# Descripción del ejercicio:

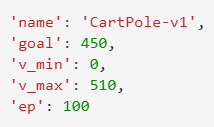
Se utilizarán los recursos utilizados en clase para resolver 2 entornos de entre los propuestos Cartpole-V1 y Acrobot-V1. Para ello se han probado varios de los ejemplos usados en las clases de Raúl Casaña y finalmente se ha optado por utilizar el ejemplo de Actor-Critic básico (AC\_v0) porque es bastante más rápido que los ejemplos DQN y ya que me interesaba entender los algoritmos, no he ido a otras variantes.

Inicialmente se iba a incluir mejoras en el notebook de clase, se introdujeron cambios en el ejemplo DQN y se intentó con C51 (gracias por la ayuda Raúl) pero dado el poco tiempo disponible y la limitación a 3 caras, finalmente opté por usar la implementación de clase para Actor-Critic (v0) (aunque también probé con el notebook de A3C, las pruebas que realicé en Acrobot no llegaron a converger) los únicos cambios son la redefinición de la configuración de los entornos, cambios en los hiperparámetros. Además, se incluye código extra para renderizar las acciones y guardar los modelos en diversos puntos del entrenamiento.

Los experimentos se empezaron a realizar en Colab pero tras muchos problemas con el Acrobot ya que era muy costoso en tiempo de resolver me he pasado a Anaconda en un portátil con Windows.

# Cartpole-V1:

Este sería una versión un poco más compleja que la vista en clase (Cartpole-V0), pero investigando en OpenAI, la documentación del entorno y StackOverflow se ha configurado así:

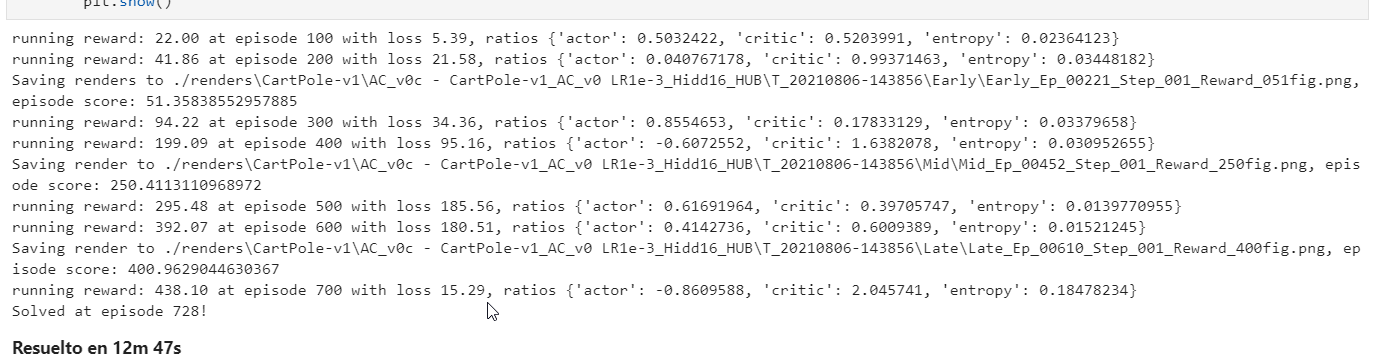
* Se ha aumentado el goal hasta los 450 puntos de media durante al menos 100 episodios.
* Los otros parámetros no se utilizan en AC-v0, serían para una posible mejora usando C51 para estimar V(s) pero finalmente se cambiaron acorde al nuevo goal pero no se utilizarán ya que no se ha integrado en AC\_v0.
* Otro cambio es la duración máx. de los episodios de 200 steps (v0) a 500 steps (V1).

## Hiperparámetros

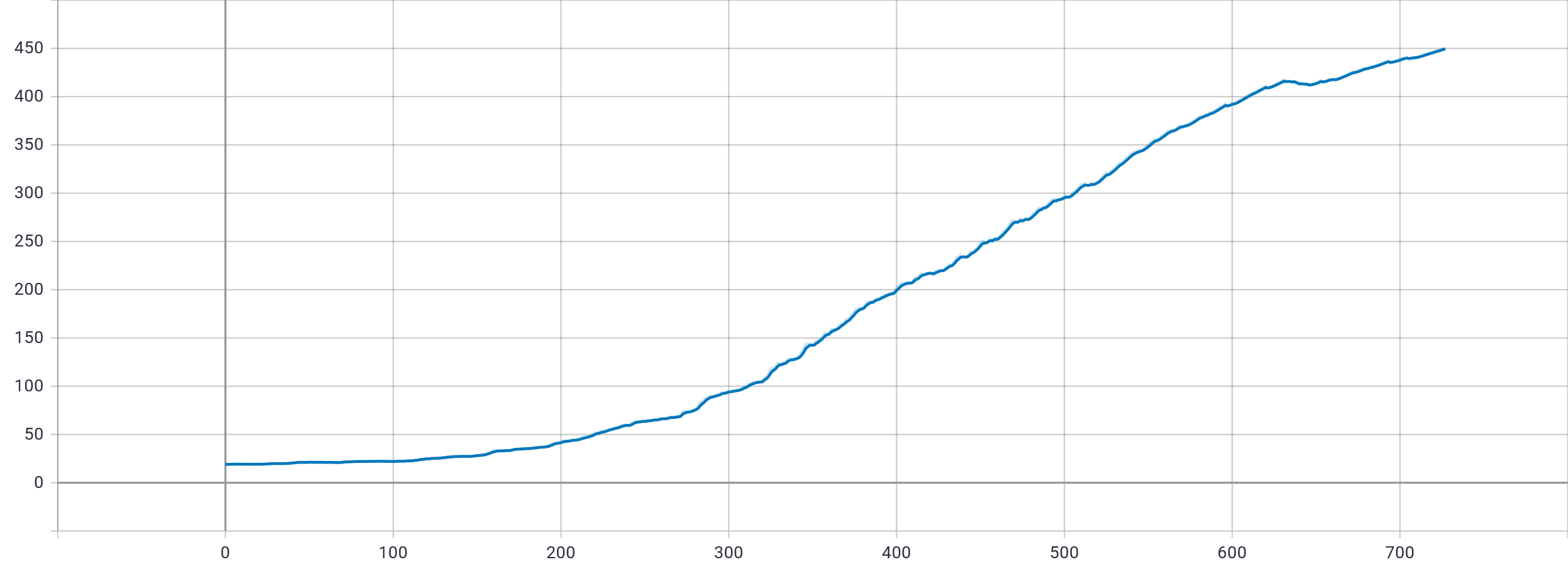
Lo único que se ha cambiado respecto del notebook original ha sido el numero máximo de steps por episodio, que se ha subido a 500 tras leer la documentación de CartPoleV1 (es un entorno un poco más difícil, pero el Actor-Critic lo supera sin problemas), además el LR se ha bajado de los 0,05 originales del notebook a 0.001.

El resto de configuración del algoritmo permanece igual incluyendo la función de loss (Huber).

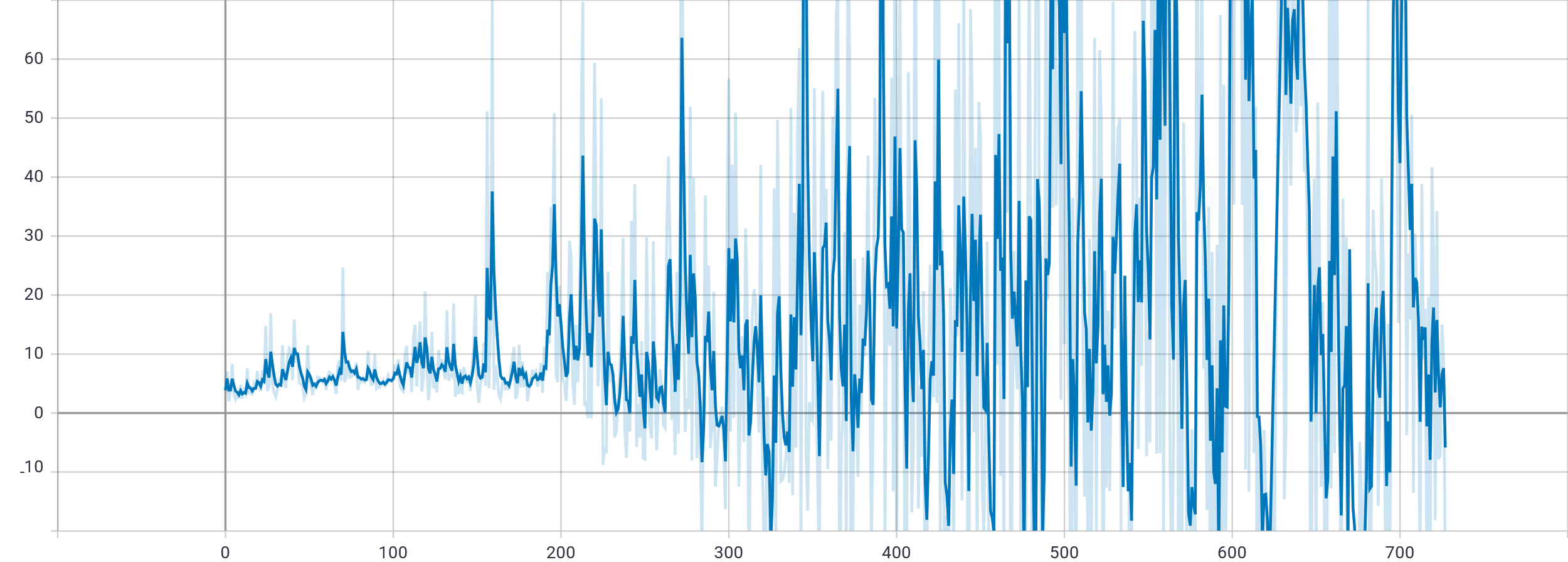
Como se puede apreciar en la traza, se ha resuelto en menos de 13 minutos a lo largo de 728 episodios



## Gráficas del Entrenamiento

**EMA Reward**  


**Total Loss:**



## Test del Agente entrenado

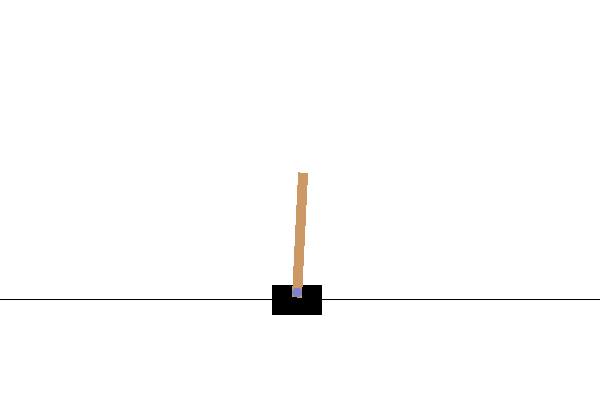
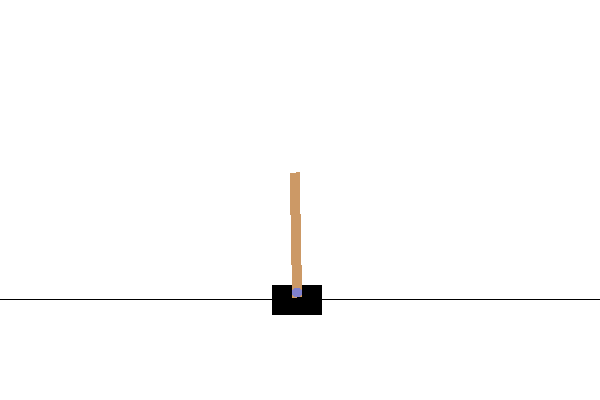
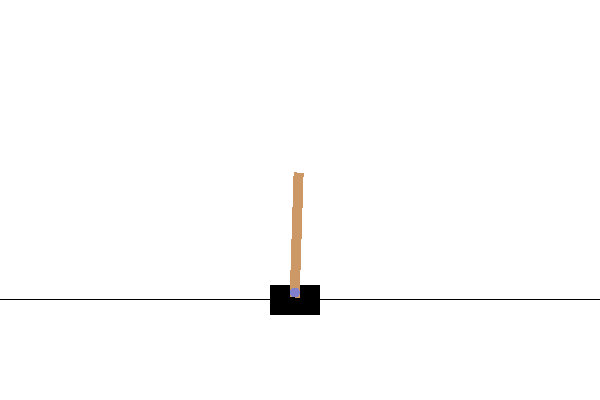
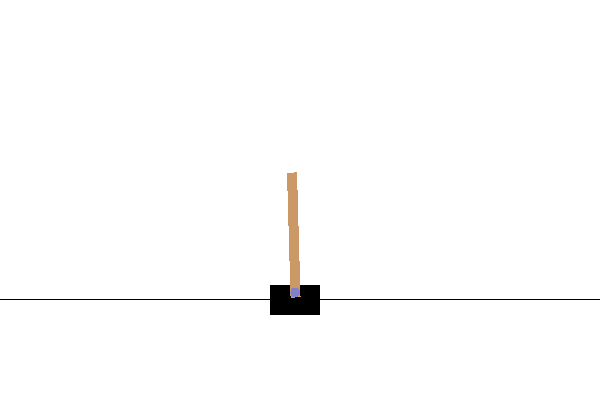
Se ha probado al agente entrenado durante 100 episodios.

La puntuación media ha sido de 480.41 con una desviación de   
+/-69.57, fijándonos en las gráficas y el histograma de los resultados de las 100 tiradas de test, podemos decir que aunque el agente superaría el problema de CartPole-V1 en un 90% de las ocasiones, podemos considerar al agente resultante con bajo bias y baja varianza, frente al agente de Acrobot-V1.

Aunque este problema es un poco más difícil que CartPole-V0, sigue siendo un problema sencillo de resolver con Actor-Critic.

## Renders del aprendizaje del agente

He capturado los renders como imágenes PNG resultantes de capturar los renders mediante matplotlib porque desde Anaconda en Windows no es compatible la librería xvfb, necesaria para guardar las animaciones como ficheros mp4 o gif, por lo que he optado por capturar los renders generados en 3 puntos del entrenamiento y el test (el juego de las 100 tiradas posterior al entrenamiento). Se han montado los Gifs a mano con GIMP.



Episodio 221, Reward 51, 104 steps

Episodio 610, Reward 400, 500 steps

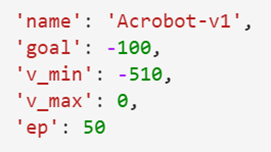
Episodio 452, Reward 250, 165 steps

Partida entrenado, Episodio 728, Reward 450, 500 steps

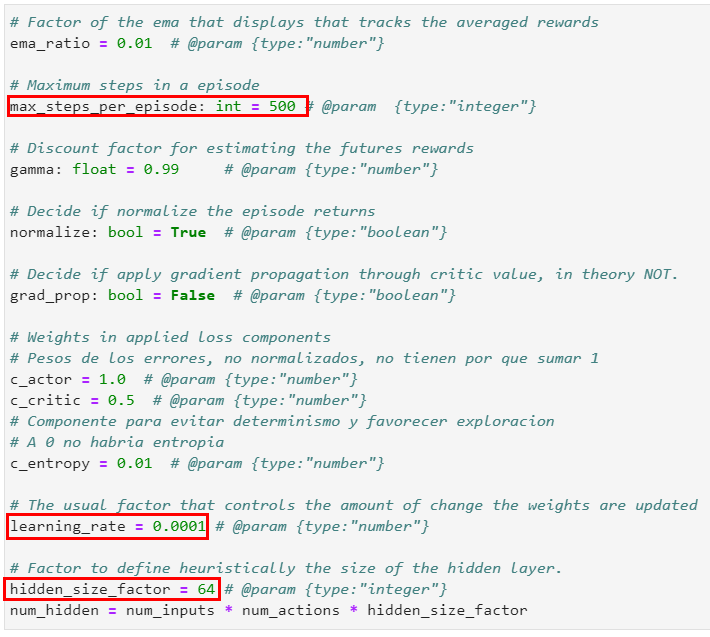
Nótese como a medida que el entrenamiento avanza, el agente controla mejor el CartPole al ser episodios más largos y con mejor puntuación.

# Acrobot-V1:

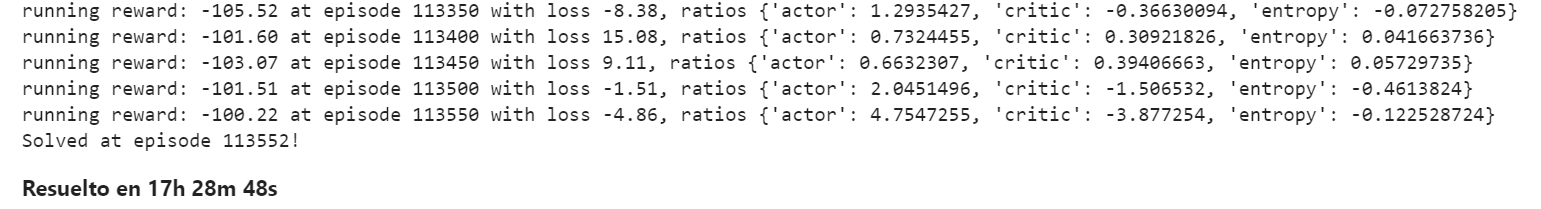
Para la resolución de este problema se ha jugado mucho con los parámetros, como se comenta arriba he tenido problemas en Colab dada la extensión en tiempo, he perdido experimentos que llevaban 15-20h trabajando tras una desconexión, por lo que me he pasado a Anaconda, y los he podido finalizar con éxito tras un entrenamiento de más de 17h.

* Objetivo llegar a los -100 puntos de media durante al menos 100 episodios.
* Los otros parámetros no se utilizan en AC-v0, serían para una posible mejora usando C51 para estimar V(s) pero finalmente se cambiaron acorde al nuevo goal pero no se utilizarán ya que no se ha integrado en AC\_v0.
* Duración máx. de los episodios a 500 steps.

## Hiperparámetros

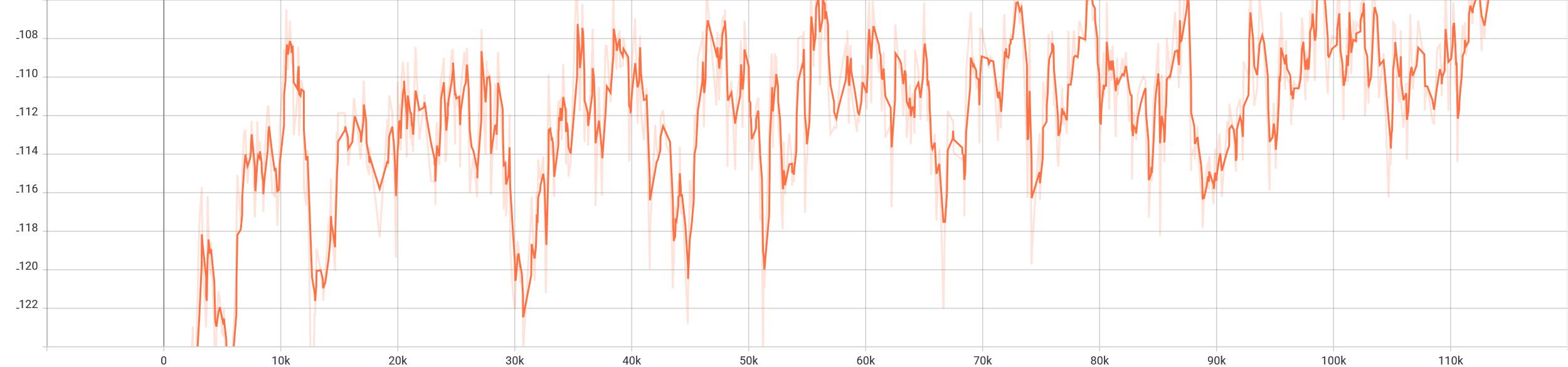
  
Aquí los cambios en los hiperparámetros han sido principalmente bajar el LR (varias pruebas, la que llegó al final en 0.0001), probar con MSE en vez de Huber como función de loss, ya que no conseguía llegar al objetivo de -100 y hubo varios experimentos muy largos interrumpidos o no finalizados por algún problema en Colab o que empezaban a inestabilizarse tras horas a -120, -110.

Además de jugar con el factor del número de neuronas de las capas ocultas que finalmente dejé en 64. Seguramente esto se pueda mejorar mucho más, alguno de los experimentos no finalizados parecía más estable.



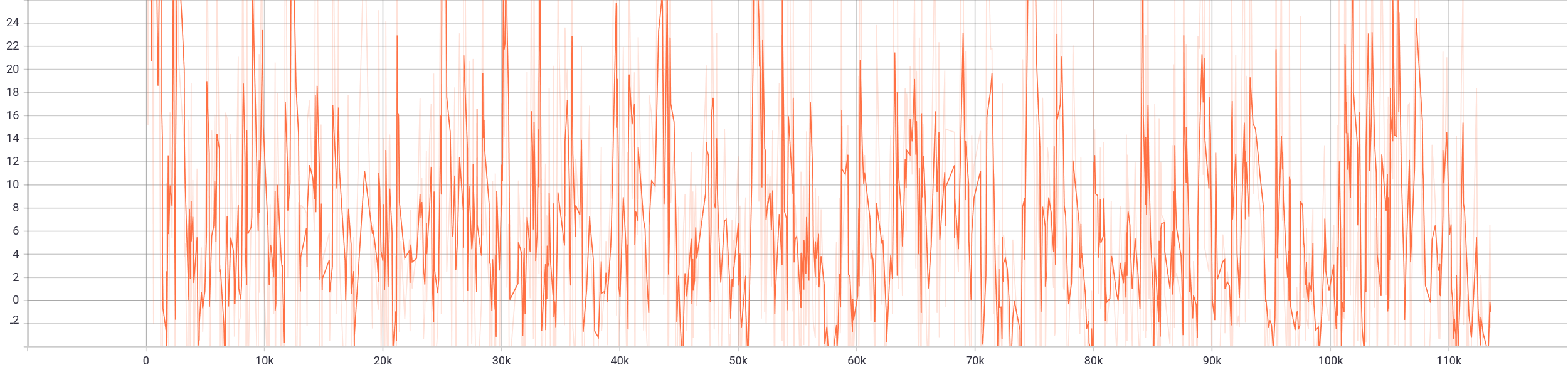
## Gráficas del Entrenamiento

**EMA Reward**

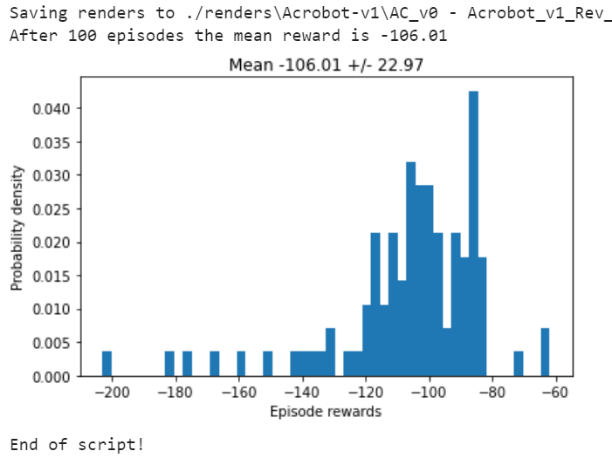


Nótese que la gráfica representa sólo la último partedurante el entrenamiento del eje de la Y, por lo que sólo se ve el rango de recompensas de -124 hasta -107. No me di cuenta al guardarla que tenía que desactivar el zoom y dada la duración del experimento no lo he podido repetir, guardé las gráficas de los logs de TensorBoard pero eran de otro experimento.

**Total Loss:** Aunque los errores oscilan se han mantenido controlados sin dispararse excesivamente.



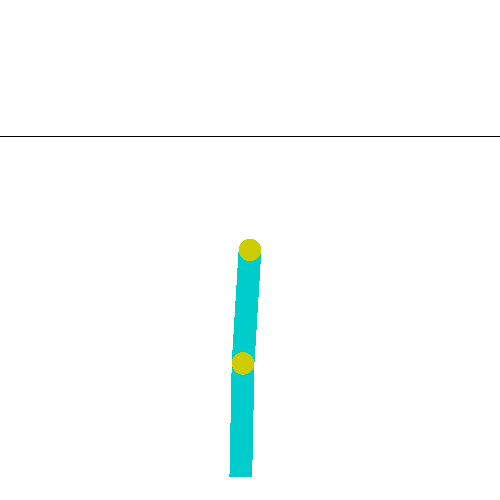
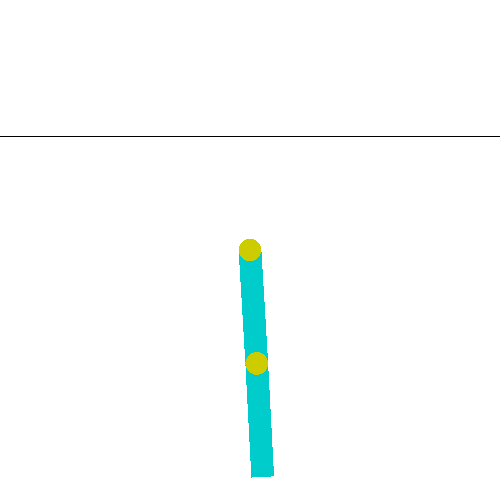
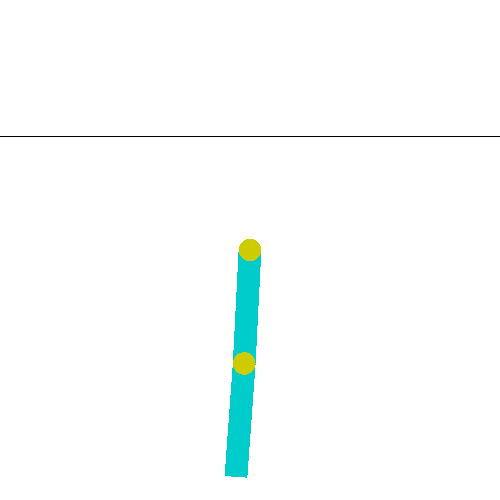
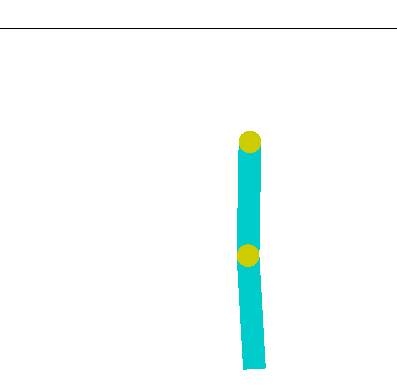
## Test del Agente entrenado

Se ha probado al agente entrenado durante 100 episodios.

La puntuación media ha sido de -106.01 con una desviación de   
+/-22.97, fijándonos en las gráficas y el histograma de los resultados de las 100 tiradas de test, podemos decir que aunque el agente ha sido capaz de llegar al objetivo de -100 puntos, tenemos a un agente con bajo bias y alta varianza, frente al agente de CartPoleV1, aquí se percibe que este problema era más complejo que el anterior, además del tiempo de entrenamiento.

## Renders del aprendizaje del agente

Gifs de los renders del comportamiento del agente a al inicio, punto medio y casi al final del entrenamiento. Además del último que es el test de los 100 episodios posteriores (una tirada random de las 100, en este caso).



Episodio 2, Reward -498, 500 steps

Partida entrenado, Episodio 113552, Reward -99, 110 steps

Episodio 2807, Reward -119, 100 steps

Episodio 421, Reward -249, 213 steps

Nótese como a medida que el entrenamiento avanza, el agente controla mejor el Acrobot al ser episodios más cortos (menor esfuerzo para llegar a la parte superior) y con mejor puntuación.

## Referencias

<https://gym.openai.com/envs/CartPole-v1/>

<https://stackoverflow.com/questions/56904270/difference-between-openai-gym-environments-cartpole-v0-and-cartpole-v1>

<https://gym.openai.com/envs/Acrobot-v1>