada la mejor recompensa.
- **La recompensa** (R_t): que identifica lo buena o mala que es la acción tomada por el agente. El objetivo del RL es el de maximizar dicha política.
- **La función de valor** $V(S_t)$ o **acción-valor** $Q(S_t)$, $Q(S_t,a_t)$ que especifica que comportamiento es bueno en el largo plazo. El valor de un estado es la recompensa que puede esperarse obtener en el futuro desde ese estado. Mientras que la recompensa determina la conveniencia inmediata de un estado, la función de valor determina la conveniencia en el largo plazo.
- El **modelo del entorno que se utiliza** para simular el comportamiento del entorno. Como veremos más adelante el uso o no del modelo en la implementación del RL identifica una familia u otra de RL.

Conocemos tres formas de abordar el aprendizaje por refuerzo:

a) **Programación dinámica** (PD) : Basada en el **concepto de optimalidad**, considerando el aprendizaje por refuerzo como un **problema de decisión secuencial**, denominado , en este caso, un **proceso de decisión de Marckov** (MDP) Richard (Bellman R. , 1958). Con esta formalización se identifican modelos deterministas (política μ: $a_t = \mu(st)$, $s_{t+1} = f(s_t , a_t)$) y $R_t = R(s_t , a_t)$ siendo el objetivo $R_t = \sum_{i=1}^{\infty} \gamma^k r_{t+k+1}$, donde $\gamma^k = factor\ de\ descuento$) y/o estocásticos ($a_t \sim \pi(\cdot|s_t)$, $s_{t+1} \sim P(\cdot|s_t , a_t)$, $J(\pi) = \int \tau\ P(\tau\ |\pi)R(\tau\ ) = E_{\tau\sim\pi}[R(\tau)]$ y el retorno esperado $\text{argmax}_\pi\ J(\pi)$) los cuales intenta optimizar la recompensa final bien por una secuencia lineal de estados-acción o por una probabilidad del mismo.

   Esta familia de algoritmos de RL **necesita** mantener **un modelo completo del entorno** (model - based). El **agente conoce el detalle del entorno** y **todas su regles de juego** tal que el **agente puede planear con antelación las acciones a tomar** y **sus recompensa**. Y cuya formulación básica viene dada por la **fórmula de Bellman** (Bellman R. , 1958):

   - **Función Valor V^π (s):** retorno esperado para el estado s siguiendo la política Π.

$$\boldsymbol{V^{\pi}(\mathbf{s})} = \sum_{\boldsymbol{a}} \boldsymbol{\pi(s,a)} * \boldsymbol{Q^{\pi}(\mathbf{s},\mathbf{a})}$$

   - **Función acción-valor Q^π (s, a): es** el retorno esperado para 's' si se toma la

acción ‘a’ siguiendo la política Π.

$$\boldsymbol{Q^{\pi}(\mathbf{s},\mathbf{a})} = \sum_{s'} \boldsymbol{P^{a}_{s,s'}} * [\, \boldsymbol{R^{a}_{s,s'}} + \Upsilon\, \boldsymbol{V^{\pi}(\mathbf{s}')}]$$

- Siendo la **ecuación de Bellman** para dichas funciones:

$$\boldsymbol{V^{\pi}(\mathbf{s})} = \sum_{a} \boldsymbol{\pi(s,a)} * \sum_{s'} \boldsymbol{P^{a}_{s,s'}} * [\, \boldsymbol{R^{a}_{s,s'}} + \Upsilon\, \boldsymbol{V^{\pi}(\mathbf{s}')}]$$
$$\boldsymbol{Q^{\pi}(\mathbf{s},\mathbf{a})} = \sum_{s'} \boldsymbol{P^{a}_{s,s'}} * [\, \boldsymbol{R^{a}_{s,s'}} + \Upsilon \sum_{a'} \boldsymbol{\pi(s',a')} * \boldsymbol{Q^{\pi}(\mathbf{s}',\mathbf{a}')}]$$

Teniendo en cuenta que buscamos la mayor recompensa, las ecuaciones de **optimalidad de Bellman** son:

$$\boldsymbol{V^{*}(\mathbf{s})} = \boldsymbol{max_{a}} \sum_{s'} \boldsymbol{P^{a}_{s,s'}} * [\, \boldsymbol{R^{a}_{s,s'}} + \Upsilon\, \boldsymbol{V^{\pi}(\mathbf{s}')}]$$
$$\boldsymbol{Q^{*}(\mathbf{s},\mathbf{a})} = \sum_{s'} \boldsymbol{P^{a}_{s,s'}} * [\, \boldsymbol{R^{a}_{s,s'}} + \Upsilon * \boldsymbol{max_{a}}\, \boldsymbol{Q^{\pi}(\mathbf{s}',\mathbf{a}')}]$$

Notar que el **problema** de utilizar la **función acción valor** en vez de la función Valor es que se **dejan muchos espacios sin explorar**. Por ello aparece las **estrategias** $\boldsymbol{\varepsilon - greedy}$ tal que en **un % el agente explora otros posibles valores** , al margen de la acción optima. Con esta estrategia **se consiguen retornos mayores a largo plazo**. Otra estrategia, para el mismos cometido, es la de utilizar la **distribución de Boltzmann**. Esta distribción **nos permite dar probabilidades de estudio diferentes atendiendo a la temperatura del sistema y la energía del estado**. La estrategia utiliza la función **Softmax** y la **estimación relativa de la función Q**. De tal forma que cuanto mayor sea el valor Q de esa acción, mayor será la probabilidad de ser elegida. $P(a_t) = \frac{e^{EtQ(st,at)}}{\sum_{a \epsilon A_{S_t}} e^{EtQ(st,a)}}$

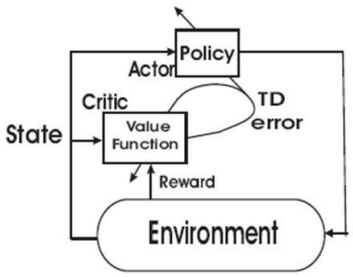
b) **Métodos de montecarlo** (MC): **Se evalua una función** valor $V(s_t)$ o acción-valor S (s_t,at) , **manteniendo una tabla con todos los pares de valores posible (s_t, a_t).** Por tanto, **no necesitan un modelo** del entorno sino que **se centran en la experiencia**. Dada la tabla de Q^{π}-values, **se identifica cual es la acción óptima en cada paso** una vez finalizado el episodio completo. En esta familia de algoritmos el agente toma decisiones por prueba y error y se va actualizando al tabla de los Q-Valores a medida que el agente avanza en su experiencia. Esto conlleva a la **necesidad de realizar numerosos episodios partiendo desde estados iniciales diferentes** y así obtenes una estimación de la recompensa esperada como la media de los diferentes estado. **EL problema de este método es que todos los estados deben ser visitados**. Además, **asume que el episodio debe finalizar para poder obtener el valor esperado**.

c) **Métodos de diferencias temporales** (TD(λ)): Podemos decir que es una **combinación de la programación dinámica** y **métodos de montecarlo**. Es decir, utilizar la recompensa obtenida en cada iteración y la evaluación de los estados vecinos. La formulación para TD(0) viene dada por la ecuación: $\boldsymbol{V(\mathbf{s})} = \boldsymbol{V(\mathbf{s})} + \boldsymbol{\alpha}\, [\boldsymbol{r} + \Upsilon \boldsymbol{V(\mathbf{s}')} - \boldsymbol{V(\mathbf{s})}]$, donde **se verifica la recompensa esperada frente a la percibida**. Una de las ventajas de este método frente a Montecarlo es que **puede reutilizar los episodios pasados** a modo de **búfer de reproducción de experiencia**.

Los métodos de **programación dinámica** se denominan métodos "**basados en modelo**" ya que un agente "imagina" las posibilidades que puede explotar con sus acciones antes de actuar. En contrapartida a los métodos "**fuera de modelo**", como son los **métodos de Monte Carlo**. En estos métodos, el agente actúa por pura intuición dada su experiencia pasada.

Además, podemos identificar otra clasificación, **dentro de los métodos de diferencias temporales**, atendiendo al **modo como actualizan su política**. Nos encontramos los método "**on-policy**", donde su mayor referente es el algoritmo **SARSA** (Whiteson, 2009), el cual **genera los episodios en base a la política actual**: $\boldsymbol{Q(s,a) = Q(s,a) + \alpha\,[r + \Upsilon Q(s',a') - Q(s,a)]}$. Y por otro lado tenemos los métodos "**off-policy**", cuyo algoritmo más utilizado es el **Q-Learning**. Los métodos "**on-policy**" se centra en la idea de **modificar la función acción valor comparando el retorno esperado con el percibido** dada una acción-estado óptimos: $\boldsymbol{Q(s,a) = Q(s,a) + \alpha\,[r + \Upsilon max_a\, Q(s',a') - Q(s,a)]}$. Las **acciones se realizan de forma aleatoria** y comprueba para dicha acción su retorno, tal que escogerá aquella acción cuyo retorno sea máximo.

Otra familia de algoritmos muy extendida son los basados en el **método del actor-crítico** (Konda, 1999). Este último método **nace por la necesidad de aplicar los modelos de aprendizaje por refuerzo en estados continuos**. Se basan en el uso de **dos algoritmos complementarios**. Por un lado, **tenemos al actor (Agente),** quien **selecciona la acción** a realizar dado el estado y cuya base es la de optimizar la política **por el método del descenso por gradiente** (Deterministic Policy Gradient (DPG)). Y, por **otro lado**, está **la crítica**, la cual **genera el valor de la función valor a partir de la actuación del actor**.



6.2. Aprendizaje Por Refuerzo Profundo. Deep Queue Network (DQN)

Los algoritmos anteriores como **Q-Learning** y **Sarsa necesitan mantener una tabla para los Q valores**, de cada uno de los estados o pares acción-estado. La cual se actualizará en cada iteración aproximándose a la política óptima del entorno. El **problema** es cuando el **número de estados y acciones es inmenso tal que dicha tabla** se vuelve inmanejable. Este hecho provoca que estos algoritmos sean **deficitarios a la hora de generalizar**, ya que **no serán capaces de predecir acciones para estados no vistos con antelación**. Para resolver este problema surgió la idea de **utilizar las redes neuronales como aproximador de funciones no lineales** para **aproximar los Q-valores** que posteriormente **se pasan a una etapa de evaluación** de la **política óptima**. E incluso **utilizar una red neuronal para calcular los parámetros de la función valor Q,** y **otra para la estimación del valor de dicha función**. Por tanto, el aprendizaje por refuerzo profundo o **Deep Reinforcement Learning (DRL) soluciona el problema de la dimensionalidad** que se producía en los algoritmos Q-learning. **Además**, permitio la introducción de los modelos de RL en problemas con **datos no**

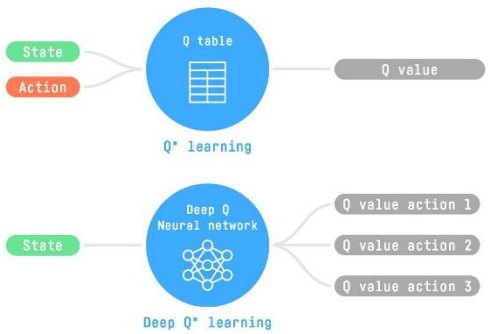


Fig 6: DL +RL : Deep Reinforcement Learning (DRL)(https://huggingface.co/blog/deep-rl-dqn)

estructurados como pasa en los problemas de **vision arificial**, **robotica** o el mundo de los **videojuegos** que fue donde comenzó todo.

Pero **los primeros pasos dentro del DLR** no dieron los frutos esperados. Si bien, la utilización de una única red neuronal para estimar los valores de la función Q (s_t, a_t) que posteriormente se pasaba a una etapa de optimización de políticas[5], dio buenos resultados. Pronto se vio, que la aproximación **conllevaba grandes problemas de inestabilidad**. En el RL, la **distribución de los datos no estacionarios** ya que, como ocurre en Q-Learning**, la política del agente se modifica en cada paso**. Además, los **valores para cada paso pueden estar correlacionados** debido al proceso de aprendizaje si tomamos los valores consecutivos. La **secuencialidad del proceso de aprendizaje**, donde las transiciones no se dan de forma independiente, **generan valores correlacionados unos con otros**. **Para evitar** todos estos problemas, una **primera aproximación** fue la expuesta por el algoritmo **Deep Queue Networks** (DQN) (Mnih et al., 2015). EL cual **toma el valor esperado para la función acción-valor como $\boldsymbol{Q(s_t, a_t) = E[R_t|s_t, a_t]}$, donde $\boldsymbol{R_t = \sum_{k=t}^{T} \gamma^k r(s_t, a_t)}$**. Y **extrapola la política codiciosa que teníamos en Q-Learning a la minimización de una función de coste**.

$$\mathrm{J}(\theta) = \boldsymbol{E}[(y_i - \mathrm{Q}(\mathrm{s}, \mathrm{a}|\, \theta))^2]$$
$$y_i = \left[\boldsymbol{r_j} + \Upsilon \boldsymbol{max_{a'}}\, \boldsymbol{Q}(\mathbf{s}', \mathbf{a}'|\, \theta')\right]$$

Y, por tanto, la función de **actualización de la función Q** queda:

$$\mathrm{Q}(\mathrm{s}, \mathrm{a}|\, \theta) = \mathrm{Q}(\mathrm{s}, \mathrm{a}|\, \theta) + \boldsymbol{\alpha}\, \boldsymbol{E}[(y_i - \mathrm{Q}(\mathrm{s}, \mathrm{a}|\, \theta))^2]$$

Esto quiere decir que para evitar la inestabilidad de la red neurona, **se opta por la utilización de dos redes en el proceso de aprendizaje**. Tal que **la primera red** (red neurona principal) **estime los valores para la función $\mathbf{Q}(\mathbf{s}, \mathbf{a}|\, \boldsymbol{\theta})$.** Los cuales se **utilizan en la función de perdida** en la **segunda red (red objetico), la cual estima los valores de la función $\boldsymbol{Q}(\mathbf{s}', \mathbf{a}'|\, \boldsymbol{\theta}')$** utilizado para **calcular el valor objetivo del siguiente estado s_{t+1}(tarjet)** de la función de pérdida de la red. El hecho de tener dos redes separadas para estimar es el uso de una segunda red durante el proceso de entrenamiento evita que la red diverga en el proceso de retroalimentación para las estmaciones de las recompensas. **Además**. **Los valores de la red objetivo no las toma siempre de la primera red**, sino que con una cierta **aleatoriedad**, los valores de la primera red **son cargados en la red objetivo**. Con esta aleatoriedad se consigue evitar tener valores correlados.

[5] El perceptron multiplayer se utilizó para desarrollar un algoritmo para jugar al backgammon en 1992 G. Tesauro, (1995)

Otra modificación del algoritmo DQN, es la utilización de un **buffer que almacena los valores de las experiencias pasadas** (***Prioritized Experience Replay***). Tal que cada periodo de tiempo escoge un lote de experiencias seleccionados al azar de dicho buffer para estimar los valores de $\boldsymbol{Q}(\mathbf{s}', \mathbf{a}' | \theta')$

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode $= 1, M$ **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do** # each step of episode
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

Experience Replay

Fixed Q-Target

Take A, observe R and S'

*1

(*1) $\theta := \theta - \alpha \frac{1}{N} \sum_{(S,A,R,S') \sim \mathcal{B}} \nabla_\theta (R + \gamma \max_a \hat{Q}_{\hat{\theta}}(S', A) - Q_\theta(S, A))^2$

Fig7: Esquema Algoritmo DQN (https://huggingface.co/blog/deep-rl-dqn)

El artículo toma como base el algoritmo DQN presentado en la fig 7 pero **con unas modificaciones**. La **primera** es una **evolución al algoritmo**, el cual utiliza como datos de entrada **valores continuos en vez de discretos**. Esto **permite introducir como entrada no sólo el conjunto de datos del estado**, sino también **la acción para tomar**. Por tanto, como entrada a la red se tendrá la **suma de la dimensión del espacio y la dimensión de la acción y genera un único valor Q**. Esto permitirá identificar **acciones más precisas**. Ya que, por ejemplo, **no sólo introduces las coordenadas en modo discreto sino también la dirección del movimiento** (fig.8).

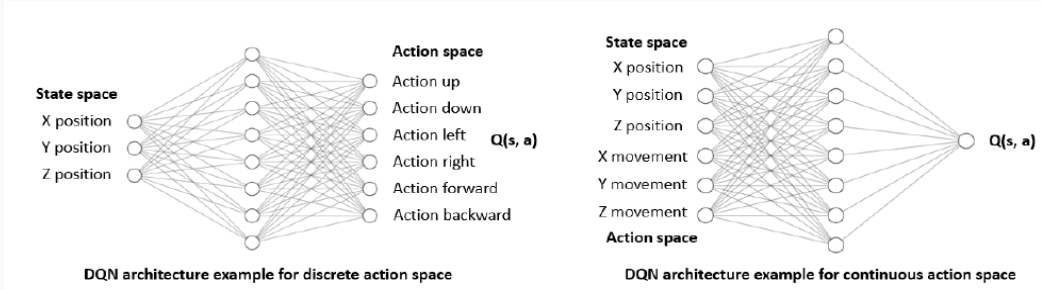


Fig 8: Arquitectura DQN espacio continuo de datos de entrada

Notar que **con valores de entrada continuos** de la acción se puede **predecir valores para la función Q mediante una distribución uniforme o mediante el método de entropía cruzada** que iterativamente **ajusta una Gaussiana para los valores de las acciones**. Tomando el **valor medio de la gaussiana para obtener el valor de Q (MCMC)**.

Otra modifcación sobre el algoritmo original es **utilizar el buffer de experiencias** mediante una **probabilida de selección**. Esta probabilidad **evitara entrenar transiciones poco relevantes con transiciones que ya están bien predichas**, . $P(i) = \frac{p_i^\alpha}{\sum_k p_i^\alpha}$ y $p_i = |\delta_i| + \epsilon$ donde ϵ es una pequeña constante positiva que evita que el caso extremo de las transiciones no se vuelva a visitar una vez que su error es cero. **A las nuevas transiciones se les asigna el mismo error TD que el más alto entre todos transiciones para incentivar su muestreo**. El exponente α determina el peso de la priorización, α = 0 correspondiente al caso uniforme.

Para evitar la sobreestimación y sobre optimismo **de las recompensas** que posee dicho algoritmo (valores-Q que aprende piensan que van a obtener una recompensa mayor de lo que en realidad obtendrá) **Hasselt propuso una modificación de la función de pérdida del DQN** para **desacoplar el valor de la acción seleccionada de la red de destino**, y que se denominó **Double DQN** (Van Hasselt, 2016). Con esta modificación, la función de actualización se queda como:

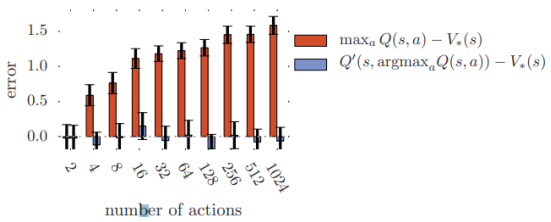


Fig9: sobreoptimismo DQN (Van Hasselt, H., Guez, A., & Silver, D. (2016, March).)

$$\mathrm{Q}(\mathrm{s},\mathrm{a}|\,\theta) = \mathrm{r} + \Upsilon \mathrm{Q}(\mathrm{s}', \boldsymbol{argmax}_{a'}\, \boldsymbol{Q}(\mathbf{s}', \mathbf{a}'|\theta\,)|\theta')$$

Quedando la función de pérdida:

$$\mathrm{J}(\theta_i) = \boldsymbol{E}\left[(\boldsymbol{r} + \Upsilon \hat{Q}(\mathrm{s}', \boldsymbol{argmax}_{a'}\, \boldsymbol{Q}(\mathbf{s}', \mathbf{a}'|\theta\,)|\widehat{\theta_i}) - \mathrm{Q}(\mathrm{s},\mathrm{a}|\,\theta_i))^2\right]$$

Es decir, **los pesos de la red principal (red θ) se reemplazan con los pesos de la red objetivo θ' para la evaluación de la política codiciosa actual**. La **actualización de la red de destino permanece sin cambios, siendo una copia periódica de la red principal**. La red principal decide cual es la mejor acción y la red objetivo evalúa dicha acción, obteniendo el valor óptimo de Q.

Otra actualización del algoritmo principal DQN es el denominado Dueling QN propuesta por Alanasary (Alansary A. e., 2019). El cual asume que la mayoría de las veces la acción realizada no importa, que **los valores Q solo son importantes en los estados clave.** Tiene **dos secuencias de capas totalmente conectadas para estimar por separado valores de estado** y las ventajas de cada acción como escalares.

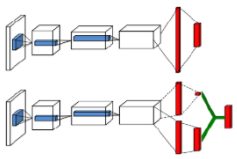


Fig 10: DQN Vs Dueling DQN

El artículo al final opta por la implementación del algoritmo Double DQN para la actualización de la función Q y las acciones a tomar. Bajo esta arquitectura base, se implementa una estrategia de múltiples agentes en colaboración. Además, realiza las siguientes asunciones:

- **Estado:** Un estado es una región de interés (ROI) que se proyecta como un voxel 45x45x45. Siendo el valor del agente el punto central del cubo. Para mejorar la estabilidad de la red y la convergencia toma como historial los últimos 4 estados.

  Cada agente comienza en una ubicación aleatoria dentro del 80% de la región interna de la imagen en el comienzo de cada episodio.

Un **agente deja de navegar cuando encuentra el hito de destino**.

Durante la inferencia, el **estado terminal** se **activa** cuando el **agente oscila alrededor de un punto objetivo**.

- **Espacio de Acción**: Se define en base a las seis direcciones en el Cartesiano 3D. El **paso se reduce atendiendo a la distancia del agente respecto al punto de referencia objetivo**. Se tienen tres niveles de paso {3, 2, 1} mm.

  El **episodio termina cuando todos los agentes llegan sus estados terminales** en la escala de 1 mm.

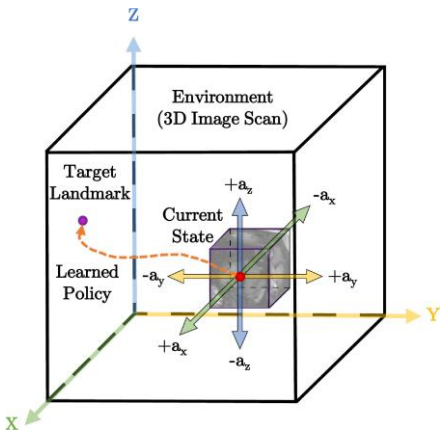


Fig11: Estado de acción del Agente

- **Recompensas**: Se calcula la distancia euclidiana entre el punto actual y el hito de destino d_t y entre el punto de interés de la paso anterior y el hito de destino d_{t-1}. Luego se calcula **la señal de recompensa**. Usando la **diferencia entre d_{t-1} y d_t,** y recortada entre −1 y 1. Esto asegura que se den recompensas positivas, si los movimientos del agente son hacia la solución objetivo.

- **Comunicación entre Agentes**: 2 tipos.
  - **Comunicación implícita:** se aprende compartiendo las capas convolucionales del modelo entre todos los agentes (Vlontzos, 2019).
  - **Comunicación explicita:** al compartir canales de comunicación en la capa totalmente conectada (Sukhbaatar, 2016). Esto se implementa promediando la salida de cada capa FC para cada agente, que luego se concatena con la entrada de la siguiente capa FC, como visto en la Fig. 2.

- **Arquitectura de red: arquitectura propuesta en el artículo de investigación** (**c-MARL**) se muestra en la figura 12.

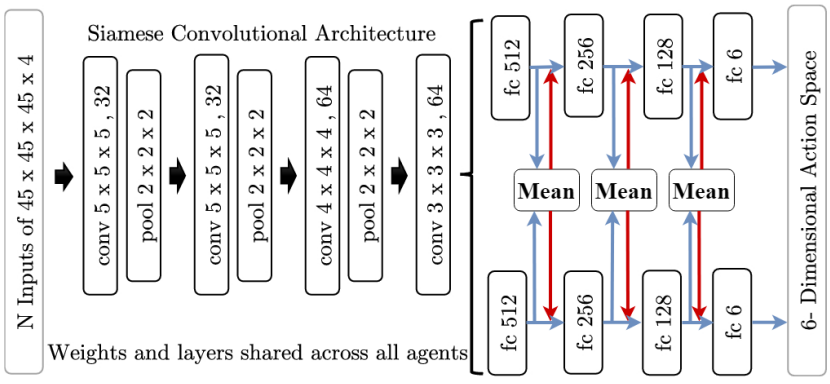


Fig 12: Arquitectura C-MARL. Leroy, G., Rueckert, D., & Alansary, A. (2020).

✓La arquitectura consta de:
- Cuatro capas de agrupación 3D convolucional
- Tres 3D máx.
- Cuatro capas FC.

- ✓ Toma como entrada un tensor de tamaño número agentes × 4 (estados pasados) × (estado actual) 45×45×45.
- ✓ Las capas convolucionales se comparten entre todos los agentes, y cada agente tiene su propia capa FC.
- ✓ La salida de todas las capas FC de cada agente se promedian y se concatenan con la entrada de la siguiente capa FC.
- ✓ El tamaño de la última capa FC es el mismo tamaño del espacio de acción.

- **El modelo se entrena minimizando la función de coste:**

$$\mathrm{J}(\theta_i) = \boldsymbol{E}\left[(\boldsymbol{r} + \Upsilon\hat{Q}(\mathrm{s}', \boldsymbol{argmax}_{\boldsymbol{a}'}\, \boldsymbol{Q}(\mathbf{s}', \mathbf{a}'|\theta)|\widehat{\theta_i}) - \mathrm{Q}(\mathrm{s}, \mathrm{a}|\,\theta_i))^2\right]$$

6.3. Aprendizaje por Refuerzo profundo Multiagente cooperativo (c-MARL)

Como se identifica en el artículo, en estudio, antes del presente **trabajo existen otros estudios que proponen otros modelos para la detección de puntos de referencia anatómicos**. Entre ellos podemos encontrar el uso de un **regression forests** (Oktay, 2016) , una **red neuronal convolucional supervisada** (CNN) (Li, 2018). En el estudio realizado por (Florin C Ghesu, 2016) se plantea la **utilización de RL para dicho cometido**. Pero es **Alansary en (Alansary A. e., 2019) quien propone emplear el modelo (DQN) y sus variantes doble DQN** (Van Hasselt, 2016) , **duelo DQN** (Wang Z. S., 2016), **y doble duelo DQN**. Por otro lado**, no es hasta la investigación de** (Vlontzos, 2019) cuando se **plantea la utilización del primer modelo multiagente para la detección de puntos de referencia**, donde **los agentes se comunican de manera eficiente compartiendo los pesos convolucionales** del modelo CNN.

Es en este estudio cunado se plantea la utilización de un modelo multiagente, fijando la arquitectura multiagente con un modelo doble DQN con datos de entrada contnuos y la utilización de un buffer donde se almacena las experiencias pasadas (Prioritized Experience Replay).

¿Pero **cómo podemos definir un modelo multiagente** y que repercusiones se tiene en el RL? Hasta ahora sólo hemos hablado de modelos donde **un agente toma una decisión sobre la acción a realizar** en un instante dado (política) **a partir de un estado actual** y una **recompensa óptima**. Por tanto, en cada instante, **el agente sólo tiene la información de su recompensa** dado el entorno (estado actual st), podemos decir que el **entorno es estacionario** y **no cambiante para el agente**. **Ahora** nos planteamos el **tener diferentes agentes** sobre el **mismo problema**, **compartiendo el entorno** y el **objetivo principal**. En este caso, en **cada instante t cada agente toma su propia decisión** (política) acorde a **una visión particular o "conjunta" de su entorno** (estado s_t^a). Por lo que en este **nuevo escenario** tenemos **diversas alternativas** según los **agentes se interrelacionan para llegar a cumplir el objetivo** del problema planteado y si **los agentes ven una porción del entorno** o **comparten la misma visión global del mismo**. **La gran diferencia** entre una situación y otra es que, en el **segundo planteamiento**, **las acciones que toman cada agente afectan al entorno global y por tanto influyen en las decisiones del resto de los agentes**. Por tanto, nos aparece un **concepto nuevo en el RL como es la interrelación entre los agentes**. Que nos lleva a **entornos puramente competitivos, entornos puramente colaborativos o entornos de interrelación mixta**. Teniendo que modificar los algoritmos base para modelizar dichas posibles situaciones. De esta nueva **situación surge una nueva familiar de modelos RL que se denominan MARL** (Multiagente reinforcement Learning), que tienen un **paralelismo intrínseco con la teoría de juegos** (Lanctot,

2017), la **lógica proposicional primer orden** y **superior** y sistemas **multiagente inteligentes**, teoría de **sistemas dinámicos complejos** y **teoría de ingeniería social**.

El principal problema de este tipo de modelos es la necesidad de **controlar la convergencia de las políticas de cada uno de los agentes**. Esto es así ya que **cada vez que un agente toma una decisión** y adopta un tipo de acción **está modificando el entorno global conjunto y** por tanto **las acciones del resto de los agentes** pueden pasar a tener una **recompensa distinta según las acciones del resto de los agentes**. Para abordar estos problemas podemos plantearnos una **arquitectura centralizada o descentralizada de las políticas**, cada una de las cuales **tiene sus ventajas e inconvenientes**. También podemos **configurar los agente**s tal que, dada una **estrategia de colaboración en el aprendizaje**, cada agente **tenga su propio objetivo** (Kim, 2020). Como vemos existe una **combinación de parámetros que llevan a configuraciones y algoritmos diferentes**. **Según el problema que estemos abordando** tendremos unas limitaciones u otras que nos indicarán que **tipo de configuración y arquitectura es más conveniente** usar. Esto conlleva a que ha **proliferado en los últimos años un sin fin de algoritmos** para resolver algunos problemas que plantea el modelo MARL, algunos ejemplos pueden ser **DRUQN** (Nguyen, 2020) , **DLCQN** (Castaneda, 2016) o **MADDPG** (Wang R. E., 2020) que intentan evitar la no convergencia y **DRQN** (Schulze, 2018) **que se centra en las observaciones parciales del buffer de experiancias pasadas**.

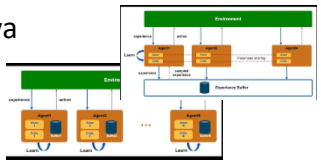


Fig13: Diferentes configuraciones MultiAgent

Existe una innumerbla base teórica en cada una de las patas que confluyen en el aprendizaje por refuerzo multiagente que **no pueden ser abordadas en este trabajo**. Si bien se ha **referenciado** en el parrafo anterior a algunos **artículos interesantes** al respeto, en la dirección https://github.com/LantaoYu/MARL-Papers y https://paperswithcode.com/task/multi-agent-reinforcement-learning **podemos encontrar diferentes trabajos muy interesantes relacionados con MARLS**, y en este otro enlace (https://github.com/markelsanz14/independent-rl-agents) **encontramos la implementación de los diferentes algoritmos comentados en el aprtado anterior**. Además, en el articulo escrito por (Littman, 1994), (Buşoniu, Multi-agent reinforcement learning: An overview. , 2010) y/o (Zhang K. Y., 2021) **. podemos localizar una revisión del estado del arte de esta familia de modelos MultiAgent-RL**.

Ahora, nos centraremos en el **modelo propuesto por el articulo**, el cual **asume un entorno conjunto en el que cada agernte colabora en el proceso de aprendizaje**. Además, como veremos posterioretne, **la configuración de los agentes se modifican para que o todos llegen a una meta comun, un punto de referencia anatomico comun. O cada agene puede aprender un punto de referencia anatomico distinto**. Es decir, **aunque exista una coalboración mutua en el aprendizaje, cada agente aprendera su politca óptima para llegar a una meta u objetivo particular**.

Como se describio en el aprtdo anterior **la arquitectura adoptada por el articulo y que se puede ver en la fig 12, adopta el algoritmo C-MARL**. En el cual mantiene **diferentes agentes que aprenden en paralelo de su entorno local, con una visión parcial del entorno. Además, actúan en colaboración**, **aportando la información de cada uno para obtener la acción optima que maximice la recompensa final**. Otra alternativa algo menos difundida es configurar múltiples competitivos[6] (Buşoniu, Multi-agent reinforcement learning: An overview. , 2010) , (Park, 2019).

[6] La utilización de múltiples agentes se puede sugerir bajo dos estrategias diferenciadas. Una estrategia de colaboración y aprendizaje mutuo como la expresada en el presente trabajo de investigación o en otros como en Zhang, K., Yang, Z., & Başar, T. (2021). En contraposición a una estrategia de competición entre los múltiples agentes (https://link.springer.com/chapter/10.1007/978-3-642-14435-6_7)

Por otro lado, **el artículo se centra en la comunicación de los agentes. En concreto, para este problema se ha utilizado el modelo CommNet** basándose en el estudio del artículo (Sainbayar Sukhbaatar, 2016) y donde **se plantea la utilización de un canal de comunicación común con una vector de comunicación continuo**. Esto produce que **un número variable de agentes pueden aprender a comunicarse mejor a través de la retropropagación**. Este método **permite eludir la necesidad de elementos de comunicación artesanal**, donde se **construyen mensajes adhoc para que los agentes se entiendan** y **aprendan a comunicarse**.

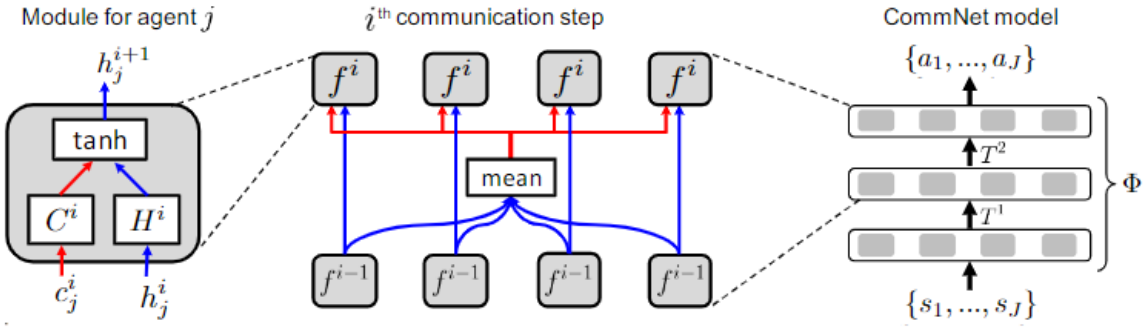


Fig14: Arquitectura CommNet.Comunicación MultiAgente

Como vemos en la figura 14, **para J agentes se generan un vector de estados** $\{s_1, ..., s_j\}$ que se mapea a un **vector de acciones** $\{a_1, ..., a_j\}$ **mediante la función lineal ϕ**, la cual se g**enera al compartir los pesos de las capas ocultas de la red de cada agente**.

El modelo y **arquitectura Multiagente c-MARL** se **comparará con una arquitectura de un único Agente**, el cual identifica los puntos de referencia anatómicos a partir de un punto inicial. Este **modelo se aproxima mediante un MDP** (proceso de Márkov parcialmente observable) con el fin de encontrar una política óptima **que nos permita identificar los puntos de referencia anatómicos que se deseen**.

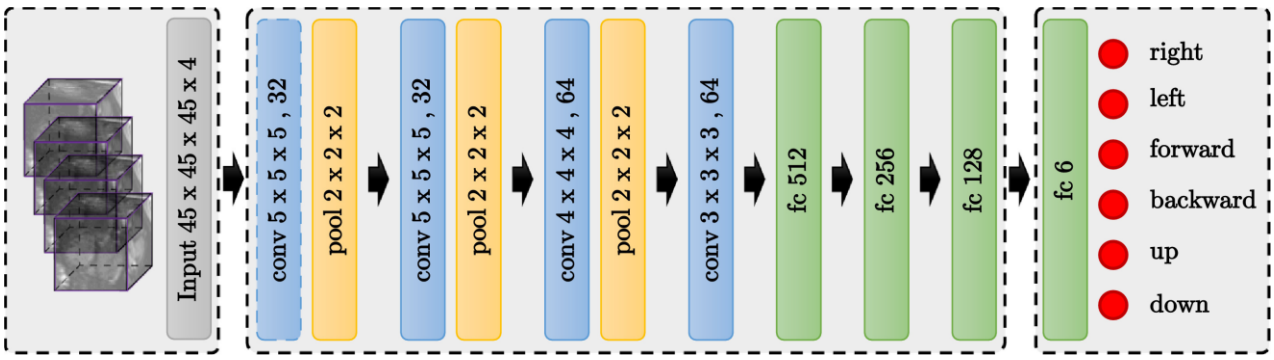


Fig15: Arquitectura Simple Agente

Este experimento **se basa en el estudio del artículo** (Florin C Ghesu, 2016) donde se utiliza un agente simple con DRL múltiple para **navegar por imágenes médicas tomando pasos**. De este artículo se saca la idea de utilizar el punto central del Voxel como las coordenadas actuales del agente. Siendo el **45x45x45 voxels la región de interés (ROI) en cada paso**. En el artículo, se demuestra que **este modo de navegar es mucho más preciso que percibir todo el entorno 3D**. Por tanto, **las acciones serán los pasos del agente**. Siendo estos pasos la i**nformación que se almacena en el buffer de experiencia reproducible.** Y que se utiliza para predecir los valores de la función Q.

El agente **recibe como entrada una ventana 3D** de s alrededor (juntamente a un historial de los 4

estados anteriores). Dicha entrada se pasa por una CNN y posteriormente por una capa completamente densa, la cual calcula como **salida las seis acciones posibles a tomar** (una por cada dirección posible del agente).

Como hemos indicado antes el agente **sigue una política codiciosa**, donde la **recompensa se evalúa atendiendo a la distancia euclídea**. Tal que si el agente **sale del entorno 3D se le penaliza con -1**. El **estado termina** cuando el agente **alcanza el punto de referencia** indicado o está en una zona cercana, **oscilando alrededor de 1500 pasos**. Para mejorar el método se ha utilizado una estrategia multiescalar, la cual aporta al agente un paso y una ventana grande y se reduce a medida que se acerca al punto de referencia. Con ello, en el experimento se demostró que **el agente simple multiescalar tiene un mejor rendimiento**. Además, durante el experimento se ha visto que **el agente es capaz de identificar puntos de referencia sin especificárselo, ya aprendido con anterioridad.** Esto hace sugerir que **el modelo llega a aprender la anatomía correcta del cerebro**.

El modelo commNet se **compara** también con una **arquitectura Multiagente inteligente** denominada **collab-DQN**. La cual se extrae del **artículo** (Athanasios Vlontzos, 2019) y que se muestra en la figura 16.

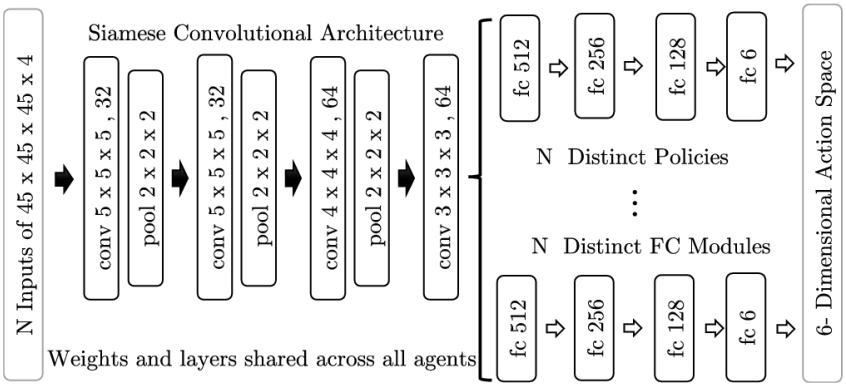


Fig16: Arquitectra MultiAgente Collab-DQN

El modelo **Collab-DQN** mantiene una **estrategia similar al modelo de un único agente**. En esta estrategia, los múltiples agentes actúan de forma independiente y por tanto se asemejan a un MPD con estados, recompensas y entorno similares. En este caso, **los agentes comparten las capas convolucionales, manteniendo independiente las capas densas.** Esto permite una comunicación indirecta entre los agentes, ya que **comparten los pesos de las capas convolucionales**. Mientras que de forma independiente cada agente aprende la política óptima para cada punto de referencia. En el **experimento se demuestra que esta arquitectura mejora la estrategia de varios agentes independientes comunicados**.

Notar la **diferencia entre esta arquitectura del modelo collab-DQN** (Fig 16) **frente al modelo C-MAR**L (fig 12), **el cual utiliza el modelo CommNet** para realizar una **comunicación explicita** en su capa densa, calculando el valor medio de dichas salidas como entradas de la segunda capa densa.

7. Experimentación y Resultados

Como hemos hecho referencia en los apartado previos se utiliza dos conjuntos de datos diferentes:

- **Eperimento I:** utiliza 32 exploraciones cerebrales de resonancia magnética ponderada en T1 de 1,5 T de la Iniciativa de neuroimagen de la enfermedad de Alzheimer (ADNI). Las imágenes ya están segmentadas y tienen un tamaño de vóxel isotrópico de 1 mm^3. Los temas seleccionados incluyen:
    - pacientes con deterioro cognitivo normal (CN),
    - deterioro cognitivo leve (DCL)
    - enfermedad de Alzheimer temprana (EA).

  Seleccionamos 8 puntos de referencia:
    - el anterior comisura (AC)
    - la comisura posterior (PC)
    - el aspecto externo
    - el inferior punta y aspecto interior del esplenio del cuerpo calloso (SCC)
    - el exterior y el aspecto interior de la Genu del cuerpo calloso (GCC)
    - el aspecto superior de pon

Los datos entregados ya están segmentados y se presenta el cerebro aislado, por lo que no se tiene ruido relacionado con otro tipo de tejidos.

Se ha dividido al azar el conjunto de datos, tal que se mantiene un 70% de los datos para la fase de entrenamiento, 15% para la fase de validación y un 15% para la fase de prueba. Notar que el código entregado solo hace distinción de dos fases, fase de entrenamiento y fase de validación.

Los resultados obtenidos respecto a las tres arquitecturas propuestas son:

	Single agent [2]	Collab-DQN [17]			C-MARL		
Landmark		3 agents	5 agents	8 agents	3 agents	5 agents	8 agents
AC	1.14 ± 0.53	1.16 ± 0.59	1.13 ± 0.64	1.21 ± 0.92	$\mathbf{1.04 \pm 0.58}$	1.12 ± 0.65	1.84 ± 0.91
PC	1.18 ± 0.55	1.25 ± 0.57	1.19 ± 0.61	1.22 ± 0.93	$\mathbf{1.13 \pm 0.66}$	1.25 ± 0.55	1.38 ± 0.64
Outer SCC	1.47 ± 0.64	1.38 ± 0.75	1.51 ± 0.77	1.46 ± 0.90	$\mathbf{1.35 \pm 0.66}$	1.62 ± 0.79	5.20 ± 13.49
Inferior SCC	2.40 ± 1.13	–	$\mathbf{1.39 \pm 0.85}$	1.53 ± 0.87	–	1.50 ± 0.89	1.87 ± 1.28
Inner SCC	1.46 ± 0.73	–	1.53 ± 0.97	2.09 ± 3.65	–	$\mathbf{1.53 \pm 0.76}$	3.56 ± 9.42

Tabla 1 Resutlados Obttenidos Experimetno I

Notar que el valor mostrado es le mejor encontrado en la fase de validación. Respecto a estos valores **se concluye que la arquitectura con múltiples agentes es mejor al de agente únicos. Y además CommNet (C-MARL) funciona mejor que el modelo collab-DQN.**

- **Experimento II: Se utilizan** 72 sujetos de ecografías de cabeza fetal en 3D del proyecto iFIND2. Todas las imágenes se vuelven a muestrear a tamaño de vóxel isotrópico con promedio dimensiones de 324 × 207 × 279 vóxeles.

  Seleccionamos el cerebelo derecho e izquierdo (RC y LC respectivamente), el cavum septum pellucidum (CSP) y el centro y cabeza anterior (CH y AH respectivamente)

puntos de referencia.

El resultado en las tres arquitecturas es:

	Single agent [2]	Collab-DQN [17]			C-MARL		
Landmark		3 agents	5 agents	8 agents	3 agents	5 agents	8 agents
RC	7.23 ± 3.54	$\mathbf{2.73 \pm 1.71}$	4.20 ± 3.76	3.39 ± 2.36	6.53 ± 4.21	4.06 ± 2.95	4.75 ± 3.28
LC	4.37 ± 1.45	$\mathbf{4.20 \pm 2.87}$	5.98 ± 8.58	5.42 ± 4.50	5.10 ± 3.66	4.43 ± 32.26	4.64 ± 3.16
CSP	9.90 ± 3.13	5.18 ± 2.05	8.02 ± 5.34	5.74 ± 5.07	5.78 ± 3.04	$\mathbf{5.13 \pm 3.51}$	7.08 ± 4.13
CH	29.43 ± 17.83	–	14.45 ± 5.25	16.83 ± 12.54	–	$\mathbf{13.00 \pm 4.97}$	16.29 ± 8.94
AH	5.73 ± 2.88	–	8.11 ± 5.22	$\mathbf{4.10 \pm 2.26}$	–	4.33 ± 2.96	8.89 ± 4.91

Tabla 2: Resutlados Obttenidos Experimetno II

Se vuelve a constatar que la arquitectura con múltiples agentes tiene un error de distancia menor que los demás, con ciertos matices para algunos puntos de referencia donde la arquitectura collab-DQN da mejores resultados.

- **Experimento III:** Los experimentos anteriores se realizan en el escenario de usar un solo agente para la detección de un punto de referencia, cada agente aprende una política para un punto de referencia diferente. En este experimento, se procede a evaluar el desempeño del uso de multiagente para la detección de mismo hito único.

  La ubicación final de los agentes se promedia al final de un episodio. Para dar una línea de base, incluimos una columna para cinco agentes individuales que buscan para el mismo hito en paralelo. El resultado indica que el modelo C-MARL da mejores resultados que los otros dos modelos.

Landmarks	Single agents [2]	Collab-DQN [17]	C-MARL
AC	0.97 ± 0.40	0.81 ± 0.36	$\mathbf{0.75 \pm 0.34}$
CSP	10.43 ± 4.28	6.66 ± 4.19	$\mathbf{5.10 \pm 4.25}$

Tabla 3: Resutlados Obttenidos Experimetno III

Por tanto, el estudio nos viene a decir que, si los agentes actúan de forma individual para obtener su propia política hacia un objetivo individual, el compartir la información de las acciones del resto de agentes, empeora la toma de decisiones individual. No aporta nada la comunicación explicita. Mientras que si ayuda la comunicación implícita, donde los agentes comparten la visión particular de su entorno local (valores para la función Q,). Mientras que, si los agentes actúan en común, hacia un mismo objetivo, el hecho de compartir la acciones individuales ayuda a la mejorar los resultados obtenidos. Algo lógico y razonable si pensamos en el hecho que siempre es mejor actuar en conjunto dentro de un grupo hacia una misma meta que actuar de forma individualista para la consecución de una misma meta.

8. Reflexiones y Modificaciones de Código

No cabe duda de que la búsqueda automática de puntos de referencia anatómicos es una tarea importantísima como fase inicial del análisis y diagnóstico por imagen, eludiendo los posibles errores humanos y facilitando dicha tarea.

La oportunidad de adentrarme en el aprendizaje por refuerzo bajo diferentes estrategias has sido una labor complicada y apasionante, en la que la falta de tiempo no me ha dejado el adentrarme más en algunos aspectos base de la técnica y matemáticas asociadas a dichos modelos. Por otra, remarcar la importancia que está teniendo hoy en día esta para del aprendizaje automático que al unirse con el Deep Learning puede ser utilizado como solución en innumerables problemas del día a día. Así lo demuestra las innumerables investigaciones y artículos que sobre todo en estos últimos cinco años, donde han proliferado algoritmos que han dado buenos resultados como el mostrado en el estudio.

Por otro lado, se echa en falta una comparativa entre este modelo DQN-Multiagente con otros tipos de modelos como los de la familia de Actor – Critico como AC3 o descenso por gradiente. Teniendo en cuenta que se ha hecho una modificación en DQN para que se puedan introducir valores continuos, y la naturaleza del modelo Actor-Crítico es dicho cometido, lo más natural hubiera sido implementar dicho modelo. En ningún lado del artículo, ni a tesis se menciona. Algo que me queda pendiente e intentare en cuanto tenga más tiempo. Entiendo, que la comparativa se centraba más en la estrategias entre los múltiples agentes para resolver un problema determinada que en la comparación con modelos diferentes. También entiendo que el hecho que el firmante sea uno de los cerebros del algoritmo Double DQN (Alansary) ha hecho que se utilice este algoritmo como base.

También no se encuentra la justificación del uso del modelo C-MARL en cuanto a su robustez frente a otros modelos o algoritmos de aprendizaje por refuerzo profundo. Hemos encontrado en esta línea un artículo más moderno (Pham, 2022) que nos habla de ello y el cual referencia por ser un artículo que puede complementar cualquier otra investigación que tenga como base dicho modelo.

Por otro lado, quiero indicar que me ha sido bastante complicado llegar a los valores que muestran las tablas del artículo. Por un lado, la configuración del entorno en los que ellos han probado los datos es muy superior al mío. Que lo he podido ejecutar con CUDA toolkit 1,6, no deja de ser un portátil que utilizo para varias cosas y no pudo dejar en exclusiva para el aprendizaje o la validación.

En cualquier caso, el aprendizaje me ha costado unos 4 días. Mientras que las ejecuciones en las validaciones pueden llegar a 20 minutos para una configuración de 5 agentes.

En cuanto al entorno, también ha sido un poco tedioso el montarlo, ya que las instrucciones no son del todo claras y faltaban librerías. Además, el hecho de no trabajar con Tensoflow (Google) sino con pytorch (Facebock) ha sido un plus más que he debido de mirarme al no haber trabajado nunca con dicho entrono. Aunque una vez montado he tenido que modificar alguna parte del código. El entorno en el cual he trabajado ha sido con

```
# CUDA 11.6
conda install pytorch==1.12.0 torchvision==0.13.0 torchaudio==0.12.0 cudatoolkit=11.6 -c pytorch -c conda-forge
```

Wheel: **Linux and Windows**

```
# CUDA 11.6
pip install torch==1.12.1+cu116 torchvision==0.13.1+cu116 torchaudio==0.12.1 --extra-index-url https://down-
load.pytorch.org/whl/cu116
```

Notar que se trabaja con pytorch y no con tensoflow. Pytotch eta desarrollado por Faceboock. En realidad, lo que Facebook está haciendo es combinar lo mejor de Caffe2 y ONNX en un nuevo

marco (PyTorch); ONNX significa Open Neural Network Exchange y es un framework interoperable al que Microsoft y AWS también contribuyen activamente proporcionando soporte para Microsoft CNTK y Apache MXNet.

En la parte de **validación ha costado más de lo que indica las especificaciones y tampoco deja muy claro que coordenadas iniciales deben tener los agentes en las diferentes configuraciones**. Por lo que no podemos identificar los valores de las tablas. Notar que en todo el articulo y en las conclusiones se afirma que **el modelo es muy sensible a los valores iniciales de los agentes**. Debes entrar en el código del script run_evaluation.py y en ese código te especifica unas coordenadas, pero para una configuración de 5 agentes. Esto hace que **las pruebas en otras configuraciones sean difíciles de reproducir**. Realmente el método utilizado es muy sensible a las coordenadas iniciales de los agentes, y si vemos que con coordenadas sin sentido los agentes no llegan a converger y se pierden, teniendo una media de error grande.

Hacer referencia a una **pequeña modificación del código** ya que el "readme.md" indica unas instrucciones para evaluar el código que no funcionan con el código presente. En concreto la variable spaw no se inicializa nunca y produce error. Es debido que han incluido un script run_evaluation.py para ejecutar la evaluación del código con los modelos entrenados por ellos. Pero que en las primeras líneas indica que no se utilice por los demás. Por tanto, se ha modificado el código en la librería DQN.py para que se pueda ejecutar sin el script run_evaluation.py

Hemos creado otro documento con los resultados y las trazas de ejecución tanto para la parte de entrenamiento como la de validación para diferentes agentes. Como nota, hacer hincapié que el aprendizaje tarda 4 días y no sé si se entrenó correctamente. Así que para la ejecución del código tal cual pone en Readme.md mejor utilizar los modelos ya aprendidos que vienen en la distribución del código.

Algo bueno del código que no he podido experimentar es todas las trazas que vienen con el entrenamiento. Permite evaluar que es lao que salió mal en cada momento. Además, se guarda información de los valores de Q va final de cada época (last_sqn.pt) y las mejores validados hasta el momento (best_dqn.pt) , etc.. y se pueden ver mediante el entorno Tensorboard. Además, se guarda los modelos en ficheros *.pt que son los que utiliza Pythorch

Además, cada vez que se ejecuta el código, se genera un log.txt y una tabla results.csv que almacena las coordenadas 3D del agente y el error de distancia al final de cada episodio.

9. Referencias

Referencias

Alansary, A. e. (2019). Evaluating reinforcement learning agents for anatomical landmark. *Med. Image Anal. 53*, 156–164.

Alansary, A. e. (2019). Evaluating reinforcement learning agents for anatomical landmark. *Med. Image Anal. 53,*, 156–164.

Anschel, O. B. (2017). Averaged-dqn: Variance reduction and stabilization for deep reinforcement learning. *In International conference on machine learning . PMLR.*, (págs. 176-185).

Athanasios Vlontzos, A. A. (2019). Multiple landmark detection using multi-agent reinforcement learning. 262–270.

Bellman, R. (1958). Dynamic programming and stochastic control processes. . 1(3):228{239).
Bellman, R. (1958). On a routing problem. Quarterly of applied mathematics, 16(1). 87-90.
Buşoniu, L. B. (2010). Multi-agent reinforcement learning: An overview. . *Innovations in multi-agent systems and applications-1*, 183-221.
Buşoniu, L. B. (2010). Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, 183-221.
Castaneda, A. O. (2016). Deep reinforcement learning variants of multi-agent learning algorithms. *Edinburgh: School of Informatics, University of Edinburgh.*
Dorri, A. K. (2018). Multi-agent systems: A survey. *Ieee Access, 6*, 28573-28593.
Florin C Ghesu, B. G. (2016). An artificial agent for anatomical landmark detection in medical images. *In International conference on medical image computing and computer-assisted interventio*, 229–237.
Gavidia, G. &.-L. (2011). Anatomía Computacional: Una Metodología Eficiente Basada en Imágenes Médicas para la Generación de Modelos 3D. doi:http://dx.doi.org/10.13140/RG.2.2.30554.54721
Gronauer, S. &. (2022). Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 895-943.
J. Lin, K. D.-G. (2020). On the Robustness of Cooperative Multi-Agent Reinforcement Learning. *IEEE Security and Privacy Workshops (SPW)*, 62-68. doi:10.1109/SPW50608.2020.00027
Kim, W. P. (2020). Communication in multi-agent reinforcement learning: Intention sharing. *In International Conference on Learning Representations.*
Konda, V. &. (1999). Actor-critic algorithms. . *Advances in neural information processing systems*, 12.
Lanctot, M. Z. (2017). A unified game-theoretic approach to multiagent reinforcement learning. *Advances in neural information processing systems*, 30.
Lazaridou, A. &. (2020). Emergent multi-agent communication in the deep learning era. arXiv preprint. doi:arXiv:2006.02419.
Leroy, G. R. (2020). Communicative reinforcement learning agents for landmark detection in brain images. (C. Springer, Ed.) *In Machine Learning in Clinical Neuroimaging and Radiogenomics in Neuro-oncology*, 177-186.
Li, Y. e. (2018). Fast multiple landmark localisation using a patch-based iterative. (G. Fichtinger, Ed.) *MICCAI 2018. LNCS, vol. 11070*, 563–571. doi:https://doi.org/10.1007/978-3-030-00928-1 64
Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. *In Machine learning proceedings*, 157-163.
McArthur, S. D. (2007). Multi-agent systems for power engineering applications—Part I: Concepts, approaches, and technical challenges. *EEE Transactions on Power systems, 22(4)*, 1743-1752.
Mnih et al. (2015). Human-level control through deep reinforcement learning. *Nature vol. 518, no. 7540*, 529-533.
Nguyen, T. T. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. . *IEEE transactions on cybernetics, 50(9),*, 3826-3839.
Oktay, O. e. (2016). Stratified decision forests for accurate anatomical landmark localization. *in cardiac images. IEEE Trans. Med. Imaging 36(1)*, 332–342.
Osband, I. B. (2016). Deep exploration via bootstrapped DQN. *Advances in neural information processing systems*, 29.
Park, Y. J. (2019). Multi-agent reinforcement learning with approximate model learning for competitive games. *PloS one, 14(9), e0222215.*
Pham, N. H. (2022). Evaluating Robustness of Cooperative MARL: A Model-based Approach. *arXiv preprint arXiv:2202.03558.*
Sainbayar Sukhbaatar, a. s. (2016). Learning multiagent communication ith backpropagation.

Advances in Neural Information Processing Systems 29, 2244–2252.
Schulze, C. &. (2018). ViZDoom: DRQN with prioritized experience replay, double-Q learning and snapshot ensembling. *In Proceedings of SAI Intelligent Systems Conference* , 1-17.
Shoham, Y. P. (2007). If multi-agent learning is the answer, what is the question?. . *Artificial intelligence, 171(7)*, 365-377.
Sukhbaatar, S. F. (2016). Learning multiagent communication with backpropagation. *In: Advances in Neural Information Processing Systems*, 2244–2252.
Tesauro, G. (1995). Temporal difference learning and td-gammon. *Communications of the ACM, vol. 38*, 58–68.
Van Hasselt, H. G. (2016). *In Proceedings of the AAAI conference on artificial intelligence Vol. 30, No. 1.*
Vlontzos, A. A. (2019). Multiple landmark detection using multi-agent reinforcement learning. (Springer, Ed.) *In: Shen, D., et al. (eds.) MICCAI 2019. LNCS, vol. 11767,*, 262–270. Obtenido de https:// doi.org/10.1007/978-3-030-32251-9 29
Wang, R. E. (2020). MADDPG for partially observable environments and limited communication. *arXiv preprint arXiv:2002.06684.*
Wang, Z. S. (2016). Dueling network architectures for deep reinforcement learning. *ernational Conference on Machine Learning,*, 1995–2003 .
Whiteson, S. (2009). A Theoretical and Empirical Analysis of Expected Sarsa.
Wooldridge, M. (1999). Multiagent systems: A modern approach to distributed artificial intelligence. 27-73.
Zhang, K. Y. (2021). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, 321-384.
Zhang, K. Y. (2021). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, 321-384.
Zhang, Y. F. (2020). Cooperative edge caching: A multi-agent deep learning based approach. *IEEE Access*, 8. doi:133212-133224

10. Enlaces/Ejemplos:

Estudio diferentes algoritmos DNQ: https://github.com/markelsanz14/independent-rl-agents

Estudio diferentes Algoritmos RL: https://tesis.usat.edu.pe/handle/20.500.12423/4948

Una implementación paralela del algoritmo de Q-Learning basada en un esquema de comunicación con caché: https://core.ac.uk/download/pdf/296349435.pdf