Universidad de Deusto
Deustuko Unibertsitatea

MUCSI - Reinforcement Learning

Proyecto final

Resolución de entorno “MiniGrid”

Unai Sainz y Lander Hernández
09-01-2023

ÍNDICE

1.- ANÁLISIS Y DETALLE DEL ENTORNO SELECCIONADO **2**

2.- ALCANCE Y OBJETIVOS **4**

3.- ANÁLISIS DE RECURSOS Y MODUS OPERANDI **5**

4.- ANÁLISIS DE RESULTADOS **11**

4.1.- Empty 11

4.2.- Lava 12

4.2.1- Lava gap 13

4.2.2.- Lava crossing 15

4.3.- Doors 18

4.4.- Dynamic Obstacles 24

5.- LÍNEAS FUTURAS DE DESARROLLO **25**

1.- ANÁLISIS Y DETALLE DEL ENTORNO SELECCIONADO

El entorno seleccionado para llevar a cabo el Proyecto Final de la asignatura es el de "MiniGrid". Se trata de una colección de entornos discretos cuadriculados, cuya resolución puede llevarse a cabo mediante la aplicación de diversas técnicas, así como mediante el empleo del algoritmo más adecuado en cada caso. Los subentornos de la colección siguen la API estándar de *Gymnasium*, y están diseñados para ser ligeros, rápidos y fácilmente personalizables.

Este entorno cuenta con un sitio web de documentación[1], así como un servidor público de discord[2], el cual es empleado para coordinar los diferentes trabajos desarrollados.

Los distintos entornos incluidos en la colección de MiniGrid, pueden ser divididos en dos categorías: Minigrid y BabyAI.

- Minigrid
  Los subentornos incluidos en esta categoría tienen su origen en la librería original de MiniGrid. Estos entornos tienen en común un agente con forma de triángulo, el cual opera en un espacio de acción discreto, sobre un mapa 2D con diferentes obstáculos (paredes, lava, obstáculos dinámicos, etc.) en función del subentorno. La misión principal del agente se basa en obtener ciertas cajas, abrir puertas tras conseguir llaves, llegar a una localización específica en el mapa, etc. En función del subentorno seleccionado. Cada entorno es configurable en términos de dimensiones y complejidad, lo cual se ha considerado útil para llevar a cabo técnicas como puede ser la de Curriculum Learning.

- BabyAI
  Los subentornos incluidos en esta categoría tienen su origen en la librería del proyecto BabyAI[3]. El objetivo de esta colección es llevar a cabo investigaciones sobre el aprendizaje de idiomas. Estos entornos son derivados de la categoría Minigrid descrita previamente, e incluyen una funcionalidad extra, la cual genera instrucciones sintéticas de aspecto natural. Por ejemplo, "coloque la bola amarilla al lado del cuadrado a su derecha". Estas instrucciones se agrupan básicamente en grupos tales como realizar el desbloqueo de puertas, o mover objetos a ubicaciones específicas.

Para los propósitos de investigación marcados en el presente Proyecto y detallados en el siguiente punto, se concluye que se llevarán a cabo simulaciones de entornos recogidos en la primera categoría descrita previamente, Minigrid, dejando de lado, para este caso concreto, las incluidas en la categoría BabyAI.

Como se ha mencionado, dentro de la categoría de Minigrid, a día de hoy existen un total de 22 subentornos, los cuales contienen una o más configuraciones registradas con OpenAI gym. Además, como se ha citado previamente, cada entorno es configurable en términos de

[1] https://minigrid.farama.org/
[2] https://discord.gg/B8ZJ92hu
[3] https://github.com/mila-iqia/babyai

dimensiones y complejidad, lo cual se ha considerado útil para llevar a cabo técnicas como puede ser la de Curriculum Learning.

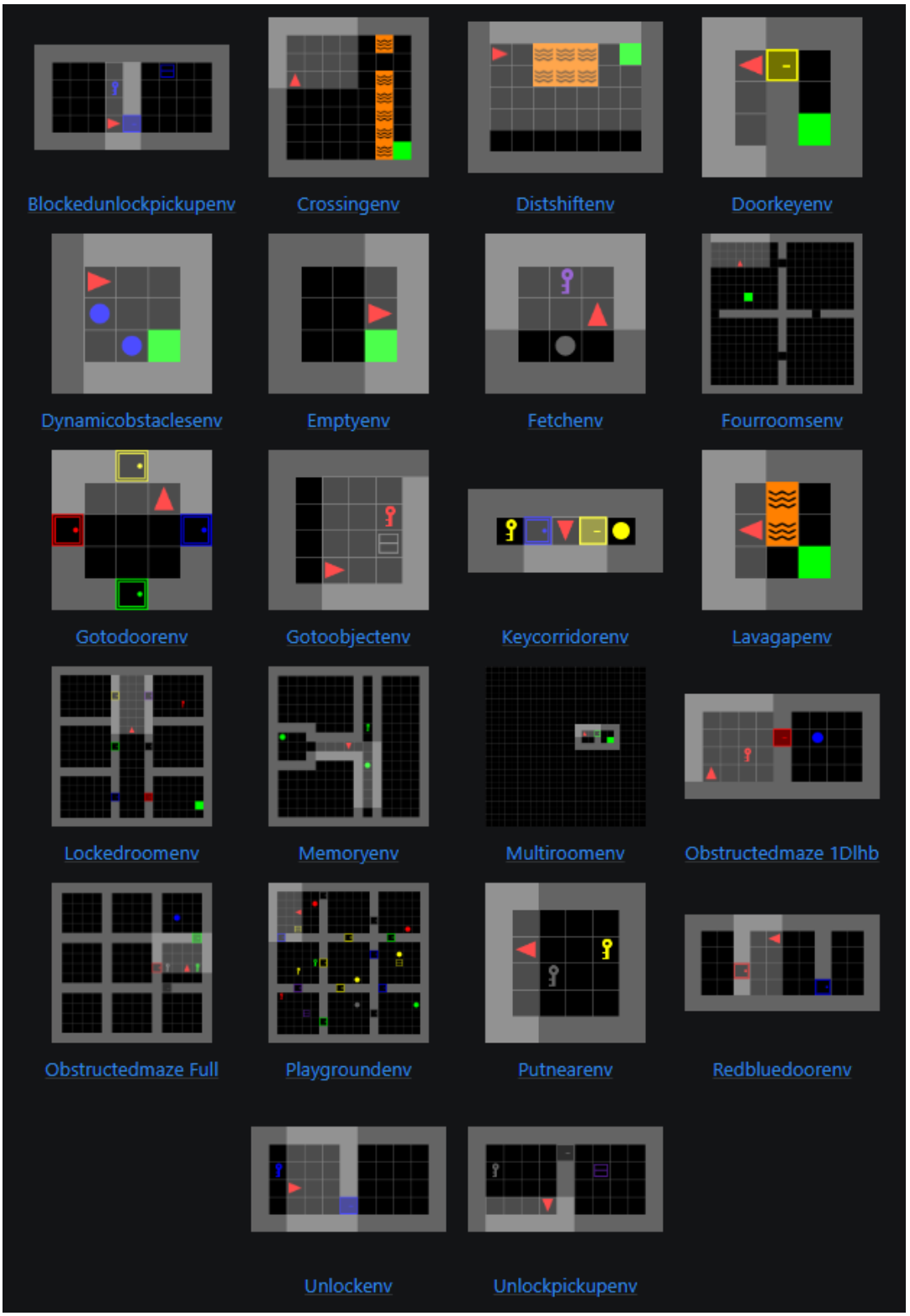

Figura 1.- Vista rápida de los distintos subentornos de MiniGrid

2.- ALCANCE Y OBJETIVOS

Como se detalla en el siguiente punto, teniendo en cuenta los diversos subentornos existentes dentro de la categoría de MiniGrid, se ha procedido a realizar una clasificación en función de, principalmente, dos factores: Misión del agente (obtener llaves, evitar obstáculos, etc.) y complejidad de resolución (ausencia de obstáculos, existencia de obstáculos dinámicos, dimensiones del entorno, etc.).

El objetivo principal de este proyecto, se basa en resolver el subentorno más complejo dentro de cada una de las misiones detectadas en estos. Es decir, para los subentornos cuya misión principal es evitar ciertos obstáculos, resolver el subentorno más complejo en este ámbito, para los subentornos cuya misión principal es obtener una llave y acceder a cierta localización, resolver el subentorno más complejo, etc.

Teniendo en cuenta que, además de lo mencionado, el hecho de que cada subentorno sea configurable ha influido en la selección del entorno, el planteamiento para lograr el objetivo principal detallado se basa en la aplicación de la técnica de Curriculum Learning. Dicha técnica implica la división del entrenamiento en varias fases. En las primeras fases se llevarán a cabo tareas más sencillas, las cuales permitirán al agente realizar una correcta comprensión y generalización del problema, para, posteriormente, en las fases finales, plantear una serie de tareas más complejas y exigentes, en las cuales el agente deberá mostrar cierto dominio para alcanzar el objetivo.

Con todo lo mencionado anteriormente, se podría concluir que el objetivo secundario más importante a lograr se trata de determinar de manera correcta la secuencia en la que las distintas misiones son exploradas. Es decir, analizar, a efectos prácticos, por ejemplo, la influencia de obtener primero un modelo capaz de resolver los subentornos referentes a la obtención de llaves y posteriormente los subentornos referentes a la llegada a localizaciones exactas o el hecho de obtenerlo de manera inversa, primero los referentes a la llegada a localizaciones exactas y posteriormente los referentes a la obtención de llaves. Dicho análisis permitirá determinar el caso en el que se obtienen resultados más favorables de cara al objetivo final a alcanzar, en comparación con el otro.

Otro de los objetivos secundarios considerados como relevantes a la hora de lograr el objetivo principal planteado anteriormente se trata de la determinación de forma acertada de la categorización en lo que respecta a complejidad dentro de cada uno de los subentornos de cada una de las misiones. Es decir, dentro de los subentornos que tienen la misma misión final, determinar de forma correcta el orden en lo referente a complejidad. Como se ha detallado anteriormente, la técnica de Curriculum Learning se basa en resolver los entornos y tareas más fácilmente alcanzables por el agente en fases iniciales y las más complejas en fases finales, por lo que el hecho de hacerlo de manera inversa dificultará, en mayor o menor medida, el aprendizaje mediante la aplicación de la técnica seleccionada.

Por último, se considera de gran importancia el hecho de tener la capacidad de determinar en qué casos tiene sentido emplear modelos entrenados para resolver ciertas misiones con intención de resolver otras, implementando un mecanismo de tipo *model-based*, en el que el aprendizaje estará más guiado por el modelo y posiblemente se llegue a una solución en menos tiempo y consumiendo un menor número de recursos. A pesar de ello, antes de

realizar el análisis de resultados, se presupone que se darán situaciones en las que el modelo final obtenido no aportará un conocimiento relevante en el proceso de resolución de otra misión diferente, por lo que la aplicación de un mecanismo de tipo *model-based* podría llegar incluso a ser contraproducente en el sentido de que el agente deberá desaprender más de lo que el modelo previamente obtenido le aporta en lo referente al proceso de aprendizaje. En ese caso se empleará un mecanismo *model-free*, en el que las experiencias del agente serán la guía del comportamiento, y no se dispone de ningún conocimiento previo del entorno ni del problema.

3.- ANÁLISIS DE RECURSOS Y MODUS OPERANDI

Para comenzar con el análisis de recursos, es importante señalar que todas las ejecuciones e instalaciones de requerimientos han sido llevadas a cabo en un servidor linux dockerizado con una imagen de linux VNC, es decir, en un software de código libre de tipo cliente servidor, que permite ver la pantalla de la máquina que actúa a modo de servidor y controlarlo en una o varias máquinas cliente, sin importar que sistema operativo pueda ejecutar el cliente o el servidor.

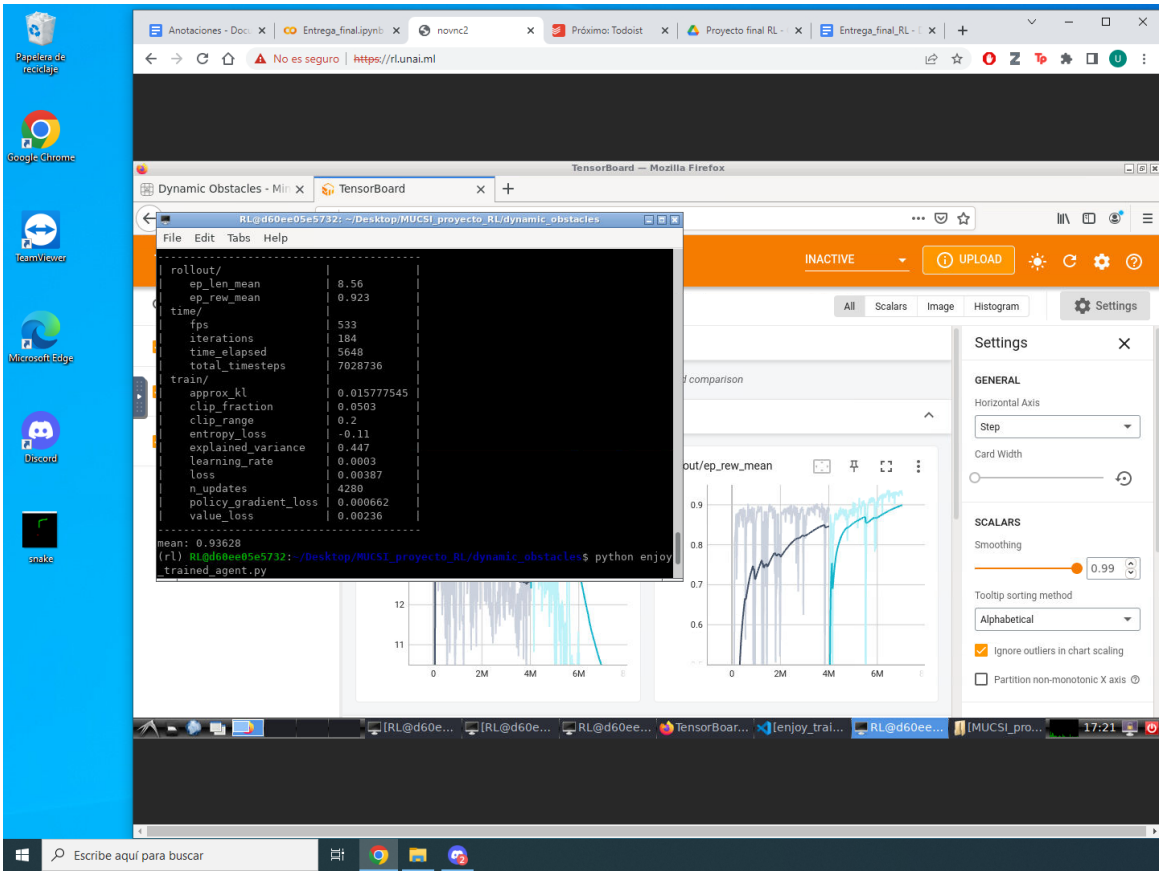

Figura 2.- Acceso web al servidor.

Como se puede ver en la imagen anterior, se ha configurado la imagen de forma que se pueda acceder desde cualquier navegador web con autenticación de usuario y contraseña. Gracias a este sencillo acceso al servidor, se ha podido comprobar en todo momento el estado del entrenamiento incluso con el equipo menos indicado para soportar estos requisitos.

Esto ha permitido, entre otras ventajas, poner en práctica la opción de que ambos integrantes del grupo puedan lanzar simulaciones partiendo de la base en la que finalizó la anterior ejecución, sin que influya el hecho de que esta fuese ejecutada por el otro integrante.

Además, la principal ventaja que aporta el hecho de realizar las simulaciones en un servidor como el mencionado anteriormente, se basa en disponer de la opción de lanzar simulaciones sin que estos procesos consuman ningún tipo de recursos de las propias máquinas de los integrantes. Este hecho ha sido catalogado como potencialmente beneficioso, tanto para procesos de computación externos a la simulación realizados de forma local, como para la propia simulación, ya que estos no compartirán recursos a pesar de ejecutarse de forma simultánea.

Por otra parte, en cuanto a los requisitos previos para llevar a cabo las ejecuciones, es importante señalar que se deben instalar los requerimientos referentes tanto a Minigrid, como a Stable Baselines 3. Para el caso de Stable Baselines 3, tras realizar ciertas pruebas sin éxito de ejecución, se consiguió llegar a una rama[4] concreta de este, que finalmente permitió realizar ejecuciones.

Con todo ello, se emplearon los siguientes comandos para llegar al objetivo, teniendo en cuenta que, al instalar Minigrid, paquetes como “gymnasium”, en su versión 0.27.0, fueron instalados. En el caso de Minigrid, se trata de la versión 2.1.0.

```
!pip install minigrid
!pip install git+https://github.com/carlosluis/stable-baselines3@fix_tests
```

Figura 3.- Comando de instalación de requisitos previos

En el caso de la instalación de Stable Baselines 3, paquetes como “gym”, en su versión 0.26.2, fueron instalados. La versión instalada en el caso de Stable Baselines 3 fue la 2.0.0a0.

A partir de la instalación de requisitos previos realizada, a fin de comenzar a realizar ejecuciones de forma organizada y con un procedimiento enfocado al alcance de objetivos principales mencionados previamente, se procedió a realizar una organización de los archivos y carpetas a crear, con intención de almacenar toda la información obtenida de las ejecuciones, para posteriormente poder obtener las conclusiones pertinentes.

Dicha organización se basa en el criterio de, principalmente, misión del agente. De esta forma, se ha clasificado entre “doors” (apertura de puertas), “empty” (subentornos vacíos), “lava” (subentornos que contengan lava como obstáculo) y “dinamic_obstacles” (subentornos que requieren que el agente aprenda a evitar obstáculos dinámicos).

[4] https://github.com/carlosluis/stable-baselines3

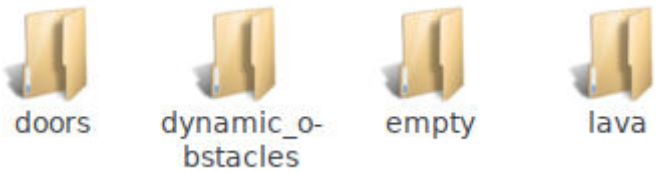

Figura 4.- Organización de carpetas general

Dentro de cada una de las rutas mencionadas, se ha generado una carpeta que contiene cada uno de los subentornos empleados para obtener el objetivo de cada misión. Por ejemplo, en el caso de “lava”, se ha generado una carpeta con el subentorno de “lava_crossing” y otra con el subentorno “lava_gap”.

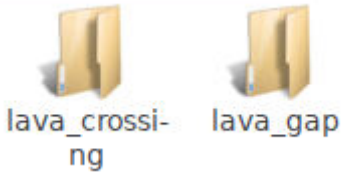

Figura 5.- Organización de carpetas dentro de cada misión

Cabe destacar que, en caso de que la misión sea alcanzada empleando un único subentorno, la carpeta se organizará directamente con la siguiente división descrita a continuación, sin hacer distinción de subentornos, como se realiza en el caso contrario.

Una vez realizada la clasificación por subentornos, dentro de cada una de las carpetas se han dispuesto las siguientes carpetas o archivos:

- logs: Carpeta que contiene los logs a partir de los cuales se podrá observar el progreso del entrenamiento mediante Tensorboard. Dentro de esta carpeta, se realiza una nueva clasificación por la configuración del subentorno empleada. Por ejemplo, para el caso concreto del subentorno “lava_crossing”, la clasificación de configuraciones quedaría dispuesta de la siguiente forma dentro de la carpeta “logs”:

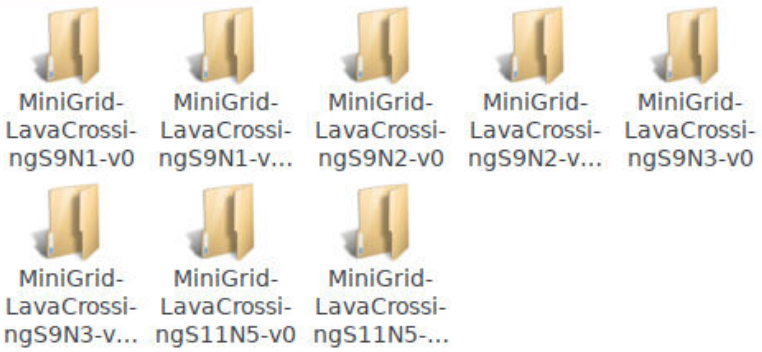

Figura 6.- Ejemplo de distribución de la carpeta logs

- models: Carpeta que contiene los modelos obtenidos. Al igual que en el caso de logs, dentro de esta carpeta, se realiza una nueva clasificación por la configuración del subentorno empleada. Cada una de las carpetas es nombrada con el prefijo del algoritmo empleado para realizar el entrenamiento. Por ejemplo, para el caso

concreto del subentorno “lava_crossing”, la clasificación de configuraciones quedaría dispuesta de la siguiente forma dentro de la carpeta “models”:

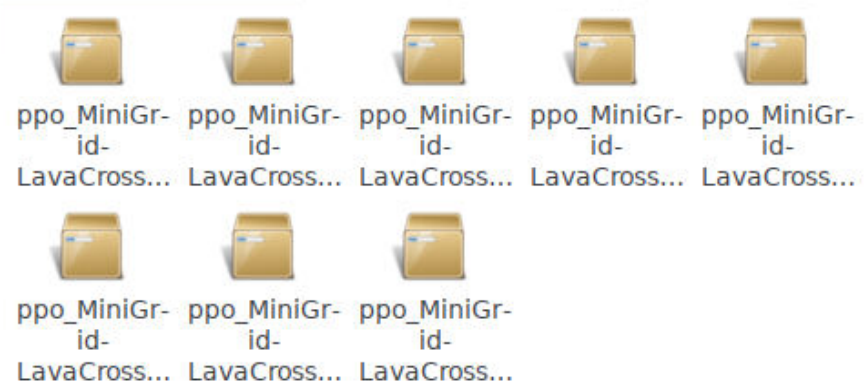

Figura 7.- Ejemplo de distribución de la carpeta logs

- enjoy_trained_agent: Script que permite lanzar el modelo y observar gráficamente la forma en la que el entorno es resuelto.

- minigrid_<mision>: Script que permite realizar el entrenamiento.

De esta forma, la distribución dentro de cada subentorno quedaría dispuesta de la siguiente manera, para el subentorno de “lava_crossing”, por ejemplo:

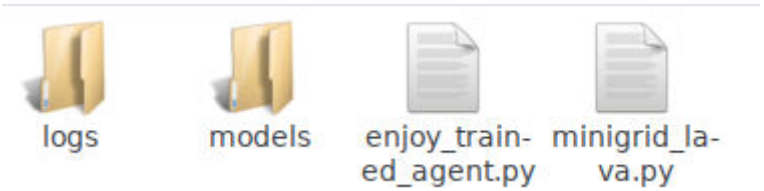

Figura 8.- Organización de carpetas dentro de cada subentorno

A modo de resumen, y a fin de comprender de manera clara y directa la clasificación realizada, a continuación se muestra un esquema de las divisiones previamente descritas:

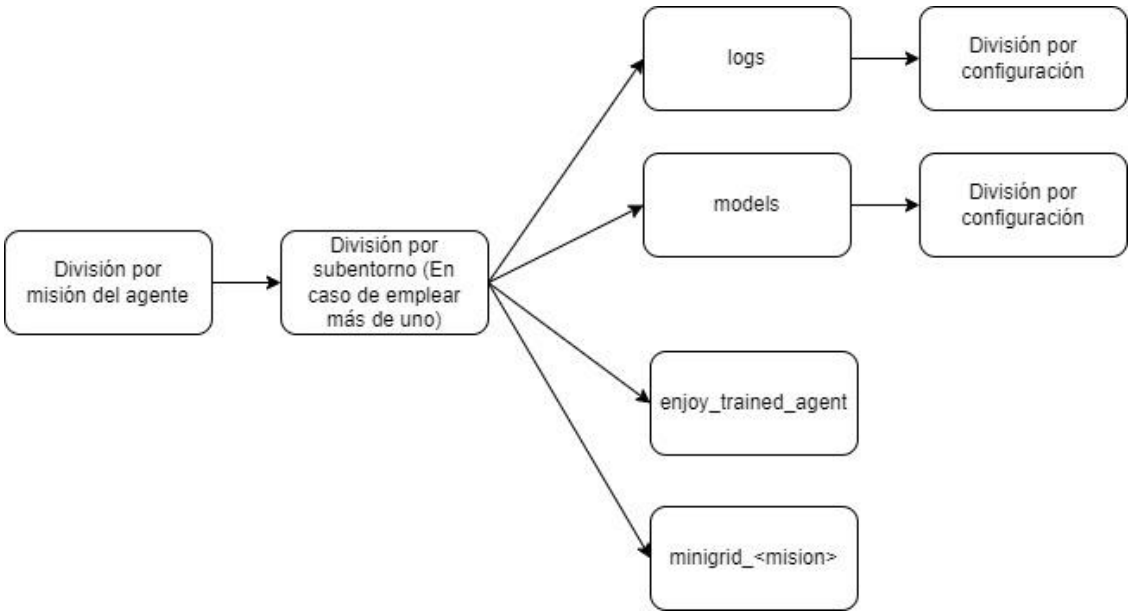

Figura 9.- Esquema de organización de las carpeta

Como se ha mencionado previamente, se va a proceder a emplear la técnica de Curriculum Learning para ir resolviendo los entornos más complejos dentro de las distintas misiones planteadas para el agente. Para ello, tal y como indica la definición de esta técnica, se comenzarán realizando tareas sencillas, para posteriormente finalizar con la realización de tareas más complejas.

De esta forma, a fin de lograr el primer objetivo final de poder resolver el subentorno más complejo dentro de la categoría de lava, el orden de las ejecuciones se ha planteado de la siguiente manera:

Para comenzar se resolverán las distintas configuraciones del entorno empty, a fin de que el agente sea capaz de reconocer que el objetivo principal es el de llegar a la celda de color verde. De esta forma, se realizarán entrenamientos con las siguientes configuraciones listadas, ordenadas de menor a mayor complejidad, orden en el que se realizarán las ejecuciones:

1. MiniGrid-Empty-5x5-v0
2. MiniGrid-Empty-Random-5x5-v0
3. MiniGrid-Empty-Random-6x6-v0
4. MiniGrid-Empty-8x8-v0
5. MiniGrid-Empty-16x16-v0

Una vez resuelto el entorno que mayor complejidad implica dentro de la misión de empty, se considera que el agente se encuentra preparado para pasar a resolver los entornos referentes a la misión de “lava”. Para resolver el entorno más complejo dentro de esta categoría, se ha planteado, primero, resolver el subentorno “lava gap”, cuya resolución se considera más sencilla que la del segundo subentorno planteado, “lava crossing”. Para resolver ambos subentornos, se ha planteado el siguiente orden de ejecución por configuraciones:

Para el subentorno “lava gap”:

1. MiniGrid-LavaGapS5
2. MiniGrid-LavaGapS6
3. MiniGrid-LavaGapS7

Para el subentorno “lava crossing”:

1. MiniGrid-LavaCrossingS9N1-v0
2. MiniGrid-LavaCrossingS9N2-v0
3. MiniGrid-LavaCrossingS9N3-v0
4. MiniGrid-LavaCrossingS11N5-v0

Una vez resuelto el entorno más complejo dentro de la categoría de lava, se plantea resolver la última de las categorías recogidas dentro del alcance del proyecto, la referente a la misión “doors”. De esta forma, para resolver el entorno más complejo dentro de esta categoría, se ha planteado, primero, resolver el subentorno “door key”, cuya resolución se considera más sencilla que la del segundo subentorno planteado, “multiroom”. Para resolver ambos subentornos, se ha planteado el siguiente orden de ejecución por configuraciones:

Para el subentorno “door key”:

1. MiniGrid-DoorKey-5x5-v0
2. MiniGrid-DoorKey-8x8-v0
3. MiniGrid-DoorKey-16x16-v0

Para el subentorno “multiroom”:

1. MiniGrid-MultiRoom-N2-S4-v0
2. MiniGrid-MultiRoom-N4-S5-v0
3. MiniGrid-MultiRoom-N6-v0

Una vez desarrollados los modelos recogidos dentro del alcance del proyecto, los referentes a las misiones de “empty”, “lava” y “door”, se plantea, a modo de extra, resolver el subentorno referente al agente cuya misión principal se basa en evitar objetos dinámicos.

Este objetivo, como se ha mencionado, se ha planteado a modo de añadido, ya que no partiría de ningún modelo desarrollado previamente, como sí ha sucedido con el resto de subentornos. El hecho de que el entrenamiento para resolver el subentorno detallado se realice sin partir de un modelo base, se debe, principalmente, a que el espacio de acciones de este subentorno es distinto al resto.

Para resolver el subentorno “dynamic obstacles” se plantean resolver sus configuraciones en el siguiente orden:

1. MiniGrid-Dynamic-Obstacles-5x5-v0
2. MiniGrid-Dynamic-Obstacles-Random-6x6-v0

En este caso, teniendo en cuenta que existe un mapa cuyas dimensiones son de 16x16, se puede concluir que el objetivo principal es distinto al del resto de subentornos resueltos, y en este caso, no se basa en conseguir resolver la configuración más compleja dentro de dicha misión, sino que, por el contrario, únicamente se busca observar la forma en la que el agente realiza el entrenamiento y logra conseguir resolver un entorno dinámico, en continuo movimiento.

4.- ANÁLISIS DE RESULTADOS

A continuación se va a realizar un análisis de los resultados obtenidos tras las distintas ejecuciones realizadas sobre los subentornos y configuraciones mencionadas previamente. Gracias a estos análisis, se podrán obtener conclusiones sobre los aspectos planteados en el apartado referente a los objetivos, como puede ser, por ejemplo, la adecuación de la técnica de Curriculum Learning para cada uno de los subentornos, el orden de ejecución planteado, la curva de aprendizaje de los modelos, etc.

Para realizar dicho análisis, a continuación se hace un resumen de los resultados obtenidos en cada subentorno, teniendo en cuenta el orden de ejecución planteado en el apartado anterior.

4.1.- Empty

El primer paso ha sido comenzar a entrenar un agente que aprenda a dirigirse al objetivo final, común en todos los mapas, un bloque verde. Para ello se han usado los mapas “empty”. Los nombres de las configuraciones describen de la siguiente manera la distribución:

a. `MiniGrid-Empty-5x5-v0`. Tablero de 5x5.
b. `MiniGrid-Empty-Random-5x5-v0`. Mismas dimensiones con spawnpoint del agente random.
c. `MiniGrid-Empty-Random-6x6-v0`. Mismo concepto con mayor tamaño de tablero (6x6).
d. `MiniGrid-Empty-8x8-v0`. Tablero aún más grande con spawnpoint estático. El agente debe aprender a guiarse sin ver el target desde el inicio.
e. `MiniGrid-Empty-16x16-v0`. Mismo concepto aumentando la dificultad/tamaño.

Todos los entrenamientos se hacen hasta llegar a los valores máximos de cada mapa. Por ejemplo, para el caso de la configuración de 16x16 (ver Figura 10), se aprecia un claro valor límite que no se puede superar.

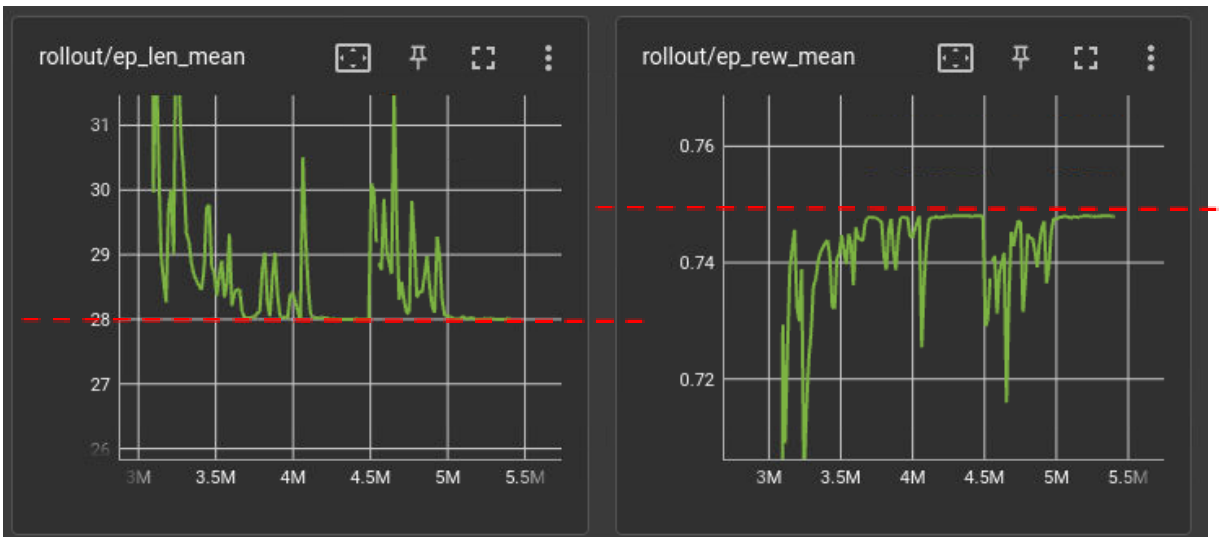

Figura 10.- Ejemplo de detención de entrenamiento en punto de saturación.

Se ha entrenado un solo modelo de forma secuencial en el orden listado anteriormente. Los resultados del entrenamiento se muestran en la Figura 11.

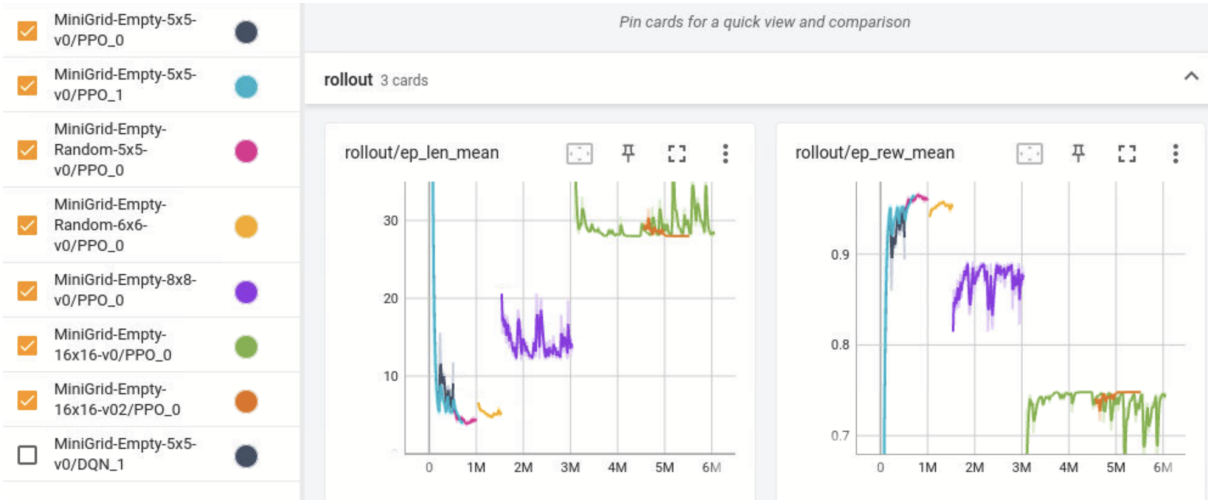

Figura 11.- Entrenamiento del modelo base "empty" mediante Curriculum Learning.

Como se puede observar en la figura anterior, hay ciertos huecos entre los entrenamientos. Esto es debido a los nuevos tamaños de los mapas y dificultades asociadas, que alargan los episodios.

Una vez logrado el modelo capaz de identificar el cubo verde e ir hasta él, es usado como base para el resto de entrenamientos.

4.2.- Lava

El objetivo de este apartado es conseguir un modelo capaz de resolver el mapa de "Lava Gap" y "Lava Crossing" partiendo del modelo desarrollado en el apartado anterior, donde se

ha entrenado a un agente en un entorno sencillo de manera progresiva a explorar hasta encontrar y alcanzar la caja verde.

Al ser la primera vez que se aplica la técnica de Curriculum Learning, y teniendo en cuenta que el entorno actual no se trata de un entorno excesivamente complejo, se ha visto conveniente replicar esta misma operación de forma paralela, llevando a cabo de manera simultánea una ejecución que no parta de un modelo base. Dentro de cada escenario, independientemente del modelo, todos parten de la dificultad anterior. Es decir, en este capítulo se van a crear 2 modelos por cada mapa: uno de ellos habiendo sido entrenado en los entornos “empty” y otro sin entrenamiento previo. Se pretende conocer las ventajas y desventajas del Curriculum Learning de forma práctica, así como analizar qué modelo da mejores resultados en términos de tiempo de entrenamiento y rendimiento al finalizar el periodo de computación.

Se presupone que la ejecución que parte de un modelo base tomará a éste como guía, por lo que el agente será capaz de predecir el siguiente estado teniendo en cuenta el actual y la acción a tomar.

En el caso en el que no se toma ningún modelo base como guía para la ejecución, el agente parte de una hoja en blanco, por lo que será a partir de la propia experiencia de la que éste realizará el aprendizaje. Este hecho, en principio, proporcionará al agente la capacidad de explorar soluciones más creativas.

4.2.1- Lava gap

En la Figura 12, mediante las gráficas de TensorBoard, se puede observar el entrenamiento que se ha seguido. La línea negra muestra el modelo con entrenamiento previo en los mapas “empty”, mientras que la azul nunca ha visto esa configuración. Ambos modelos se han entrenado hasta 1,5 M de iteraciones (nótese la línea roja discontinua), punto donde se han guardado los modelos y se han vuelto a cargar para seguir su entrenamiento en el siguiente mapa de mayor dificultad.

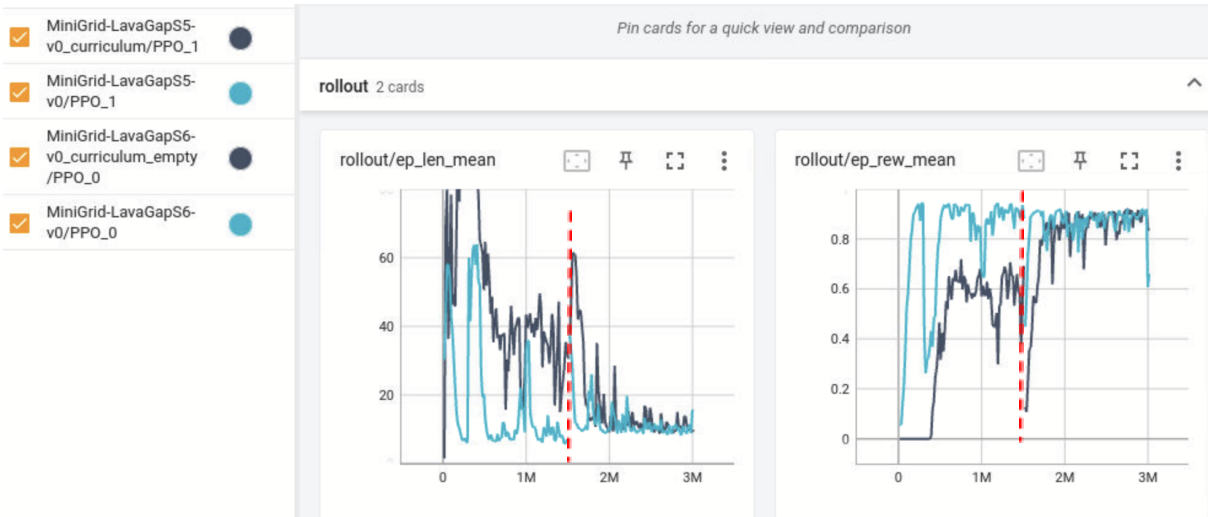

Figura 12.- Entrenamientos con y sin Curriculum Learning para Lava Gap.

Como se puede observar, los dos modelos han logrado llegar al mismo punto al final de los dos entrenamientos. Sin embargo, el modelo sin curriculum learning ha aprendido más rápido, e incluso mejor, para el caso de “MiniGrid-LavaGapS5” (cuyo entrenamiento acaba en 1,5 M de iteraciones, previo a aumentar la dificultad con el siguiente mapa). La caída de recompensa en el cambio de mapa también es mucho menor para el modelo entrenado con Curriculum Learning.

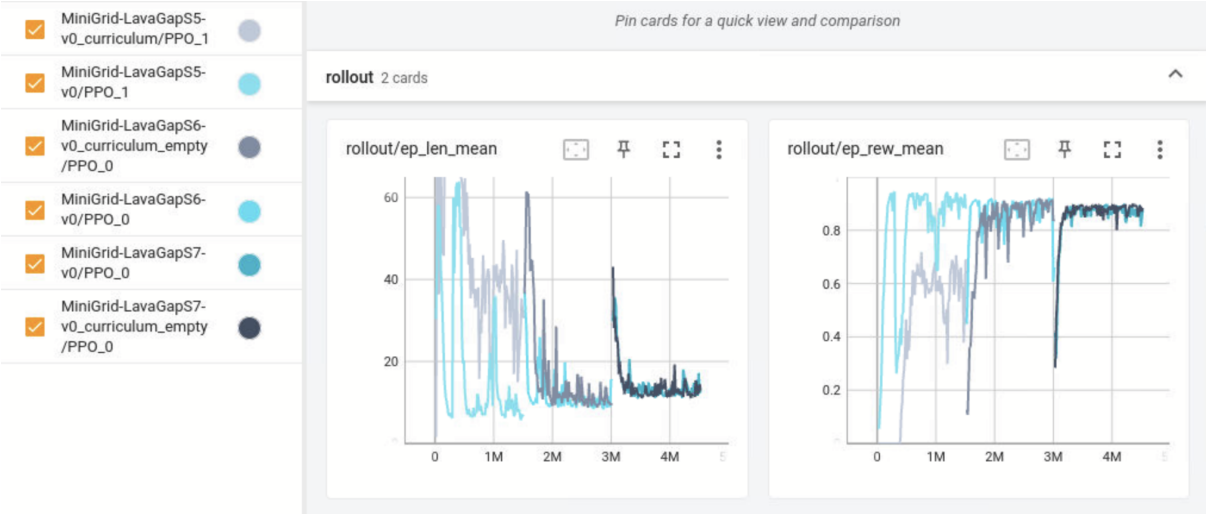

Figura 13.- Entrenamientos con y sin Curriculum Learning finalizados para Lava Gap.

Al analizar los modelos (ver Figuras 12 y 13), se puede apreciar que la media de reward en la evaluación del modelo con curriculum learning es notablemente mayor que el modelo sin aprendizaje previo.

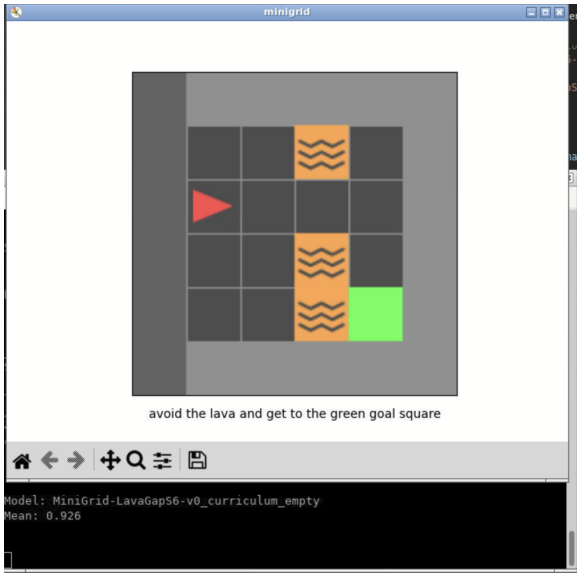

Figura 14.- Análisis del modelo con Curriculum Learning.

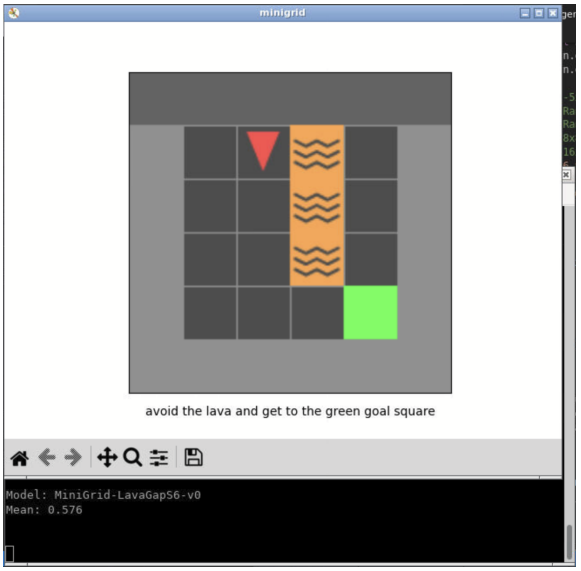

Figura 15.- Análisis del modelo sin Curriculum Learning.

Tras finalizar por completo el entrenamiento para las configuraciones de Lava Gap, se puede apreciar en la Figura 14 y 15 que, el Curriculum Learning utilizado para saltar entre configuraciones del mismo juego ha sido todo un éxito. Sin embargo, el inicio resulta algo más lento para el modelo con aprendizaje previo, resultado totalmente inesperado. Si que es cierto que, al inicio, para la línea negra (modelo con Currículum), por mucho que tenga cierta latencia en aprender, este progreso es mucho más robusto y estable que su competidor.

4.2.2.- Lava crossing

Una vez obtenido un modelo que sabe trabajar con la lava y moverse en un entorno pequeño hasta llegar al objetivo, se intenta subir el nivel de dificultad mediante los mapas de “Crossing”. Se ha seguido con la metodología de Curriculum Learning para hacer un entrenamiento completo a lo largo de todos los mapas disponibles de “Crossing”. Dentro del entrenamiento, para seguir explorando las capacidades de Curriculum Learning, se vuelven a hacer dos modelos: uno de ellos sin base previa, y otro partiendo del modelo desarrollado en el punto anterior.

Para estas configuraciones, la variable N indica el número de filas/columnas de lava, mientras que el parámetro S sigue indicando el número de celdas por cada dimensión del entorno.

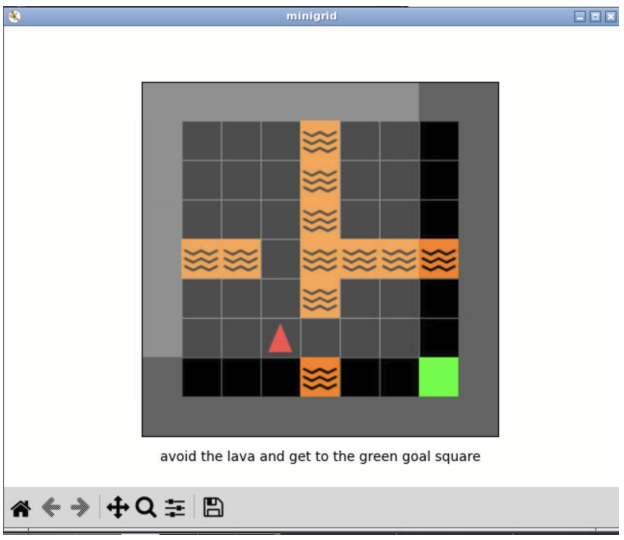

Figura 16.- Lava Crossing N=2, S=9

A medida que se han ido creando modelos, se han ido probando, tanto en los mapas en los que se han entrenado, como en configuraciones de mayor dificultad. Por ejemplo, se ha testado el modelo “MiniGrid-LavaCrossingS9N2-v0” en el entorno “MiniGrid-LavaCrossingS9N3-v0”, de mayor dificultad, y el agente ha sabido resolverlo. Aun así, se ha seguido entrenando para obtener un modelo más robusto. Lo mismo ha sucedido con el S11N5 usando el modelo entrenado en S9N3, como se muestra en la Figura 17.

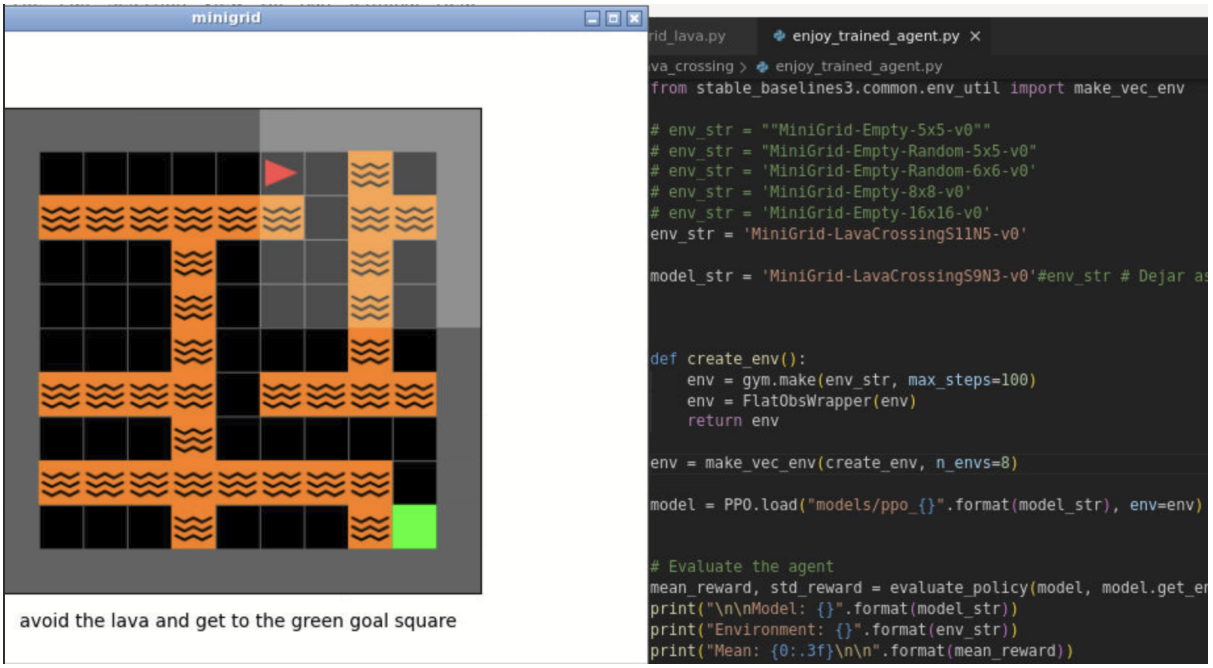

Figura 17.- Lava Crossing N=5, S=11

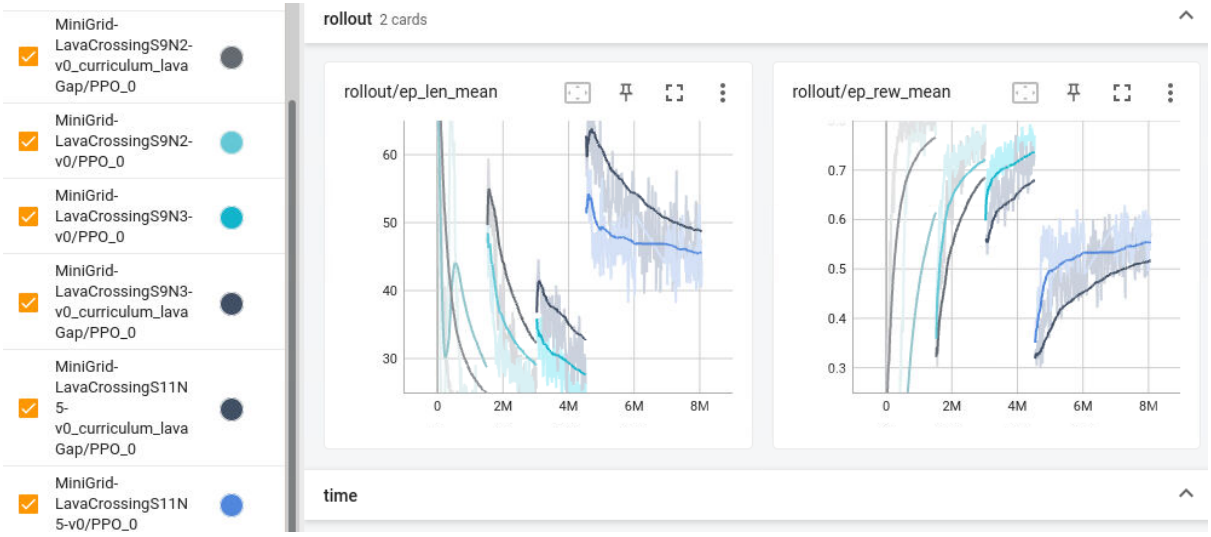

Figura 18.- Lava Crossing N=5, S=11

El entrenamiento completo para Lava Crossing se puede ver en la Figura 18. Se aprecian perfectamente las distintas etapas de entrenamiento, pues entre configuración y configuración, hasta que el agente empieza a aprender en el nuevo mapa al que se expone, hay una pérdida de reward y alargamiento de episodios (también porque el tablero es más grande). Esta pérdida se manifiesta con una discontinuidad en la línea de aprendizaje en el gráfico.

Las líneas azules son las correspondientes al modelo entrenado únicamente en Lava Crossing, mientras que las líneas oscuras parten del modelo desarrollado en Lava Gap. Como se puede apreciar, no hay gran diferencia de resultados entre los dos modelos. De hecho, el modelo sin aprendizaje previo se ha comportado algo mejor, a excepción de la etapa inicial donde ha tardado más en aprender y lo ha hecho de forma más inestable.

Como conclusión se puede obtener que, tanto en Lava Gap, como en Lava Crossing, se han adoptado técnicas de Curriculum Learning, bien para subir la dificultad entre mapas distintos como para un mismo mapa. Está claro que aplicar esta técnica dentro de un conjunto de mapas iguales es una clara fortaleza para que el modelo llegue a la máxima dificultad sin problemas. Sin embargo, el salto entre mapas no ha dado tan buenos resultados. Esto se debe a que las reglas del juego entre estos mapas cambian ligeramente. El agente, por mucho que sepa moverse, tiene que “transformar” el aprendizaje anterior a un nuevo paradigma, es decir, el aprendizaje no es del todo incremental y eso “confunde” al agente y alarga el tiempo de aprendizaje.

En otras palabras, el Curriculum Learning se ha comportado de manera excelente para un conjunto de entornos en los que las reglas del juego no cambian, sólo la dificultad o configuración.

4.3.- Doors

El objetivo de este capítulo es lograr completar 2 entornos distintos de Doors. El primero de ellos, “Door Key” y el siguiente “Multi Room”. Esta vez no se realiza la comparación entre aplicar Curriculum Learning o no, ya que se han obtenido las respectivas conclusiones en las anteriores simulaciones. En este apartado se aplicará esta técnica, tanto para los saltos de dificultad entre mapas, como para el inicio del entrenamiento.

Partiendo de la base “MiniGrid-Empty-Random-6x6” se ha seguido el entrenamiento en las configuraciones “MiniGrid-DoorKey-5x5”, “MiniGrid-DoorKey-8x8” y “MiniGrid-DoorKey-16x16” consecutivamente, tal y como se puede apreciar en la Figura 19.

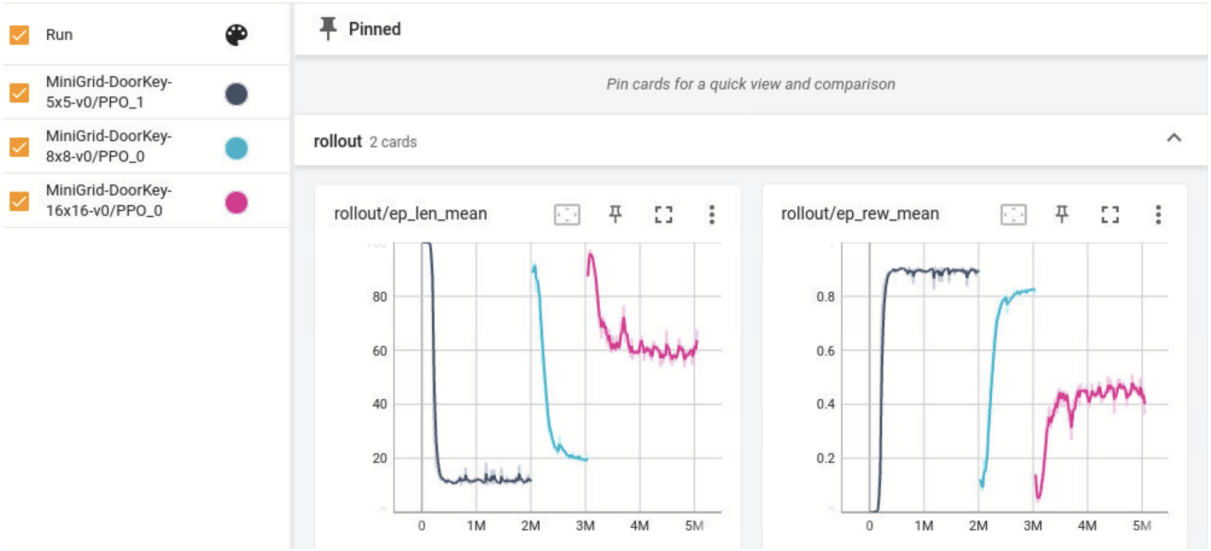

Figura 19.- Estadísticas de entrenamiento para Curriculum Learning en MiniGrid-DoorKey.

Como viene siendo habitual, al cambiar de entorno y al ampliar el tablero, los episodios duran más y la recompensa obtenida es menor. Aun así, se ha logrado completar el entrenamiento hasta resolver la configuración más complicada.

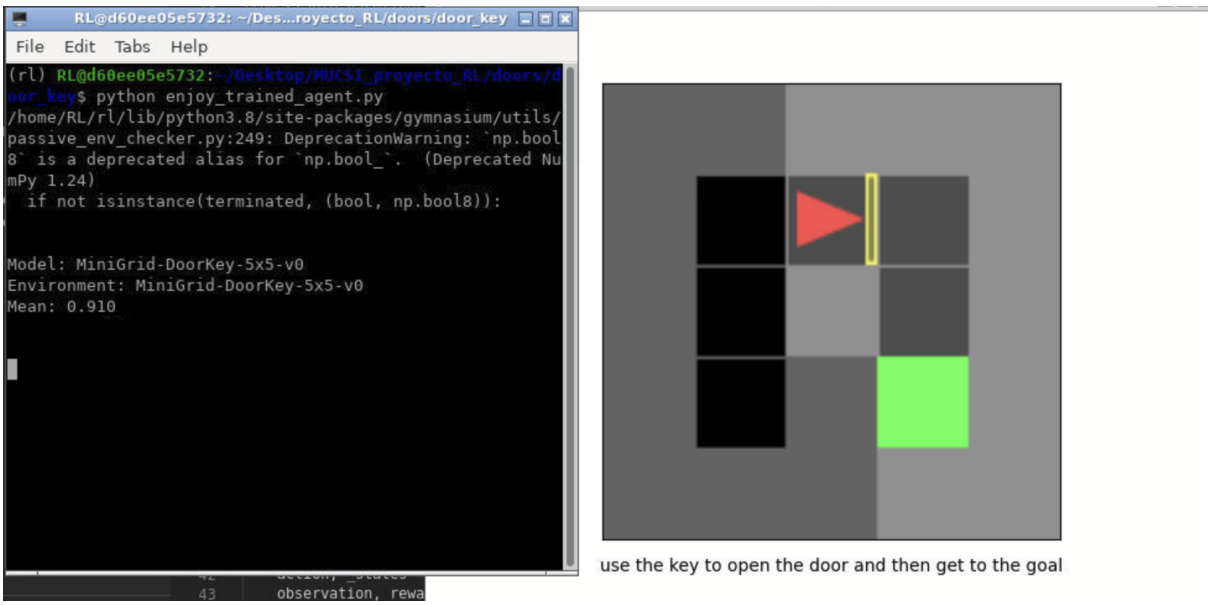

Figura 20.- Modelo desarrollado en MiniGrid-DoorKey.

Tras la resolución del entorno anterior (ver Figura 20), como se ha detallado previamente, se plantea resolver la configuración más complicada del proyecto, Multi Room. De hecho, como se puede apreciar en la Figura 21, la propia documentación de MiniGrid avisa de que es extremadamente difícil resolver el entorno sin usar Curriculum Learning.

Description

This environment has a series of connected rooms with doors that must be opened in order to get to the next room. The final room has the green goal square the agent must get to. This environment is extremely difficult to solve using RL alone. However, by gradually increasing the number of rooms and building a curriculum, the environment can be solved.

Figura 21.- Explicaciones de la documentación oficial de MiniGrid para Multi Room.

Se ha seguido desarrollando un Curriculum Learning, partiendo del aprendizaje “empty”, que ha dado muy buenos resultados, rápidos y sólidos para los 2 primeros mapas. En cuanto se ha continuado con el entrenamiento para resolver el último entorno, la media obtenida ha sido de 0.

Para hacer frente a este problema, se ha determinado que subir el número máximo de pasos podía ser una buena solución, ya que, al ser un mapa con tantas habitaciones y tanto espacio por recorrer, es posible que el agente llegue al límite sin poder llegar al final.

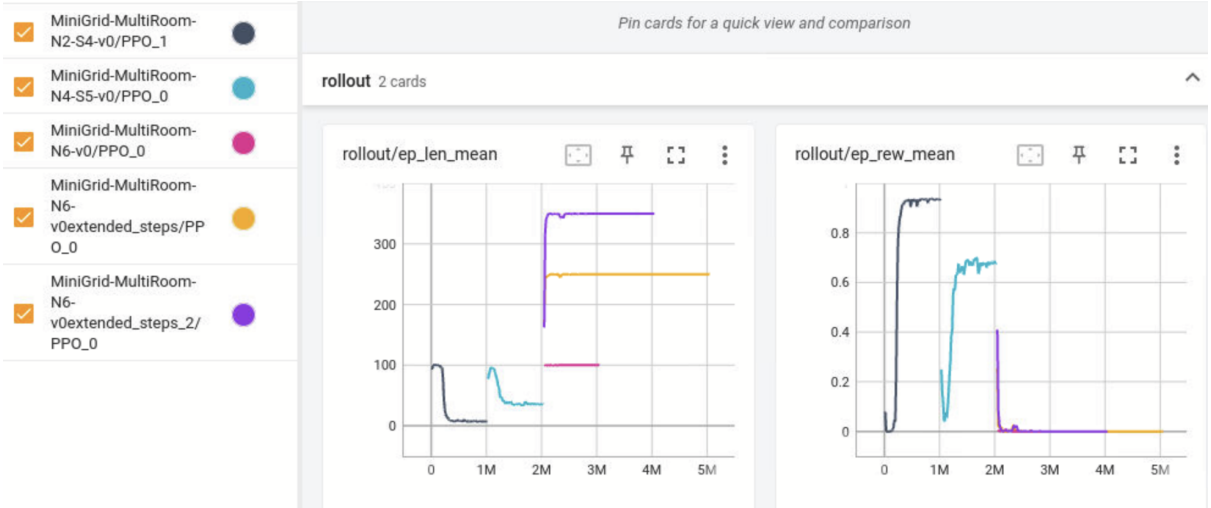

Figura 22.- Logs de entrenamiento en TensorBoard para los intentos con PPO.

Como se puede observar en la Figura 17, los 2 primeros entrenamientos han dado buenos resultados, mientras que los siguientes, todos para el último entorno, solo han conseguido consumir todos los pasos configurados para esos intentos.

En este punto se decide probar con el algoritmo DQN, en lugar de emplear el PPO. Al no ser posible reutilizar los entrenamientos anteriores, se comienza de nuevo sin base previa.

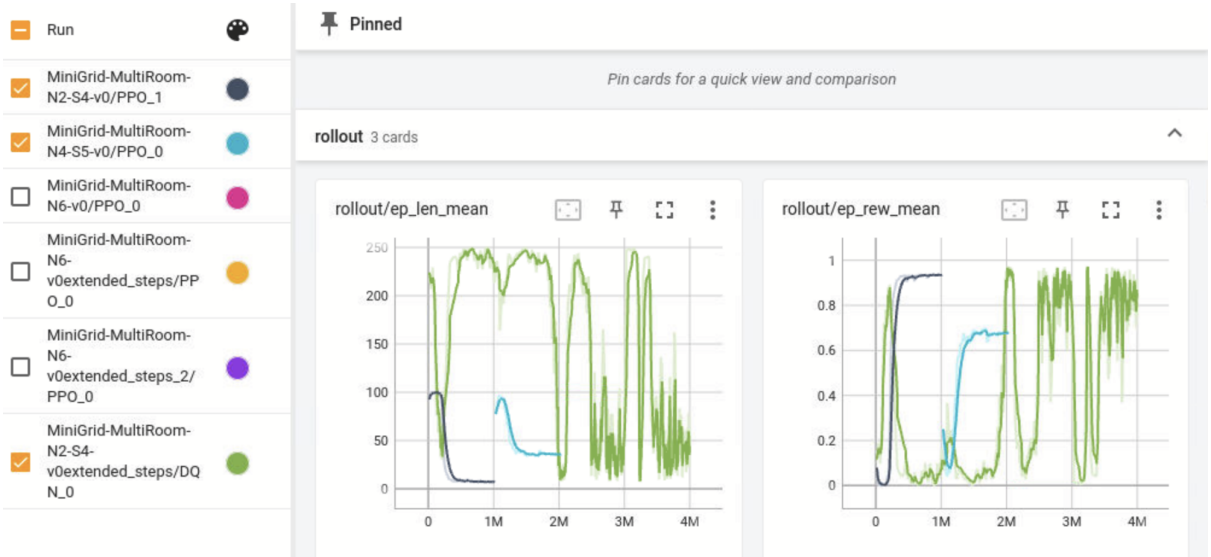

Figura 23.- Logs de entrenamiento DQN vs PPO.

Como se puede observar en la Figura 23, el entrenamiento ha requerido de más tiempo para llegar a unos resultados similares del equivalente en PPO. Sin embargo, a la hora de evaluar el agente, su calificación media es de 0.009, y éste se queda quieto delante de la puerta una vez abierta.

Figura 24.- Evaluación modelo DQN.

Tras este intento fallido, se concluye que es posible que el agente no haya aprendido que el objetivo es llegar al cubo verde a través de las puertas. De ahí que se decide crear un modelo base del que el agente pueda partir para repetir el entrenamiento.

Figura 25.- Comparación de creación de modelo Empty con DQN vs PPO.

La Figura 25 muestra el resultado del proceso de entrenamiento, comparando el aprendizaje mostrado anteriormente con el algoritmo PPO y el actual con DQN. Como se puede observar, DQN ha requerido de muchísimos más episodios para llegar al mismo punto que los modelos con PPO. En cualquier caso, se ha obtenido el modelo empty mediante DQN con éxito.

Una vez obtenido el modelo base con el que poder hacer curriculum learning usando DQN, se procede a ello.

La Figura 26 muestra los resultados obtenidos durante el entrenamiento. Las líneas negras son las mostradas en la Figura 23, es decir, los entrenamientos con PPO. Las líneas verdes son los entrenamientos con DQN. Siguiendo el patrón anterior, los nuevos entrenamientos han tardado mucho más en llegar a algún resultado, además de necesitar un número de pasos mayor para poder efectuar el entrenamiento correctamente.

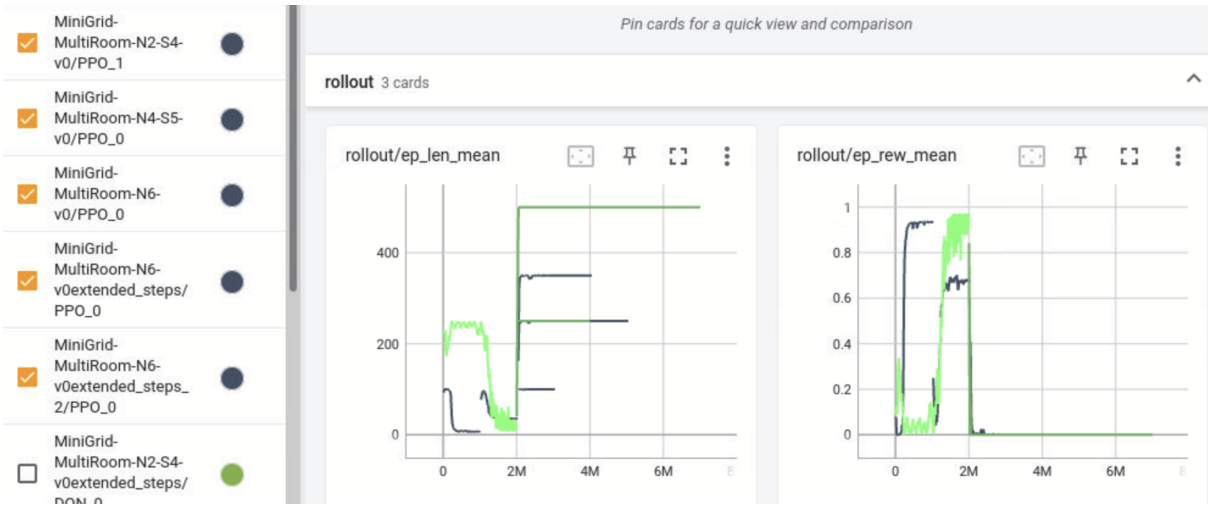

Figura 26.- Comparación entrenamiento mapa final con DQN vs PPO.

Como se puede observar, el entrenamiento con DQN ha sido todo un fracaso, al no poder ni igualar los resultados de PPO. El algoritmo no ha sido capaz de resolver el entorno de 4 habitaciones, cosa que con PPO si ha sido posible. A pesar de jugar con parámetros de número máximo de pasos e iteraciones de entrenamiento, se ha abandonado el intento al considerarse un derroche computacional.

En cualquier caso, se ha logrado un modelo con PPO capaz de resolver un entorno con 4 habitaciones de 5x5 (ver Figura 27).

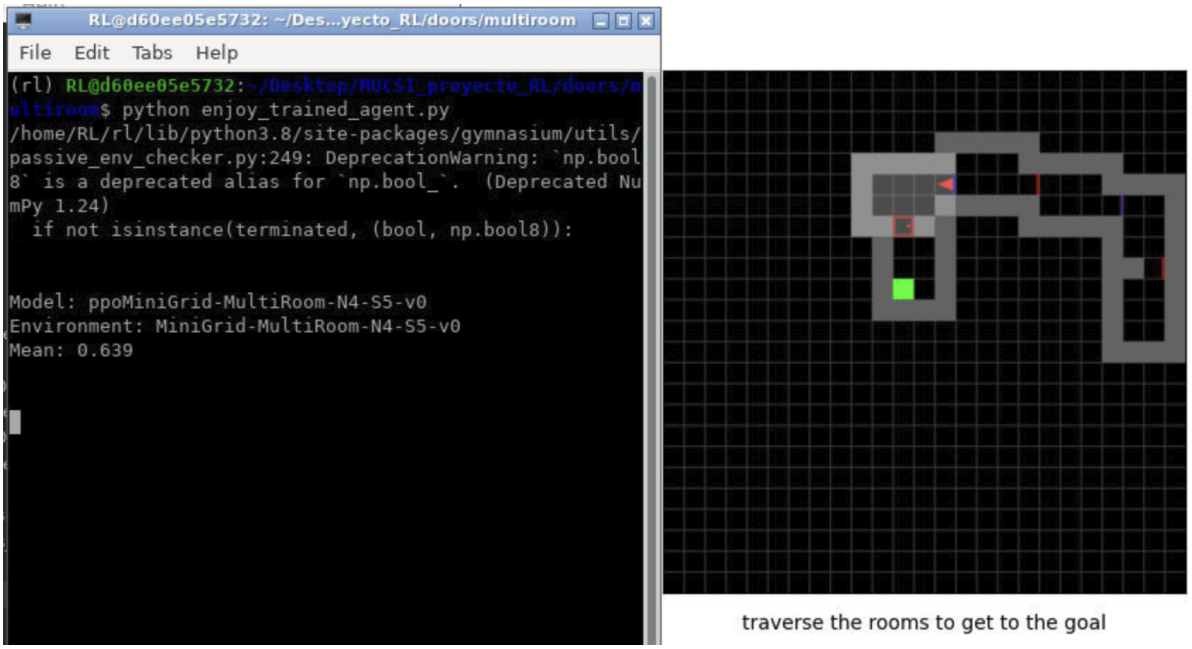

Figura 27.- Ejecución del agente en el entorno entrenado.

También se ha probado a evaluar al agente capaz de operar en el entorno de 4 habitaciones en el entorno irresoluble en los entrenamientos, el de 6 habitaciones. Como se indica en la consola de la Figura 28, el resultado es nulo. El agente se queda quieto en el punto de partida.

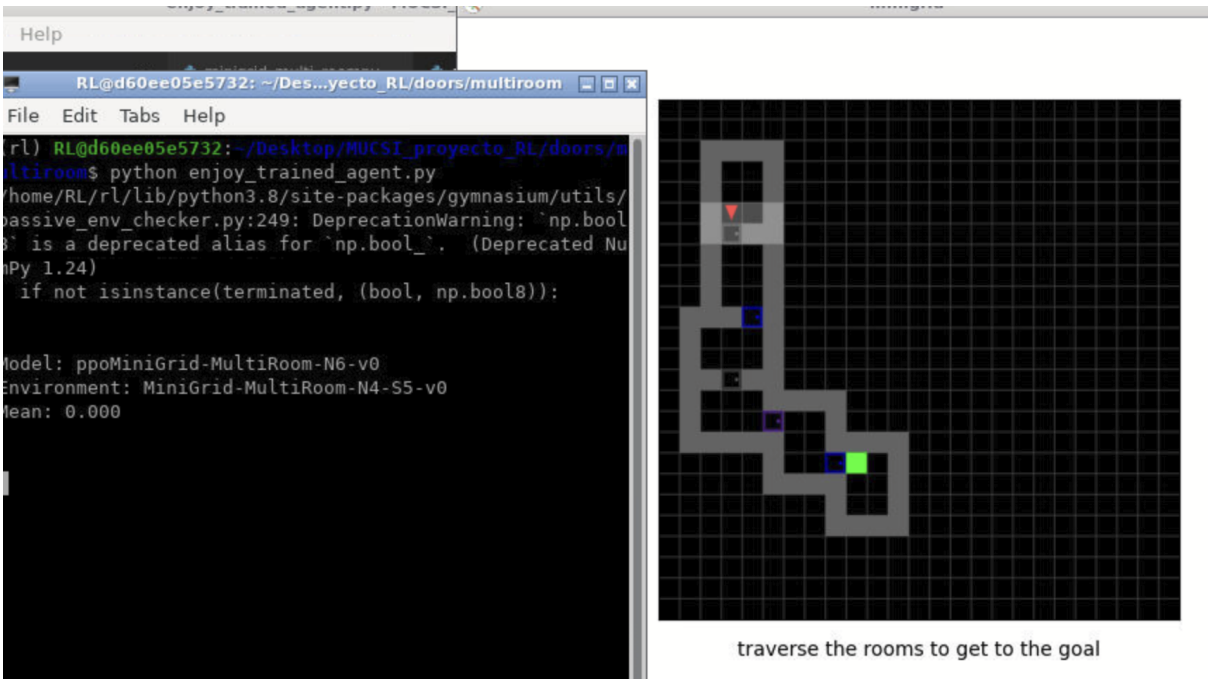

Figura 28.- Ejecución del agente en el entorno de 6 habitaciones.

4.4.- Dynamic Obstacles

A diferencia del resto de configuraciones, como ya se ha comentado anteriormente, esta contiene ciertos objetos dinámicos que el agente debe esquivar para lograr llegar al objetivo.

A pesar de tener el extra de dificultad de los “adversarios”, y que no ha sido posible partir de la base “empty”, el agente ha logrado aprender y resolver todos los entornos mediante el algoritmo PPO.

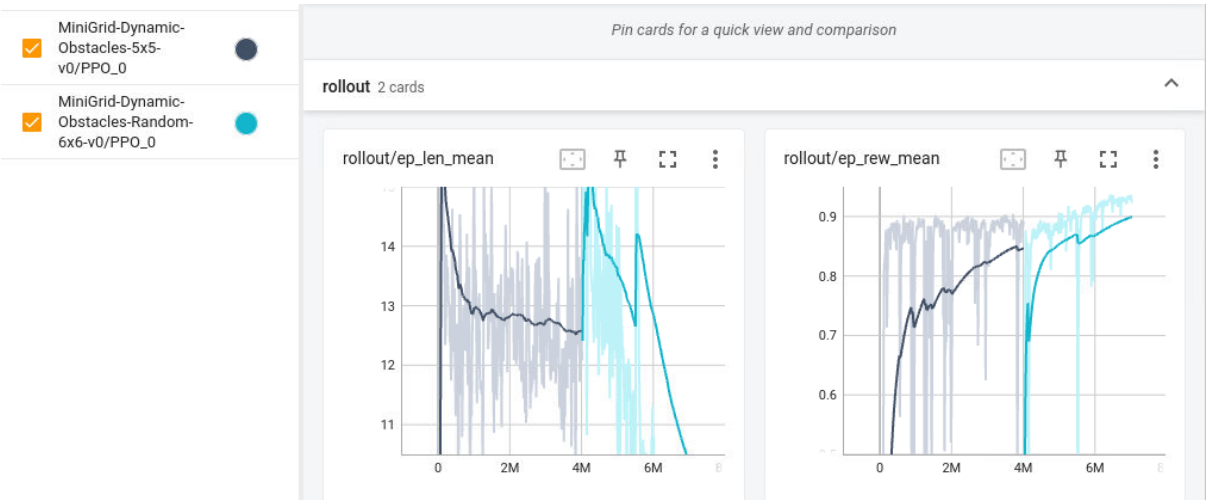

Figura 29.- Logs de entrenamiento en TensorBoard para los intentos con PPO.

Como se puede observar en la ilustración anterior, se ha logrado que el agente resuelva el mapa de 6x6 casillas con una recompensa media excelente. También es cierto que la curva de aprendizaje es más lenta, habiendo requerido 7 millones de iteraciones.

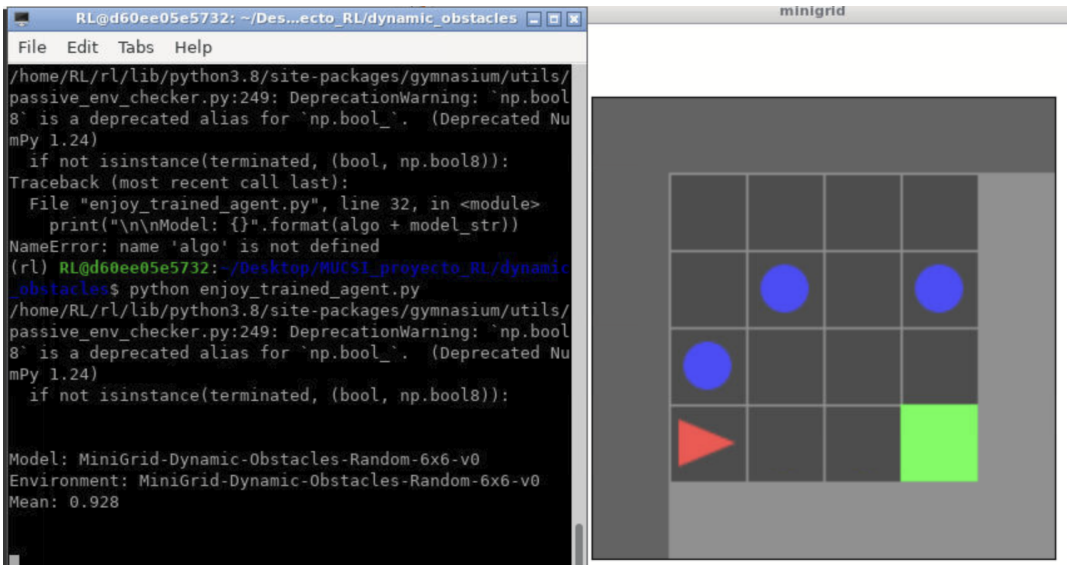

Figura 30.- Evaluación y visualización del modelo entrenado.

5.- LÍNEAS FUTURAS DE DESARROLLO

Como se ha podido observar, en la extensión del presente documento se ha mencionado la opción de resolver una serie de entornos que se basan en lograr que el agente obedezca una orden proporcionada por el usuario. Tras finalizar con el análisis de resultados, de la misma forma, ha podido comprobarse como no se ha detallado ningún tipo de resultados obtenidos para la resolución de dichos entornos. A modo de líneas futuras de desarrollo, se podría plantear la opción de resolver dichos subentornos, como pueden ser, por ejemplo, los referentes a “Fetchenv”, “Gotodoorenv” o “keycorridorenv”.

El entrenamiento del agente para la resolución de dichos subentornos, al igual que en el planteamiento del presente documento, se realizaría empleando una técnica de Curriculum Learning. El objetivo final, se basaría en resolver el entorno considerado como más complejo dentro de dicha categoría, en la que la misión del agente se trata de obedecer órdenes. Dicho subentorno se trata del denominado “Locked Room”.

El motivo principal por el cual, finalmente, y a pesar de haberlo incluido en el alcance inicial del proyecto, se ha decidido dejar la resolución de este tipo de entornos fuera de este, ha sido, básicamente, la limitación de tiempo para el desarrollo del proyecto.

A dicha limitación de tiempo, debe añadirse el hecho de que la ejecución del entorno seleccionado, Minigrid, ha sido la acción que mayor problema ha proporcionado en el desarrollo del proyecto. Dicho problema ha provocado un retraso de, aproximadamente, 3 semanas, en las que, finalmente y tras multitud de intentos, se logró llegar a ejecutar el código sin ningún tipo de error de dependencias o similares. Hasta ese momento no se logró comenzar a entrenar ningún modelo en ninguno de los subentornos descritos.

A modo de conclusión, se obtuvo que el motivo principal del error se debía a que el entorno de ejecución desde el que se estaba probando era Windows, y, teniendo en cuenta que el soporte a todos los errores no explicaba que el entorno no podía ejecutarse en este sistema operativo, tras intentarlo sin éxito, y sin saber el motivo del error, decidimos probar otro entorno de atari.

Se trataba del MS Pac Man. Tras conseguir preparar un entorno apto para ejecutar el minijuego y volcar este entorno al servidor, nos dimos cuenta de que no había prácticamente opciones con las que jugar. Al parecernos demasiado sencillo, decidimos volver a intentarlo con MiniGrid.

Fue en ese momento, tras horas desesperantes de investigar errores, descubrimos que el entorno no era compatible con windows gracias al profesorado de la asignatura. Las nuevas versiones de gym/gymnasium siguieron dando problemas hasta conseguir, gracias a los docentes, el código que solucionaba esta incompatibilidad, y logrando, de esta forma, tras la realización de los entrenamientos de los distintos modelos, el objetivo principal marcado al inicio del proyecto, el cual se basaba en resolver el entorno más complejo dentro de cada misión del agente.

También han surgido más problemas de menor gravedad, como la corrupción contínua de archivos *.ipynb con las que se pretendía trabajar al principio y el crash de VS Code en el servidor.

Por otra parte, es importante mencionar que, a pesar de haber intentado resolver el subentorno referente a “Doorkey- Multiroom” probando con dos algoritmos diferentes como pueden ser el de PPO, siguiendo un Curriculum Learning, y el de DQN, no se ha conseguido resolver el entorno de la manera esperada, debido a su complejidad. Teniendo en cuenta lo mencionado, no ha sido posible llegar a la solución óptima deseada debido a la limitación temporal para el desarrollo del proyecto. Por ello, otra línea futura de desarrollo podría basarse en volver a plantear una resolución adecuada para dicho subentorno, mediante el desarrollo de un modelo que aprenda que el objetivo es llegar al cubo verde a través de las puertas, mediante el desarrollo de un modelo base, del que el agente pueda partir para repetir el entrenamiento. También se plantea probar entrenamientos con otros algoritmos como DQN+HER.

Por último, comentar que, el objetivo final a desarrollar en un futuro dentro del presente proyecto enfocado a Minigrid, se basa en obtener un modelo capaz de resolver entornos de cualquier tipo, de ahí el entrenamiento diferenciado por categorías o misiones a completar por el agente, a fin de generar un modelo que sea capaz de resolver tanto entornos cuya misión se trate de obedecer órdenes, como entornos cuya misión se base en esquivar objetos dinámicos, etc.