PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

APRENDIENDO MODELOS *SPARSE* PARA ALGORITMOS DE *DEEP REINFORCEMENT LEARNING* BASADOS EN *POLICY GRADIENT*

CHRISTIAN MELÉNDEZ SALINAS

Tesis para optar al grado de
Magíster en Ciencias de la Ingeniería

Profesor Supervisor:
HANS LOBEL

Santiago de Chile, Enero 2021

© MMXX, CHRISTIAN MELÉNDEZ SALINAS

DocuSign Envelope ID: 5723CD81-E9C0-400B-940F-80A717A4C0B1

PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA

APRENDIENDO MODELOS *SPARSE* PARA ALGORITMOS DE *DEEP REINFORCEMENT LEARNING* BASADOS EN *POLICY GRADIENT*

CHRISTIAN MELÉNDEZ SALINAS

Miembros del Comité:

HANS LOBEL

ÁLVARO SOTO

IVÁN LILLO

RODRIGO CÁDIZ

DocuSigned by: Hans Lobel 892460F6E47C4A7...
DocuSigned by: Alvaro Soto DA096D318D8B4F2...
DocuSigned by: Iván Lillo FC24E7F0A55840E...
DocuSigned by: Rodrigo Cádiz E0AFA235D0AD428...

Tesis para optar al grado de
Magíster en Ciencias de la Ingeniería

Santiago de Chile, Enero 2021

© MMXX, CHRISTIAN MELÉNDEZ SALINAS

AGRADECIMIENTOS

Mi familia ha sido un pilar fundamental a lo largo de mi desarrollo personal, educacional y profesional. Ellos me han formado, me han inculcado valores y convicciones en las que creo firmemente y me han apoyado en todas las decisiones que he tomado. Por eso quiero agradecerles, a todos: tías/os, primos/as, abuelas/os, en especial a aquellos que estuvieron muy cerca, con los que convivo día a día: mi madre, mi padre, mi hermano y mi abuela.

Quiero agradecer a todas esas personas que se cruzaron en mi camino, las cuales, quizás sin saberlo, aportaron su granito de arena en formar la persona que soy, me motivaron a seguir adelante y compartimos buenos momentos. Quiero dar especial agradecimiento a unas personas que fueron especiales para mí durante mi paso por la universidad por su gran amistad y compañerismo: Javiera San Martin, José Gramsch y Franco Bernasconi. También quiero agradecer a Anakena Pakarati por todo su apoyo y amistad incondicional durante gran parte de mi vida universitaria.

Por último quiero agradecer al grupo IALab por el gran espacio que se generó para investigar, en especial por el grupo humano que se formó. Quiero dar especiales agradecimientos a mi profesor guía, Hans Lobel, por todo el apoyo y motivación que me entregó durante la investigación.

ÍNDICE DE CONTENIDOS

AGRADECIMIENTOS iii

ÍNDICE DE FIGURAS vi

ÍNDICE DE TABLAS viii

ABSTRACT ix

RESUMEN x

1. INTRODUCCIÓN 1

2. TRABAJOS RELACIONADOS 4

3. MARCO TEÓRICO 7
   3.1. Aprendizaje reforzado 7
   3.2. *Policy Optimization* 10

4. APRENDIENDO MODELOS SPARSE 11
   4.1. *Proximal Policy Optimization* 11
   4.2. Arquitectura del modelo 14
   4.3. Técnicas de regularización *sparse* 15
      4.3.1. Regularización L_1 y L_2 15
      4.3.2. Regularización estructural: *Sparse Group Lasso* 17
      4.3.3. Regularización basada en distribución de probabilidad: *Set KL-divergence* 19
   4.4. Regularización aplicada a PPO 21

5. EXPERIMENTOS 23
   5.1. Ambientes de simulación 23
   5.2. Entrenamiento 24

6. RESULTADOS 27
6.1. Entrenamiento 27
6.2. Evaluación de las políticas encontradas durante entrenamiento 28
6.3. *Sparsity* 31
6.4. Rendimiento vs *Sparsity* 35
6.5. Especialización de neuronas 38
6.5.1. *Clustering* de neuronas 38
6.5.2. Comportamiento de los grupos de neuronas 40
6.5.3. Cuantificando el grado de especialización de los grupos de neuronas . 43
6.5.4. Discusión 44
7. CONCLUSIONES 46
8. TRABAJO FUTURO 48
REFERENCIAS 50
ANEXO 56
A. BÚSQUEDA DE HIPERPARÁMETROS 57
A.1. Hiperparámetros de PPO 57
A.2. Coeficientes de regularización 60
B. VARIABILIDAD DE EXPERIMENTOS 61
B.1. Variabilidad durante entrenamiento 62
B.2. Variabilidad durante evaluación de *checkpoints* 66
B.3. Variabilidad durante evaluación de *sparsity* 70
C. COMPARACIÓN REGULARIZACIÓN SOBRE PESOS Y ACTIVACIONES 74

ÍNDICE DE FIGURAS

3.1 Interacción agente-ambiente en un MDP 8

4.1 Estructura de red neuronal para algoritmo PPO 13

4.2 Esquema de aplicación de regularizaciones sobre los pesos y activaciones del modelo 18

4.3 Esquema de aplicación de regularización SGL sobre los pesos del modelo . . 19

5.1 Imágenes de los ambientes de simulación 25

6.1 Recompensa promedio durante entrenamiento de los modelos con las distintas regularizaciones 29

6.2 Recompensa promedio de los *checkpoints* guardados durante entrenamiento de las distintas regularizaciones 30

6.3 Evaluación del nivel de *sparsity* alcanzado por las regularizaciones 33

6.4 Visualización de las activaciones de los modelos entrenados 36

6.5 Rendimiento vs *Sparsity* 37

6.6 Reducción de dimensionalidad de las activaciones de las neuronas de un agente durante la ejecución de su política 39

6.7 Serie de tiempo de las activaciones de los grupos neuronas de un agente durante la ejecución de la política 42

6.8 Ejemplo del cálculo del porcentaje de activación de neuronas durante ejecución de la política 42

6.9 Cálculo de sobrelape entre grupo de neuronas 43

B.1 Variabilidad durante entrenamiento del ambiente *BipedalWalker* 62

B.2 Variabilidad durante entrenamiento del ambiente *Ant* 63

B.3 Variabilidad durante entrenamiento del ambiente *HalfCheetah* 64

B.4 Variabilidad durante entrenamiento del ambiente *Walker2D* 65

B.5 Variabilidad durante evaluación de *checkpoints* en ambiente *BipedalWalker* . 66

B.6 Variabilidad durante evaluación de *checkpoints* en ambiente *Ant* 67

B.7 Variabilidad durante evaluación de *checkpoints* en ambiente *HalfCheetah* . . 68

B.8 Variabilidad durante evaluación de *checkpoints* en ambiente *Walker2D* 69

B.9 Variabilidad durante evaluación de *sparsity* en ambiente *BipedalWalker* . . . 70

B.10 Variabilidad durante evaluación de *sparsity* en ambiente *Ant* 71

B.11 Variabilidad durante evaluación de *sparsity* en ambiente *HalfCheetah* 72

B.12 Variabilidad durante evaluación de *sparsity* en ambiente *Walker2D* 73

C.1 Recompensa y valor de regularización en ambiente *BipedalWalker* 74

C.2 Recompensa y valor de regularización en ambiente *Ant* 75

C.3 Recompensa y valor de regularización en ambiente *HalfCheetah* 75

C.4 Recompensa y valor de regularización en ambiente *Walker2D* 76

ÍNDICE DE TABLAS

5.1 Descripciones de los ambientes de entrenamiento 23

6.1 Recompensa promedio de los últimos 5 *checkpoints* de cada experimento . . 31

6.2 Nivel de *sparsity* logrado por las regularizaciones 34

6.3 Sobrelape promedio entre grupos de neuronas durante la ejecución de la política 44

A.1 Hiperparámetros del ambiente *BipedalWalker* 57

A.2 Hiperparámetros del ambiente *Ant* 58

A.3 Hiperparámetros del ambiente *HalfCheetah* 59

A.4 Hiperparámetros del ambiente *Walker2D* 60

A.5 *Grid Search* de coeficientes regularizadores 61

ABSTRACT

Neural networks models have been widely used in the last decade, mainly because of their great versatility and capacity of achieving good performance while solving various problems. One of the causes of this phenomenon seems to be their hierarchical structure and large number of parameters as they provide the models with considerable expressive power. However, this expressive power can be detrimental, because it can generate: i) overfitting, ii) redundant parameters for the task being solved and iii) unnecessary computation. One way of reducing model complexity is through sparse regularization, which is a penalization in the objective function of the optimization problem that forces the use of fewer parameters or neurons. Ways of reducing model complexity in neural networks through sparse regularization have been explored in supervised learning, nevertheless, it hasn't been explored in policy gradient based deep reinforcement learning. This work studies the possibility of obtaining less complex models in policy gradient based deep reinforcement learning. This is done by comparing different types of sparse regularization, focusing on obtaining less complex models regarding neuron use. Results indicate it is possible to find models that use a reduced number of neurons through sparse regularization and that applying it over neuron activations has better results in performance and sparsity level. Also, it is shown that less complex models have more specialized neurons that could help to interpret models regarding the role that group of neurons has in the agent policy.

Keywords: sparsity, deep reinforcement learning, neural networks, regularization

RESUMEN

Los modelos de redes neuronales han sido ampliamente utilizados durante la última década, debido principalmente a su gran versatilidad y capacidad de obtener un alto rendimiento al resolver diversos problemas. Una de las posibles causas de este fenómeno parece ser la estructura jerárquica y la gran cantidad de parámetros que poseen, ya que les otorga un gran poder expresivo. Sin embargo, esta gran complejidad puede ser perjudicial, ya que puede generar: i) *overfitting*, ii) parámetros redundantes para la tarea que se está resolviendo y iii) cómputo innecesario. Una forma de reducir la complejidad del modelo es a través de regularización *sparse*, la cual consiste en una penalización dentro de la función objetivo del problema de optimización que fuerza el uso de menos parámetros o neuronas. Se han explorado formas de reducir la complejidad de los modelos de redes neuronales a través de regularización en contextos de aprendizaje supervisado, sin embargo, no se ha explorado el efecto que tiene en un contexto de aprendizaje reforzado basado en *policy gradient*. El presente trabajo estudia la posibilidad de obtener modelos menos complejos en aprendizaje reforzado utilizando algoritmos basados en *policy gradient*. Esto se hace comparando distintos tipos de regularización *sparse*, enfocándose en la obtención de modelos menos complejos en cuanto al uso de neuronas. Los resultados de este estudio indican que sí es posible encontrar modelos que utilicen una baja cantidad de neuronas a través de regularización *sparse*, siendo aquella aplicada sobre las activaciones la que obtuvo mejores resultados en cuanto a rendimiento y nivel de *sparsity*. Además, se muestra que modelos menos complejos poseen neuronas más especializadas que podrían ayudar a interpretar modelos en cuanto al rol que cumplen grupos de neuronas dentro de una política.

Palabras Claves: *sparsity*, Aprendizaje reforzado profundo, redes neuronales, regularización.

1. INTRODUCCIÓN

Los modelos de redes neuronales han sido ampliamente utilizados durante la última década, debido principalmente a su gran versatilidad y su capacidad de obtener un alto nivel de rendimiento en diversos problemas. Dentro de los posibles motivos que explican este rendimiento, la gran cantidad de parámetros que usan estos modelos parece ser clave, ya que les otorga un gran poder expresivo que les permite detectar y generar una amplia gama de patrones. Sin embargo, esta enorme cantidad de parámetros puede llegar a ser perjudicial en algunos contextos, ya que podría provocar, por ejemplo, *overfitting* (Hawkins, 2004) o se podría terminar con un modelo con demasiados parámetros, los cuales podrían llegan a ser redundantes en la tarea que se quiere resolver y aumentan el cómputo necesario. Es por esto que es deseable que los modelos sean menos complejos, es decir, que utilicen una menor cantidad de parámetros o una menor cantidad de neuronas.

Una forma de lograr modelos menos complejos es a través de la generación de modelos *sparse*, lo cual puede definirse de dos formas: un modelo que tiene pocos pesos o neuronas activas; o un modelo cuyas representaciones ocupan una baja cantidad de neuronas activas. Esta propiedad de los modelos ha demostrado ser útil en el área de inteligencia artificial desde hace mucho tiempo, tanto en modelos conexionistas, como lo son las redes neuronales (Frankle & Carbin, 2019; French, 1991; Lobel, Vidal, & Soto, 2015, 2020; Louizos, Welling, & Kingma, 2018; Scardapane, Comminiello, Hussain, & Uncini, 2016; Sutton, 1996), como en modelos clásicos, como regresión (Friedman, 2012; Tibshirani, 1996). Esto se debe a que ha ayudado en múltiples aspectos del proceso de entrenamiento de estos modelos, como la simplificación de modelos complejos al remover parámetros que no son relevantes para la tarea realizada (Friedman, 2012; Scardapane et al., 2016; Tibshirani, 1996), ayudar al problema del aprendizaje continuo, en donde un modelo debe aprender continuamente distintas tareas (Aljundi, Rohrbach, & Tuytelaars, 2019; Ororbia, Mali, Kifer, & Giles, 2019; Srivastava, Berman, Blaschko, & Tuia, 2019) o incluso encontrar subestructuras dentro de un modelo que tengan un rendimiento superior al modelo original (Frankle & Carbin, 2019).

Uno de los contextos en los cuales las redes neuronales también son ampliamente usadas es en aprendizaje reforzado (RL, por sus siglas en inglés), el cual consiste en aprender un comportamiento basado en una función de recompensa previamente definida. En RL se ha explorado el efecto que tiene aplicar *sparsity* al proceso de entrenamiento de una red neuronal y una de las metodologías utilizadas es la aplicación de regularizaciones que fomenten esta propiedad: regularización basada en normas convexas, regularizaciones distribucionales o *dropout* (Hernandez-Garcia & Sutton, 2019; Liu, Kumaraswamy, Le, & White, 2019). El objetivo de estos estudios no era simplificar el modelo en cuanto a la cantidad de parámetros o neuronas activas, sino que las representaciones (activaciones de las neuronas) que aprendiera el modelo sean *sparse*, lo cual significa que pocas neuronas estén activadas para una observación dada y que observaciones muy distintas entre sí, tengan distintas neuronas activadas. De hecho, Hernandez-Garcia y Sutton mencionan que las neuronas "muertas" (no usadas por el modelo) son perjudiciales para el rendimiento.

Sin embargo, disminuir la cantidad de parámetros usados por el modelo, como se dijo anteriormente, puede traer beneficios, como la obtención de modelos menos complejos y eficientes en cuanto al uso de neuronas, prevenir el *overfitting*, mejorar la interpretación del modelo al eliminar neuronas redundantes o permitir el aprendizaje continuo. Por lo que surge la siguiente interrogante: ¿es posible encontrar, durante entrenamiento, un modelo que utilice pocas neuronas, es decir, que posea un gran número de neuronas que no aporten a la representación de los datos, en un contexto de aprendizaje reforzado sin que afecte el rendimiento?

El presente trabajo tiene como objetivo estudiar el impacto de la aplicación de regularización *sparse* mediante normas convexas al proceso de entrenamiento de un agente en un contexto de aprendizaje reforzado, pero, a diferencia de trabajos anteriores, se buscará que esta regularización simplifique el modelo haciendo que ocupe una menor cantidad de neuronas. Esto se realizará aplicando distintos tipos de regularización estructural al proceso de entrenamiento de agentes en ambientes de simulación de control continuo y se analizará el impacto de estas comparada con regularizaciones utilizadas en trabajos anteriores. Dado

que se utilizarán ambientes de control continuo, se utilizará el algoritmo *Proximal Policy Optimization* (Schulman, Wolski, Dhariwal, Radford, & Klimov, 2017) basado en *policy gradient*, ya que es uno de los algoritmos que mejores resultados ha obtenido en este tipo de ambientes. Los resultados de este estudio indican que es posible entrenar modelos que utilicen un bajo porcentaje de neuronas ($\sim 45\%$) en un contexto de aprendizaje reforzado y que logren un desempeño similar a un modelo entrenado sin regularización que induzca un menor uso de parámetros. Además, se hace un estudio cualitativo en donde se confirma que un modelo más simple permite saber con mayor facilidad el rol de las neuronas en la política del agente.

El documento se estructura de la siguiente forma: en la sección 2 se describen los trabajos relacionados en donde se ha ocupado *sparsity*. En la sección 3 se presenta el marco teórico. En la sección 4 se describe el modelo, las regularizaciones utilizadas y cómo fueron introducidas al proceso de entrenamiento. En la sección 5 se describen los experimentos realizados. En la sección 6 se exponen los resultados obtenidos y su análisis. En la sección 8 se describen las posibles extensiones al trabajo realizado y en la sección 7 se presentan las conclusiones.

2. TRABAJOS RELACIONADOS

La regularización *sparse* en redes neuronales ha sido ocupada tanto en aprendizaje supervisado como reforzado con diversos objetivos. En aprendizaje supervisado se ha utilizado regularización *sparse* para disminuir la complejidad de los modelos de redes neuronales, de tal manera que menos pesos o grupos de pesos sean ocupados por el modelo (Lobel et al., 2020; Louizos et al., 2018; Scardapane et al., 2016). Esto permite tener modelos con menos neuronas redundantes (que codifiquen la misma información), computacionalmente más eficientes y con mejor generalización.

También se ha utilizado regularización *sparse* en el problema de aprendizaje continuo, el cual consiste en aprender tareas de manera secuencial sin olvidar las tareas anteriores (*Catastrophic Forgetting*). Aljundi et al. (2019) descubren que aplicar regularización *sparse* a nivel de las representaciones (activaciones de las neuronas) es más beneficioso para aprender tareas de manera secuencial que aplicar la regularización a nivel de pesos. Con esto se logra tener un modelo con neuronas ”libres” al terminar de aprender una tarea para que puedan ser utilizadas en las próximas tareas. Otros trabajos han aplicado *sparsity* para resolver este problema con metodologías diferentes a la regularización, pero con la misma finalidad de tener más capacidad libre en el modelo para ocuparla en tareas nuevas (Sokar, Mocanu, & Pechenizkiy, 2020; Srivastava et al., 2019).

En el caso de aprendizaje reforzado, se ha utilizado regularización *sparse* mediante normas convexas con el fin de realizar selección de *features* para mejorar la generalización y prevenir el *overfitting* (Le, Kumaraswamy, & White, 2017; Painter-Wakefield & Parr, 2012; Qin, Li, & Janoos, 2014; Song, Li, Jin, & Hirasawa, 2020). Aunque en estos casos no se ocupan redes neuronales, dejan en claro que la reducción de dimensionalidad de las *features* utilizadas durante entrenamiento ayudan a mejorar el rendimiento y estabilidad de los algoritmos de aprendizaje reforzado.

Otro uso de la regularización *sparse* en aprendizaje reforzado es prevenir el *Catastrophic Interference*, el cual consiste en la dificultad que tiene una red neuronal para recordar información antigua cuando está aprendiendo con nuevas observaciones del agente. Esto ocurre cada vez que el agente cambia su política y comienza a explorar estados que las políticas anteriores no fueron capaces de alcanzar. Liu et al. (2019) estudian una forma de evitar este problema a través de distintas regularizaciones *sparse* (normas convexas, distribucionales y *dropout*) aplicadas a la última capa de una red neuronal de dos capas. La finalidad de esta regularización, a diferencia de los usos mencionados anteriormente, no es eliminar pesos ni neuronas, sino que forzar que las representaciones que aprende el modelo para estados muy distintos entre sí, no se sobrelapen, es decir, que distintas neuronas se activen para estados distintos. Hernandez-Garcia y Sutton (2019) complementan el estudio anterior utilizando DQN (Mnih et al., 2013) como algoritmo de optimización. Ambos trabajos concluyen que *sparsity* a nivel de representaciones (activaciones neuronales) ayudan a disminuir el *Catastrophic Interference*. Otros trabajos también buscan resolver este problema a través de *sparsity* pero aplicando transformaciones *sparse* al *input* (Ghiassian, Yu, Rafiee, & Sutton, 2018) mediante métodos que fuerzan la inhibición de neuronas (Rafati & Noelle, 2019) o a través de la modificación de la función de activación del modelo (Pan, Banman, & White, 2020).

Como se puede ver, *sparsity* ha sido ampliamente usado en distintos contextos. Sin embargo, en el área de aprendizaje reforzado no se ha explorado el efecto que tiene aplicar regularización *sparse* con el objetivo de que el modelo ocupe la menor cantidad de neuronas posibles, como se ha hecho en el caso de aprendizaje supervisado. Hernandez-Garcia y Sutton (2019) mencionan que los modelos con neuronas "muertas" (neuronas no utilizadas por el modelo) que generan las regularizaciones estudiadas, tienen peor rendimiento que los modelos sin neuronas "muertas", sin embargo, no hacen un análisis a fondo de esto. Además, modelos con pocas neuronas activas se enfocarían en las *features* que realmente importan para el problema (*feature selection*), serían menos complejos, tendrían menor cantidad de neuronas redundantes e incluso podría utilizarse la capacidad

libre de estos para nuevas tareas. El presente trabajo busca estudiar esta área no explorada con el objetivo de analizar el impacto de las regularizaciones *sparse* que inducen a modelos tener menos neuronas activas en el contexto de aprendizaje reforzado.

3. MARCO TEÓRICO

3.1. Aprendizaje reforzado

Según Sutton y Barto (2018), "aprendizaje reforzado" (*Reinforcement Learning* en inglés) es aprender qué hacer en algún escenario para maximizar una recompensa numérica, es decir, se busca encontrar la mejor acción dada cierta situación en algún contexto. Este contexto puede ser realista, como un robot caminando, o abstracto, como decidir cuándo es mejor comprar acciones en la bolsa. El aprendizaje parte sin información sobre las acciones que se deben tomar, por lo tanto, una de las dificultades de este tipo de aprendizaje es que se deben explorar distintas combinaciones de acciones hasta encontrar aquellas que generen mayor recompensa.

Formalmente, el problema de aprendizaje reforzado puede representarse a través de un *Markov Decision Process* (MDP), el cual está definido por la tupla (S, A, P, R, ρ_0) en donde:

- S es el conjunto de estados.
- A es el conjunto de acciones.
- $P : S \times A \to \mathcal{P}(S)$ es la función de probabilidad de transición, en donde $P(s'|s, a)$ corresponde a la probabilidad de transicionar al estado s' cuando se está en el estado s y se toma la acción a.
- $R(s, a, s') : S \times A \times S \to \mathbb{R}$ es la función de recompensa.
- ρ_0 es la distribución del estado inicial.

Al ente que toma las decisiones se le llama "agente" y a todo con lo que debe interactuar se le llama "ambiente". A las reglas que debe seguir el agente para decidir qué acción tomar se le llama política (π) y, para el caso estocástico, se define como $\pi : S \times A \to [0, 1]$. La figura 3.1 muestra la relación entre el agente y su ambiente en un MDP. Como se puede ver, el agente interactúa con el ambiente a través de una acción (A_t), la cual condiciona

el próximo estado (S_{t+1}) y recompensa (R_{t+1}) que el ambiente genera. Luego, el agente toma otra decisión basada en estas nuevas variables y se repite el ciclo.

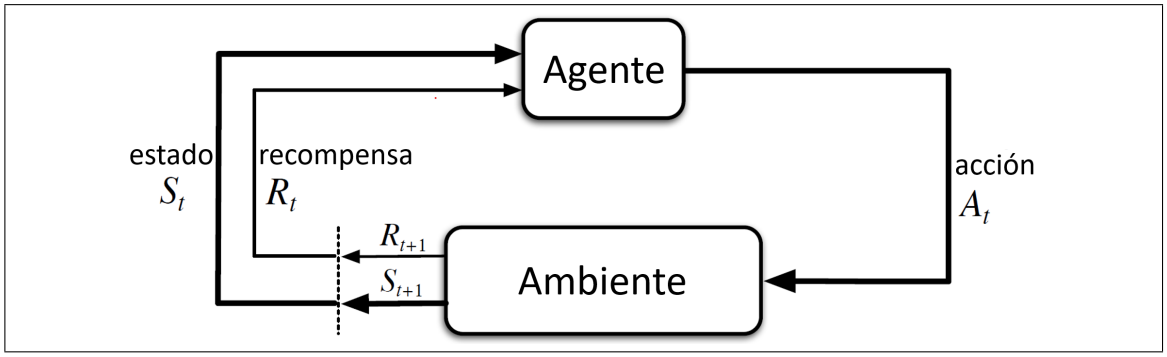


Figura 3.1. Interacción agente-ambiente en un MDP. Adaptada de Sutton y Barto (2018).

Dado lo anterior, el objetivo del aprendizaje reforzado es maximizar la esperanza de la suma de las recompensas descontadas que obtiene el agente:

$$\max_{\pi} \mathbb{E}[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1})|\pi] \tag{3.1}$$

En donde $\gamma \in [0, 1]$ es la tasa de descuento. Existen diversos métodos para optimizar la ecuación 3.1, desde soluciones exactas, en donde se busca un mapeo explícito de estados a acciones, hasta soluciones aproximadas, en donde se utiliza una función para mapear estados a acciones (o estimar la recompensa que se obtendrá a futuro) cuando el dominio de los estados es extremadamente grande. Sin embargo, la mayoría se basa en la función de valor y la función de acción-valor (también llamada *Q-function* en inglés) las cuales se definirán a continuación basándose en el trabajo de Achiam (2018). Para simplificar la notación, las esperanzas a continuación se calculan sobre las distribuciones $s_{t+1} \sim P(\cdot|s_t, a_t)$ y $a_t \sim \pi(\cdot|s_t)$

- **Función de valor**: recompensa esperada partiendo desde el estado s y después actuando según la política π.

$$V^{\pi}(s) = \mathbb{E}[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s] \tag{3.2}$$

- **Función de acción-valor**: recompensa esperada partiendo desde el estado s y acción a y después actuando de acuerdo a la política π.

$$Q^{\pi}(s, a) = \mathbb{E}[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s, a_0 = a] \tag{3.3}$$

- **Función de valor óptima**: recompensa esperada partiendo desde el estado s y después actuando según la política óptima.

$$V^{*}(s) = \max_{\pi} \mathbb{E}[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s] \tag{3.4}$$

- **Función de acción-valor óptima**: recompensa esperada partiendo desde el estado s y acción a y luego actuando según la política óptima.

$$Q^{*}(s, a) = \max_{\pi} \mathbb{E}[\sum_{t=0}^{H} \gamma^t R(s_t, a_t, s_{t+1}) | s_0 = s, a_0 = a] \tag{3.5}$$

La función de valor y la función de acción-valor obedecen la ecuación de Bellman, por lo que pueden expresarse como

$$V^{\pi}(s) = \mathbb{E}_{s' \sim S, a \sim \pi}[\gamma R(s, a, s') + \gamma V^{\pi}(s')] \tag{3.6}$$

$$Q^{\pi}(s, a) = \mathbb{E}_{s' \sim S}[R(s, a, s') + \gamma \mathbb{E}_{a' \sim \pi}[Q^{\pi}(s', a')]] \tag{3.7}$$

y para el caso de optimalidad:

$$V^{*}(s) = \max_{a} \mathbb{E}_{s' \sim S}[R(s, a, s') + \gamma V^{*}(s')] \tag{3.8}$$

$$Q^{*}(s, a) = \mathbb{E}_{s' \sim S}[R(s, a, s') + \gamma \max_{a'} Q^{*}(s', a')] \tag{3.9}$$

En base a estas últimas ecuaciones han surgido diversas metodologías para encontrar la política óptima, como la aplicación de programación dinámica (cuando el dominio de

los estados es pequeño), *Q-Learning* (Watkins & Dayan, 1992) o la optimización directa de la política, la cual es la que se ocupará en el presente trabajo.

3.2. *Policy Optimization*

Policy Optimization, como su nombre lo dice, es una metodología de optimización para aprendizaje reforzado en donde se optimiza directamente la política. Esto quiere decir que el modelo resultante puede decidir qué acción tomar solo basado en el estado actual. Este tipo de algoritmos se basan en el cálculo de un estimador del gradiente de la política (más conocido como *policy gradient*) y, tal como lo plantean Schulman et al. (2017), el estimador más usado es:

$$\hat{g} = \hat{\mathbb{E}}_t[\nabla_\theta \log \pi_\theta(a_t|s_t)\hat{A}_t] \tag{3.10}$$

En donde π_θ es una política estocástica parametrizada por θ y $\hat{A}_t$ es el estimador de la función *advantage*, definida como $A(s,a) = Q(s,a) - V(s)$, la cual mide cuánto mejor es tomar la acción a con respecto al promedio de las distintas acciones. Intuitivamente, lo que hace este gradiente es fomentar una secuencia de acciones que obtengan, en promedio, una mayor recompensa, ya que como $\hat{A}_t$ mide qué tan mejor es elegir la acción con respecto al promedio de las demás acciones, si la secuencia de acciones que el agente escogió en promedio dio un valor de $\hat{A}_t$ positivo, entonces el gradiente tratará de hacer más probables esas secuencias de acciones. Empíricamente, la esperanza de la ecuación 3.10 corresponde a un promedio de un conjunto de observaciones obtenidas desde la ejecución de la política del agente. El estimador de la ecuación 3.10 es usado para actualizar los parámetros del modelo que se está entrenando:

$$\theta_{k+1} = \theta_k + \alpha\hat{g} \tag{3.11}$$

4. APRENDIENDO MODELOS *SPARSE*

Como se dijo en la sección 1, se utilizarán ambientes de simulación de control continuo para estudiar el efecto de aplicar regularizaciones *sparse* al proceso de entrenamiento en aprendizaje reforzado. Se eligen estos agentes principalmente debido a que, en la práctica, la información que debe manejar un agente en la mayoría de los casos es continua. El algoritmo que será utilizado es *Proximal Policy Optimization* (PPO) (Schulman et al., 2017) debido a que es uno de los que mejores resultados ha obtenido en ambientes de control continuo. De hecho, ha logrado aprender exitosamente políticas muy complejas (Andrychowicz et al., 2020; OpenAI et al., 2019).

A continuación se detallará el algoritmo a utilizar, la arquitectura de la red neuronal y las distintas regularizaciones que se utilizarán para imponer *sparsity* en el modelo.

4.1. *Proximal Policy Optimization*

Entre los algoritmos más recientes y con mejores resultados que se basan en *policy gradient* se encuentra *Proximal Policy Optimization* (PPO) (Schulman et al., 2017) el cual busca simplificar y mejorar en cuanto a complejidad a *Trust Region Policy Optimization* (TRPO) (Schulman, Levine, Abbeel, Jordan, & Moritz, 2015). Lo que buscan estos trabajos, además de proponer un algoritmo que optimiza directamente la política, es evitar que la política cambie radicalmente al momento de actualizar los parámetros del modelo, previniendo así que el agente disminuya drásticamente el rendimiento en caso que la actualización de los parámetros lleve a una mala política. A continuación, se describirá PPO, el algoritmo de optimización usado en el presente trabajo.

PPO actualiza los parámetros del modelo según la ecuación:

$$\theta_{k+1} = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{s,a\sim\pi_{\theta_k}}[L(s, a, \theta_k, \theta)] \tag{4.1}$$

La función L está definida por tres "sub-funciones":

$$L(s, a, \theta_k, \theta) = L^{CLIP}(s, a, \theta_k, \theta) - c_1 L^{VF}(s, a, \theta) + c_2 S[\pi_\theta](s) \tag{4.2}$$

En donde c_1 y c_2 son hiperparámetros de constantes positivas. Cada una de estas funciones serán explicadas a continuación.

La función L^{CLIP} se define como:

$$L^{CLIP}(s, a, \theta_k, \theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}, \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon\right) A^{\pi_{\theta_k}}\right) \tag{4.3}$$

Esta es la función objetivo principal de PPO. En ella, $A^{\pi_{\theta_k}}$ es la función *advantage*, la cual formalmente se define como $A^{\pi_{\theta_k}} = Q^{\pi_{\theta_k}}(s, a) - V^{\pi_{\theta_k}}(s)$ y cuantifica qué tan mejor, en cuanto a recompensa obtenida, es la acción a tomar a bajo el estado s con respecto al promedio de todas las acciones actuando bajo la política π_{θ_k}. Este valor es calculado usando el estimador *Generalized Advantage Estimation* (GAE) (Schulman, Moritz, Levine, Jordan, & Abbeel, 2016), el cual hace uso de las recompensas obtenidas por el agente y un estimador de la función de valor, el cual es detallado más adelante. El término $\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}$ mide el rendimiento de la política con los nuevos parámetros con respecto a la política anterior usando los datos de la política anterior. La función clip mantiene la razón $\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}$ en el intervalo $[1-\epsilon, 1+\epsilon]$, lo cual evita cambios muy bruscos de este valor. Por último, la función $\min$ toma el menor valor entre el término truncado y el término real, lo cual hace que se ignoren los cambios en la razón entre las política antigua y la nueva solo cuando la función *advantage* es positiva y se incluye cuando es negativa. De esta forma se penaliza más cuando la política está empeorando.

La función L^{VF} se define como:

$$L^{VF}(s, \theta) = (V_{\pi_\theta}(s) - \hat{V}^{true})^2 \tag{4.4}$$

En donde $V_{\pi_\theta}(s)$ es una estimación de la función de valor y $\hat{V}^{true}$ es una estimación de la verdadera función de valor utilizando las recompensas obtenidas en las trayectorias

ejecutadas por el agente. Para poder estimar $(V_{\pi_\theta}(s)$ y la acción a tomar por el agente, el modelo utilizado (una red neuronal en este caso), debe tener como *output* dos valores: un real que estima $V(s)$ y la acción a tomar por el agente. Si las acciones que debe tomar el agente son discretas, el *output* de la política corresponde a una *softmax* entre las posibles acciones que puede tomar el agente. Si las acciones son continuas, como es el caso de los ambientes utilizados en este trabajo, el *output* corresponde a un vector que indica el valor que debe tener cada componente de la acción que debe tomar el agente y, generalmente, están codificadas según una distribución normal multivariada con una matriz de covarianza diagonal. La figura 4.1 muestra un esquema de una red neuronal utilizada en PPO. Primero, cuenta con una arquitectura compartida que recibe el estado y luego se separa en dos subredes: una para el estimador de la función de valor (subred superior) y otra para la política (subred inferior). En el caso de la figura, estas subredes solo contienen una capa, pero podrían tener un número arbitrario de capas.

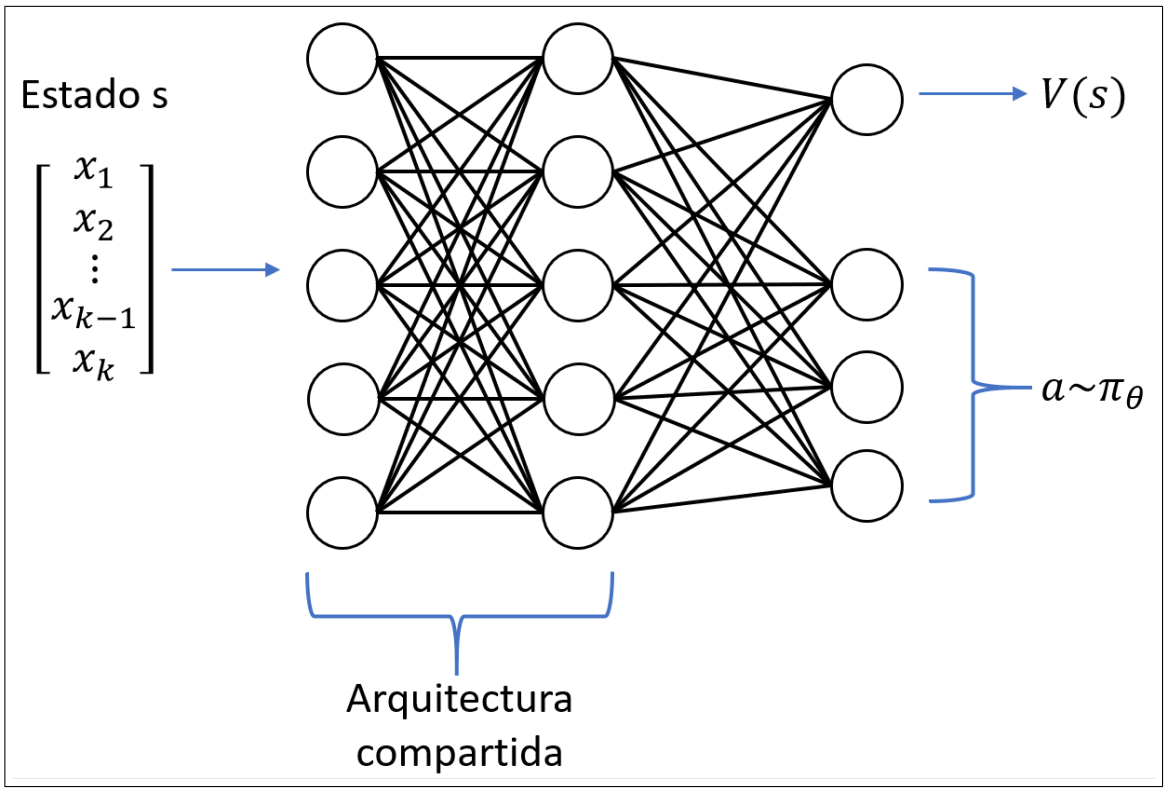


Figura 4.1. Estructura red neuronal para algoritmo PPO.

Finalmente la función $S[\pi_\theta](s)$ mide la entropía de la política. Como las acciones que debe tomar el agente son continuas codificadas mediante una distribución normal multivariada, este valor puede ser calculado. Esta función tiene la finalidad de fomentar la exploración de nuevas políticas por el modelo, debido a que se busca maximizar este valor, lo cual fuerza a que la política tenga más probabilidades de tomar acciones poco comunes.

El pseudocódigo del algoritmo PPO se muestra en el algoritmo 1 (Schulman et al., 2017). Un aspecto importante es que se pueden tener diversos agentes (actores) recopilando información para el entrenamiento, de esta forma se puede paralelizar el entrenamiento y que se converja a una buena política más rápido.

Algoritmo 1: Proximal Policy Optimization

for *iteración=1,2,...* **do**
  for *actor=1,2,...,N* **do**
    Ejecutar política $\pi_{\theta_{old}}$ en el ambiente por T *timesteps*.
    Calcular los estimadores de la función *advantage* $\hat{A}_1, ..., \hat{A}_T$
  end
  Optimizar función objetivo 4.2 con respecto a θ durante K épocas y un *minibatch* de tamaño $M \leq NT$
end

4.2. Arquitectura del modelo

Antes de describir las regularizaciones *sparse* que serán estudiadas, se describirá el modelo que se utilizará en los experimentos. Este modelo tiene como arquitectura compartida una red neuronal de dos capas ocultas con 128 y 256 neuronas (véase figura 4.1) con activaciones ReLU (Nair & Hinton, 2010). Estas capas son las que serán regularizadas, ya que es la principal estructura del modelo y el *output* de esta se utiliza tanto para el estimador de la función del valor como para la decisión de la acción que debe tomar el

agente. En adelante se referirá a los parámetros de esta estructura como θ_i y a las activaciones como y_i donde i corresponde a la capa de la estructura compartida. Cuando se hable de los pesos de una neurona, estos serán representados como $\theta_i[:, k]$ en donde i indica la capa de la estructura compartida y k la neurona en particular. Por ejemplo, $\theta_1[:, 23]$ hace referencia a los pesos de la neurona 23 de la primera capa de la estructura compartida, la cual tiene 128 neuronas.

Para el estimador de la función de valor se ocupó una capa de una neurona, ya que solo se necesita calcular un escalar. Para el estimador de la acción a tomar por el agente se ocupó una capa de N neuronas, donde N es la dimensionalidad de la acción que puede realizar el agente. En ambos casos el *input* que reciben son las *features* que se obtienen de la arquitectura compartida. Para la estimación de la función de valor y para la decisión de la acción a tomar se escogió una estructura mínima (solo una capa) debido a que no se quería diferenciar el nivel de *sparsity* para cada estimador, sino que se quería analizar su efecto en el modelo en su totalidad.

4.3. Técnicas de regularización *sparse*

Para obtener modelos que utilicen una menor cantidad de neuronas se compararán distintas regularizaciones. Algunas de estas explícitamente buscarán llevar a cero parámetros o neuronas (regularización L_1 y *Sparse Group Lasso*), mientras otras tienen como objetivo aprender representaciones *sparse* (*Set KL-Divergence*), las cuales fueron estudiadas por Hernandez-Garcia y Sutton (2019).

4.3.1. Regularización L_1 y L_2

Las regularizaciones L_1 y L_2 han sido ampliamente usadas en inteligencia artificial, tanto para disminuir la cantidad de operaciones realizadas y parámetros usados por los modelos en aprendizaje supervisado (Han, Pool, Tran, & Dally, 2015; Yang et al., 2019) como para reducir el *overfitting* (Gupta, Gupta, Ojha, & Singh, 2018; Ying, 2019). Este

trabajo centra el estudio de estas regularizaciones para la simplificación de modelos en un contexto de aprendizaje reforzado, lo cual se puede lograr aplicando estas penalizaciones tanto a las activaciones como los pesos de una red neuronal, sin embargo, en la mayoría de los trabajos se estudia su efecto sobre los parámetros.

Las regularizaciones L_1 y L_2 aplicada sobre los pesos se define como:

$$L_1^{Pesos} = \sum_{i=0}^{L} ||\theta_i||_1 \tag{4.5}$$

$$L_2^{Pesos} = \sum_{i=0}^{L} ||\theta_i||_2^2 \tag{4.6}$$

En donde L es la cantidad de capas de la red neuronal, θ_i son los parámetros de la capa i de la red neuronal, $|| \cdot ||_1$ es la norma L_1 y $|| \cdot ||_2^2$ es la norma L_2. Intuitivamente, la norma L_1 dirige los valores de los pesos a cero, mientras que la norma L_2 mantiene los valores de los pesos pequeños. Esto tiene dos efectos distintos sobre la red, ya que por una parte, la regularización L_1 fomenta que los valores de los pesos sean cero, lo cual significa que el modelo podría prescindir de ese parámetro, ya que todo *input* multiplicado por ese parámetro siempre tendrá el valor cero. Por otra parte, la regularización L_2 fomenta que el valor absoluto de los parámetros no sea muy grande. La figura 4.2a muestra visualmente la aplicación de estas regularizaciones sobre los pesos.

Para el caso de la regularización sobre las activaciones, sea x el input de la red neuronal (un estado u observación del ambiente), g la función de activación y $y_i = g(\theta_i^\intercal x)$ las activaciones de la capa i, las regularizaciones quedan definidas por:

$$L_1^{Activaciones} = \sum_{i=0}^{L} ||y_i||_1 \tag{4.7}$$

$$L_2^{Activaciones} = \sum_{i=0}^{L} ||y_i||_2^2 \tag{4.8}$$

Este caso es análogo al de los pesos de la red, sin embargo, aplicado sobre el *output* de la neurona. Nótese que en este caso, la norma L_1 busca que una neurona tenga un *output* cercano a cero, esto implicaría que la información codificada por esa neurona no le sirve al modelo, ya que para cualquier *input*, el resultado será cero. Dado esto, uno podría prescindir de todos los pesos de esa neurona y de todos los pesos que reciben como *input* la información de esa neurona, ya que su valor siempre será cero. Esto, a diferencia de la regularización aplicada sobre los pesos, podría llevar a una mayor simplificación del modelo.

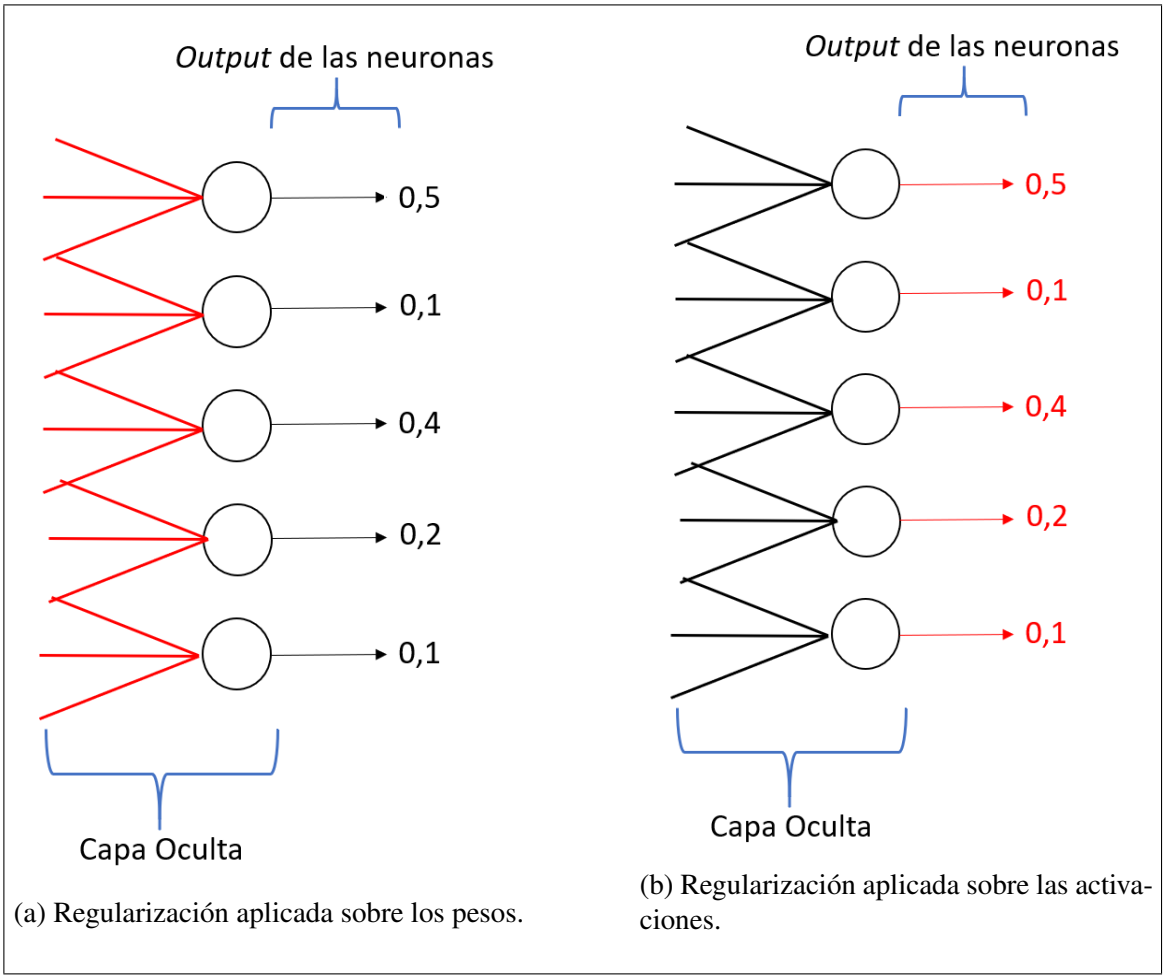
La norma L_2 tiene un objetivo similar al caso aplicado a los parámetros: evitar que el *output* de las neuronas tenga un valor absoluto muy grande. Dado esto, se espera que la regularización L_1 fomente modelos con mayor nivel de *sparsity* que la norma L_2. La figura 4.2b muestra visualmente la aplicación de estas regularizaciones sobre las activaciones.

4.3.2. Regularización estructural: *Sparse Group Lasso*

Group Lasso es un método utilizado principalmente en regresión que tiene como finalidad seleccionar grupos de variables que no son relevantes para el modelo (Yuan & Lin, 2006). De esta idea nace la regularización *Sparse Group Lasso* en redes neuronales (Scardapane et al., 2016), la cual busca que un modelo utilice la menor cantidad de neuronas haciendo que grupos de pesos sean cero. Para este trabajo, se ocupará una variación de *Sparse Group Lasso*, la cual busca en que todos los pesos de una neurona sean cero. Esto indicaría indicaría que la información codificada por esta neurona no es relevante para el modelo y asemejaría el efecto que produce $L_1^{Activaciones}$, pero, esta vez, a través de los pesos de una neurona.

Sea $\theta_i[:, k]$ los pesos de la neurona k de la capa i, la regularización *Sparse Group Lasso* queda definida por:

$$SGL^{Pesos} = \sum_{i=0}^{L} \sum_{k=0}^{N_i} \sqrt{|\theta_i[:, k]|} \cdot ||\theta_i[:, k]||_2 + \sum_{i=0}^{L} ||\theta_i||_1 \tag{4.9}$$



(a) Regularización aplicada sobre los pesos.

(b) Regularización aplicada sobre las activaciones.

Figura 4.2. Esquema de aplicación de regularizaciones sobre los pesos y activaciones del modelo. En rojo se muestra el componente regularizado.

En donde $|\cdot|$ corresponde a la cantidad de elementos y N_i es la cantidad de neuronas de la capa i. En la ecuación 4.9, el primer término está agrupando todos los pesos de una neurona y llevándolos a un valor pequeño, mientras que el segundo término busca llevar los valores de los parámetros a cero. En conjunto, se busca que todos los parámetros de una neurona sean cero, por lo que el *output* no aportaría información al modelo y, por ende, se podría prescindir de esa neurona. La figura 4.3 muestra visualmente la aplicación de esta regularización sobre los pesos.

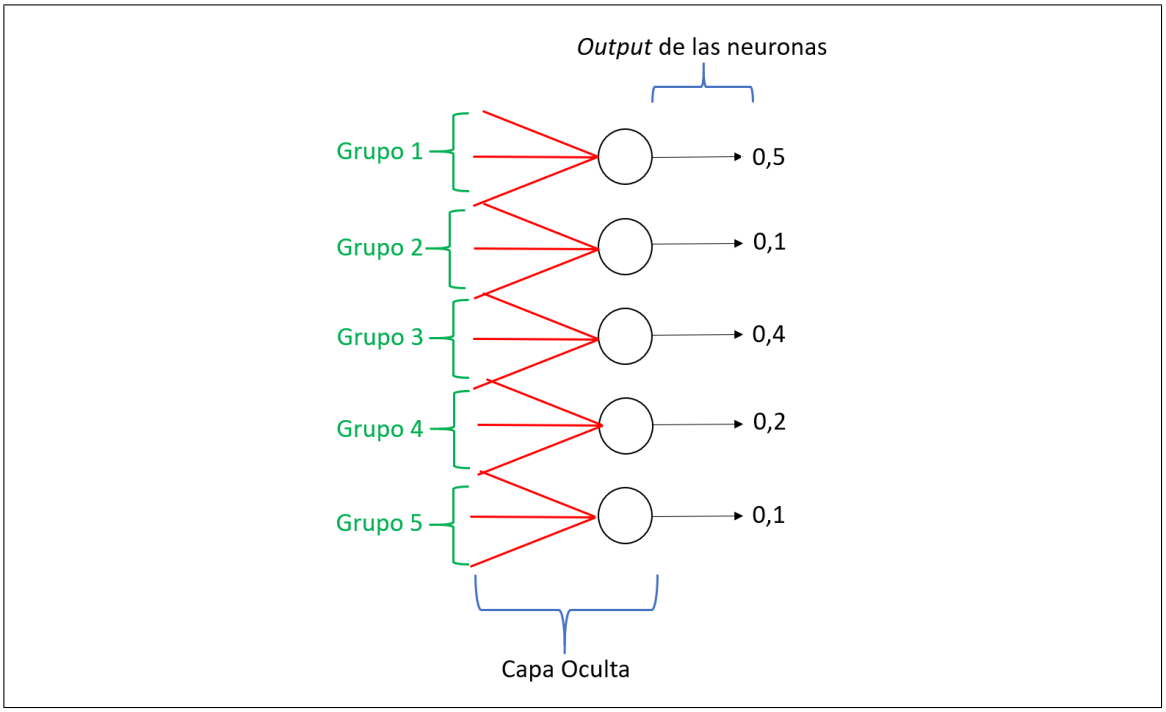


Figura 4.3. Esquema de aplicación de regularización SGL sobre los pesos el modelo. El componente rojo representa los pesos, mientras que en verde se muestran los grupos de pesos regularizados.

4.3.3. Regularización basada en distribución de probabilidad: *Set KL-divergence*

Esta regularización busca modelar las activaciones de las neuronas mediante una distribución de probabilidad de la familia exponencial de parámetro natural β, el cual especifica el grado de *sparsity*. Formalmente, se modela la activación de una neurona k de la capa i de una red neuronal como $y_{i,k} \sim p_\beta$.

Este tipo de regularización ha demostrado que es capaz de generar representaciones *sparse* en RL (Hernandez-Garcia & Sutton, 2019; Liu et al., 2019), sin embargo, como se dijo en la sección 2, el objetivo de esta regularización es generar representaciones que no se sobrelapen entre sí y no generar neuronas inactivas. A pesar de esto, se incluye en el presente estudio para analizar más a fondo su efecto en cuanto a la disminución de la cantidad de parámetros usados.

Para lograr que las activaciones de una neurona se comporte como una distribución de la familia exponencial, la regularización penaliza de acuerdo a cuánto difiere la distribución empírica ($p_{\hat{\beta}}$) de la distribución objetivo (p_β). Esto se mide usando la KL-divergencia entre ambas distribuciones, $KL(p_\beta||p_{\hat{\beta}})$. Sin embargo, dado que es difícil que las activaciones de una neurona sigan exactamente una distribución en particular, Liu et al. (2019) definen una medida de distancia en donde se busca que la distribución empírica no se aleje de un conjunto de distribuciones en vez de una distribución única. A esta medida le denominan *Set KL-divergence* (SKL) que se define como:

$$SKL(Q_B||p_\eta) := \min_{p \in Q_B} KL(p||p_\eta) \tag{4.10}$$

En donde p_η es una distribución de la familia exponencial de una dimensión con parámetro natural η, $B = [\eta_1, \eta_2]$ es un conjunto convexo en el espacio de parámetros naturales y $Q_B = \{p_\eta : \eta \in B\}$, es decir, el conjunto de distribuciones que poseen parámetro natural en el intervalo B. Ellos demuestran que, bajo estas condiciones, cuando se tiene $B = (0, \beta]$ y una distribución exponencial, SKL puede reescribirse como:

$$SKL_{exp}(Q_B||p_\eta) = \begin{cases} \log \hat{\beta} + \frac{\beta}{\hat{\beta}} - \log \beta - 1 & \text{if } \hat{\beta} > \beta \\ 0 & \text{otherwise} \end{cases} \tag{4.11}$$

En donde el parámetro $\hat{\beta}$ es estimado mediante el promedio de las activaciones de una neurona. Esta penalización busca que las activaciones de las neuronas se comporten como una distribución exponencial, por ende, mientras menor sea el parámetro β, se espera que las activaciones de las neuronas se concentren en valores menores a β, lo cual produciría que la mayoría de las activaciones sean pequeñas y, así, se obtendrían representaciones *sparse* dentro del modelo. Esta penalización se calcula para cada neurona y se suman para obtener el valor final que se añade a la función de pérdida.

4.4. Regularización aplicada a PPO

Para incluir cualquiera de las regularizaciones anteriores al proceso de entrenamiento del agente usando el algoritmo PPO, se añadió el valor de la regularización a la función objetivo de PPO definida en la ecuación 4.2:

$$L(s, a, \theta_k, \theta) = L^{CLIP}(s, a, \theta_k, \theta) - c_1 L^{VF}(s, a, \theta) + c_2 S[\pi_\theta](s)] - c_3 L^{REG} \quad (4.12)$$

En donde L^{REG} corresponde a cualquiera de las regularizaciones definidas anteriormente y c_3 es el coeficiente que mide qué tanto aportará la regularización a la función objetivo de la optimización. Nótese que solo para el caso de la regularización SKL es necesario dos parámetros: c_3 y β. Este último corresponde al parámetro natural de la distribución exponencial.

Los rangos de valores del parámetro c_3 variará según la regularización aplicada, ya que todas poseen escalas distintas dependiendo de lo que se esté penalizando. Por ejemplo, la regularización L_2^{Pesos} tendrá una escala muy distinta a $L_2^{Activaciones}$, debido a que los valores de los pesos y los de las activaciones podrían llegar a diferir bastante. Por esto, se hará un proceso de selección de rangos de valores que puede tomar c_3 para cada regularización antes de decidir los parámetros finales para evitar así inestabilidades en el entrenamiento o que el agente no pueda aprender correctamente una política debido a que la penalización es muy alta comparada con las otras funciones a optimizar.

Además de variar c_3 debido a la escala de los valores que pueden tomar las regularizaciones, este parámetro debe tomar en cuenta los rangos de valores en los que se mueven las otras funciones de pérdida, ya que estas no tienen un comportamiento estable como el caso de aprendizaje supervisado, sino que varían durante entrenamiento a medida que el agente aprende nuevas políticas. Por ejemplo, la función de pérdida L^{VF}, la cual mide qué tan bien se está estimando la función de valor, no mantiene un decrecimiento constante, sino que cada vez que el agente actualiza su política y se encuentra con estados que no ha visto antes, el valor de esta función probablemente aumentará debido a que no estimará

bien la función de valor dado que se tratan de estados nuevos. Por lo tanto, oscila durante gran parte del entrenamiento. Lo mismo pasa con la función de pérdida L^{CLIP}, ya que ocupa tanto el estimador de la función de valor como la razón entre la probabilidad de una acción en la política nueva y la antigua, cuyos valores también van cambiando a medida que el agente aprende su política.

Uno de los detalles en PPO que reduce la variabilidad de las funciones de pérdida a medida que el agente aprende, es la normalización aplicada a las recompensas y observaciones que obtiene el agente durante el proceso de entrenamiento. Esto hace que, durante entrenamiento, a pesar de que el agente vaya aumentando la recompensa obtenida y vaya viendo distintos estados, los rangos de valores de estos términos se mantienen relativamente constantes, por lo que los pesos no necesitan cambiar tanto para adaptarse a los nuevos estados y la función de pérdida mantiene un rango de valores similares que hace el entrenamiento más estable.

5. EXPERIMENTOS

5.1. Ambientes de simulación

Los ambientes de simulación ocupados para entrenar los modelos se basan en el control locomotor de un agente/robot, ya que tienen la complejidad de tener como *input* estados continuos y las acciones que puede realizar son continuas, lo cual se asemeja más a un ambiente real. Se eligieron agentes con distinta cantidad de articulaciones y formas corporales para poder analizar el efecto de la regularización *sparse* en diversos contextos. Estos ambientes poseen un estado que no está representados por imágenes, debido a que se quería eliminar la dificultad de la tarea de reconocimiento visual y poder centrarse en la capacidad del agente para aprender una política. Por lo tanto, se utilizaron los ambientes *BipedalWalker* de la librería *Gym* (Brockman et al., 2016) y los ambientes *Ant*, *HalfCheetah* y *Walker2D* de la librería *PyBullet* (Coumans & Bai, 2016–2019). En la tabla 5.1 se muestra una breve descripción de las dimensionalidades de los estados y acciones y cantidad de extremidades de cada ambiente. Cada componente de la acción que realiza el agente es el torque aplicado a las articulaciones del robot.

Tabla 5.1. Dimensionalidad de las observaciones y las acciones y cantidad de extremidades de los distintos ambientes utilizados.

Ambiente	Dim. Observación	Dim. Acciones	Extremidades
BipedalWalker	24	4	2
Ant	28	8	4
HalfCheetah	26	6	2
Walker2D	22	6	2

Ant, *HalfCheetah* y *Walker2D* son similares entre sí en cuanto al objetivo de la simulación, ya que los tres deben avanzar lo más rápido posible sobre un plano liso. La diferencia entre ellos son los puntos de contacto con el suelo, la cantidad de articulaciones

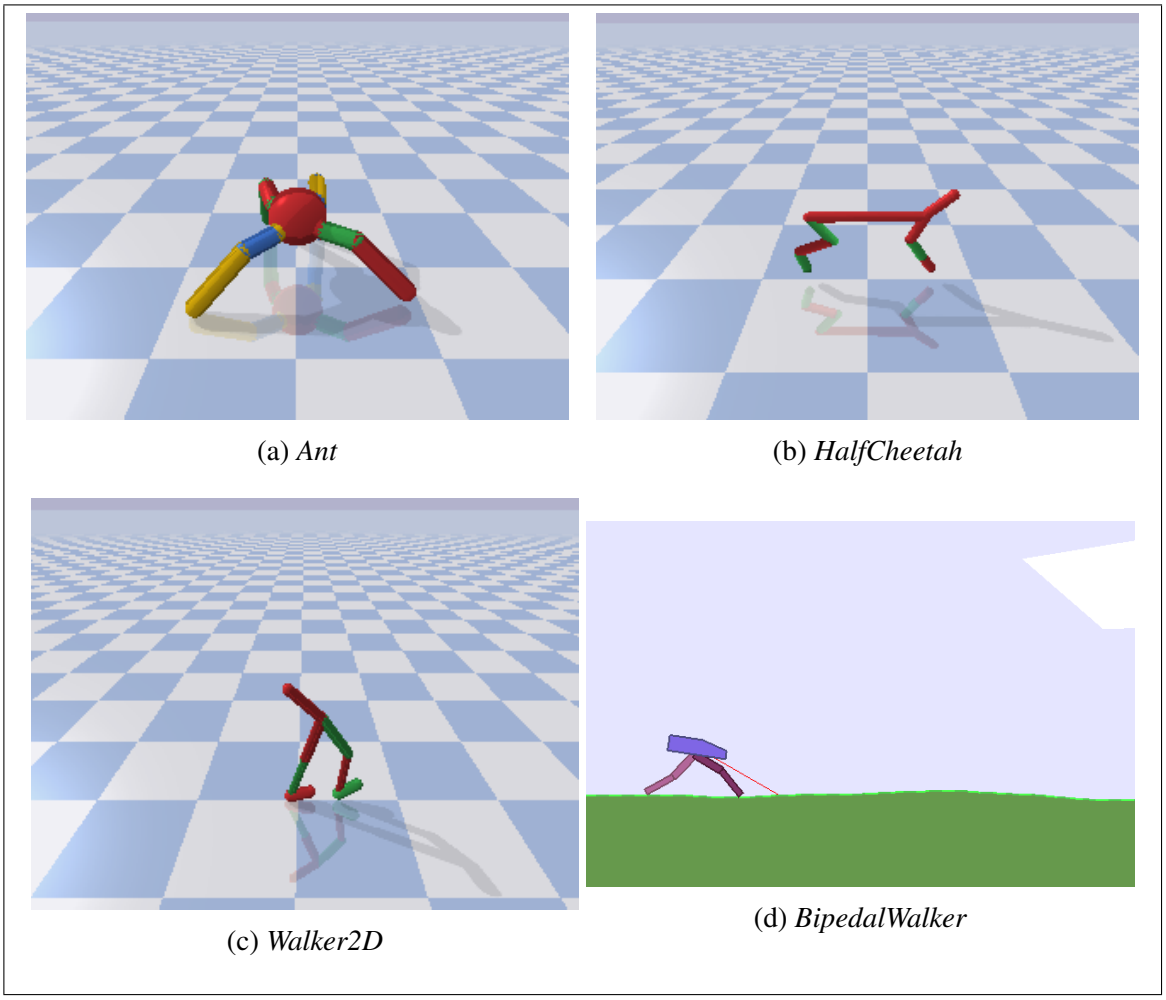
y la estructura corpórea, lo cual hace variar tanto la dimensionalidad de la observación como la dimensionalidad de la acción. Los estados de estos ambientes están determinados por la velocidad angular de las articulaciones, la posición del agente y la velocidad de este. A pesar de que el objetivo sea similar, la política a aprender varía mucho entre ellos, ya que *Ant* posee cuatro extremidades, *HalfCheetah* dos extremidades con cuerpo horizontal y *Walker2D* dos extremidades con el cuerpo vertical.

BipedalWalker es distinto de los ambientes anteriores debido a que el terreno es irregular y no posee coordenadas como observación. Además, el objetivo es avanzar de manera eficiente por una determinada distancia en donde se penaliza si es que pierde el equilibrio y se bonifica si llega al final del recorrido. El estado está representado por las velocidades angulares y posiciones de las articulaciones, las velocidades del cuerpo y 10 mediciones de un "radar láser". La figura 5.1 muestra imágenes de todos los ambientes.

5.2. Entrenamiento

Los modelos fueron entrenados utilizando la implementación de PPO de la librería *stable-baselines* (Hill et al., 2018), cuya función de pérdida fue modificada de acuerdo a la ecuación 4.12. Los hiperparámetros del algoritmo que fueron utilizados se basaron en los *benchmarks* realizados por los autores de esta librería (Raffin, 2018).

Para cada ambiente se entrenaron modelos con regularización y sin regularización *sparse*. Los hiperparámetros propios del algoritmo PPO se mantuvieron constantes para cada ambiente de simulación. Para medir el desempeño del agente durante entrenamiento, se registró el promedio de las recompensas obtenidas por los ambientes de entrenamiento durante los últimos 100 episodios en cada iteración de PPO. Esta curva de entrenamiento fue utilizada para definir los mejores coeficientes de regularización, los cuales fueron encontrados a través de *grid search* entre distintos valores (para mayor detalle, véase apéndice A). Los coeficientes que se presentarán en los resultados fueron los que obtuvieron mejores curvas de entrenamiento y mayor nivel de *sparsity*, lo cual se traduce en el



(a) *Ant*

(b) *HalfCheetah*

(c) *Walker2D*

(d) *BipedalWalker*

Figura 5.1. Imágenes de los ambientes de simulación.

coeficiente de regularización más grande sin disminuir considerablemente el rendimiento durante entrenamiento. Cabe destacar que este trabajo se centró en obtener el mayor coeficiente de regularización posible y no en obtener el mejor rendimiento posible. Esto con el fin de verificar si es posible aprender representaciones *sparse*. Se repitieron 5 veces los experimentos con los hiperparámetros finales para tomar en cuenta la variabilidad del algoritmo PPO, por lo tanto, todos los gráficos de la sección 6 muestran la recompensa promedio obtenido entre las 5 ejecuciones del algoritmo. Las desviaciones estándar de estas ejecuciones se encuentran en el apéndice B.

La curva de entrenamiento del agente representa un rendimiento aproximado, debido a que es una muestra pequeña que podría tener mucha variabilidad y que mezcla distintas políticas dado que esta está en constante cambio. Por esto, se guardaron periódicamente *checkpoints* del modelo para ser evaluados una vez terminado el entrenamiento. Estos *checkpoints* fueron evaluados durante 300 episodios con distintas *seeds* para lograr un buen estimador del rendimiento y su variabilidad.

6. RESULTADOS

6.1. Entrenamiento

El primer aspecto que se analizó, con respecto a la aplicación de regulación *sparse* al proceso de entrenamiento, es el rendimiento durante entrenamiento y la velocidad de convergencia del algoritmo. El objetivo de esto es comprobar si la aplicación de regularización a PPO gatilla un aumento o disminución del rendimiento del agente con respecto al algoritmo sin regularización y si la convergencia se ve afectada positiva o negativamente. Cabe destacar que, durante entrenamiento, el modelo obtiene una muestra de la distribución de la acción que puede tomar, por lo que la recompensa obtenida podría ser menor que la que obtendría si eligiera determinísticamente la mejor acción.

En la figura 6.1 cada punto de las curvas corresponde a la recompensa promedio de los últimos 100 episodios de los ambientes de simulación utilizados durante entrenamiento. En todos los ambientes, la curva del modelo sin regularización se mantiene por sobre casi todas las curvas de los modelos con regularización, lo cual indicaría, en primera instancia, que el no tener ninguna restricción extra, ayuda al agente a aprender más rápido y mejor. La regularización que mejor curva de entrenamiento obtiene en casi todos los casos es $L_2^{Activaciones}$, la cual promueve activaciones pequeñas, evitando así que se sobreajusten valores dentro del modelo para que ciertas activaciones tengan un valor muy diferente al de otras activaciones dentro de la misma capa. La regularización que sigue en cuanto a rendimiento durante entrenamiento es SKL_{exp}, la cual promueve representaciones *sparse* en las activaciones de las neuronas sin forzar el apagado estas (lo cual logra, como se verá en la sección 6.3). Esto último puede ser la razón del buen rendimiento comparado con otras regularizaciones en que explícitamente se trata de llevar valores a cero tanto en pesos como en activaciones.

Las regularizaciones que explícitamente intentan disminuir la complejidad del modelo, $L_1^{Activaciones}$, L_1^{Pesos} y SGL^{Pesos}, tienen una curva de rendimiento que demora más

en converger y, cuando lo hacen, llegan a un rendimiento inferior que las otras regularizaciones. Sin embargo, esto no significa que en evaluación el rendimiento sea inferior, como ser verá en la sección 6.2. Este bajo rendimiento durante entrenamiento puede deberse a que el modelo no tiene la total libertad de expresión para lograr una mejor política. En general, $L_1^{Activaciones}$ es la que logra converger a una mejor recompensa y es la única de este tipo de regularizaciones que es aplicada sobre las activaciones, lo cual indicaría que lograr *sparsity* a nivel estructural, es más efectivo a través de las activaciones, como también se vio en el caso supervisado cuando se exploraba el problema de aprendizaje continuo (Aljundi et al., 2019).

La regularización L_2^{Pesos} es un caso particular debido a que no intenta llevar valores a cero en los pesos, sino que promueve valores pequeños en estos. Sin embargo, no logra un buen rendimiento durante entrenamiento, lo cual indicaría que aplicar regularización a nivel de pesos en un contexto de aprendizaje reforzado, haga más difícil que el agente aprenda una buena política.

6.2. Evaluación de las políticas encontradas durante entrenamiento

Un aspecto que llamó la atención durante el análisis del proceso de entrenamiento es que habían casos en que las curvas de entrenamiento de los distintos tipos de regularización no se comportaban igual que las evaluaciones de los *checkpoints* del agente registrados durante entrenamiento. Por esto, se analizaron las curvas de las evaluaciones de estos *checkpoints*, las cuales se construyeron a partir de un promedio de la recompensa obtenida durante 300 episodios para cada *checkpoint*, como se dijo en la sección 5.2. Además, se tomaron en cuenta las 5 ejecuciones del algoritmo para introducir la variabilidad de PPO (véase apéndice B.2).

La figura 6.2 muestra las evaluaciones de los 50 *checkpoints* guardados durante entrenamiento para cada tipo de regularización y para cada ambiente. Lo primero que llama

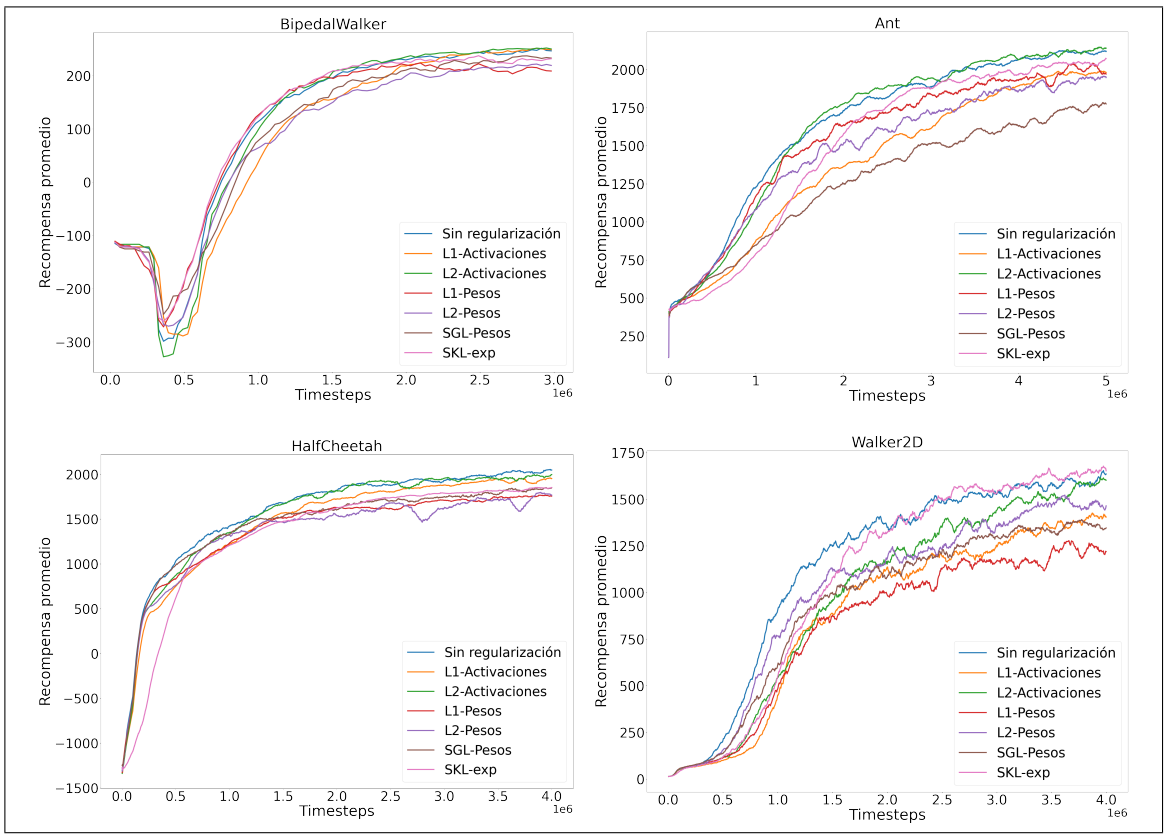


Figura 6.1. Recompensa promedio durante el entrenamiento de los distintos ambientes de simulación con las regularizaciones estudiadas. Cada curva corresponde al promedio entre 5 ejecuciones del algoritmo.

la atención, es que el comportamiento es más errático que lo calculado durante entrenamiento. Esto puede deberse a que las curvas de entrenamiento muestran un promedio de los últimos 100 episodios, los cuales pueden integrar diversas políticas en esta métrica, ya que cambia constantemente. En cambio, en la figura 6.2 se muestra una evaluación de una política en particular. Este comportamiento errático puede ayudar a ver que ciertas regularizaciones generan políticas más inestables que otras.

En *BipedalWalker* y *Walker2D* el modelo sin regularización se comporta de manera más inestable, lo cual hace que regularizaciones que durante entrenamiento parecían converger a una menor recompensa, como es el caso de $L_1^{Activaciones}$, igualen su valor o incluso

lo superen. Más aún, $L_1^{Activaciones}$ logra una convergencia muy similar al modelo sin regularización durante evaluación a pesar de que en esta regularización se busque disminuir la complejidad del modelo.

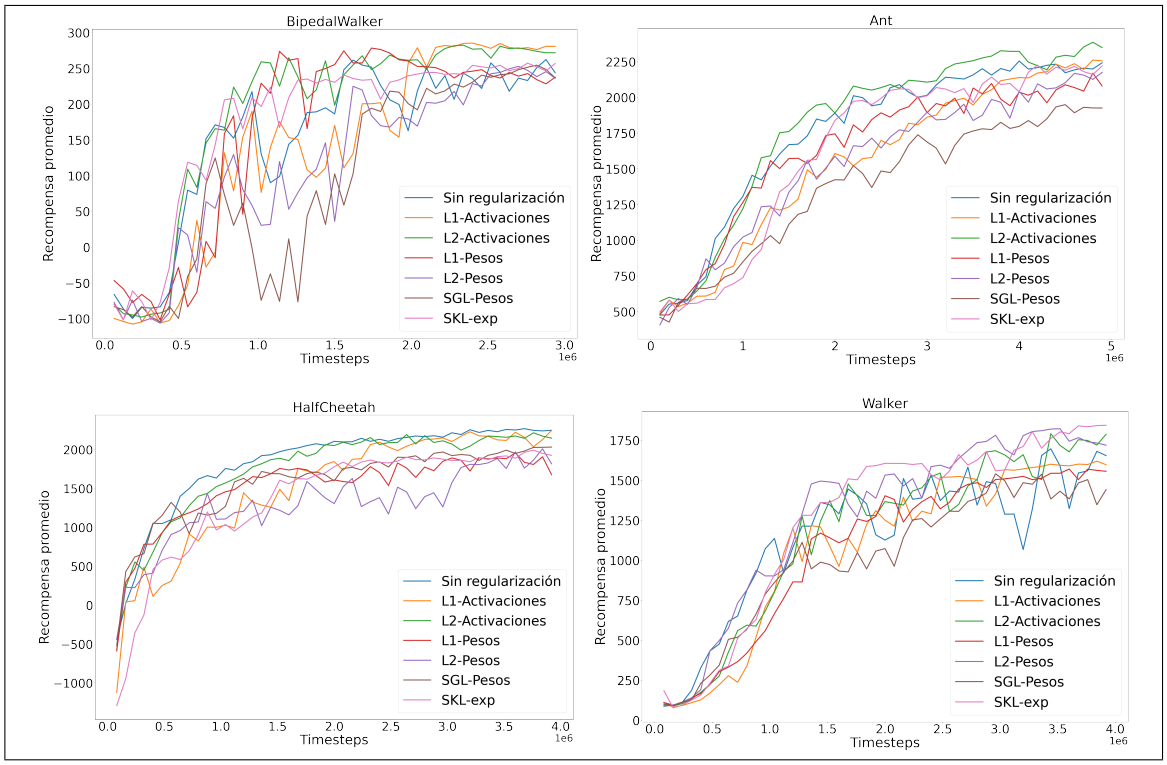


Figura 6.2. Recompensa promedio de los *checkpoints* guardados durante entrenamiento para cada regularización. Para lograr un buen estimador del rendimiento, cada *checkpoint* fue evaluado durante 300 episodios. Cada curva corresponde al promedio entre las 5 ejecuciones de PPO.

Como se dijo en la sección 5.2, cada ejecución de PPO se repitió 5 veces debido a que distintas ejecuciones se pueden comportarse diferente. La variabilidad de las repeticiones en cuanto al rendimiento de los *checkpoints* se puede ver en el apéndice B.2, sin embargo, para poder comparar las evaluaciones más fácilmente, en la tabla 6.1 se muestra el promedio y desviación estándar del rendimiento de los últimos 5 *checkpoints* de cada ambiente. Esto corresponde al último 10% del entrenamiento, lo cual muestra el grado de convergencia de las distintas configuraciones. Como se puede ver, las regularizaciones basadas en normas convexas sobre las activaciones, $L_1^{Activaciones}$ y $L_2^{Activaciones}$, logran

una mayor recompensa promedio y menor variabilidad al converger, excepto para el caso de *HalfCheetah*, en el cual logra uno de los mejores resultados pero no logra superar al modelo sin regularización. Esto indicaría que las regularizaciones sobre las activaciones, en un contexto de aprendizaje reforzado basado en métodos de gradiente, logran mejores resultados y más estables que las regularizaciones aplicadas a los pesos. De hecho, la regularización SGL^{Pesos} que tiene el mismo objetivo de disminuir la cantidad de neuronas que ocupa el modelo que $L_1^{Activaciones}$, logra un rendimiento considerablemente inferior que $L_1^{Activaciones}$ y, como se verá en la sección 6.3, en general logra menor *sparsity*.

Tabla 6.1. Recompensa promedio de los últimos 5 *checkpoints* de cada ambiente tomando en cuenta las 5 ejecuciones de PPO. Esto corresponde al último 10% del entrenamiento. Cada *checkpoint* fue evaluado durante 300 episodios.

Regularización	*BipedalWalker*	*Ant*	*HalfCheetah*	*Walker2D*
Sin regularización	245 ± 116	2202 ± 273	2250 ± 440	1553 ± 539
$L_1^{Activaciones}$	279 ± 72	2231 ± 222	2157 ± 369	1602 ± 199
$L_2^{Activaciones}$	274 ± 71	2334 ± 164	2170 ± 388	1738 ± 329
L_1^{Pesos}	236 ± 113	2090 ± 277	1837 ± 468	1553 ± 509
L_2^{Pesos}	244 ± 109	2146 ± 293	1884 ± 501	1741 ± 437
SGL^{Pesos}	248 ± 107	1925 ± 244	2003 ± 417	1432 ± 470
SKL_{exp}	252 ± 104	2194 ± 178	1943 ± 209	1831 ± 283

6.3. *Sparsity*

Una vez confirmado que es posible aplicar regularizaciones *sparse* tanto a nivel estructural como a nivel de representaciones, se analizó el nivel de *sparsity* estructural logrado por cada una de las regularizaciones, es decir, cuán menos complejos son los modelos en cuanto al uso de neuronas. A continuación se explicarán y presentarán los experimentos realizados para hacer este análisis.

Para analizar el grado de *sparsity* que logran las regularizaciones estudiadas, se eligió el mejor agente encontrado durante entrenamiento para cada regularización y se evaluó el desempeño de este a medida que se forzaba el "apagado" de las neuronas. En este

experimento, “apagar” una neurona significa asignar el valor 0 a los pesos de la neurona y, para el caso de las regularizaciones aplicadas a las activaciones, también se asignó el valor 0 al *bias*, debido a que estas regularizan el *output* final de una neurona. El orden en que se apagaban las neuronas varía entre las regularizaciones aplicadas a los pesos y las regularizaciones aplicadas a las activaciones, debido a que las regularizaciones aplicadas a los pesos no regularizaban el *bias*. A continuación se detalla la forma en que se definió este orden:

- **Regularización sobre los pesos**: el orden se definió de acuerdo a la suma absoluta de los valores de los pesos de cada neurona. Se asumió que las neuronas que tenían menor suma absoluta, eran menos importantes para el cálculo de la acción del agente y, por ende, eran las primeras en ser desactivadas, asignando el valor 0 a los pesos de esas neuronas.

- **Regularización sobre las activaciones**: el orden se definió de acuerdo a la suma absoluta de los valores de las activaciones del modelo durante la ejecución de la política del agente durante N *timesteps*. Se asumió que las neuronas que tienen una menor suma absoluta de activaciones eran menos importantes para el cálculo de la acción del agente y, por ende, eran las primeras en ser desactivadas, asignando el valor 0 tanto a los pesos como al *bias* de esas neuronas.

Un aspecto importante a tomar en cuenta es que, tanto los valores de los pesos como los valores de las activaciones de las neuronas, variaban en cada capa, por lo que estos fueron normalizados. Así, el valor 1 correspondía al máximo valor de la suma absoluta de los pesos o activaciones y el valor 0 correspondía al valor mínimo. De esta forma fue posible apagar neuronas independientemente de la capa a la que pertenecían.

Dado este orden, se iteró evaluando el desempeño del agente a medida que se apagaba el 5% de las neuronas hasta que todas las neuronas eran apagadas. La figura 6.3 muestra el resultado de este experimento y en el apéndice B.3 se pueden ver los resultados con sus respectivas desviaciones estándar, ya que cada evaluación corresponde al promedio de 300 episodios. Como se puede ver, las regularizaciones que explícitamente disminuyen la

complejidad del modelo, $L_1^{Activaciones}$, L_1^{Pesos} y SGL^{Pesos}, son las que logran una menor cantidad de uso de neuronas sin disminuir el rendimiento del agente. Además, si tomamos en cuenta el rendimiento de convergencia mostrado en la tabla 6.1, $L_1^{Activaciones}$ es la que mejor rendimiento tiene y la que menos neuronas utiliza. L_1^{Pesos} y SGL^{Pesos} también logran obtener un modelo menos complejo, pero no en todos los casos. Probablemente esto se deba a que es más difícil para PPO conducir todos los parámetros de una neurona a cero que conducir la activación de una neurona a cero.

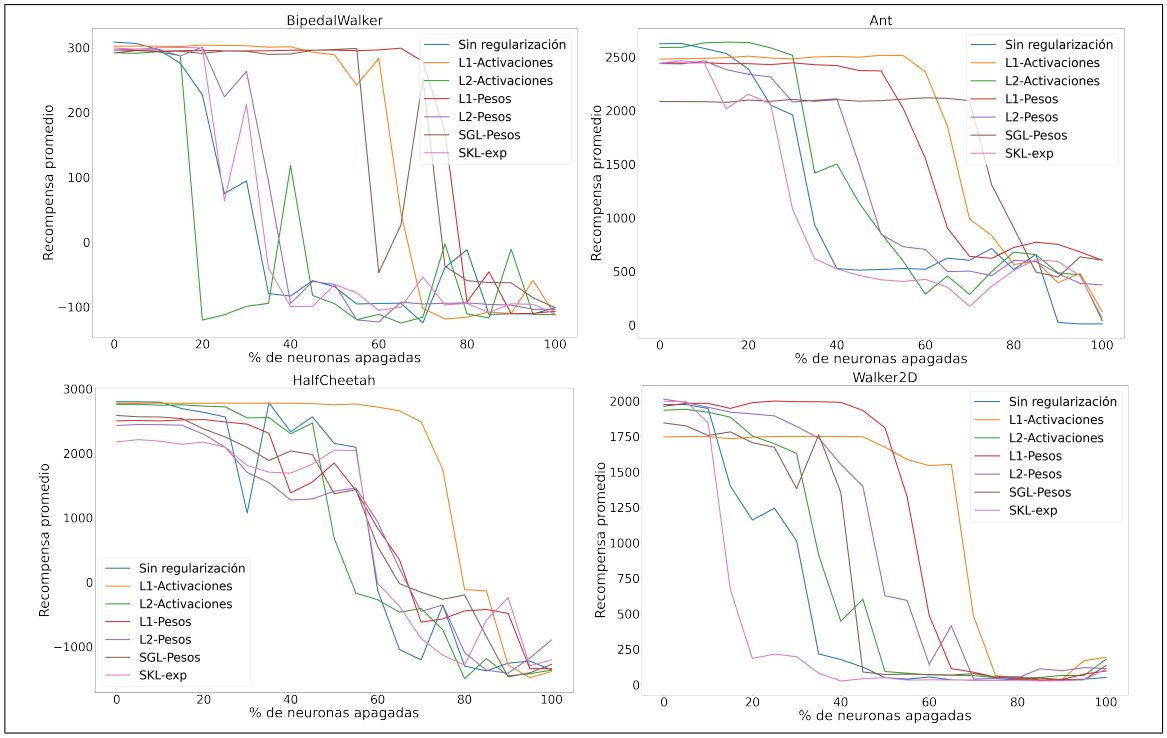


Figura 6.3. Rendimiento de los agentes a medida que se forzaba el apagado de las neuronas con menor activación. Para hacer este experimento, se escogió el modelo que mejor rendimiento obtuvo entre las 5 ejecuciones de los experimentos. En cada iteración un 5% de las neuronas fueron "apagadas" y el agente fue evaluado por 300 episodios.

Para complementar la figura 6.3, la tabla 6.2 muestra el nivel de *sparsity* alcanzado por las regularizaciones sin disminuir considerablemente la recompensa promedio. Se puede ver que $L_1^{Activaciones}$ logra un alto nivel de *sparsity* consistente en todos los ambientes, luego le sigue L_1^{Pesos} y por último SGL^{Pesos}. Lo cual confirma que una regularización

sparse aplicada a las activaciones logra un alto nivel de *sparsity* manteniendo un buen rendimiento.

Es importante notar que la regularización SKL_{exp} es la que menos *sparsity* a nivel estructural alcanza, lo que coincide con los resultados de Liu et al. (2019) y Hernandez-Garcia y Sutton (2019), los cuales afirman que esta regularización previene que hayan muchas neuronas muertas en el modelo.

Tabla 6.2. Porcentaje de las neuronas que pueden "apagarse" sin que el agente disminuya su rendimiento. Se escogió el modelo que mejor rendimiento obtuvo entre las 5 ejecuciones de PPO. Se consideró que el agente no disminuye su rendimiento si la recompensa promedio es mayor o igual que el 95% de la recompensa máxima obtenida por el agente.

Regularización	*BipedalWalker*	*Ant*	*HalfCheetah*	*Walker2D*
Sin regularización	10%	15%	35%	10%
L1-Activaciones	50%	55%	65%	50%
L2-Activaciones	15%	30%	25%	15%
L1-Pesos	65%	50%	30%	45%
L2-Pesos	20%	15%	15%	15%
SGL-Pesos	55%	70%	15%	35%
SKL-exp	20%	10%	20%	5%

Estos buenos resultados de la regularización $L_1^{Activaciones}$ tanto a nivel de *sparsity* como de rendimiento, en comparación con aquellas regularizaciones aplicadas sobre los pesos, como L_1^{Pesos}, puede deberse a que la regularización sobre las activaciones otorga una mayor "libertad" a los pesos para cambiar sus valores a medida que el agente se encuentra con nuevos estados, ya que no se penaliza el valor de cada peso para que este sea pequeño, sino que el *output* final de la neurona. En cambio, la regularización aplicada sobre los pesos restringe el valor de estos durante todo el entrenamiento haciendo que estos cada vez sean más pequeños, lo cual otorga menos flexibilidad para adaptarse a nuevos estados que podría encontrar el agente. El apéndice C muestra gráficos comparando $L_1^{Activaciones}$ y SGL^{Pesos} en cuanto a la recompensa y valor de la penalización a lo largo del entrenamiento.

Para verificar de manera cualitativa el grado de *sparsity* de los modelos, se escogió la regularización $L_1^{Activaciones}$ (por sus buenos resultados) y se visualizaron las activaciones de las neuronas del modelo durante un episodio del agente en cada ambiente. Para poder analizar mejor esta visualización, se ordenaron las neuronas de acuerdo a su similitud a lo largo del episodio. Para hacer esto, se redujo la dimensionalidad de las series de tiempo generadas por cada neurona (activaciones) a solo una componente usando *t-SNE* (Maaten & Hinton, 2008) y luego se ordenaron de acuerdo al valor que cada neurona obtuvo. La figura 6.4 muestra este resultado: activaciones sin regularización a la izquierda y con regularización $L_1^{Activaciones}$ a la derecha. Como era esperado, para el caso sin regularización, gran parte de las neuronas sí están siendo activadas durante la ejecución de la política, mientras que para el caso con regularización, es posible ver que gran parte de las neuronas están "apagadas", es decir, no poseen activación alguna (*output* 0). Por lo tanto, visualmente se puede confirmar que la política del agente puede reducirse a un pequeño número de neuronas.

6.4. Rendimiento vs *Sparsity*

Dado que se logró entrenar agentes con un alto porcentaje de *sparsity*, se exploró la relación entre el rendimiento del agente y el nivel de *sparsity* que obtuvo el modelo. Para esto, se utilizó la regularización que mejores resultados obtuvo tanto a nivel de *sparsity* como de rendimiento: $L_1^{Activaciones}$. El experimento consistió en ejecutar 5 veces el algoritmo PPO para una serie de coeficientes de regularización en cada ambiente. Luego, tal como se hizo para la tabla 6.1, se evaluaron los últimos 5 *checkpoints* de cada ejecución y se calculó el promedio y desviación estándar de la recompensa promedio obtenida. Para calcular el porcentaje de *sparsity*, se eligió el *checkpoint* que mejor rendimiento obtuvo en evaluación (entre las 5 ejecuciones del algoritmo) y, tal como se hizo para la tabla 6.2, se fueron apagando las neuronas menos importantes sin que el agente disminuyera más del 5% de su rendimiento máximo.

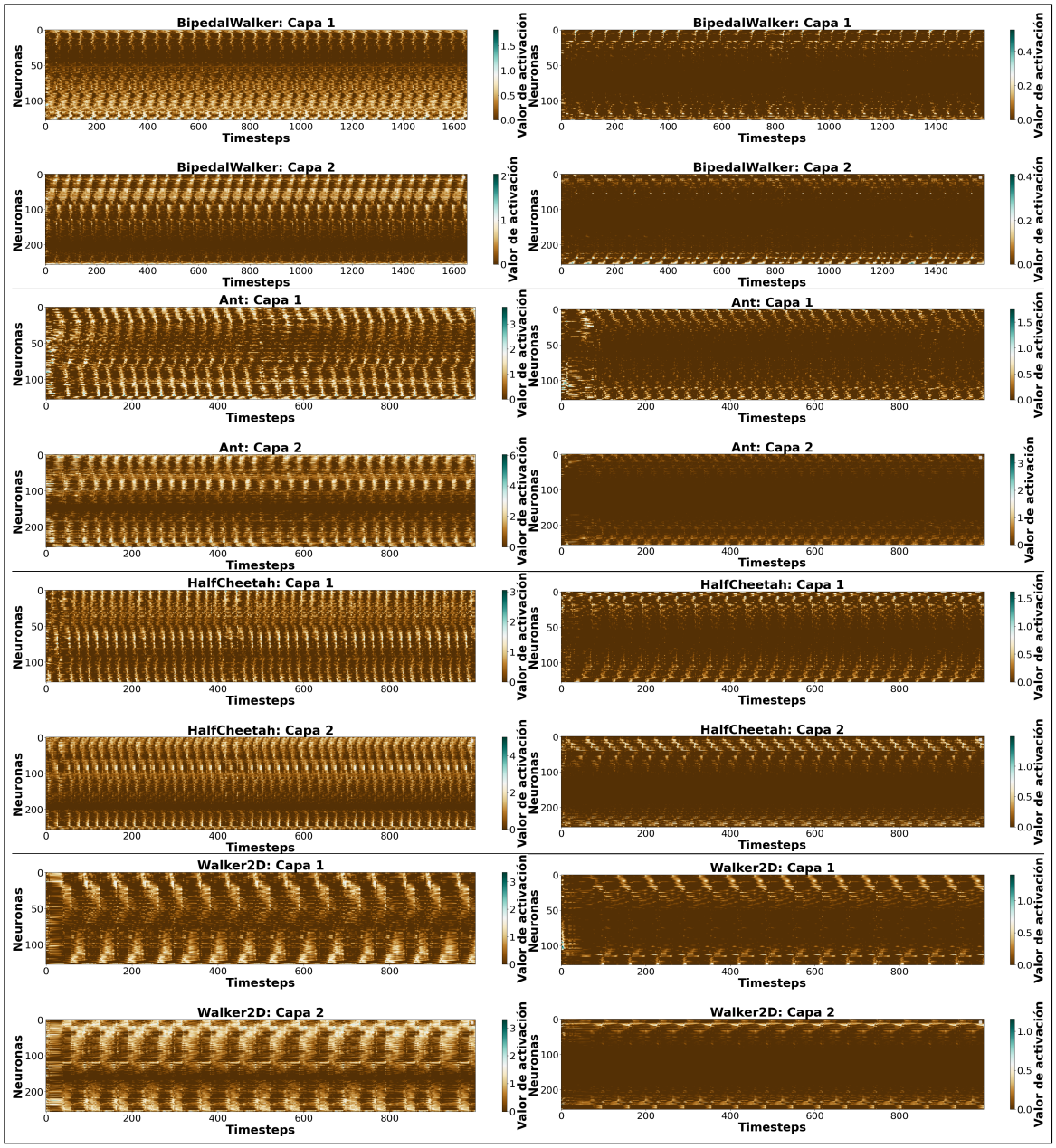


Figura 6.4. Activaciones de las neuronas durante un episodio de cada ambiente utilizando regularización $L_1^{Activaciones}$. A la izquierda se muestra el agente sin regularización y a la derecha el agente con regularización.

La figura 6.5 muestra el resultado de este experimento. En ella es posible ver que el agente, en todos los ambientes, mantiene o mejora levemente la recompensa promedio que obtiene a medida que el coeficiente de regularización aumenta. En algunos casos,

como *BipedalWalker* y *Walker2D* ayuda a disminuir la variabilidad del rendimiento del agente. Sin embargo, cuando el coeficiente de regularización llega a valores muy altos, el rendimiento disminuye considerablemente. A su vez, el nivel de *sparsity* también crece, esto se debe a que la penalización de la regularización cada vez es mayor, por lo que durante entrenamiento se fomenta más el uso de menos neuronas.

Estos resultados indican que la aplicación de regularización *sparse* en aprendizaje reforzado mantiene o mejora el rendimiento de un agente, como también puede disminuir la variabilidad en evaluación. Sin embargo, si el valor de la penalización es muy alto, perjudica el proceso de entrenamiento, lo cual se debe a que la regularización domina el gradiente al comienzo del entrenamiento, dejando pocas neuronas activas para aprender la política y estas no son suficientes para que el agente tenga un buen desempeño.

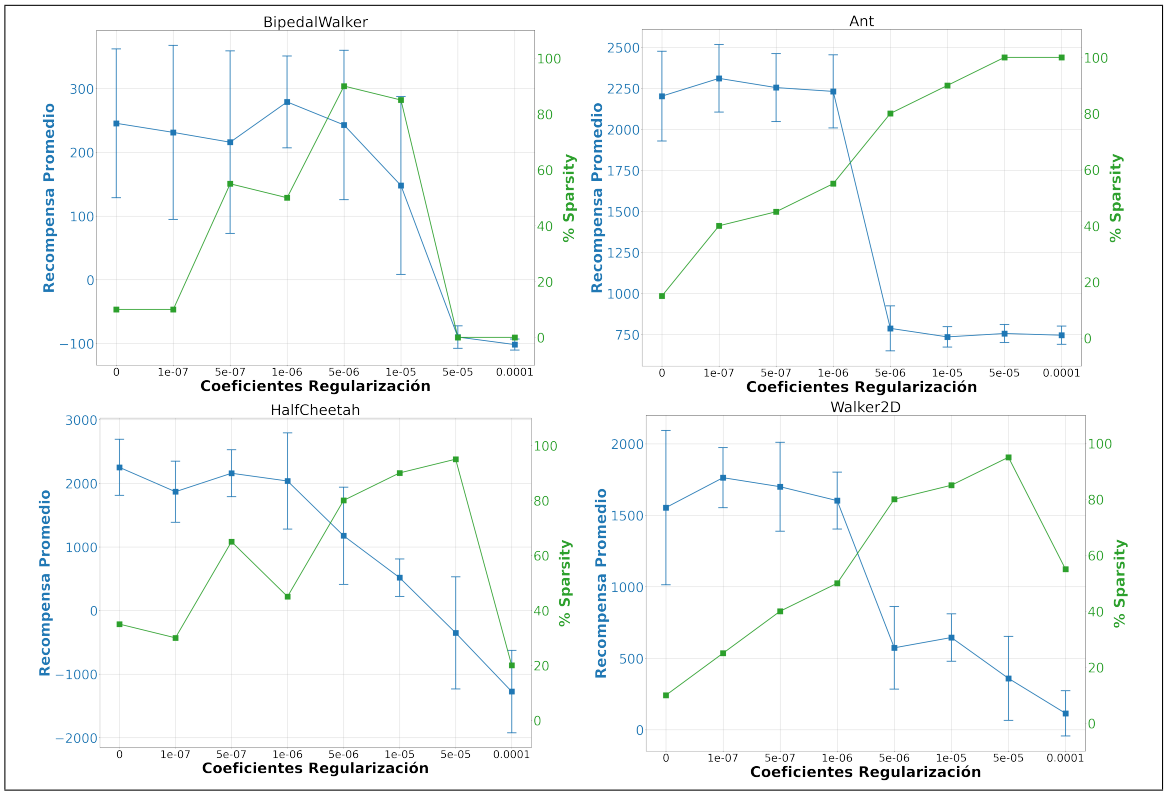


Figura 6.5. Rendimiento promedio de los agentes con su desviación estándar y el porcentaje de *sparsity* a medida que se aumenta el coeficiente de regularización de $L_1^{Activaciones}$.

6.5. Especialización de neuronas

Dado que el modelo regularizado posee menos neuronas involucradas en la toma de decisiones de la política del agente, estas podrían haberse especializado en aspectos específicos de la política. Por lo tanto, también se exploró el efecto que tenía la regularización *sparse* en el comportamiento de las neuronas durante la ejecución de una política. Esto podría ayudar a determinar las neuronas que se encargan de algún comportamiento del agente, en donde "comportamiento" hace referencia a un conjunto de acciones continuas con algún objetivo en particular (por ejemplo, levantar la extremidad izquierda). Además, podría permitir entender mejor la política aprendida a bajo nivel. La hipótesis es la siguiente: neuronas que tienen patrones de activación similares a lo largo de la ejecución de la política de un agente, están encargadas de un determinado comportamiento del agente.

6.5.1. *Clustering* de neuronas

Para validar esta hipótesis, se trabajó con el ambiente *BipedalWalker*, ya que es un ambiente más simple que los demás. Con él, se ejecutó una política aprendida con y sin regularización *sparse* durante 200 *timesteps*. Luego, se redujo la dimensionalidad de las activaciones de cada neurona a 2 dimensiones utilizando t-SNE (Maaten & Hinton, 2008). Con esto se pudo visualizar la separabilidad de las neuronas en cuanto a sus patrones de activaciones durante la ejecución de la política. Sobre el resultado de la reducción de dimensionalidad, se agruparon neuronas utilizando el algoritmo de *clustering k-means* (Arthur & Vassilvitskii, 2006), con lo cual se buscaba categorizar grupos de neuronas con patrones de activación similares.

El resultado se puede ver en la figura 6.6. De esta imagen se puede ver que, a pesar de que en ambos casos los grupos de neuronas no son explícitamente separables, el agente con regularización *sparse* tiene grupos de neuronas más definidos que el agente sin regularización *sparse*. Esto da indicios de que el modelo con regularización *sparse* posee

grupos de neuronas con patrones de activación similares que pueden diferenciarse de los demás. El modelo sin regularización *sparse* posee pocos grupos de neuronas con patrones de activaciones diferenciables, sobre todo en la capa 2, representada en la imagen inferior izquierda. Esto indica que las activaciones de distintas neuronas poseen patrones diferentes, pero estos patrones cambian más "suavemente" que el caso regularizado, llevando a muchos patrones de activación intermedios.

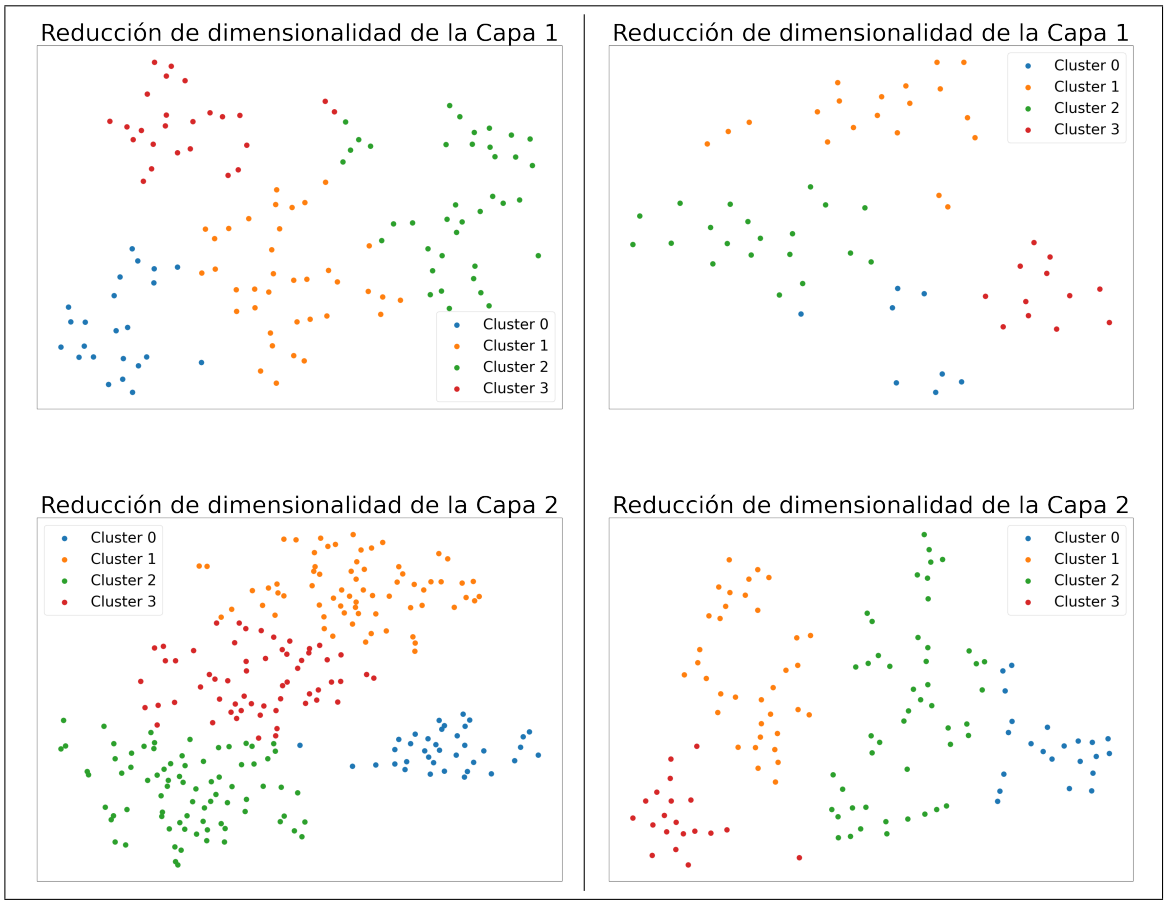


Figura 6.6. Reducción de dimensionalidad de las activaciones de las neuronas de un agente durante la ejecución de su política por 200 *timesteps*. A la izquierda el agente sin regularización *sparse* y a la derecha el agente con regularización *sparse*. Cada color corresponde a un *cluster* definido por el algoritmo *k-means*. Para ambos modelos y ambas capas se escogieron 4 *clusters*.

6.5.2. Comportamiento de los grupos de neuronas

Para visualizar qué comportamientos codifican los grupos de neuronas de la figura 6.6, se grabó al agente ejecutando las políticas *sparse* y *no-sparse* y se midió, por capa, cuánto aportaba cada grupo de neuronas a la activación total en cada *timestep*.[1] De esta forma se podría ver en tiempo real el grupo de neuronas que está más activado en cada instante y, por lo tanto, podría observarse si es que los grupos se especializan en algún comportamiento determinado.

La figura 6.7 muestra capturas de pantalla del resultado de este experimento para el caso sin regularización (izquierda) y con regularización (derecha). En ellas se muestra cuánto aporta cada grupo de neuronas ("Porcentaje de Activación") a la activación total que tuvo una capa, valor que se definirá a continuación. Sea $y_{ij}^t = g(\theta_i[:, j]^\intercal x)$ el *output* de la neurona j de la capa i en el *timestep* t, donde g es la función de activación (*ReLu*), N_i es la cantidad de neuronas de la capa i y L es la cantidad de capas del modelo. Se definen los siguientes términos:

- "**Activación Total**" de la capa i:

$$\mathrm{AT}_i^t = \sum_{j=1}^{N_i} y_{ij}^t \tag{6.1}$$

- "**Porcentaje de Activación**" del grupo k de la capa i en el *timestep* t, en donde $\mathbb{C}_{ik}$ es el conjunto de neuronas pertenecientes al grupo k de la capa i:

$$\mathrm{PA}_{ik}^t = \frac{\sum_{j=1}^{N_i} \mathbb{1}(j \in \mathbb{C}_{ik}) \cdot y_{ik}^t}{\mathrm{AT}_i^t} \tag{6.2}$$

Un ejemplo del cálculo del "Porcentaje de Activación" se muestra en la figura 6.8. Con estas definiciones, se puede ver en la figura 6.7 que la principal diferencia entre los agentes es que el modelo con regularización *sparse* (derecha) presenta *peaks* de activación de grupos de neuronas más definidos que el modelo sin regularización. Esto indica que,

[1] Los vídeos se encuentran en el siguiente en los siguientes links: regularización *sparse* (`https://youtu.be/3FfJw2O61lI`), sin regularización (`https://youtu.be/qGganAHchV8`)

dependiendo de lo que tenga que hacer el agente, hay grupos de neuronas específicos que se activan más para llevar a cabo el comportamiento requerido. A pesar de esto, las contribuciones entre grupos de neuronas no tienen límites estrictos, sino que cambian de a poco, lo cual muestra que, como se trata de un ambiente continuo, los comportamientos que debe realizar el agente tienen versiones intermedias que involucran el uso simultáneo de grupos de neuronas con patrones de activaciones distintos.

Analizando este fenómeno de manera cualitativa, si se ve con detalle el comportamiento del agente con regularización para cada grupo de neuronas de la primera capa[2], se puede observar que el grupo de neuronas verdes tiene *peaks* de activación siempre que el agente se impulsa con la extremidad trasera, lo cual indicaría que ese grupo de neuronas tiene mayor incidencia en la realización de esa acción. También se puede ver que el grupo rojo se activa más cuando el agente flecta la extremidad delantera. Sin embargo, para la capa 2 no es tan claro el rol de cada grupo de neuronas en cuanto a comportamientos del agente, esto podría deberse a que la información de esta capa es utilizada para calcular dos valores distintos entre sí: la función de valor y la acción a tomar por el agente.

Estas observaciones indican que los grupos de neuronas encontrados en el agente con regularización *sparse* efectivamente se están encargando de un comportamiento en particular del agente. Para el caso del agente sin regularización[3] también es posible hacer en algunos casos esta distinción, pero las contribuciones a los comportamientos de los distintos grupos de neuronas están más distribuidos entre ellos que el caso regularizado, sin que alguno destaque. De hecho, en la primera capa, el grupo de neuronas naranjo está constantemente activado a lo largo de la ejecución de la política, lo cual indica que incide en la toma de decisiones de todos los comportamientos del agente.

[2] `https://youtu.be/u1hJOKwAdGY`
[3] `https://youtu.be/qGganAHchV8`

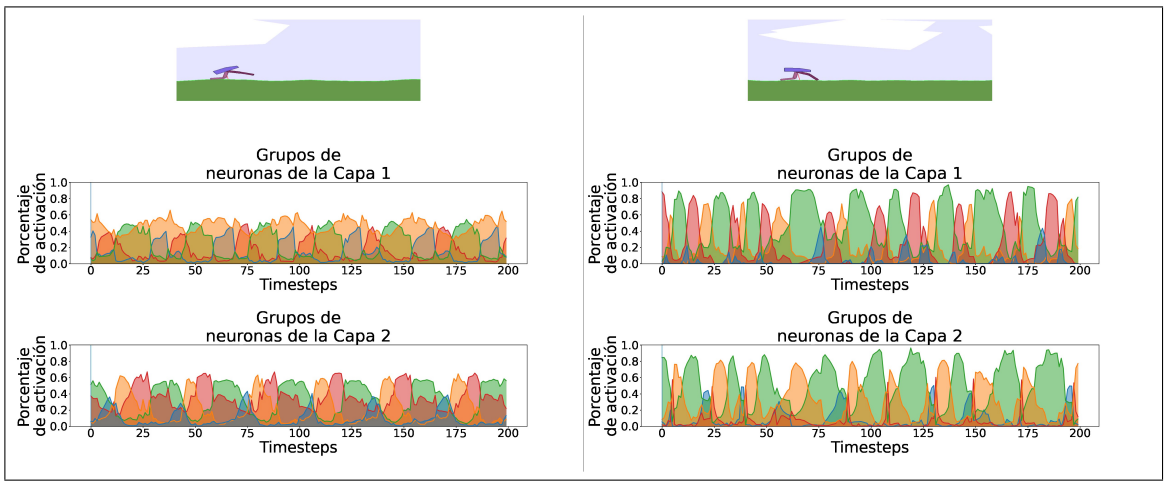


Figura 6.7. Serie de tiempo de las contribuciones de cada grupo de neuronas a la activación total de las capas del modelo durante la ejecución de 200 *timesteps* de la política entrenada. El porcentaje de activación corresponde al porcentaje de lo que aporta cada grupo de neuronas con respecto a la suma total de la activación de una capa. Cada color corresponde a un grupo de neuronas distinto. A la izquierda, el agente sin regularización y a la derecha, el agente con regularización.

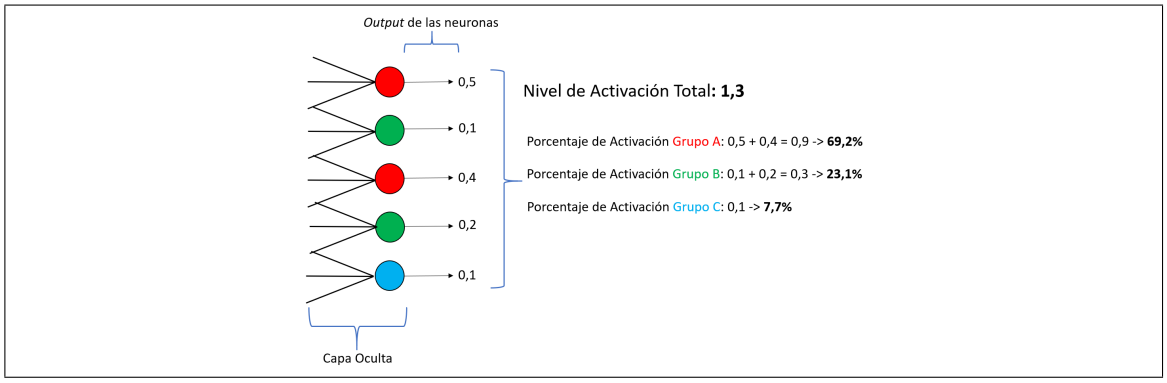


Figura 6.8. Ejemplo del cálculo del porcentaje de activación de los grupos de neuronas. Se muestran 3 grupos de neuronas denotados por su color, el *output* de cada neurona (*output* de la función de activación *ReLu*) se suma para cada grupo y se calcula su porcentaje con respecto a la activación total de la capa. Este último valor corresponde al porcentaje de activación.

6.5.3. Cuantificando el grado de especialización de los grupos de neuronas

Es posible cuantificar qué tan especializados están los grupos de neuronas por medio del "sobrelape" de contribuciones a la activación total de una capa en cada *timestep*. Siguiendo con las notaciones de la ecuación 6.2, se definieron los siguientes valores:

- "**Sobrelape**" en *timestep* t de la capa i:

$$S_i^t = \sum_{k \in \mathbb{C}_i} (\mathrm{PA}_{ik}^t) - \max \{\mathrm{PA}_{ik}^t | k \in \mathbb{C}_i\} \tag{6.3}$$

- "**Sobrelape Promedio**" de la capa i, en donde T es la cantidad de *timesteps* que ejecuta la política:

$$\mathrm{SP}_i = \frac{1}{T} \sum_{t=0}^{T} S_i^t \tag{6.4}$$

Nótese que en la ecuación 6.3 se resta el porcentaje de activación máximo al porcentaje de activación, esto se hace porque se asume que el grupo con mayor contribución a la activación de la capa oculta es el grupo de neuronas predominante para tomar la acción en ese *timestep*. Un esquema del cálculo del sobrelape se muestra en la figura 6.9. El sobrelape promedio (ecuación 6.4) es el promedio del sobrelape de una capa en particular. Esta medida es la que indica qué tan especializados están los grupos de neuronas, debido a que mientras menor sobrelape exista, más predominante es un grupo de neuronas específico para un comportamiento del agente.

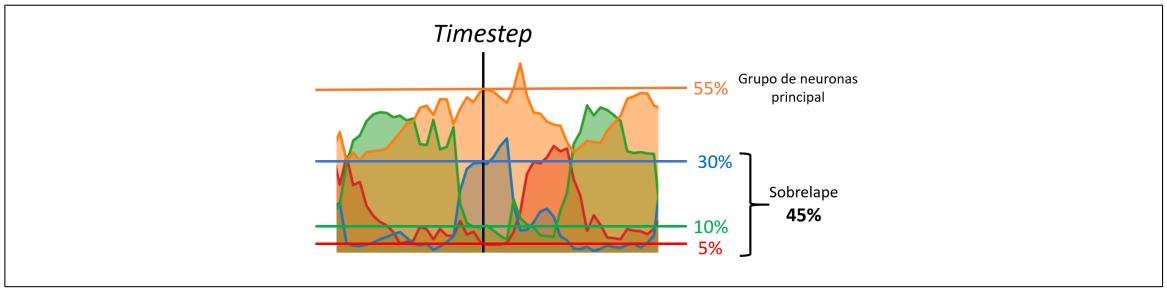


Figura 6.9. Cálculo de sobrelape entre grupos de neuronas.

Los resultados de aplicar estas medidas a la ejecución del agente de la figura 6.7 se presentan en la tabla 6.3. Como era esperado, el agente con regularización *sparse* presenta menor cantidad de sobrelape promedio a lo largo de la ejecución de la política. Esto se condice con el gráfico de los porcentajes de activación de la figura 6.7, en donde se ve que el modelo con regularización posee grupos de neuronas que predominan durante algunos *timesteps*, los cuales corresponderían a un comportamiento particular del agente.

Tabla 6.3. Sobrelape promedio entre grupos de neuronas durante la ejecución de un modelo entrenado con y sin regularización en el ambiente *BipedalWalker* durante 200 *timesteps*

Tipo de Modelo	Sobrelape Promedio Capa 1	Sobrelape Promedio Capa 2
Sin regularización	0.51 ± 0.06	0.47 ± 0.08
L1-Activaciones	0.31 ± 0.17	0.33 ± 0.16

Una de las diferencias notorias entre el modelo sin regularización y el modelo con regularización *sparse* es la variabilidad del sobrelape promedio. El modelo sin regularización *sparse* posee mayor sobrelape promedio pero menor variabilidad, esto se debe a que presenta un sobrelape constante a lo largo de la ejecución de la política, como se puede ver en la figura 6.7, en donde la contribución de los grupos de neuronas dominantes solo alcanza alrededor del 60% de la activación total. En cambio, el modelo con regularización *sparse* presenta constantemente *peaks* de porcentajes de activación de grupos de neuronas que llegan a bordear el 80% de contribución, y, cuando el comportamiento del agente comienza a cambiar, se produce una transición entre grupos de neuronas que producen momentáneamente un sobrelape, lo cual lleva a una mayor variabilidad en el sobrelape promedio.

6.5.4. Discusión

Los resultados obtenidos en esta sección se pueden relacionar con el concepto de interferencia que ha sido presentado en trabajos recientes (Hernandez-Garcia & Sutton, 2019; Liu et al., 2019) en los cuales se ha utilizado regularizaciones *sparse* y en donde se mostró

que a menor interferencia, los modelos tenían mejor rendimiento. Estos trabajos medían la interferencia en las representaciones (activaciones del modelo) para grupos de estados distintos. Esto lo hacían dividiendo los estados del ambiente del agente arbitrariamente en distintos grupos y medían las neuronas que se activaban para cada grupo. Si una neurona se activaba para dos grupos distintos de estados, entonces existía interferencia.

En este estudio también se analizó la interferencia en modelos *sparse* pero desde el punto de vista del sobrelape (definido en la ecuación 6.3) de activaciones del modelo y, en vez de dividir los estados del agente en grupos distintos, se dividieron las neuronas en grupos distintos dependiendo de la similaridad de los patrones de activación durante la ejecución de una política. En este caso, se mostró que modelos *sparse* (en cuanto al uso de neuronas) mantenían o mejoraban levemente el rendimiento del agente y, a su vez, diminuían el sobrelape entre grupos de neuronas distintos, por lo que en ambos casos se llegan a conclusiones similares vista desde puntos de vista distintos.

Finalmente, estos resultados indican que un modelo con gran cantidad de neuronas puede reducirse a un grupo más acotado de neuronas que se especialicen en ciertos aspectos del control del agente, lo cual disminuye la redundancia de la información y podría ayudar a entender mejor el rol de las neuronas dentro de una política.

7. CONCLUSIONES

En el presente trabajo se estudió el efecto que tiene aplicar regularización *sparse* estructural en un contexto de aprendizaje reforzado utilizando un algoritmo basado en *policy gradient* (PPO). Con esto se busca disminuir la complejidad del modelo, ya sea a través de los pesos del modelo o las activaciones de las neuronas, y analizar el efecto que tiene. Las regularizaciones estudiadas que explícitamente tenían este propósito fueron $L_1^{Activaciones}$, L_1^{Pesos} y SGL^{Pesos}. Además, se estudiaron otras regularizaciones que no buscaban explícitamente disminuir la complejidad del modelo, pero que han sido usadas en trabajos anteriores con el fin de obtener representaciones *sparse* a nivel de activaciones del modelo. Esto, con el fin de comparar el nivel de *sparsity* entre regularizaciones con objetivos distintos. Estas regularizaciones fueron $L_2^{Activaciones}$, L_2^{Pesos} y SKL_{exp}.

De todas las regularizaciones estudiadas, aquella que obtuvo mejores resultados en cuanto a disminuir la complejidad del modelo final, manteniendo un buen rendimiento del agente, fue $L_1^{Activaciones}$, la cual busca que el *output* de las neuronas sean cero. En contraste, las regularizaciones L_1^{Pesos} y SGL^{Pesos}, las cuales también buscan que una neurona no sea utilizada por el modelo forzando que los valores de sus pesos sean cero, logran disminuir la complejidad del modelo pero sin mantener un buen rendimiento como $L_1^{Activaciones}$. Esto indica que en un contexto de aprendizaje reforzado, utilizando un algoritmo basado en *policy gradient* (PPO), se obtienen mejores resultados disminuyendo la complejidad del modelo a través de las activaciones que a través de los pesos del modelo.

Las regularizaciones que buscaban obtener representaciones *sparse* a nivel de representaciones del modelo, $L_2^{Activaciones}$, L_2^{Pesos} y SKL_{exp}, efectivamente no logran disminuir la complejidad del modelo de manera significativa comparado con el modelo sin regularización. De hecho, Liu et al. (2019) proponen la regularización SKL_{exp} con el fin de obtener representaciones *sparse* en el modelo sin neuronas que no sean utilizadas por el modelo, lo cual efectivamente se cumple, ya que obtiene incluso menos *sparsity* estructural que el modelo sin regularización.

Por último, se pudo observar que en modelos menos complejos obtenidos mediante regularización *sparse* estructural, las neuronas se especializan más en ciertos comportamientos de la política, lo cual permite encontrar con mayor facilidad grupos de neuronas que se activan solo cuando el comportamiento que codifican es necesario. Para el caso no regularizado, estos grupos son más difíciles de distinguir debido a que las funcionalidades de las neuronas son más diversas. Con esta capacidad de diferenciar a qué comportamiento corresponde cada neurona, sería posible obtener modelos de aprendizaje reforzado más interpretables a nivel neuronal, pudiendo entender el rol de las neuronas dentro de un modelo.

8. TRABAJO FUTURO

Como se vio en las secciones anteriores, es posible entrenar un agente utilizando el algoritmo PPO de aprendizaje reforzado basado en *policy gradient* para encontrar una política que logre un buen rendimiento en cuanto a la recompensa promedio utilizando una menor cantidad de neuronas. Esto significa que el modelo ahora posee capacidad libre, la cual podría utilizarse para extender las capacidades del modelo.

Una de las aplicaciones en donde sería posible ocupar las neuronas libres es en un contexto de aprendizaje continuo, en donde el agente podría aprender una tarea utilizando la menor cantidad de neuronas posibles y luego aprender otra tarea ocupando principalmente las neuronas no utilizadas por la tarea anterior. De esta forma, se esperaría que el modelo sea capaz de mantener lo aprendido en ambas tareas. Además, la política aprendida en las primeras tareas podría ayudar a descubrir nuevas políticas más rápido en las tareas siguientes. Esta metodología ha sido explorada en aprendizaje supervisado (Aljundi et al., 2019), pero no en aprendizaje reforzado. En el presente trabajo se hicieron pruebas preliminares de esta metodología, sin embargo, uno de los grandes problemas es el hecho de que las neuronas libres del modelo tienen un *output* muy cercano a cero, por lo que al comenzar a aprender una nueva tarea, es difícil para el algoritmo de optimización "revivir" estas neuronas para ser ocupadas en una nueva tarea.

Otra línea de investigación que podría apoyarse con modelos menos complejos es la explicabilidad en inteligencia artificial. Como se vio en la sección 6.5, un agente que utilice una menor cantidad de neuronas, logra especializar neuronas en ciertos comportamientos de la política, por lo que podrían aislarse estructuras dentro del modelo encargadas de alguna función del agente, logrando analizar a un nivel más bajo las políticas aprendidas y ver cómo se relacionan las distintas representaciones internas. En el presente trabajo se utilizaron técnicas simples de *clustering* y reducción de dimensionalidad para analizar este fenómeno de especialización, pero podrían explorarse más metodologías y lograr mejores interpretaciones del rol de las neuronas de un modelo en una política. Incluso podría

explorarse el uso de memoria en conjunto con estos modelos, como lo son las redes neuronales recurrentes, por ejemplo, las redes recurrentes *Long-Short Term* (Hochreiter & Schmidhuber, 1997).

REFERENCIAS

Achiam, J. (2018). Spinning Up in Deep Reinforcement Learning.

Aljundi, R., Rohrbach, M., & Tuytelaars, T. (2019). Selfless sequential learning. In *International conference on learning representations.* Retrieved from `https://openreview.net/forum?id=Bkxbrn0cYX`

Andrychowicz, M., Baker, B., Chociej, M., Józefowicz, R., McGrew, B., Pachocki, J. W., ... Zaremba, W. (2020). Learning dexterous in-hand manipulation. *Int. J. Robotics Res.*, *39*(1). Retrieved from `https://doi.org/10.1177/0278364919887447` doi: 10.1177/0278364919887447

Arthur, D., & Vassilvitskii, S. (2006). *k-means++: The advantages of careful seeding* (Tech. Rep.). Stanford.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). *Openai gym.*

Coumans, E., & Bai, Y. (2016–2019). *Pybullet, a python module for physics simulation for games, robotics and machine learning.* `http://pybullet.org`.

Frankle, J., & Carbin, M. (2019). The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International conference on learning representations.* Retrieved from `https://openreview.net/forum?id=rJl-b3RcF7`

French, R. M. (1991). Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. In *Proceedings of the 13th annual cognitive science society conference* (Vol. 1, pp. 173–178).

Friedman, J. H. (2012). Fast sparse regression and classification. *International Journal of*

Forecasting, *28*(3), 722–738.

Ghiassian, S., Yu, H., Rafiee, B., & Sutton, R. S. (2018). Two geometric input transformation methods for fast online reinforcement learning with neural nets. *CoRR*, *abs/1805.07476*. Retrieved from http://arxiv.org/abs/1805.07476

Gupta, S., Gupta, R., Ojha, M., & Singh, K. P. (2018). A comparative analysis of various regularization techniques to solve overfitting problem in artificial neural network. In B. Panda, S. Sharma, & N. R. Roy (Eds.), *Data science and analytics* (pp. 363–371). Singapore: Springer Singapore.

Han, S., Pool, J., Tran, J., & Dally, W. J. (2015). Learning both weights and connections for efficient neural networks. *CoRR*, *abs/1506.02626*. Retrieved from http://arxiv.org/abs/1506.02626

Hawkins, D. M. (2004). The problem of overfitting. *J. Chem. Inf. Model.*, *44*(1), 1–12. Retrieved from https://doi.org/10.1021/ci0342472 doi: 10.1021/ci0342472

Hernandez-Garcia, J. F., & Sutton, R. S. (2019). Learning sparse representations incrementally in deep reinforcement learning. *CoRR*, *abs/1912.04002*. Retrieved from http://arxiv.org/abs/1912.04002

Hill, A., Raffin, A., Ernestus, M., Gleave, A., Kanervisto, A., Traore, R., ... Wu, Y. (2018). *Stable baselines*. https://github.com/hill-a/stable-baselines. GitHub.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, *9*(8), 1735–1780. Retrieved from https://doi.org/10.1162/neco.1997.9.8.1735 doi: 10.1162/neco.1997.9.8.1735

Le, L., Kumaraswamy, R., & White, M. (2017). Learning sparse representations in reinforcement learning with sparse coding. In *Proceedings of the 26th international joint conference on artificial intelligence* (p. 2067–2073). AAAI Press.

Liu, V., Kumaraswamy, R., Le, L., & White, M. (2019). The utility of sparse representations for control in reinforcement learning. In *Proceedings of the aaai conference on artificial intelligence* (Vol. 33, pp. 4384–4391).

Lobel, H., Vidal, R., & Soto, A. (2015). Learning shared, discriminative, and compact representations for visual recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, *37*(11), 2218–2231. Retrieved from https://doi.org/10.1109/TPAMI.2015.2408349 doi: 10.1109/TPAMI.2015.2408349

Lobel, H., Vidal, R., & Soto, A. (2020). Compactnets: Compact hierarchical compositional networks for visual recognition. *Comput. Vis. Image Underst.*, *191*, 102841. Retrieved from https://doi.org/10.1016/j.cviu.2019.102841 doi: 10.1016/j.cviu.2019.102841

Louizos, C., Welling, M., & Kingma, D. P. (2018). Learning sparse neural networks through l0 regularization. In *International conference on learning representations.* Retrieved from https://openreview.net/forum?id=H1Y8hhg0b

Maaten, L. v. d., & Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(Nov), 2579–2605.

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

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In J. Fürnkranz & T. Joachims (Eds.), *Proceedings of the 27th international conference on machine learning (icml-10), june 21-24, 2010, haifa, israel* (pp. 807–814). Omnipress. Retrieved from https://icml.cc/Conferences/2010/papers/432.pdf

OpenAI, Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., ... Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning. Retrieved from https://

arxiv.org/abs/1912.06680

Ororbia, A., Mali, A., Kifer, D., & Giles, C. L. (2019). *Lifelong neural predictive coding: Sparsity yields less forgetting when learning cumulatively.*

Painter-Wakefield, C., & Parr, R. (2012). *Greedy algorithms for sparse reinforcement learning.*

Pan, Y., Banman, K., & White, M. (2020). *Leaky tiling activations: A simple approach to learning sparse representations online.*

Qin, Z., Li, W., & Janoos, F. (2014). Sparse reinforcement learning via convex optimization. In *Icml* (p. 424-432). Retrieved from http://proceedings.mlr.press/v32/qin14.html

Rafati, J., & Noelle, D. C. (2019). Learning sparse representations in reinforcement learning. *CoRR*, *abs/1909.01575*. Retrieved from http://arxiv.org/abs/1909.01575

Raffin, A. (2018). *Rl baselines zoo.* https://github.com/araffin/rl-baselines-zoo. GitHub.

Scardapane, S., Comminiello, D., Hussain, A., & Uncini, A. (2016). Group sparse regularization for deep neural networks. *CoRR*, *abs/1607.00485*. Retrieved from http://arxiv.org/abs/1607.00485

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015, 07–09 Jul). Trust region policy optimization. In F. Bach & D. Blei (Eds.), (Vol. 37, pp. 1889–1897). Lille, France: PMLR. Retrieved from http://proceedings.mlr.press/v37/schulman15.html

Schulman, J., Moritz, P., Levine, S., Jordan, M. I., & Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In Y. Bengio & Y. LeCun

(Eds.), *4th international conference on learning representations, ICLR 2016, san juan, puerto rico, may 2-4, 2016, conference track proceedings.* Retrieved from http://arxiv.org/abs/1506.02438

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017, July). Proximal Policy Optimization Algorithms. *arXiv e-prints*, arXiv:1707.06347.

Sokar, G., Mocanu, D. C., & Pechenizkiy, M. (2020). *Spacenet: Make free space for continual learning.*

Song, T., Li, D., Jin, Q., & Hirasawa, K. (2020, Nov). Sparse proximal reinforcement learning via nested optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *50*(11), 4020-4032. doi: 10.1109/TSMC.2018.2865505

Srivastava, S., Berman, M., Blaschko, M. B., & Tuia, D. (2019). Adaptive compression-based lifelong learning. In *Bmvc* (p. 153). Retrieved from https://bmvc2019.org/wp-content/uploads/papers/0649-paper.pdf

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems 8* (pp. 1038–1044). MIT Press. Retrieved from http://papers.nips.cc/paper/1109-generalization-in-reinforcement-learning-successful-examples-using-sparse-coarse-coding.pdf

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, *58*(1), 267–288.

Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine learning*, *8*(3-4), 279–292.

Yang, C., Yang, Z., Khattak, A. M., Yang, L., Zhang, W., Gao, W., & Wang, M. (2019). Structured pruning of convolutional neural networks via L1 regularization. *IEEE Access*, *7*, 106385–106394. Retrieved from https://doi.org/10.1109/ACCESS.2019.2933032 doi: 10.1109/ACCESS.2019.2933032

Ying, X. (2019, feb). An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, *1168*, 022022. Retrieved from https://doi.org/10.1088/1742-6596/1168/2/022022 doi: 10.1088/1742-6596/1168/2/022022

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *68*(1), 49–67.

ANEXO

A. BÚSQUEDA DE HIPERPARÁMETROS

A.1. Hiperparámetros de PPO

Los hiperparámetros del algoritmo PPO escogidos están basados en los *benchmarks* hechos por los autores de la librería utilizada (Raffin, 2018) y son diferentes para cada ambiente. A continuación se presentarán los parámetros para cada ambiente. Los coeficientes c_1 y c_2 están definidos en la ecuación 4.12. Los hiperparámetros que tienen como prefijo "lin_" tienen como valor inicial el indicado por la tabla y linealmente de acuerdo al avance del entrenamiento, eventualmente tomando un valor cercano a cero.

Tabla A.1. Hiperparámetros del ambiente *BipedalWalker*

Hiperparámetro	Valor
c_1	0.5
c_2	$1e^{-3}$
N épocas (K)	30
N total de *timesteps*	$3e^6$
N *timesteps* por iteración	2048
N ambientes de entrenamiento	16
Batch size	2048
N *mini-batches*	16
Learning rate	$2.5e^{-4}$
λ	0.95
γ	0.99
ϵ	$lin\_0.2$
Valor máximo de la norma del gradiente	0.5
Normalización de input	No
Normalización de recompensa	Sí

Tabla A.2. Hiperparámetros del ambiente *Ant*

Hiperparámetro	Valor
c_1	0.5
c_2	0.0
N épocas (K)	10
N total de *timesteps*	$5e^6$
N *timesteps* por iteración	256
N ambientes de entrenamiento	8
Batch size	2048
N *mini-batches*	1
Learning rate	$2.5e^{-4}$
λ	0.95
γ	0.99
ϵ	0.2
Valor máximo de la norma del gradiente	0.5
Normalización de input	Sí
Normalización de recompensa	Sí

Tabla A.3. Hiperparámetros del ambiente *HalfCheetah*

Hiperparámetro	Valor
c_1	0.5
c_2	0.0
N épocas (K)	10
N total de *timesteps*	$4e^6$
N *timesteps* por iteración	2048
N ambientes de entrenamiento	1
Batch size	2048
N *mini-batches*	1
Learning rate	$3e^{-4}$
λ	0.95
γ	0.99
ϵ	0.2
Valor máximo de la norma del gradiente	0.5
Normalización de input	Sí
Normalización de recompensa	Sí

Tabla A.4. Hiperparámetros del ambiente *Walker2D*

Hiperparámetro	Valor
c_1	0.5
c_2	0.0
N épocas (K)	10
N total de *timesteps*	$4e^6$
N *timesteps* por iteración	1024
N ambientes de entrenamiento	4
Batch size	2048
N *mini-batches*	2
Learning rate	$lin\_2.5e^{-4}$
λ	0.95
γ	0.99
ϵ	0.1
Valor máximo de la norma del gradiente	0.5
Normalización de input	Sí
Normalización de recompensa	Sí

A.2. Coeficientes de regularización

Para la búsqueda del coeficiente de regularización que forzara mayor *sparsity* en el modelo y que no disminuyera el rendimiento con respecto al modelo sin regularización, primero se hizo una muestra de 10 coeficientes entre $1e^{-11}$ y $1e^{-3}$ para saber en qué rangos la regularización no induce errores numéricos durante entrenamiento y en qué punto la pérdida de la regularización comienza a disminuir, es decir, en qué punto la regularización tiene un gradiente significativo.

Luego, se hizo una búsqueda más fina del mejor hiperparámetro de acuerdo a los resultados anteriores. A continuación se muestran la lista de hiperparámetros utilizados para cada regularización.

Tabla A.5. *Grid Search* de coeficientes regularizadores

Regularización	Hiperparámetros
L1-Activaciones	$1e^{-7}, 5e^{-7}, 1e^{-6}, 5e^{-6}, 1e^{-5}, 5e^{-5}, 1e^{-4}$
L2-Activaciones	$1e^{-11}, 5e^{-11}, 1e^{-10}, 5e^{-10}, 1e^{-9}, 5e^{-9}, 1e^{-8}$
L1-Pesos	$1e^{-6}, 5e^{-6}, 1e^{-5}, 5e^{-5}, 1e^{-4}, 5e^{-4}, 1e^{-3}$
L2-Pesos	$1e^{-6}, 5e^{-6}, 1e^{-5}, 5e^{-5}, 1e^{-4}, 5e^{-4}, 1e^{-3}$
SGL-Pesos	$1e^{-6}, 5e^{-6}, 1e^{-5}, 5e^{-5}, 1e^{-4}, 5e^{-4}, 1e^{-3}$
SKL-Exp	$c_3 : 1e^{-3}, 1e^{-2}, 1e^{-1}, 1, 10$ — $\beta : 0.1, 0.2$

B. VARIABILIDAD DE EXPERIMENTOS

En esta sección se presentan todos los experimentos realizados que se muestran en la sección 6 con sus respectivas desviaciones estándar. Esto permite apreciar el grado de variabilidad que presenta la ejecución del algoritmo PPO con las distintas regularizaciones *sparse*, la variabilidad del rendimiento de los *checkpoints* obtenidos y la variabilidad del rendimiento a medida que se dejan de ocupar neuronas (*sparsity*).

B.1. Variabilidad durante entrenamiento

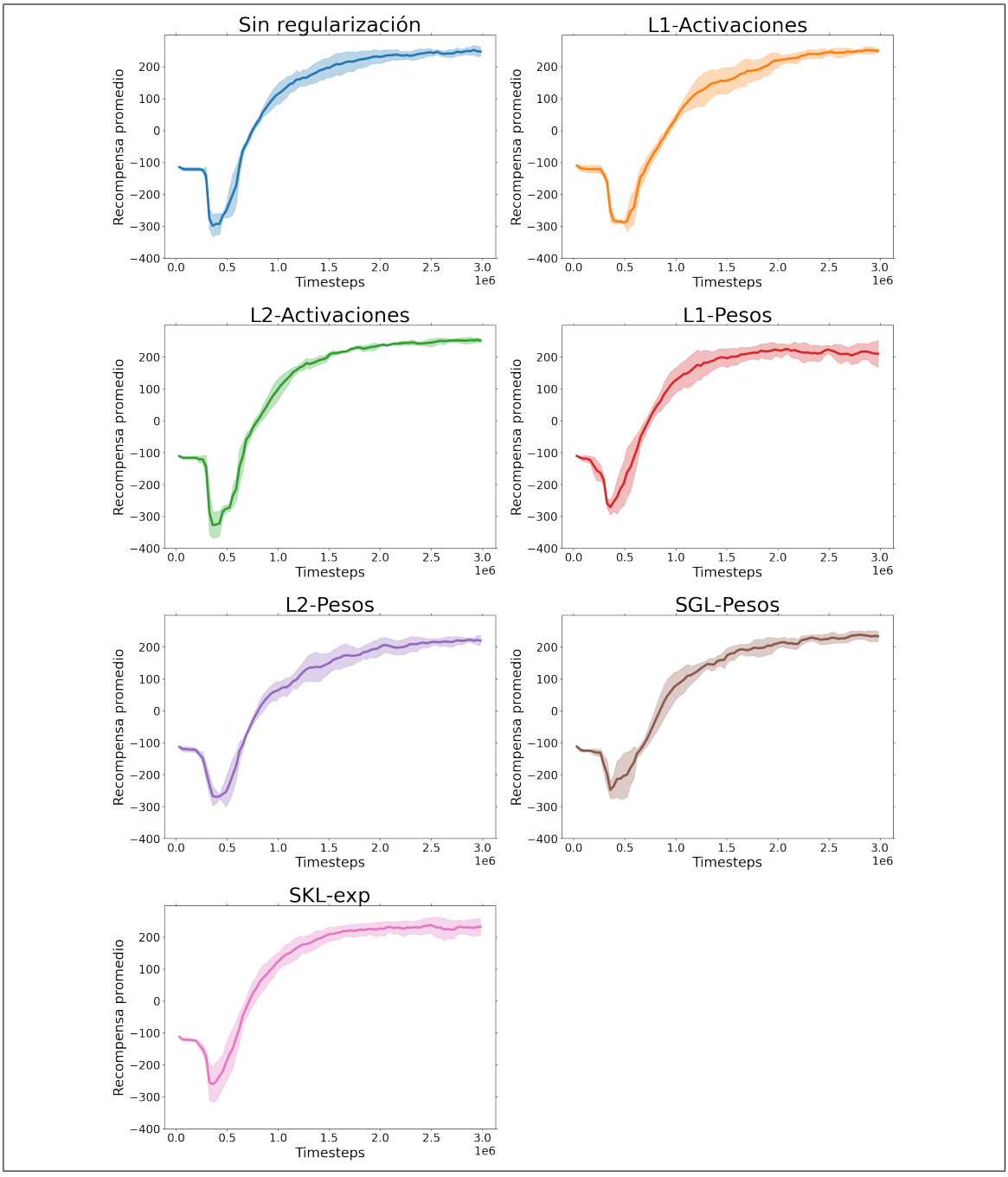


Figura B.1. Recompensa promedio con desviación estándar durante entrenamiento del ambiente *BipedalWalker* tomando en cuenta las 5 ejecuciones de PPO.

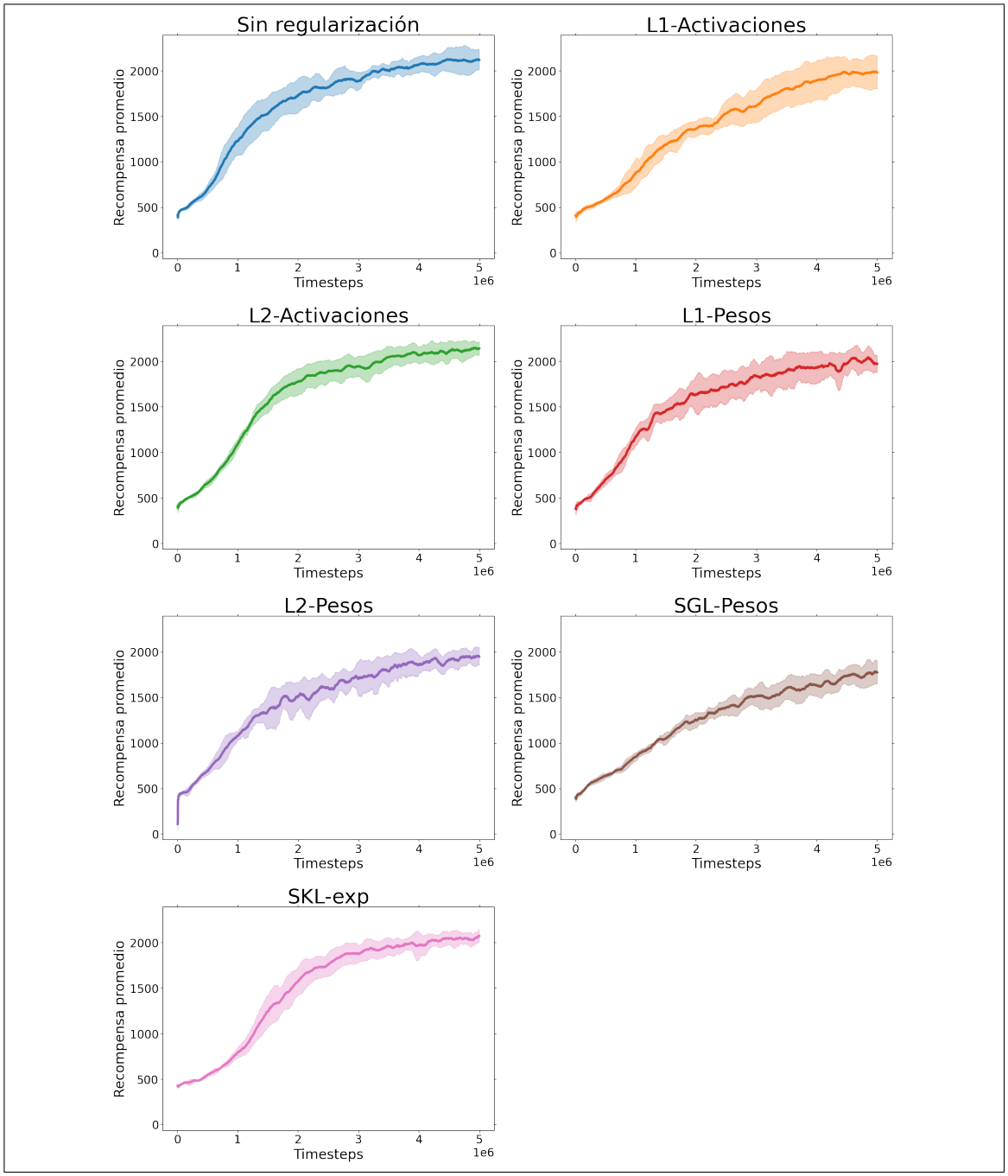


Figura B.2. Recompensa promedio con desviación estándar durante entrenamiento del ambiente *Ant* tomando en cuenta las 5 ejecuciones de PPO.

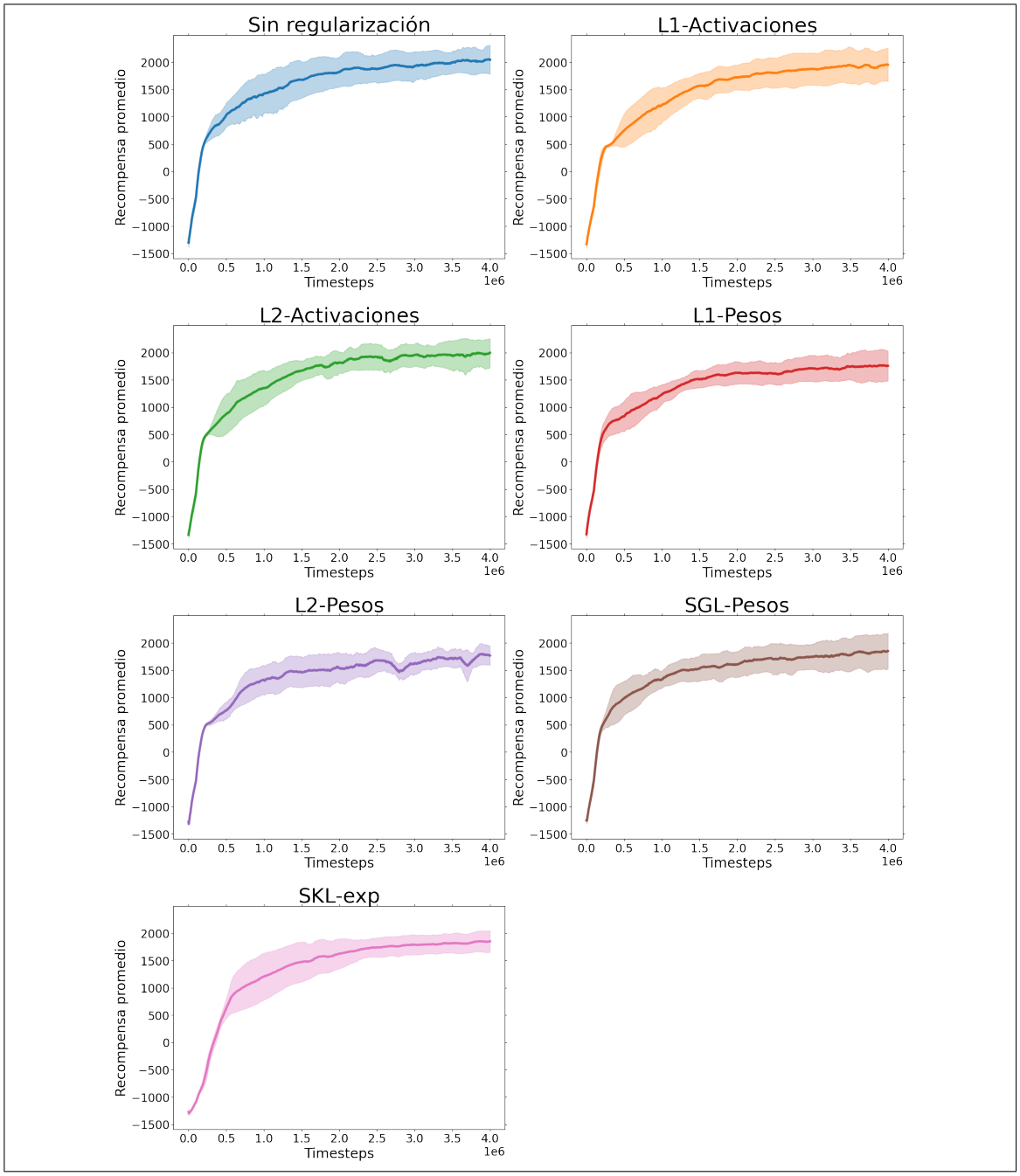


Figura B.3. Recompensa promedio con desviación estándar durante entrenamiento del ambiente *HalfCheetah* tomando en cuenta las 5 ejecuciones de PPO.

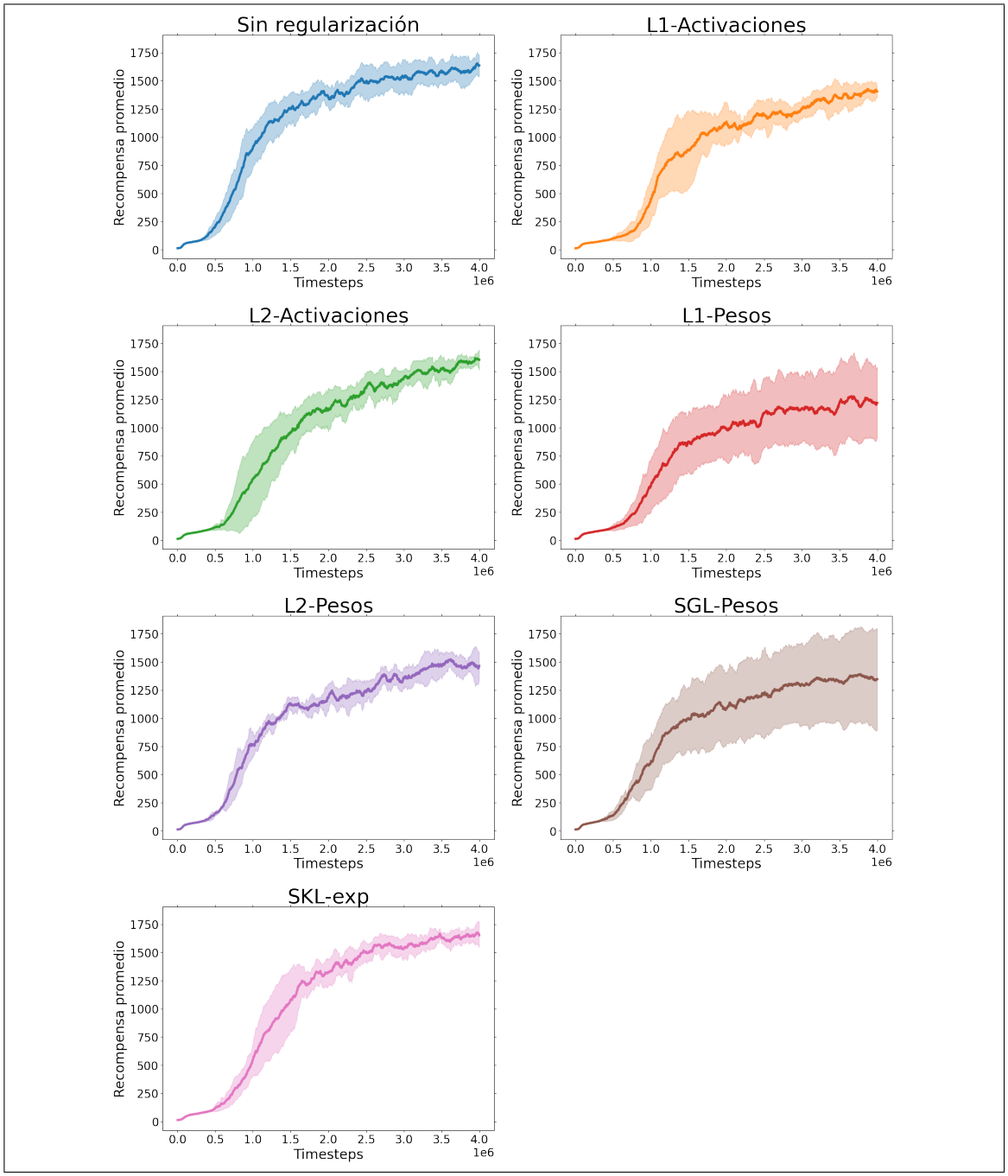


Figura B.4. Recompensa promedio con desviación estándar durante entrenamiento del ambiente *Walker2D* tomando en cuenta las 5 ejecuciones de PPO.

B.2. Variabilidad durante evaluación de *checkpoints*

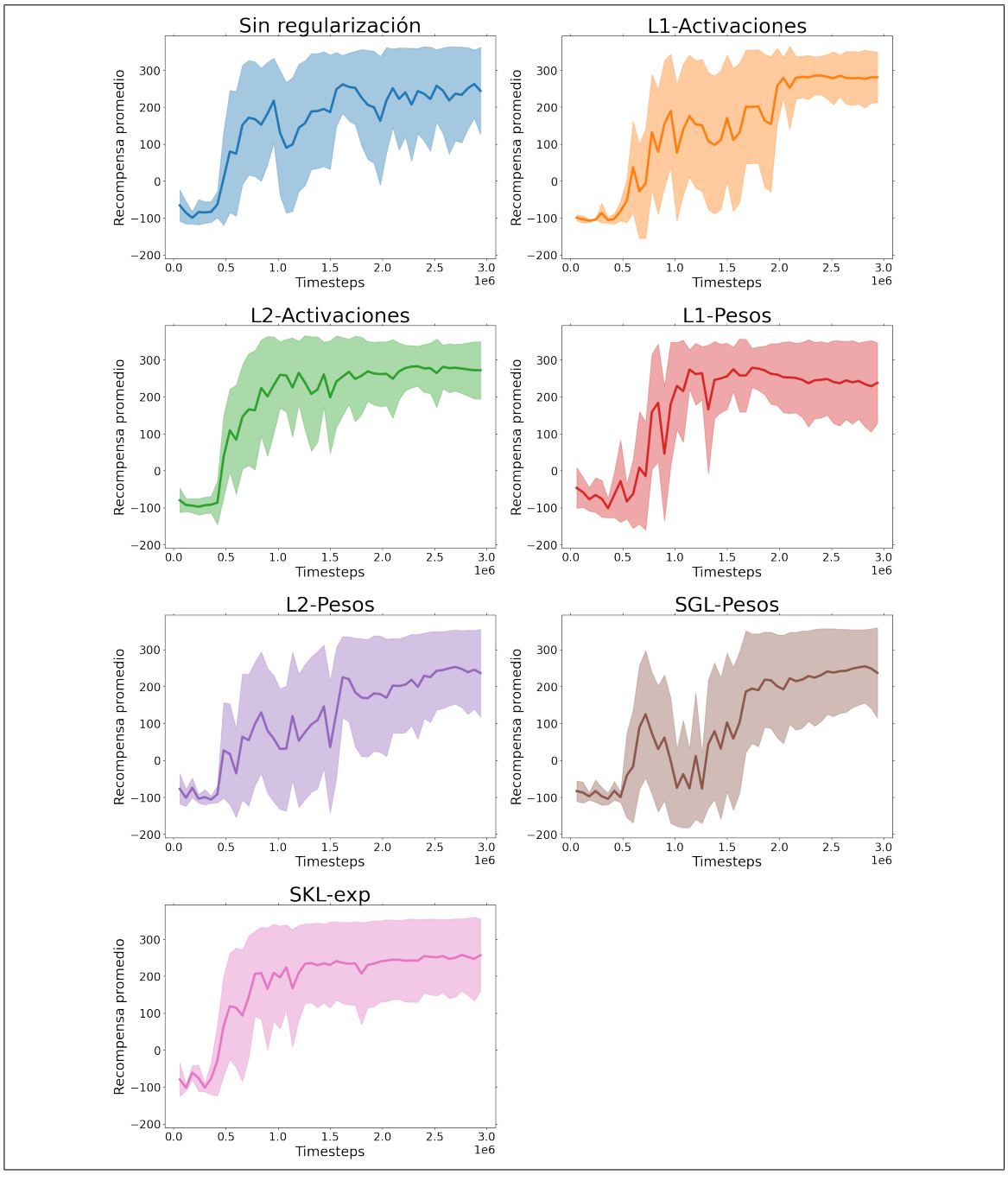


Figura B.5. Recompensa promedio con desviación estándar durante evaluación de *checkpoints* del ambiente *BipedalWalker* tomando en cuenta las 5 ejecuciones de PPO.

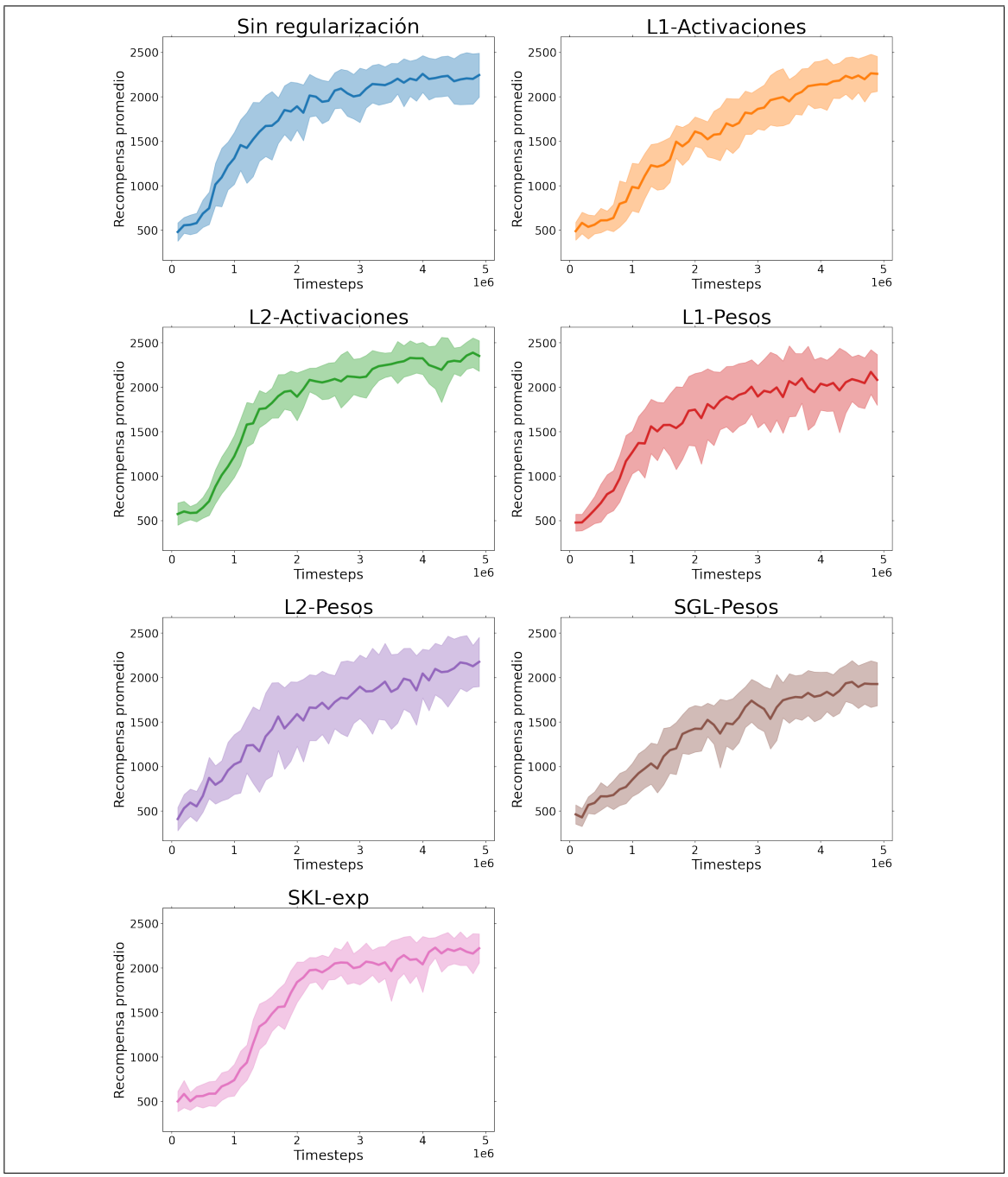


Figura B.6. Recompensa promedio con desviación estándar durante evaluación de *checkpoints* del ambiente *Ant* tomando en cuenta las 5 ejecuciones de PPO.

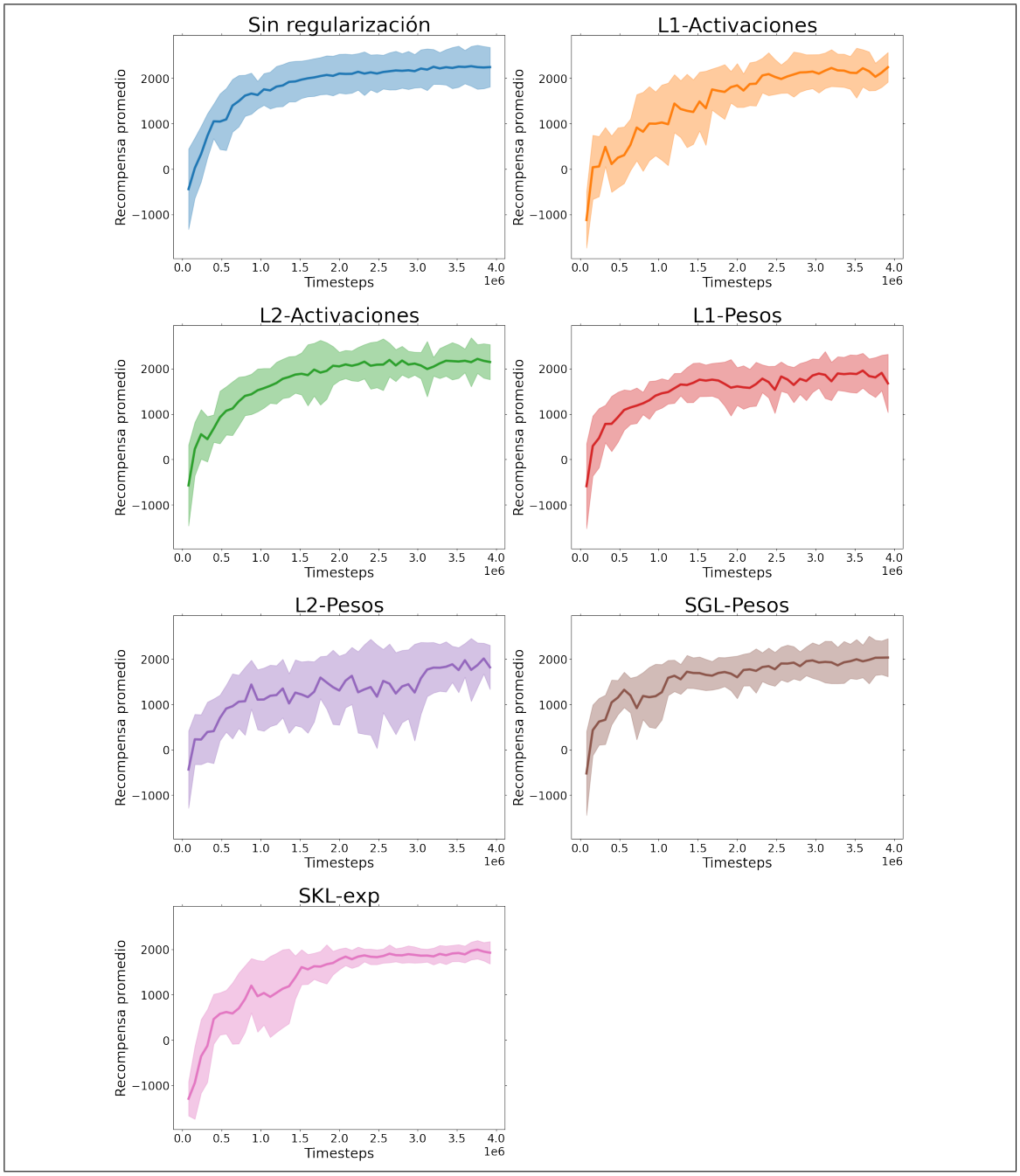


Figura B.7. Recompensa promedio con desviación estándar durante evaluación de *checkpoints* del ambiente *HalfCheetah* tomando en cuenta las 5 ejecuciones de PPO.

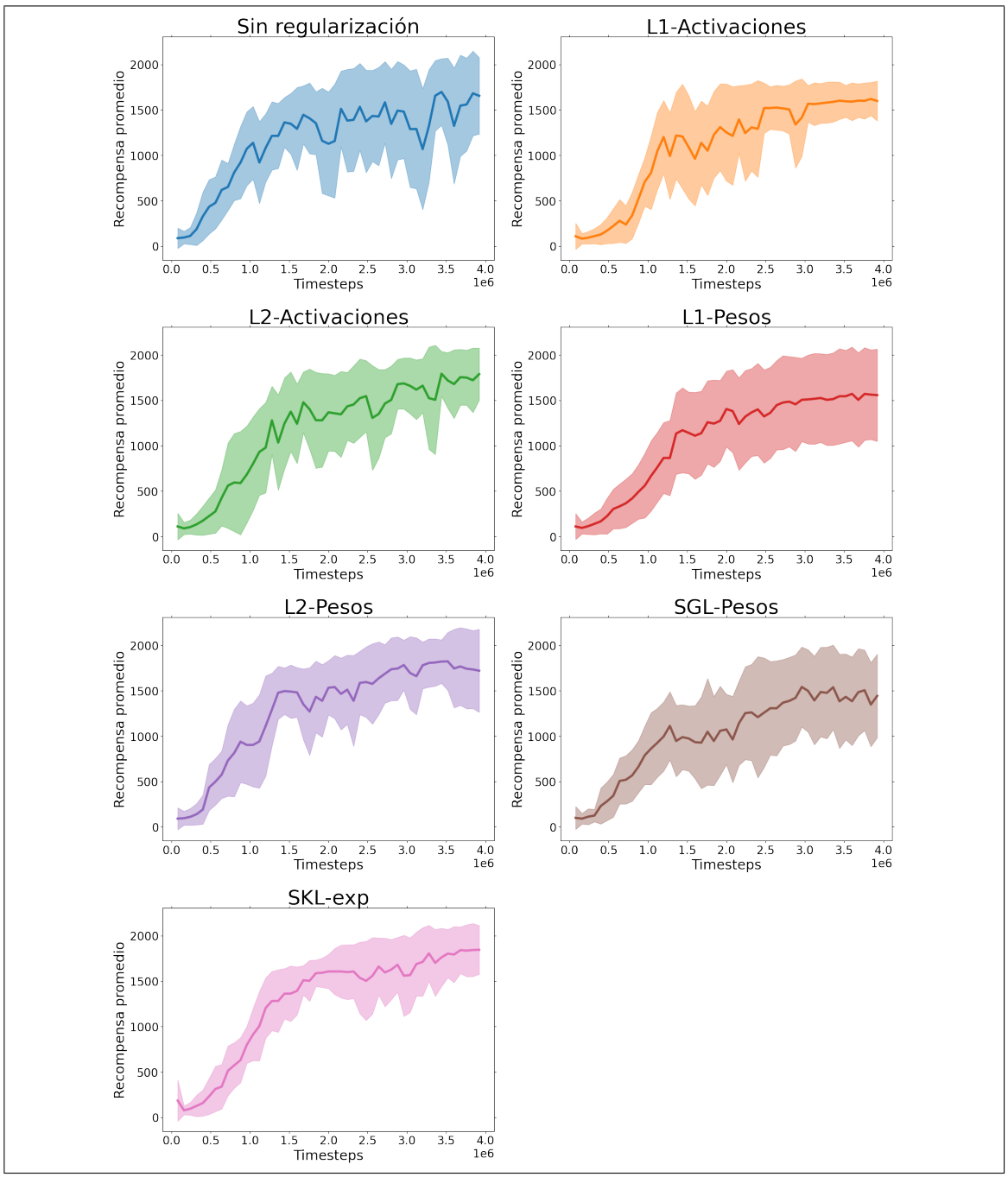


Figura B.8. Recompensa promedio con desviación estándar durante evaluación de *checkpoints* del ambiente *Walker2D* tomando en cuenta las 5 ejecuciones de PPO.

B.3. Variabilidad durante evaluación de *sparsity*

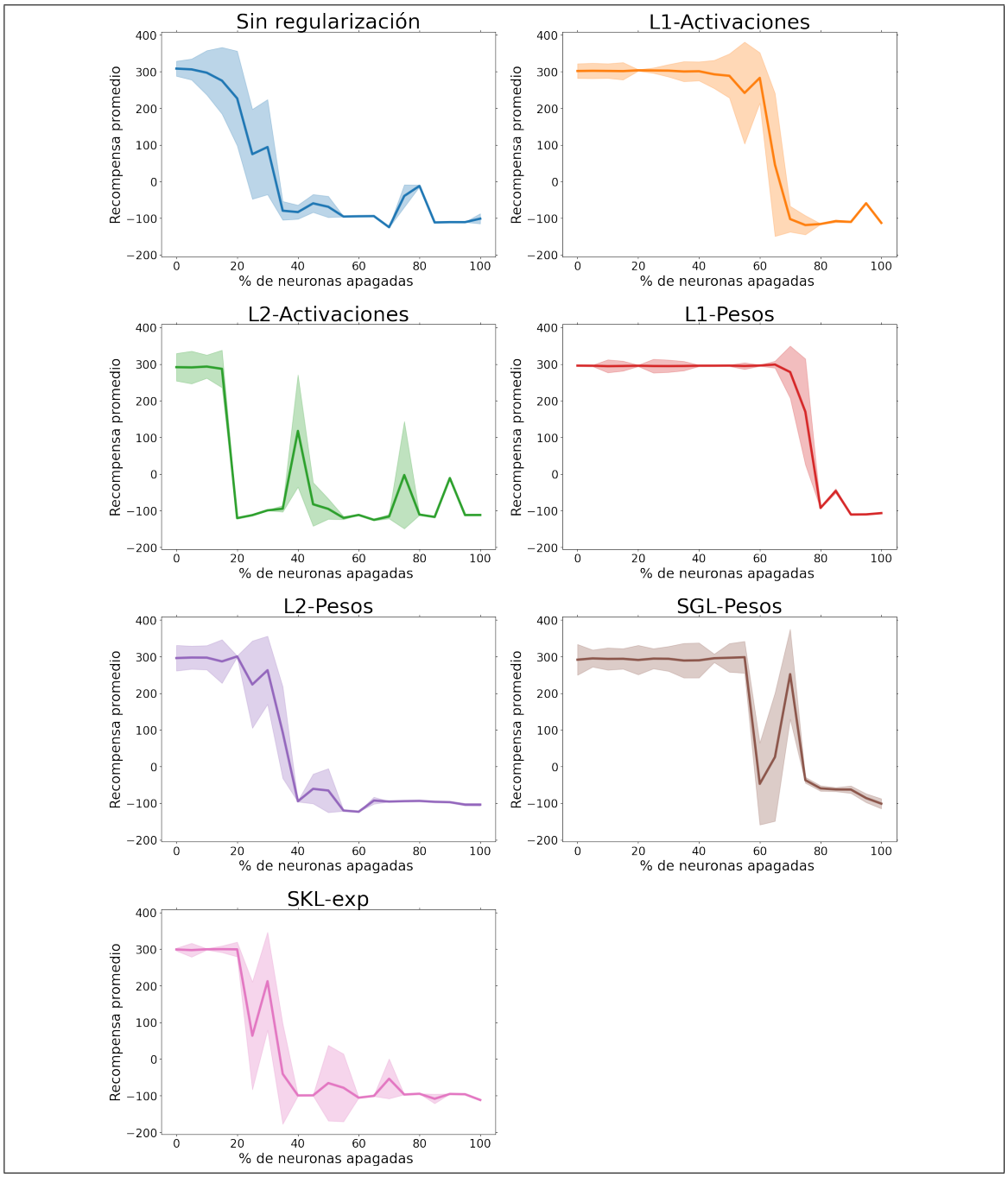


Figura B.9. Recompensa promedio con desviación estándar durante evaluación de *sparsity* del ambiente *BipedalWalker*.

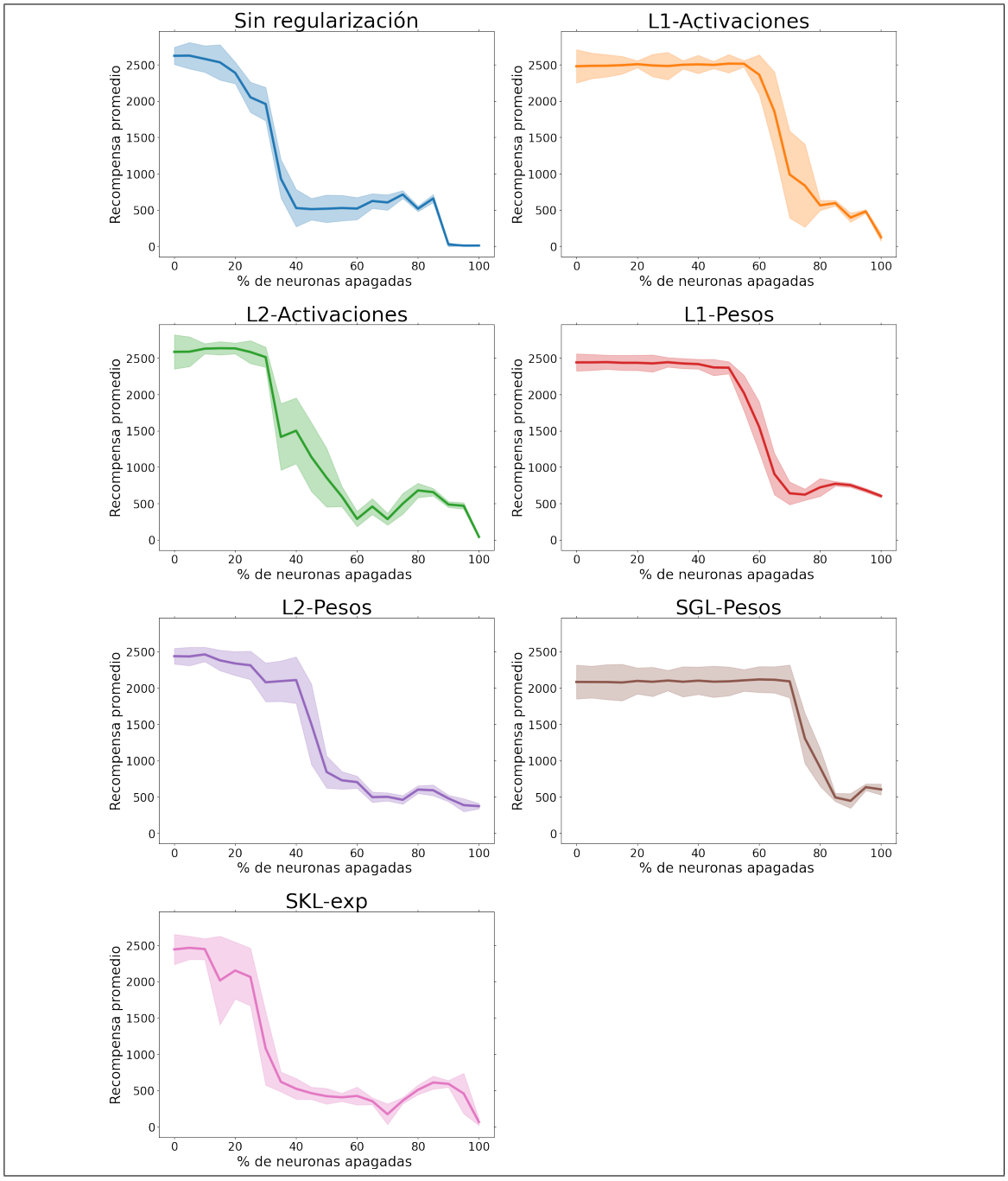


Figura B.10. Recompensa promedio con desviación estándar durante evaluación de *sparsity* del ambiente *Ant*.

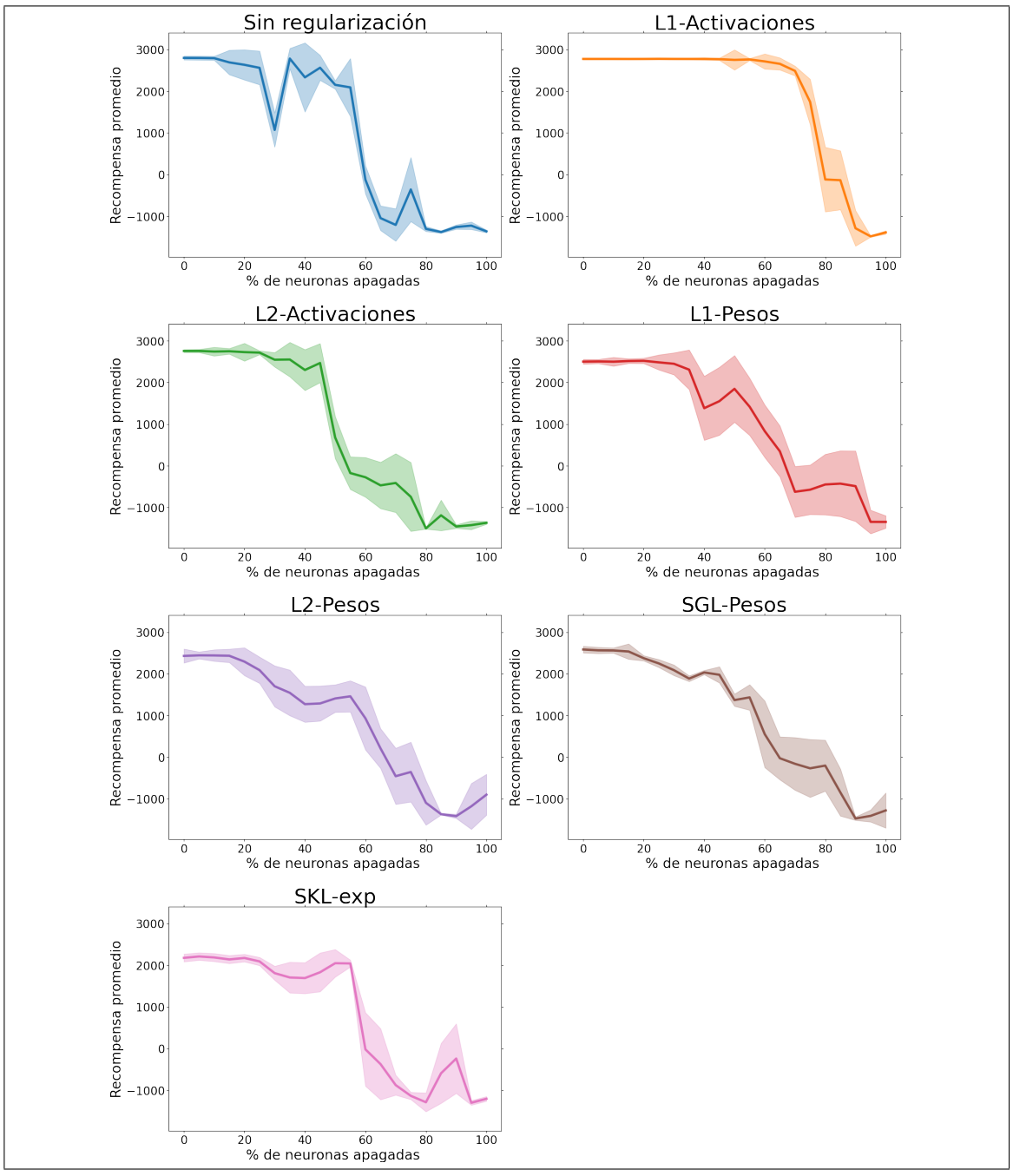


Figura B.11. Recompensa promedio con desviación estándar durante evaluación de *sparsity* del ambiente *HalfCheetah*.

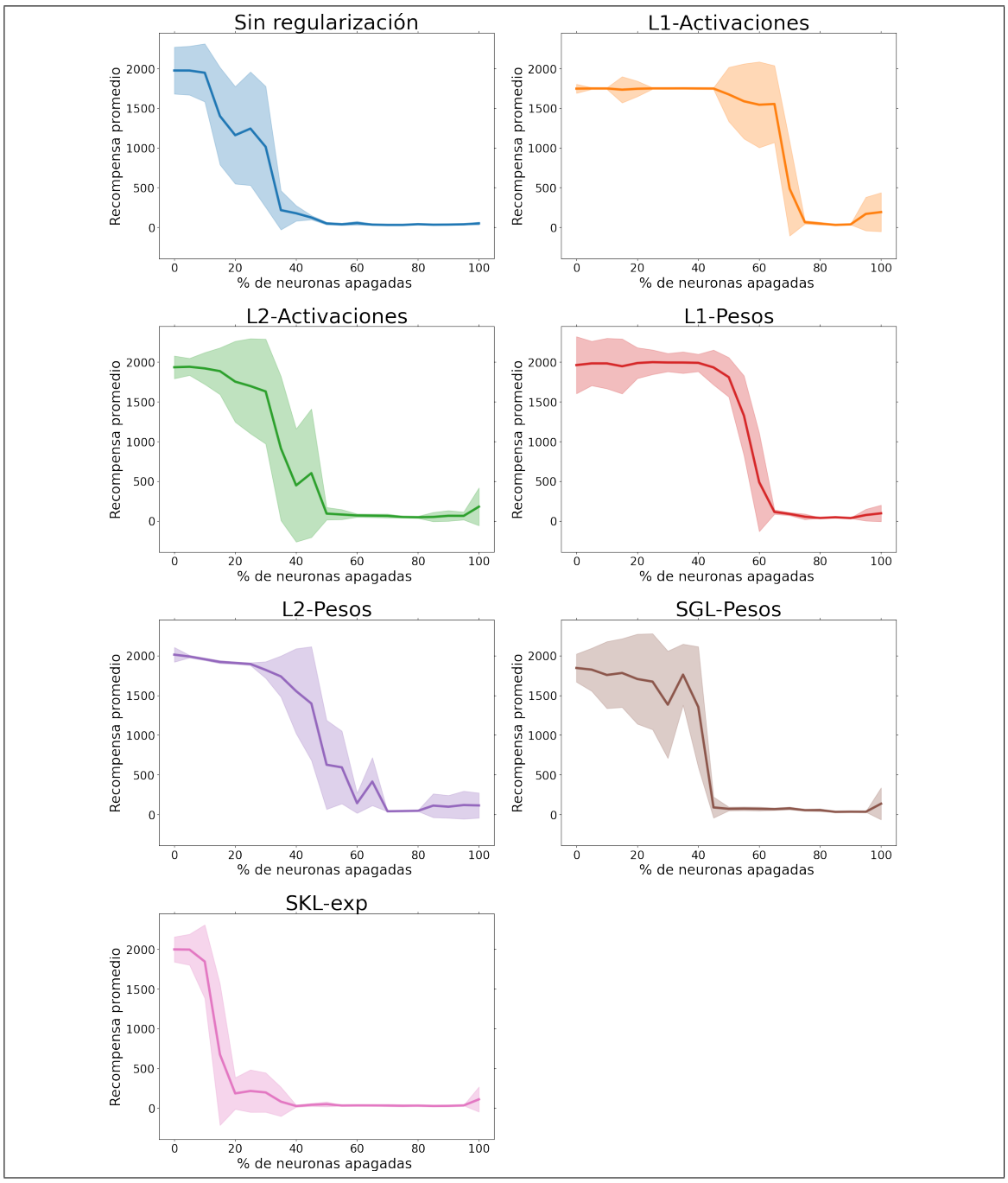
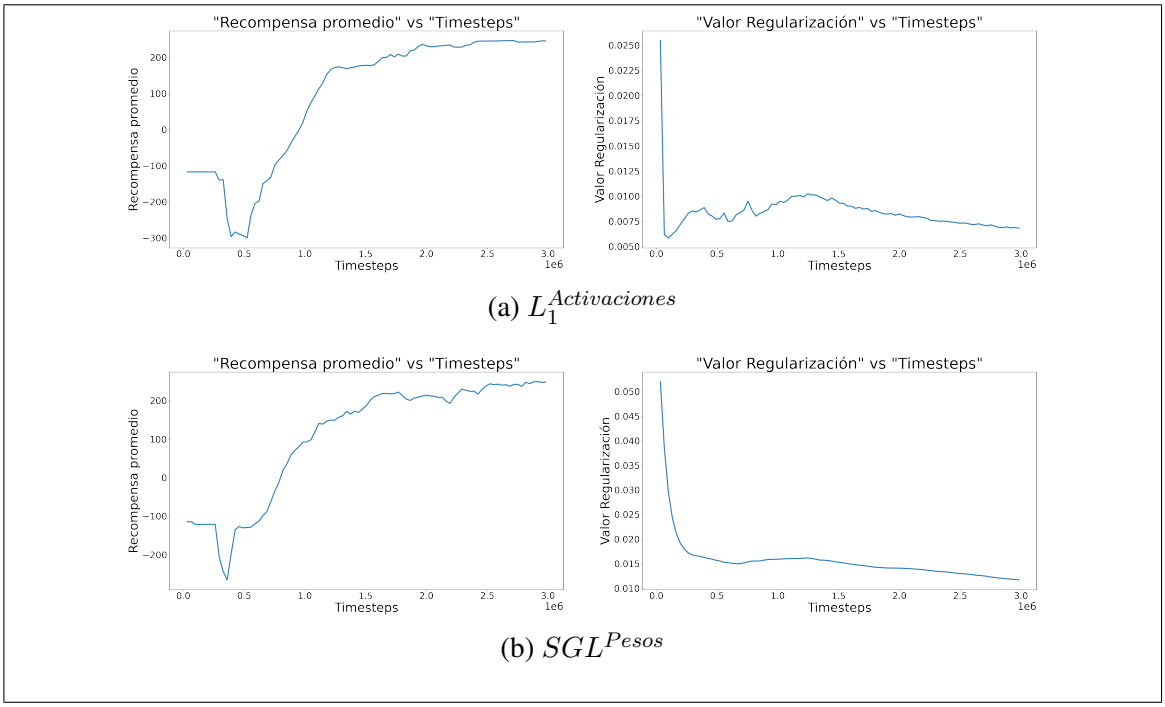


Figura B.12. Recompensa promedio con desviación estándar durante evaluación de *sparsity* del ambiente *Walker2D*.

C. COMPARACIÓN REGULARIZACIÓN SOBRE PESOS Y ACTIVACIONES

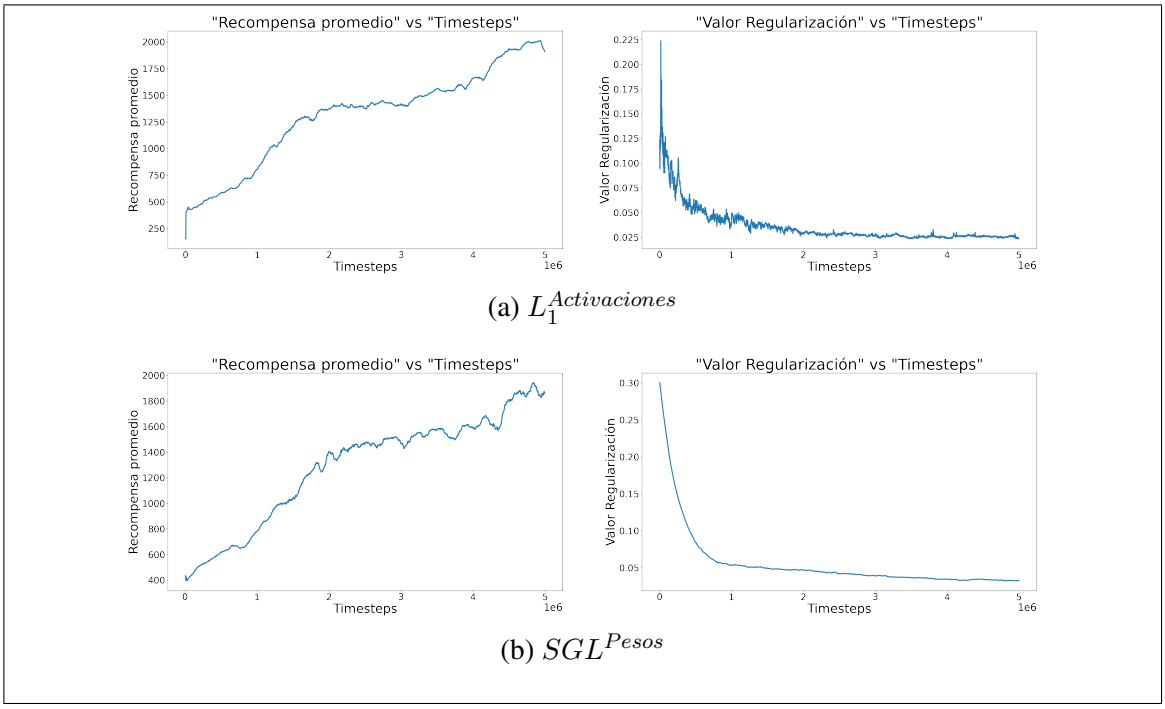
A continuación se muestra una comparación entre la regularización aplicada sobre los pesos y aplicada sobre las activaciones de una ejecución del algoritmo PPO sobre cada ambiente. Para esta comparación se utilizó $L_1^{Activaciones}$ y SGL^{Pesos}. Se puede ver que durante entrenamiento, la penalización aplicada sobre las activaciones en algunos momentos aumenta su valor, posiblemente debido a que el agente se encuentra con nuevos estados. Esto indica que esta penalización, a pesar de buscar siempre disminuir el valor de las activaciones, otorga una mayor libertad al modelo y sus pesos para poder adaptarse a cambios de estados, lo cual puede darse cuando la recompensa cambia rápidamente. Sin embargo, la regularización aplicada sobre los pesos siempre busca que los valores de los pesos sean más pequeños (curva sin aumento considerable del valor de la penalización), restringiendo su poder expresivo y, por ende, dificultando la adaptación del agente.



(a) $L_1^{Activaciones}$

(b) SGL^{Pesos}

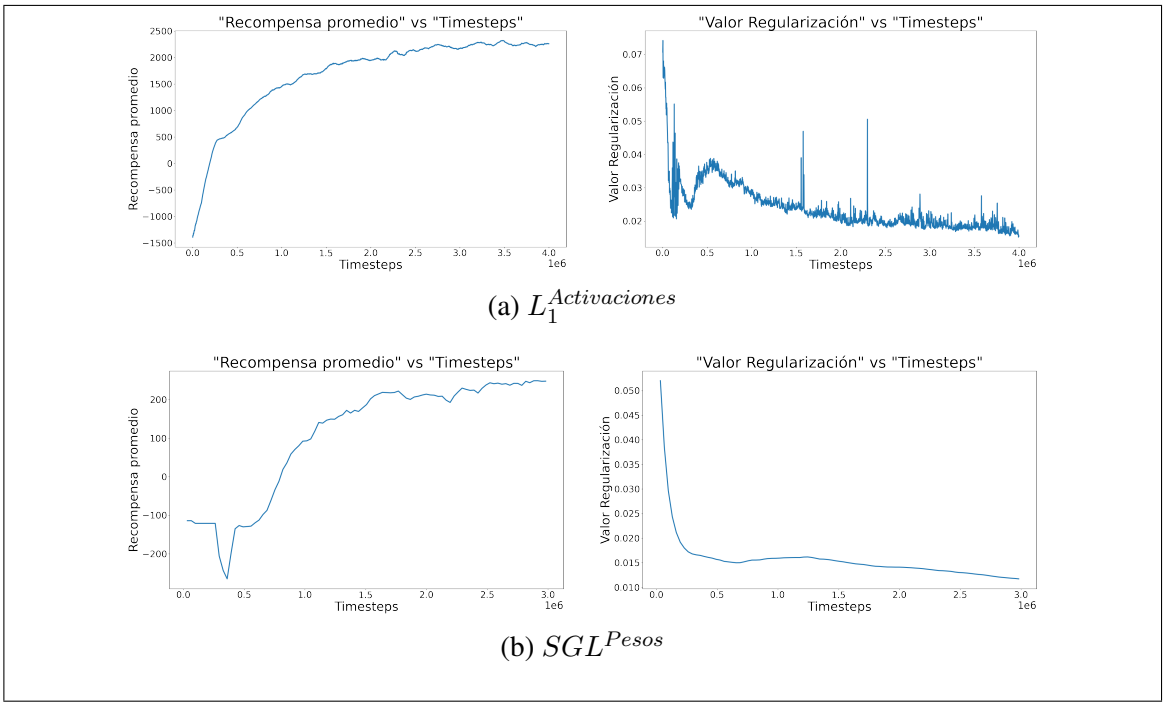
Figura C.1. Recompensa y valor de regularización en ambiente *BipedalWalker*.



(a) $L_1^{Activaciones}$

(b) SGL^{Pesos}

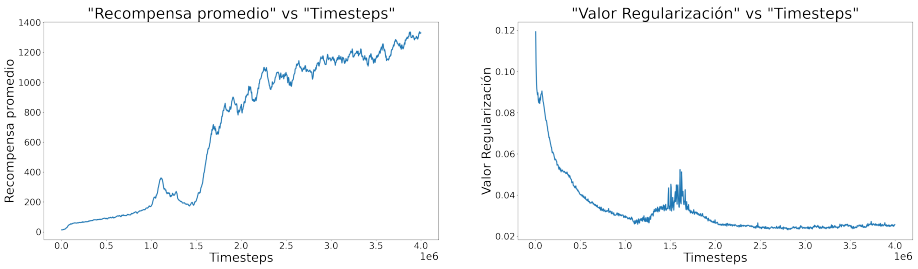
Figura C.2. Recompensa y valor de regularización en ambiente *Ant*.



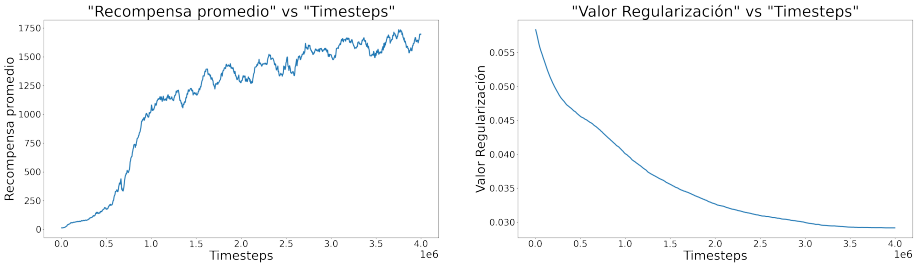
(a) $L_1^{Activaciones}$

(b) SGL^{Pesos}

Figura C.3. Recompensa y valor de regularización en ambiente *HalfCheetah*.



(a) $L_1^{Activaciones}$



(b) SGL^{Pesos}

Figura C.4. Recompensa y valor de regularización en ambiente *Walker2D*.