GRADO EN INGENIERÍA MATEMÁTICA E INTELIGENCIA ARTIFICIAL

TRABAJO FIN DE GRADO

ESTUDIO COMPARATIVO Y DESARROLLO DE ALGORITMOS DE IMITATION LEARNING: ANÁLISIS Y APLICACIÓN

Autor: Antonio Lorenzo Díaz-Meco

Director: Lucía Güitta López

Co-Director: Álvaro Jesús López López

Madrid

Junio de 2025

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

ESTUDIO COMPARATIVO Y DESARROLLO DE ALGORITMOS DE IMITATION LEARNING: ANÁLISIS Y APLICACIÓN

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2024/25 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Antonio Lorenzo Díaz-Meco Fecha: 09/ 06/ 2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Lucía Güitta López Fecha: 09/ 06/ 2025

Agradezco a mi directora del TFG por su tiempo y esfuerzo. También a mi familia porque sin su apoyo nada de esto habría sido posible.

ESTUDIO COMPARATIVO Y DESARROLLO DE ALGORITMOS DE IMITATION LEARNING: ANÁLISIS Y APLICACIÓN

Autor: Lorenzo Díaz-Meco, Antonio.
Director: Güitta López, Lucía.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

Resumen

Este trabajo compara seis algoritmos de Imitation Learning (BC, BCO, GAIL, GAIfO, AIRL, SQIL) en entornos de control continuo y discreto. Se implementaron BCO, GAIfO y SQIL desde cero para espacios continuos, cubriendo un vacío en librerías estándar. Los resultados muestran que BC supera a métodos complejos cuando hay suficientes demostraciones, alcanzando el 91.2% del rendimiento experto, mientras que AIRL es el más estable (~80%) ante variaciones del número de trayectorias. Se identifica una "zona crítica" alrededor de 20 trayectorias donde casi todos los algoritmos degradan su rendimiento. El trabajo proporciona guías empíricas para elegir algoritmos según datos y estabilidad.

Palabras clave: Imitation Learning, Reinforcement Learning, Behavioral Cloning, Aprendizaje Adversarial, Eficiencia muestral

1. Introducción

El diseño manual de funciones de recompensa en Aprendizaje por Refuerzo (RL) es el mayor obstáculo para su aplicación en tareas complejas, ya que necesita definir explícitamente todos los comportamientos deseados e indeseados, haciéndolo inviable en entornos realistas y de alta dimensionalidad [1]. El Imitation Learning (IL) surge como una alternativa que permite que los agentes aprendan directamente a partir de demostraciones expertas. No se necesita definir recompensas manualmente. Sin embargo, la literatura sobre IL está dividida en varios enfoques metodológicos (*Behavioral Cloning,* métodos inversos, adversariales y basados en observaciones), cada uno tiene sus requisitos y limitaciones y, además, existen pocas comparativas homogéneas que orienten sobre su eficiencia y estabilidad en contextos diferentes. Para acabar con esta falta de consenso, este trabajo implementa y compara seis algoritmos representativos de las principales familias metodológicas de IL: Behavioral Cloning (BC), Behavioral Cloning from Observation (BCO), Generative Adversarial Imitation Learning (GAIL), Generative Adversarial Imitation from Observation (GAIfO), Adversarial Inverse Reinforcement Learning (AIRL) y Soft Q Imitation Learning (SQIL). Todo esto bajo un protocolo experimental controlado en dos de los entornos más usados como referencias para la comparación de algoritmos de RL, CartPole-v1 (discreto) y HalfCheetah-v4 (continuo) [2]. Conseguir demostraciones es muy costoso, por lo que el objetivo principal es evaluar su eficiencia muestral y generalización, y así guiar la selección de métodos según los datos disponibles y la complejidad del entorno.

2. Metodología y contribuciones técnicas

La selección de algoritmos realizada proporciona una amplia cobertura del campo, abarcando métodos supervisados, adversariales, de aprendizaje por refuerzo inverso y basados solo en observaciones. Otro criterio seguido para escogerlos era tener implementaciones disponibles o documentación suficiente para implementarlos.

La contribución más significativa consiste en la implementación desde cero de BCO, GAIfO y SQIL para entornos continuos, ya que estas implementaciones no existían previamente en librerías estándar. Mientras que BC, GAIL y AIRL se importaron de la librería *imitation*, los tres algoritmos implementados requirieron un desarrollo completo y validación basado en los artículos originales y sus resultados reportados.

3. Experimentos

El diseño experimental usa un protocolo homogéneo para garantizar comparabilidad y reproducibilidad que se observa en la Ilustración 1. CartPole-v1 se usa como entorno de validación por su simplicidad y convergencia rápida, mientras que HalfCheetah-v4 representa un benchmark estándar en literatura de control continuo de alta dimensionalidad [2].

El experimento consiste en la variación del número de trayectorias expertas (5, 10, 20, 50, 100) para evaluar la eficiencia muestral y la robustez ante datos escasos. Todos los algoritmos se entrenan durante 2M de pasos con la misma seed fija para reproducibilidad. La optimización de hiperparámetros se realizó con Optuna y su búsqueda bayesiana [3].

Se usó la recompensa acumulada y la desviación estándar como métrica principal para buscar la estandarización con la literatura [2]. Esta métrica permite comparar directamente el rendimiento de cada método con el del experto, que sigue una política óptima generada con *Soft Actor Critic* (SAC).

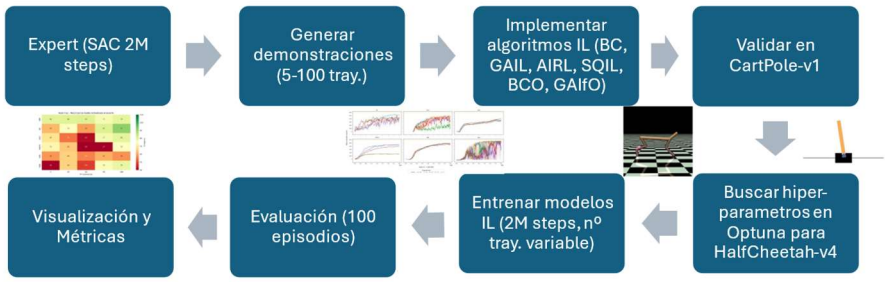

Ilustración 1. Pipeline experimental resumido

4. Resultados

La Ilustración 2 muestra los resultados de los experimentos. Estos resultados revelan que, la creencia de que los algoritmos cuanto más complejos mejor rinden, es falsa en este caso. BC alcanza el rendimiento más alto con abundantes datos, obteniendo un 91.2% del rendimiento experto con 100 trayectorias, superando significativamente a AIRL (~80%) y cuestionando la necesidad métodos más complejos cuando se tienen suficientes demostraciones.

El hallazgo más importante es la identificación de una "zona crítica" alrededor de 20 trayectorias, donde todos los algoritmos menos AIRL empeoran su rendimiento. BC cae un 15%, GAIfO un 39% y BCO tiene su peor resultado. Esta zona representa un umbral en el que el conjunto es demasiado diverso como para mantener coherencia interna, pero no lo suficiente para cubrir de forma adecuada el espacio de estados.

AIRL es el método más consistente. Se mantiene en un 80% del experto independientemente del número de trayectorias, lo que lo convierte en la opción más robusta ante datos variables. SQIL es el que más depende del volumen de los datos, pasa de un 38% a un 79% al tener acceso a más trayectorias. Esto confirma que su señal de recompensa +1/0 necesita que el espacio de estados esté muy cubierto o no será estable. Los métodos que se basan en observaciones (BCO y GAIfO) muestran su gran sensibilidad a la cantidad y distribución de las demostraciones, afectando su capacidad para inferir acciones.

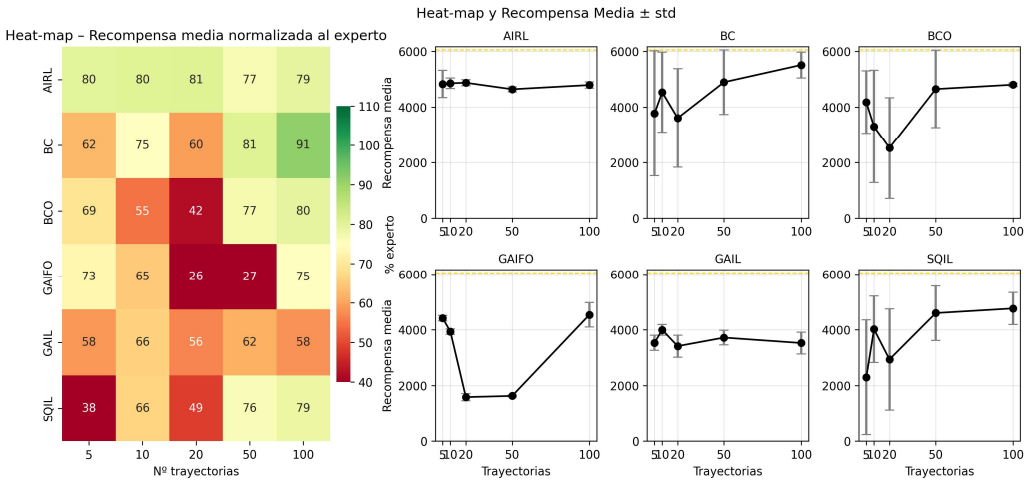

Ilustración 2 – Recompensa media normalizada y desviación estándar

5. Conclusiones

Este trabajo demuestra que no existe un algoritmo universalmente superior en IL, existe un equilibrio entre estabilidad y rendimiento máximo. Con muchas demostraciones, BC alcanza el mayor rendimiento, pero AIRL es el más consistente independientemente de las trayectorias. Las implementaciones desde cero de BCO, GAIfO y SQIL para entornos continuos, junto al protocolo experimental homogéneo, ofrecen herramientas útiles a la comunidad.

Las líneas futuras más prometedoras son el desarrollo de métodos híbridos que junten la estabilidad de AIRL con el rendimiento de BC, técnicas específicas para mitigar la "zona crítica" de alrededor de las 20 trayectorias, y validar estos resultados en entornos reales para comprobar su transferibilidad desde los entornos simulados (sim2real), expandiendo así el impacto del trabajo.

6. Referencias

[1] M. Zare, P. M. Kebria, A. Khosravi, y S. Nahavandi, «A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges», IEEE Trans. Cybern., vol. 54, n.o 12, pp. 7173-7186, dic. 2024, doi: 10.1109/TCYB.2024.3395626.

[2] N. Gavenski, F. Meneguzzi, M. Luck, y O. Rodrigues, «A Survey of Imitation Learning Methods, Environments and Metrics», 2024, arXiv: arXiv:2404.19456. doi: 10.48550/arXiv.2404.19456.

[3] T. Akiba, S. Sano, T. Yanase, T. Ohta, y M. Koyama, «Optuna: A Next-generation Hyperparameter Optimization Framework», en Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage AK USA: ACM, 2019, pp. 2623-2631. doi: 10.1145/3292500.3330701.

COMPARATIVE STUDY AND DEVELOPMENT OF IMITATION LEARNING ALGORITHMS: ANALYSIS AND APPLICATION

Author: Lorenzo Díaz-Meco, Antonio.
Director: Güitta López, Lucía.
Collaborating Entity: ICAI – Pontifical University of Comillas

Abstract

This work compares six Imitation Learning algorithms (BC, BCO, GAIL, GAIfO, AIRL, SQIL) in both continuous and discrete control environments. BCO, GAIfO and SQIL were implemented from scratch for continuous spaces, filling a gap in standard libraries. The results show that BC outperforms more complex methods when there are enough demonstrations, achieving 91.2% of expert performance, while AIRL is the most stable (~80%) against changes in the number of trajectories. A "critical zone" around 20 trajectories is identified, in which almost all algorithms degrade markedly. This work provides empirical guidelines for selecting algorithms considering the available data and stability requirements.

Keywords: Imitation Learning, Reinforcement Learning, Behavioral Cloning, Adversarial Learning, Sample Efficiency

1. Introduction

Manually designing reward functions in Reinforcement Learning (RL) is the biggest problem for applying it to complex tasks because one must explicitly define all desirable and undesirable behaviors, making it unfeasible for realistic and high-dimensionality environments [1]. Imitation Learning (IL) appears as an alternative that allows agents to learn directly from expert demonstrations. Manually defining the rewards is not necessary. However, the literature of IL is divided into various methodological approaches (Behavioral Cloning, inverse methods, adversarial, and based on observations), each one has their requirements and limitations and, also, not many comparative studies exist that can guide practitioners on efficiency and stability on different contexts. To address this lack of consensus, this work implements and compares six representative IL algorithms: Behavioral Cloning (BC), Behavioral Cloning from Observation (BCO), Generative Adversarial Imitation Learning (GAIL), Generative Adversarial Imitation from Observation (GAIfO), Adversarial Inverse Reinforcement Learning (AIRL) y Soft Q Imitation Learning (SQIL), under a controlled experimental protocol in two benchmark RL environments: CartPole-v1 (discrete) and HalfCheetah (continuous) [2]. Collecting demonstrations is costly, so the primary objective is to evaluate sample efficiency and generalization to provide guidance with method selection based on data availability and environment complexity.

2. Methodology and Technical Contributions

The selected algorithms cover the main IL families: supervised, adversarial, inverse-RL, and observation-only methods. Another criterion for selection was having either existing implementations or sufficient documentation to allow faithful re-implementation.

The most significant contribution is the implementation from scratch of BCO, GAIfO, and SQIL for continuous environments, as they were not previously available in standard

libraries. BC, GAIL and AIRL were imported from the *imitation* library, while the three newly implemented algorithms required full development and validation against their original papers and reported results.

3. Experiments

The experimental design uses a uniform protocol that guarantees comparability and reproducibility, as shown on Illustration 3. CartPole-v1 is used as a validation environment because of its simplicity and fast convergence, whereas HalfCheetah-v4 represents a standard benchmark in the high-dimensional continuous control literature [2].

The experiment consists of varying the number of expert trajectories (5, 10, 20, 50, 100) to evaluate sample efficiency and robustness with limited data. All the algorithms are trained for 2M steps with the same fixed seed for reproducibility. The optimization of hyperparameters was done with Optuna and its Bayesian search [3].

The accumulated reward and standard deviation was used as the main metric to try to standardize the literature [2]. This metric allows a direct comparison of each methods performance with the expert, that follows an optimal policy generated with *Soft Actor Critic* (SAC).

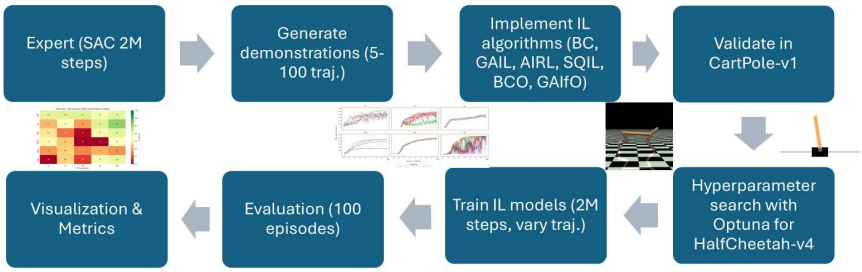

Illustration 3. Experimental Pipeline overview

4. Results

Illustration 4 shows the results of the experiments. These results reveal that the belief that the more complex an algorithm is, the better it performs is false in this case. BC achieves the highest performance with abundant data, achieving 91.2% of expert performance with 100 trajectories, significantly surpassing AIRL (~80%) and questioning the need for more complex methods when enough demonstrations are available.

The most important finding is the identification of a “critical zone” around 20 trajectories, where all the algorithms except AIRL perform worse. BC performance fell by 15%, GAIfO by 39% and BCO had its worst result. This zone represents a threshold where the dataset is too diverse to maintain internal coherence, but not enough to adequately cover the space of states.

AIRL is the most consistent method. It maintains 80% of expert performance regardless of the number of trajectories, which makes it the most robust option against variable data. SQIL depends the most on the data volume, going from 38% to 79% when more data is available. This confirms that its reward signal +1/0 needs the space of states to be thoroughly covered or it won't be stable. The methods that are based on observations (BCO and GAIfO) show high sensitivity to the amount and distribution of demonstrations, affecting their capacity to infer actions.

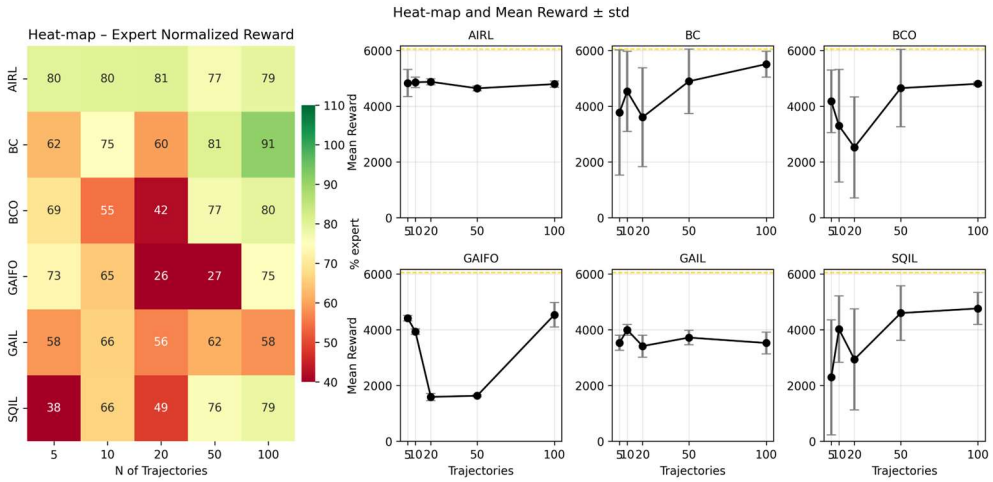

Illustration 4 – Normalized mean reward and standard deviation

5. Conclusions

This work demonstrates that a universally superior algorithm in IL does not exist, there is a trade-off between stability and performance. With many demonstrations, BC achieves the best performance, but AIRL is the most consistent regardless of the trajectories. The implementation of BCO, GAIFO and SQIL for continuous environments from scratch, with the uniform experimental protocol provide useful tools for the community.

The most promising future research directions are the development of hybrid methods that combine the stability of AIRL with the performance of BC, specific techniques to mitigate the "critical zone" around 20 trajectories, and validating these results on real environments to demonstrate their transferability from simulated environments (sim2real), expanding the impact of this work.

6. References

[1] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, «A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges», IEEE Trans. Cybern., vol. 54, n.o 12, pp. 7173-7186, dec. 2024, doi: 10.1109/TCYB.2024.3395626.

[2] N. Gavenski, F. Meneguzzi, M. Luck, and O. Rodrigues, «A Survey of Imitation Learning Methods, Environments and Metrics», 2024, arXiv: arXiv:2404.19456. doi: 10.48550/arXiv.2404.19456.

[3] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, «Optuna: A Next-generation Hyperparameter Optimization Framework», in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage AK USA: ACM, 2019, pp. 2623-2631. doi: 10.1145/3292500.3330701.

Índice de la memoria

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

En los últimos años, el diseño de recompensas manuales para agentes de IA se ha vuelto cada vez más complejo, especialmente en entornos continuos, frenando muchas aplicaciones de aprendizaje por refuerzo (RL). Imitation Learning (IL) ofrece otra vía: en lugar de premiar comportamientos, el agente aprende directamente a partir de demostraciones de un experto. Este trabajo revisa el estado del arte, implementa y compara seis algoritmos de IL en CartPole-v1 y HalfCheetah-v4, estos algoritmos son: Behavioral Cloning (BC), Behavioral Cloning from Observation (BCO), Generative Adversarial Imitation Learning (GAIL), Generative Adversarial Imitation from Observation (GAIfO), Adversarial Inverse Reinforcement Learning (AIRL) y Soft Q Imitation Learning (SQIL).

Además, se aportan implementaciones propias de BCO, GAIfO y SQIL en entornos continuos y un protocolo de evaluación homogéneo para ser reproducido fácilmente. Las implementaciones y scripts de experimentación están en un repositorio abierto, facilitando a la comunidad replicar los resultados y extender el análisis a otros entornos o variaciones de hiperparámetros, fomentando la transparencia y la comparabilidad de futuros estudios.

A continuación, se describe el contexto y los retos que convierten al Imitation Learning en un área con demanda creciente.

1.1. CONTEXTO Y MOTIVACIÓN

La irrupción del IL contrasta con la fragmentación metodológica: abundan algoritmos con requisitos de datos, estabilidad y costes muy distintos. Entre ellos destacan los supervisados (BC), los que infieren la función de recompensa (IRL), los adversariales (AIL) y los que se basan en observaciones (IfO). La literatura ofrece resultados aislados, faltan comparativas homogéneas que orienten cuándo conviene cada técnica [1], [2].

IL se ha consolidado como una línea de investigación clave en robótica, conducción autónoma y videojuegos, ya que permite transferir conocimientos de un experto a un agente sin especificar manualmente la función de recompensa [1]. Pero esas demostraciones son muy costosas de obtener, por lo que la eficiencia muestral (número mínimo de trayectorias expertas necesarias) es un criterio fundamental en entornos donde generar esas demostraciones es más complicado. Por otro lado, la estabilidad del aprendizaje puede variar drásticamente entre dominios continuos y discretos. Por ello, este trabajo escoge un entorno discreto y simple para validar las implementaciones y uno continuo más complejo, que exige configuraciones exhaustivas.

1.2. OBJETIVOS

El **objetivo general** es implementar y comparar los seis algoritmos escogidos de IL en dos entornos de Gymnasium: CartPole-v1 y HalfCheetah-v4.

Los **objetivos específicos** consisten en implementar desde cero BCO, GAIfO y SQIL, evaluar la eficiencia muestral variando el número de trayectorias expertas (5, 10, 20, 50, 100) y detectar si existe una "zona crítica", es decir, un punto a partir del cual añadir demostraciones no mejora casi el rendimiento.

Esto plantea varias preguntas de investigación como la rapidez de cada algoritmo en alcanzar al experto en entornos de distinta complejidad, la existencia de un umbral a partir del cual los métodos adversariales superan a los supervisados y cuál es el método que varía menos su recompensa sin sacrificar la eficiencia muestral.

Las pruebas se limitan a simulación, no se aborda transferencia sim2real ni demostraciones subóptimas ni aprendizaje multi-agente.

1.3. ESTRUCTURA DEL TRABAJO

Las secciones de este trabajo se dividen en: 2. Estado del Arte y criterios de comparación, 3. Metodología, que abarca los entornos y algoritmos

implementados, 4. Experimentos, 5. Resultados, y, por último, 6. Conclusiones. De este modo, el lector recorre primero el marco teórico, después los detalles de implementación y, al final, el análisis empírico.

2. ESTADO DEL ARTE

El **Imitation Learning (IL)** conecta el aprendizaje por refuerzo con el aprendizaje supervisado. Es una alternativa a situaciones en las que es demasiado complejo programar comportamientos o definir funciones de recompensa para cada tarea. Con este enfoque los agentes aprenden mediante la observación, reciben recompensas según las acciones realizadas e imitación de un experto [1].

La motivación fundamental está en usar demostraciones óptimas para acelerar el proceso de aprendizaje usando conocimiento experto, y así mejorar la eficiencia muestral de este tipo de algoritmos [2]. El campo enfrenta **desafíos críticos** que han marcado su evolución. El **problema del *covariate shift*** es una de las limitaciones centrales: el agente entrena con estados generados por la política experta, pero en ejecución utiliza su propia distribución de estados, acumulando errores que degradan el rendimiento a largo plazo. Otras limitaciones son **la dependencia de demostraciones óptimas** y que no **escala** bien en entornos complejos [1].

La evolución desde 1989 hasta 2024 (véase Figura 1) incluye varios hitos clave. En 1989 apareció Behavioral Cloning y se aplicó en conducción autónoma. En 1998 se desarrolló Inverse Reinforcement Learning. En 2008 llegó Maximum Entropy IRL. En 2011 se propuso DAgger. En 2016 surgieron los métodos adversariales con GAIL Y en 2022 se reconoció la necesidad de métodos robustos que manejen demostraciones subóptimas y discrepancias de dominio [1].

Esta línea del tiempo nos da las bases para analizar las familias algorítmicas que hay en la actualidad, sus aplicaciones más comunes y retos pendientes en IL que justifiquen que se siga investigando este campo.

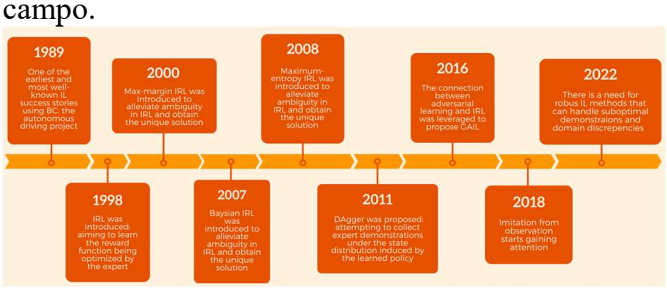

Figura 1. Evolución metodológica IL (Zare et al. 2023)

2.1. TAXONOMÍA DE MÉTODOS

2.1.1. BEHAVIORAL CLONING

Behavioral Cloning (BC) es el enfoque más directo en Imitation Learning. Trata el problema como aprendizaje supervisado donde el agente aprende a mapear estados a acciones usando demostraciones complejas [1]. Sin embargo, BC enfrenta el **problema de *covariate/distribution shift***, que consiste en que, durante la ejecución, el agente genera una distribución de estados diferente a la del experto y así, acumula errores que degradan su rendimiento a largo plazo [4].

Para mitigar esto, **Dataset Aggregation (DAgger)** introduce un enfoque interactivo que entrena de forma iterativa una política recopilando datos bajo la distribución del agente y solicitando correcciones del experto [5]. DAgger reduce el error compuesto (de T^2 a T) pero a costa de depender de **consultar al experto de manera frecuente**, lo que supone una carga cognitiva significativa, especialmente en entornos complejos [1].

Para hacerlo más eficiente aparecen variantes como **SafeDAgger** y **LazyDAgger**. SafeDAgger usa una política de seguridad auxiliar para decidir cuándo darle el control al experto y así reducir intervenciones innecesarias [6]. LazyDAgger limita las transiciones entre agente y experto usando un presupuesto de intervención, dándole prioridad a estados nuevos o de riesgo [7]. Estas extensiones reducen la carga humana un 40% comparado con la carga de DAgger [2].

Soft Q Imitation Learning (SQIL) es un enfoque diferente que combina *BC* con *RL*. Asigna **recompensas fijas**: +1 a transiciones expertas, 0 al resto. Optimiza una política usando RL, así el agente trata de volver a estados expertos cuando se desvía [8]. Esto evita consultar al experto cuando se desvía y también generaliza mejor que BC puro, pero necesita interaccionar más con el entorno [2].

Esta evolución refleja la búsqueda de equilibrio entre eficiencia muestral, carga experta y estabilidad en BC y sienta las bases para los métodos híbridos modernos [1].

2.1.2. INVERSE REINFORCEMENT LEARNING (IRL)

Inverse Reinforcement Learning (IRL) trata de **inferir la función de recompensa que hay detrás de las demostraciones expertas**. Asume que el experto actúa de forma óptima según esa función [1]. A diferencia de Behavioral Cloning, IRL entrena la política con algoritmos de *Reinforcement Learning (RL)*. Así permite que el agente explore de forma correctiva en distribuciones de estados no expertos, reduciendo el *covariate shift* [9].

Los métodos de **Maximum Margin** son los pioneros en IRL. El **problema de ambigüedad** en IRL se basa en que una sola demostración puede ser explicada por diferentes funciones de recompensa [1]. Por ello, este método busca una función de recompensa que justifique la política experta mejor que otras por un margen, maximizando la diferencia entre el valor esperado de las características del experto y otras políticas [10]. Sin embargo, asumen optimalidad estricta y, por tanto, son sensibles a demostraciones subóptimas [1].

Para manejar la ambigüedad y suboptimalidad se creó **Maximum Entropy IRL (MaxEntIRL)**, que introduce un enfoque probabilístico para manejarlo. Al maximizar la entropía de la distribución de trayectorias consigue explicar demostraciones estocásticas sin sobreajustar a ruido usando la distribución de Boltzmann sobre las trayectorias. Aunque es efectivo en espacios discretos, para usarlo en entornos continuos necesita aproximaciones costosas [11].

Guided Cost Learning (GCL) supera esto integrando optimización de políticas en tiempo real con redes neuronales para recompensas no lineales permitiendo estados crudos y mejorando la eficiencia muestral [12].

Los métodos **bayesianos (BIRL)** usan las demostraciones como evidencia para modelar una distribución posterior sobre funciones de recompensa [13]. Aunque BIRL maneja incertidumbre de forma inherente, es tan costoso computacionalmente que está limitado a espacios de baja dimensión [1]. Extensiones como **AVRIL** usan inferencia variacional para entornos complejos, pero introducen sesgos de aproximación [14].

Esta evolución metodológica, de modelos deterministas a probabilísticos, prioriza la generalización sobre el coste computacional. Tienen desafíos clave como la dependencia de modelos de transición conocidos y el supuesto implícito de optimalidad experta, crítico en aplicaciones reales con demostraciones subóptimas [1].

2.1.3. ADVERSARIAL IMITATION LEARNING (AIL)

Adversarial Imitation Learning (AIL) surge como alternativa escalable a IRL, evitando estimar explícitamente funciones de recompensa mediante el **entrenamiento adversarial entre un generador (política) y un discriminador**. Su objetivo es igualar la distribución de trayectorias del agente con las del experto, resolviendo así las limitaciones de eficiencia muestral y generalización [1].

Generative Adversarial Learning (GAIL) combina redes generativas adversariales (GANs) con demostraciones expertas. Tiene un generador (política del agente) que aprende a replicar el comportamiento experto mientras que un discriminador distingue entre trayectorias generadas y reales, quitando la necesidad de recompensas explícitas. GAIL establece la base, pero es inestable y sufre *mode collapse* por la divergencia de Jensen-

Shannon [15]. En entornos complejos como la conducción autónoma, GAIL tiende a generar comportamientos limitados que no capturan la diversidad humana [2].

InfoGAIL introduce variables latentes z que codifican factores no observados (ej. estilos de conducción en el caso de la conducción autónoma), así maximiza la información compartida entre z y las trayectorias. Esto hace que pueda generar comportamientos diversos usando píxeles crudos, pero a su vez, es más costoso computacionalmente ya que requiere modelos auxiliares para inferir z [16].

Adversarial Inverse Reinforcement Learning (AIRL) junta AIL con IRL, consiste en aprender una función de recompensa estructurada $r_\phi(s,a) = \log D_\phi(s,a)$ donde D_ϕ es el discriminador [17]. Esto hace que se las recompensas sean transferibles entre entornos, y así resuelve el *distribution shift* mejor que GAIL, pero requiere ajustar más hiperparámetros para evitar ser muy inestable [1].

Discriminator-Actor-Critic (DAC), es un algoritmo *off-policy* que separa el crítico del actor y hace uso de un *buffer* de repetición, lo que reduce las interacciones con el entorno en un 40% [18]. Sin embargo, al depender de muestras antiguas puede introducir sesgos en entornos dinámicos [1].

Primal Wasserstein Imitation Learning (PWIL) usa la distancia de Wasserstein en vez de la divergencia Jensen-Shannon, cambiando a un enfoque primal-dual que ayuda a la estabilidad [19]. Consigue políticas más robustas en entornos complejos de MuJoCo, pero a cambio de tener que hacer aproximaciones numéricas costosas y ajustar delicadamente los parámetros [1].

Esta evolución muestra la búsqueda de equilibrar la eficiencia, estabilidad e interpretabilidad. InfoGAIL y AIRL se centran en la diversidad y transferibilidad, por otro lado, DAC y PWIL se centran en optimizar la eficiencia muestral y la estabilidad. Gracias a esto se expanden las aplicaciones de AIL en robótica y sistemas autónomos [1].

2.1.4. IMITATION FROM OBSERVATION (IFO)

Imitation from Observation (IfO) aborda el desafío de aprender políticas que imiten usando solo trayectorias de estados del experto, sin tener acceso a las acciones que hay detrás. Este paradigma surge para aprovechar demostraciones disponibles en formatos no estructurados (ej. videos humanos), donde las acciones específicas no pueden observarse o registrarse [1].

Behavioral Cloning from Observation (BCO) es el enfoque pionero y divide el problema en dos etapas: aprender un modelo de dinámica inversa usando exploración autosupervisada para inferir acciones a partir de pares de estados y aplicar *Behavioral Cloning* normal sobre esas acciones inferidas. Sin embargo, hereda el problema del *covariate shift* del BC clásico, por lo que su dependencia de acciones estimadas imperfectamente hace que acumule errores en trayectorias largas [20].

Generative Adversarial Imitation from Observation (GAIfO) usa un marco adversarial para evitar inferir explícitamente las acciones. Se inspira en GAIL y usa una política que se optimiza para engañar a un discriminador entrenado para distinguir las transiciones (s, s') del experto de las del agente. Así elimina la necesidad de modelos de dinámica inversa y mejora la estabilidad. GAIfO necesita interactuar online con el entorno para generar trayectorias del agente, y eso complica su utilización en entornos costosos [21]

Time-Contrastive Networks (TCN) se basa en métricas de aprendizaje auto-supervisado y usa videos multivista sincronizados. Atrae en el espacio de *embedding* estados co-temporales de diferentes vistas (ancla y positivo) y repele estados que estén cerca temporalmente, pero diferentes en función (negativos). Así aprende sin importar la vista ni variables del entorno como iluminación u

oclusiones. Captura atributos funcionales, por ejemplo ángulo de inclinación en tareas de vertido [22]. A diferencia de GAIfO, TCN no necesita interactuar con el entorno, pero sí necesita datos multivista sincronizados para el entrenamiento [1].

Esta progresión muestra cómo se trata de balancear el requerimiento de datos, complejidad computacional y generalización, expandiendo el alcance práctico del IfO en robótica y sistemas autónomos.

2.2. PANORAMA DE ENTORNOS/MÉTRICAS

2.2.1. TAXONOMÍA DE ENTORNOS

En Imitation Learning los entornos se clasifican en tres categorías según su propósito evaluativo. Los **entornos de validación** aparecen en el 68% de los estudios, son simulaciones simples como MuJoCo (Hopper, Walker2D) y Gym (CartPole), sirven para probar ideas básicas sin añadir complejidades. Los **entornos de precisión** necesitan acciones discretas exactas sin dependencia temporal, son tareas como manipulación con brazos robóticos. Los **entornos secuenciales** evalúan las consecuencias a largo plazo de las acciones, como CARLA o SuperTuxKart para conducción autónoma [2]

El análisis de 66 entornos utilizados en literatura revela un comportamiento similar a la ley de Zipf: seis entornos (Walker2D, brazo robótico MuJoCo, Hopper, CartPole, MountainCar y HalfCheetah) representan más del 80% del uso total. Preocupantemente, 5 de 6 son entornos de validación, además, el 63% de entornos aparecen solo una vez en la literatura, evidenciando una falta de protocolos experimentales estandarizados [2].

Concentrar los estudios en entornos de validación da lugar a evaluaciones incorrectas, ya que los agentes aprenden a sobre-imitar sin entender la tarea subyacente. La falta de diversidad y consenso sobre qué entornos usar complica comparar los métodos y aumenta la brecha entre simulación y realidad [2].

2.2.2. TAXONOMÍA DE MÉTRICAS

En Imitation Learning hay tres tipos de métricas: **comportamiento**, **dominio** y **modelo**. Las **métricas de comportamiento** (73% de uso) miden qué tan cerca está el agente del experto, destaca la recompensa acumulada (68% de trabajos), que asume implícitamente optimalidad experta y puede distorsionar resultados si el experto no lo es [2]. La tasa de éxito episódico soluciona eso en parte, pero ignora aspectos como la seguridad.

Las **métricas de dominio** (22% de estudios) son específicas por tarea, por ejemplo, infracciones de tráfico en conducción autónoma. Pero hay tantas variantes en un mismo dominio que comparar resultados se vuelve difícil [2].

Las **métricas de modelo** miden componentes internos, como la precisión de dinámicas inversas, pero no reflejan con rendimiento real. Los mapas de saliencia son muy usados, pero son subjetivos y costosos de validar [2].

Existen carencias críticas en este sector, por ejemplo, solo el 12% de trabajos incluyen métricas cualitativas (evaluación humana, tests de Turing), que son vitales para aplicaciones centradas en humanos. Menos del 10% miden seguridad o robustez ante cambios distribucionales, que también es relevante en entornos críticos. La falta de estandarización se evidencia en que la mayoría de métodos no usan semillas diferentes para evitar el filtrado de datos [2].

2.3. DESAFÍOS Y ÁREAS SUBEXPLORADAS

El IL enfrenta desafíos y áreas subexploradas que hacen más difícil su uso en la práctica. La **dependencia de demostraciones óptimas** es un gran problema, ya que los métodos asumen que las demostraciones expertas son óptimas, lo que lleva a políticas frágiles cuando las muestras tienen ruido o no son óptimas [2]. En áreas como la robótica médica este problema empeora porque las demostraciones casi nunca son perfectas [1].

La **seguridad y robustez** son áreas críticas que no se tienen en cuenta habitualmente. Solo el 12% de los trabajos incluyen métricas explícitas de

seguridad, y menos del 5% evalúan el rendimiento con distribuciones cambiantes o perturbaciones en el entorno [2]. En aplicaciones como la conducción autónoma esto es muy importante ya que errores compuestos pueden tener consecuencias catastróficas.

En cuanto a **escalabilidad**, existe una brecha entre métodos de validación (ej. CartPole) y complejos (ej. HalfCheetah). El 89% de los métodos se evalúan solo en simulaciones simples, mostrando gran degradación del rendimiento al transferirse a entornos físicos [1], [2].

Algunas áreas están subexploradas. Los **sistemas multi-agente** casi no se evalúan porque pocos entornos permiten interacciones entre varios agentes [2]. El **aprendizaje en línea con restricciones computacionales** está poco cubierto, casi no hay métodos que consideren memoria o energía en dispositivos embebidos [1]. También faltan **métricas cualitativas**, solo el 12% de los trabajos incluyen evaluaciones humanas, que son muy relevantes para tareas que requieran interpretabilidad [2].

2.4. PROBLEMAS ABORDADOS

Este trabajo aborda dos desafíos clave identificados en la literatura: **eficiencia muestral** y **generalización entre dominios**. La eficiencia muestral responde al hallazgo de que la mayoría de métodos requieren más de mil demostraciones para rendimiento óptimo [1]. La generalización entre entornos aborda la brecha de que solo el 12% de algoritmos se evalúan en varios dominios [2].

Al implementar y comparar 6 algoritmos (BC, GAIL, AIRL, SQIL, BCO, GAIfO) bajo **condiciones controladas** (mismo número de steps, número de trayectorias variable), se analiza cómo cada método escala desde entornos simples (CartPole-v1) a complejos (HalfCheetah-v4), ambos son simulados así que la brecha *sim2real* no se abordará.

SQIL y AIRL mitigan la dependencia de optimalidad experta mediante recompensas estructuradas y regularización adversarial. BC y GAIL comparan paradigmas clásicos supervisados contra adversariales para ver cuánto se degrada cada uno bajando el número de trayectorias. BCO y GAIfO exploran IfO, que son métodos recientes que están cobrando importancia con la cantidad de datos no estructurados a los que se tiene acceso actualmente [1].

La elección de usar la **recompensa acumulada** como métrica principal sigue críticas recientes sobre la necesidad de estandarización [2], mientras que la variación en el número de trayectorias expertas explora directamente la dependencia de optimalidad del experto y el verdadero cuello de botella del IL, conseguir demostraciones expertas de calidad [1].

3. METODOLOGÍA

3.1. ENTORNOS

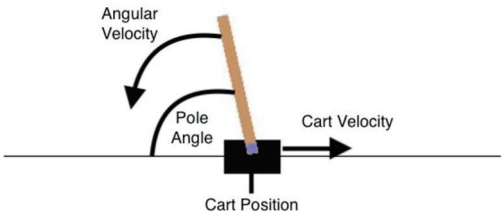

Figura 2. CartPole-v1

CartPole-v1 es un entorno de control clásico con espacio de observación de 4 dimensiones en el que se incluye la posición del carro, su velocidad, ángulo del poste y velocidad angular del poste (véase Figura 2). El espacio de acciones es **discreto** y tiene dos posibles acciones: mover el carro hacia la izquierda o derecha. El objetivo consiste en mantener un poste vertical equilibrado sobre un carro en movimiento. Se termina el episodio si el poste se inclina más de 15 grados de la vertical, se mueve más de 2.4 unidades del centro o se alcanzan los 500 pasos. La recompensa es de +1 cada segundo que el poste permanece equilibrado [23].

HalfCheetah-v4 es un entorno de control continuo más complejo de la suite MuJoCo. Tiene un espacio de observación de 17 dimensiones e incluye información sobre posición, orientación,

velocidades lineales y angulares del cuerpo del guepardo. El espacio de acciones es **continuo** y tiene 6 dimensiones que corresponden a los torques que se aplican a las articulaciones del guepardo. El objetivo es maximizar la velocidad a la que se desplaza hacia adelante mientras minimiza la energía gastada en los movimientos. La función de recompensa combina la velocidad de avance con una penalización por usar excesivamente los actuadores [24]. El entorno se puede observar en la Figura 3.

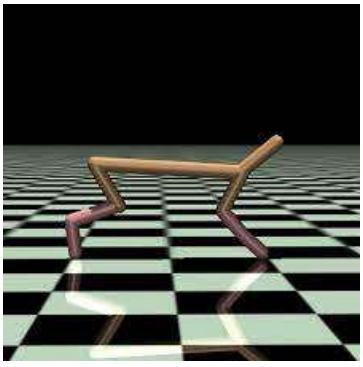

Figura 3. HalfCheetah-v4

3.2. BEHAVIORAL CLONING (BC)

El **Behavioral Cloning (BC)** se define como un método de **aprendizaje supervisado** en el contexto del aprendizaje por imitación, donde el objetivo es replicar la política experta minimizando una función de pérdida entre acciones predichas y demostradas. Su formulación matemática se basa en el principio de máxima verosimilitud: dado un conjunto de demostraciones $\mathcal{D} = (s_i, a_i)_{i=1}^{N}$, se optimizan los parámetros θ de una política $\pi_\theta(a \mid s)$ para maximizar la probabilidad logarítmica de las acciones expertas condicionadas a los estados observados [25]:

$$\theta^* = \arg\max_\theta \frac{1}{N}\sum_{i=1}^{N} \log \pi_\theta(a_i \mid s_i)$$

En problemas de control continuo, la pérdida práctica incluye, además, un término de regularización L2 sobre los pesos [25] :

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N} \log \pi_\theta(a_i \mid s_i) + \lambda|\theta|_2^2$$

La arquitectura de la política difiere según el espacio de acciones. Para entornos **discretos** como CartPole-v1, se usa una capa *softmax*. Esta capa genera una distribución categórica sobre las acciones posibles. Para espacios **continuos** como HalfCheetah-v4, modelamos la política con una distribución gaussiana. La red neuronal predice la media $\mu_\theta(s)$ y la varianza $\sigma_\theta(s)$, que puede ser fija o aprendida [25]. Gracias a esta flexibilidad en su arquitectura se puede adaptar BC a dominios tan diversos como la conducción autónoma y manipulación robótica [1].

El entrenamiento se implementa mediante descenso de gradiente estocástico con mini-lotes. Convergemos cuando se llega al número máximo de épocas o no varía la pérdida. Véase Algoritmo 1 donde se detalla el proceso de aprendizaje.

Algoritmo 1 BC

Require: Demostraciones $D = (s_i, a_i)_{i=1}^{N}$, tasa de aprendizaje η, regularización λ
Ensure: Parámetros óptimos θ^*

```
Inicializar θ~N(0,σ²)
Repetir
  Extraer mini-lote B ⊂ D
  Para cada (s,a) ∈ B:
    l ← −log(π_theta(a|s))
  Fin
  L ← (1/|B|) Σ_{(s,a)∈B} l + λ‖θ‖²
  θ ← θ − η · ∇_θ L
Hasta que se cumpla el criterio de convergencia
Retornar θ* ← θ
```

Algoritmo 1. Behavioral Cloning.

BC es un algoritmo **estrictamente offline**, esto se debe a que para la actualización de θ sólo se emplea demostraciones pregrabadas $\mathcal{D}$ sin consultas al entorno durante el entrenamiento [4].

No obstante, BC tiene dos limitaciones teóricas críticas. La primera es el ***covariate shift***, que analiza Ross et al. Este fenómeno surge porque durante el

entrenamiento, la política aprende únicamente sobre la distribución de estados visitados por el experto, pero durante la ejecución, pequeños errores pueden llevar al agente a estados que no han sido observados en las demostraciones. Ross et al. demostraron que un clasificador con tasa de error ε en la distribución experta genera una pérdida proporcional a $T^2\varepsilon$ en su propia distribución de estados. Esa dependencia hace que no suela ser aplicado a tareas de larga duración [5].

La segunda limitación es el ***problema copycat*** que identificó Chuang et al. en entornos con observaciones parciales. Ocurre cuando el agente desarrolla dependencias *sparse* entre acciones sucesivas (e.g, $a_{t-1} \approx a_t$) y omite información contextual importante. En conducción autónoma, esto puede llegar a provocar colisiones si, por ejemplo, el agente acelera en lugar de frenar ante un obstáculo ya que la política interpreta erróneamente la acción previa como la que predice la actual. Chuang et al. propuso arquitecturas de red neuronal *"copycat-free"* que emplean módulos de extracción de memoria especializados para evitar el flujo de información espuria desde acciones previas [4].

3.3. *GENERATIVE ADVERSARIAL IMITATION LEARNING (GAIL)*

El **aprendizaje por imitación adversarial** surge como una alternativa innovadora a las limitaciones de *Behavioral Cloning*, particularmente el problema de *distribution/covariate shift* que afecta el rendimiento cuando el agente se desvía de las trayectorias demostradas. Ho y Ermon presentaron ***Generative Adversarial Imitation Learning (GAIL)***. GAIL reformula el problema de imitación como un juego adversarial inspirado en las **Redes Generativas Adversariales (GANs)** [15]. Aprende políticas óptimas sin reconstruir explícitamente la función de recompensa. Así reduce la ineficiencia computacional de los métodos clásicos de *Inverse Reinforcement Learning* [1].

Su formulación matemática se basa en un juego **min-max** entre una política generadora π_θ y un discriminador D_w donde el objetivo consiste en encontrar el punto de silla de la siguiente expresión [15]:

$$\min_\theta \max_w E_{(s,a)\sim\mathcal{D}}[\log D_w(s,a)] + E_{(s,a)\sim\pi_\theta}\left[\log\left(1 - D_w(s,a)\right)\right] - \lambda H(\pi_\theta)$$

donde $\mathcal{D}$ representa las demostraciones expertas, $H(\pi_\theta)$ denota la entropía de la política como término regularizador, y $\lambda \geq 0$ controla la fuerza de esta regularización. El discriminador $D_w: S \times A \to (0,1)$ aprende a distinguir entre pares estado-acción que provienen del experto de los que son generados por la política en el entrenamiento [15].

La clave del método está en la transformación de la salida del discriminador en una señal de recompensa para el entrenamiento por refuerzo. La recompensa se define como $\tilde{r}(s,a) = -\log\left(1 - D_w(s,a)\right)$, y proporciona recompensas altas cuando el discriminador se equivoca y clasifica las acciones del agente como expertas [26]. Esta formulación garantiza que minimizar la divergencia Jensen-Shannon entre las medidas de ocupación de la política aprendida y del experto lleven a imitar exactamente el comportamiento demostrado [1].

La arquitectura del discriminador se basa en una MLP con activación *tanh* y *batch normalization* que hace más estable el entrenamiento. La política usa el esquema actor-crítico y se optimiza con algoritmos de RL como PPO o TRPO [15]. El proceso de entrenamiento se puede ver en el Algoritmo 2. Alterna entre actualizar el discriminador y optimizar la política usando el algoritmo escogido con la recompensa que sale del discriminador [27].

Algoritmo 2 GAIL

Require: Demostraciones expertas $\mathcal{D}$, hiperparámetros de entrenamiento

Ensure: Política entrenada π_θ

Inicializar parámetros de política θ y discriminador w

Mientras no convergencia:

  Recoger rollouts (s_t, a_t) ejecutando π_θ // Muestreo interactivo

```
Actualizar w maximizando:
E_{(s,a)~D}[logD_w(s,a)] + E_{(s,a)~π_θ}[log(1 − D_w(s,a))]
Calcular señal de recompensa:
ṙ(s,a) ←− log(1 − D_w(s,a))        // Transformación adversarial
Optimizar θ usando PPO/TRPO con ṙ(s,a)      // Actualización de política
Retornar π_θ
```

Algoritmo 2. Generative Adversarial Imitation Learning.

GAIL hereda la **inestabilidad** típica del entrenamiento adversarial. Se dan curvas de entrenamiento oscilantes y sensibilidad extrema a hiperparámetros como la frecuencia de actualización entre discriminador y política. Su fragilidad puede causar *mode collapse* o divergencia del entrenamiento, por ello hay que ajustar cuidadosamente los parámetros y arquitecturas [28]. Para poder abordar estas limitaciones, han surgido variantes como AIRL, que incorporan términos de regularización adicionales para mejorar la estabilidad [26], y métodos más recientes como DRAIL que integran modelos de difusión para producir recompensas más robustas y suaves [28].

3.4. ADVERSARIAL INVERSE REINFORCEMENT LEARNING (AIRL)

Adversarial Inverse Reinforcement Learning (AIRL) fue concebido como una **extensión de *GAIL*** que aborda una gran limitación: la incapacidad de **recuperar funciones de recompensa transferibles** y robustas a cambios en la dinámica del entorno. Mientras que *GAIL* aprende políticas que aprenden eficazmente, su discriminador converge a una salida uniforme de 0.5 en el óptimo, y eso impide poder extraer una función de recompensa útil. *AIRL* reformula el problema adversarial para recuperar a la vez una política de imitación y una función de recompensa que generalice a entornos con dinámicas modificadas. Su formulación matemática consiste en que *AIRL* plantea el siguiente objetivo adversarial [17]:

$$\min_{\theta} \max_{\phi,w} E_{\mathcal{D}}\left[\log D_{\phi,w}(s,a,s')\right] + E_{\pi_\theta}\left[\log\left(1 - D_{\phi,w}(s,a,s')\right)\right]$$

donde $\mathcal{D}$ representa las demostraciones expertas y π_θ la política del agente. La novedad está en la estructura que tiene el discriminador [17]:

$$D_{\phi,w}(s,a,s') = \sigma\left(f_{\phi,w}(s,a,s')\right),$$

$$f_{\phi,w}(s,a,s') = g_w(s,a) + \gamma\, h_\phi(s') - h_\phi(s)$$

Esta descomposición separa la función de recompensa $g_w(s,a)$ del término de *reward shaping* $\gamma h_\phi(s') - h_\phi(s)$, donde h_ϕ aproxima la función de valor. Coincide con la formulación de IRL de máxima entropía causal, ya que el *shaping* $\gamma\, h_\phi(s') - h_\phi(s)$ emula el cambio de valor en el objetivo de entropía máxima [17]. Así optimiza simultáneamente la recompensa y la entropía causal de la política. La arquitectura garantiza que, en el óptimo, g_w recupere la función de recompensa real hasta una constante y que h_ϕ capture los efectos dinámicos del entorno de entrenamiento [10].

La señal de recompensa para la política es derivada directamente del discriminador. Se implementa en la práctica como [17]:

$$\tilde{r}(s,a,s') = \log D_{\phi,w}(s,a,s') - \log\left(1 - D_{\phi,w}(s,a,s')\right)$$

Teóricamente esta señal coincide con $f_{\phi,w}(s,a,s') - \log \pi_\theta(a|s)$ tras la derivación de máxima entropía causal, pero en el entrenamiento real se añade el término de entropía $-\log \pi_\theta$ como regularizador aparte en el objetivo de la política [17].

El discriminador usa una red neuronal con tres capas totalmente conectadas, activaciones *ReLU* y *batch normalization* [17]. La política, al igual que en GAIL, utiliza arquitecturas estándar de actor-crítico o algoritmos como TRPO/PPO [15], [17]. El entrenamiento, como se puede ver en el Algoritmo 3, sigue un bucle iterativo [17] en el que primero muestrea trayectorias con π_θ, luego actualiza el discriminador $D_{\phi,w}$ con descenso de gradiente y optimiza π_θ usando la recompensa derivada $\tilde{r}$.

Algoritmo 3 AIRL

Require: Demostraciones expertas $\mathcal{D}$, hiperparámetros α, β, λ

Ensure: Política entrenada π_θ, función de recompensa g_w

1 Inicializar θ (política), w, φ (discriminador)

2 **Mientras no convergencia:**

3 Muestrear trayectorias $\tau \sim \pi_\theta$ // Muestreo interactivo

4 Actualizar discriminador (φ, w) vía gradiente ascendente:

$$(\varphi, w) \leftarrow (\varphi, w) + \alpha \nabla \left[E_{(s,a,s')\sim\mathcal{D}} \left[log\, D_{\varphi,w}(s,a,s') \right] + E_{(s,a,s')\sim\pi_\theta} \left[log \left(1 - D_{\varphi,w}(s,a,s') \right) \right] \right]$$

5 Calcular recompensa: $\dot{r}(s,a,s') \leftarrow -log \left(1 - D_{\varphi,w}(s,a,s') \right)$ // Transformación adversarial

6 Optimizar π_θ usando RL con $\dot{r} + \lambda H(\pi_\theta)$ // Actualización de política entrópica

7 **Retornar** π_θ, g_w

Algoritmo 3. Adversarial Reinforcement Imitation Learning

Sobre sus limitaciones, AIRL hereda la **inestabilidad** inherente al entrenamiento adversarial, requiriendo ajustar cuidadosamente la tasa de actualización entre discriminador y generador (como en GAIL), así como del coeficiente de entropía [17], [28]. La convergencia puede ser sensible a la inicialización y la calidad de las demostraciones expertas, de forma particular en los entornos con alta dimensionalidad donde es desafiante estimar la función de valor h_ϕ [17].

Varias extensiones de AIRL mejoran la aplicabilidad y superan sus limitaciones. Augmented AIRL incorpora recompensas semánticas de dominio y estabiliza tareas complejas, como la conducción autónoma, con señales de éxito, colisión y márgenes de seguridad [29]. La versión *off-policy* de AIRL aumenta la eficiencia muestral. Emplea un buffer de replay y corrección de importancia [30]. DYNAIL adapta AIRL a entornos cambiantes. Regulariza la divergencia de las medidas de ocupación [26].

3.5. SOFT Q IMITATION LEARNING (SQIL)

Soft Q Imitation Learning (SQIL) es un método que combina la simplicidad del *Behavioral Cloning (BC)* con el aprendizaje dinámico de *Reinforcement Learning (RL)*, y además, evita el entrenamiento adversarial de *GAIL* [1]. A diferencia de BC, que sufre *covariate shift* al ignorar las transiciones de estado, *SQIL* incentiva al agente a imitar demostraciones expertas y regresar a los estados demostrados usando recompensas binarias. Esto hace que supere las limitaciones de *BC* sin necesitar aprender funciones de recompensa, lo que facilita su implementación en entornos de alta dimensionalidad [8].

Su formulación matemática se basa en que *SQIL* asigna **recompensas *sparse*** según el origen de las transiciones, si $(s,a) \in \mathcal{D}_{\text{demo}}$ la recompensa es $r(s,a) = +1$, pero si es una experiencia propia $r(s,a) = 0$, donde $\mathcal{D}_{\text{demo}}$ contiene pares estado-acción expertos [8]. La actualización de la función Q sigue la ecuación de Bellman suave:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r(s,a) + \gamma \log \sum_{a'} \exp\left(Q(s',a') \right) - Q(s,a) \right]$$

Este operador $r + \gamma \log \sum_{a'} \exp$ Q proviene de SAC y añade entropía para promover la exploración. Proposición 1 de Reddy et al. demuestra que, con recompensas 1,0 la política óptima de SQIL aproxima la medida de ocupación experta $\rho_\pi(s,a) \approx \rho_{Exp}(s,a)$. De este modo, SQIL puede verse como BC regularizado por el mismo error de Bellman aplicado tanto a demostraciones como a datos propios [8].

El proceso de entrenamiento (Véase Algoritmo 4) **intercala muestras expertas y generadas** (inicialmente 50%) y optimiza Q_θ mediante el error cuadrático de Bellman, que es la función de pérdida global [8].

Algoritmo 4 SQIL

Require: Demostraciones expertas $\mathcal{D}_{demo}$, tasa de aprendizaje α, factor de descuento γ

Ensure: Política entrenada $\pi(a|s) \propto exp(Q_\theta(s,a))$

1 Inicializar $Q_\theta(s,a)$ aleatoriamente

2 Llenar buffer de experiencias con $\mathcal{D}_{demo}$ (recompensa = +1)
3 **Mientras no convergencia:**
4 Muestrear lote (s, a, s') del buffer (50 % demostraciones, 50 % propias)
5 Calcular $y \leftarrow r + \gamma log \sum_{a'} exp(Q_\theta(s', a'))$ // Actualización suave
6 Actualizar $\theta \leftarrow \theta - \alpha \nabla_\theta [Q_\theta(s, a) - y]^2$
7 **Retornar** $\pi(a|s) \propto exp(Q_\theta(s, a))$ //Política Boltzmann

Algoritmo 4. Soft Q Imitation Learning.

SQIL utiliza dos redes MLP idénticas para aproximar Q_θ. En control continuo, la actualización se implementa sobre la cabeza Q de SAC, lo que reutiliza su *temperature* α y su *target network* estable. La exploración *off-policy* permite reutilizar demostraciones sin necesidad de acceder continuamente al experto [8].

Tiene varias limitaciones, entre ellas la relevancia del **balance experto/exploración**, un desbalance >70/30 en el *buffer* degrada el rendimiento hasta un 37%. La **convergencia asintótica** no garantiza igualar la medida de ocupación de estados del experto, solo aproximaciones locales [31]. La **escalabilidad** en espacios de alta dimensión depende en gran medida de la elección inicial de recompensas *sparse* [8].

DSQIL añade un discriminador adversarial que ajusta las recompensas de forma dinámica y así mejora un 15% en tareas complejas [32]. *IQ-Learn* reformula la pérdida como una f-divergencia convexa, así duplica la eficiencia muestral en MuJoCo [31].

3.6. BEHAVIORAL CLONING FROM OBSERVATION (BCO)

Behavioral Cloning from Observation (BCO) aborda la limitación fundamental del aprendizaje por imitación tradicional que requiere acceso a acciones expertas explícitas. BCO usa exclusivamente **trayectorias de estados observados**, haciendo que se pueda aprender usando videos de YouTube u otras fuentes de demostración sin anotación de acciones [20]. Esta capacidad es crucial para aprovechar la vasta cantidad de datos de comportamiento disponibles en formato visual.

El algoritmo BCO(α) es la formulación completa del método, y BCO estándar representa el caso especial $\alpha = 0$. La metodología consta de **dos fases**. La **fase de preentrenamiento** consiste en recopilar un conjunto de demostraciones previas $\mathcal{D}_{pre}$ explorando aleatoriamente, el agente sigue una política estocástica para generar transiciones (s_t^a, a_t, s_{t+1}^a). Después, se entrena un modelo de dinámicas inversas $\mathcal{M}_\theta: S^a \times S^a \rightarrow \mathrm{A}$ aplicando estimación de máxima verosimilitud sobre las transiciones que se han extraído [20].

La **fase de *Behavioral Cloning*** usa el modelo inverso entrenador para inferir, en cada transición de $\mathcal{D}_{\text{demo}}$, la acción $\widetilde{a_t} = \mathcal{M}_\theta(s_t, s_{t+1})$. Luego se optimiza la política imitadora $\pi_\phi(a|s)$ usando *BC* estándar sobre los pares $(s_t, \widetilde{a_t})$ que se han inferido [20].

BCO(α) extiende esta formulación **iterando las mismas fases** cuando $\alpha > 0$. En cada iteración k, el agente realiza $M_k = \alpha \mathcal{I}^{\text{pre}}$ interacciones adicionales usando la política actual $\pi_\phi^{(k)}$, reentrenando posteriormente tanto el modelo inverso como la política con los datos ampliados [20]. Este proceso permite el refinamiento progresivo a costa de interacción post-demostración controlada.

La lógica completa de entrenamiento se encuentra en el Algoritmo 5 que, usando la distinción $\alpha > 0$ diferencia entre el BCO estándar y BCO(α).

Algoritmo 5 BCO

Require: Conjunto de demostraciones $\mathcal{D}_{demo}$, parámetro α, iteraciones pre-demostración $|\mathcal{I}^{pre}|$
Ensure: Política entrenada π_φ
1 Inicializar política π_φ y modelo inverso $\mathcal{M}_\theta$
2 Fijar $I = |\mathcal{I}^{pre}|$
3 **Si** $\alpha = 0$ // BCO estándar
4 **Para** t = 1 hasta I
5 Generar transiciones (s_t^a, s_{t+1}^a, a_t) con π_φ
6 Almacenar en $\mathcal{D}_{pre}$

```
    Fin Para
    Entrenar M_θ en D_pre vía máxima verosimilitud
    Inferir acciones Ã_demo = M_θ(s_t, s_{t+1})_{t=1}^N
    Entrenar π_φ en (S_demo, Ã_demo) vía BC
  Sino    // BCO(α) con mejora iterativa
    Mientras π_φ no converge
      Para t = 1 hasta I
        Generar transiciones (s_t^a, s_{t+1}^a, a_t) con π_φ^(k)
        Actualizar D_pre
      Fin Para
      Reentrenar M_θ con D_pre ampliado
      Re-inferir Ã_demo con nuevo M_θ
      Actualizar π_φ^(k+1) vía BC con Ã_demo
      Fijar I = α · |J^pre|  // Interacciones post-demostración
    Fin Mientras
  Fin Si
  Retornar π_φ
```

Algoritmo 5. Behavioral Cloning from Observation

La arquitectura usa MLP específicas. El modelo inverso $\mathcal{M}_\theta$ tiene 2 capas de 256 neuronas con activación ReLU. La política se modela como distribución gaussiana en espacios de acción continuos. En BCO(α) con $\alpha = 0.1$, logran el 83% del rendimiento experto con solo un 10% de interacciones extra en entornos complejos como Ant-v1 [20].

Sus dos limitaciones más importantes son la dependencia crítica de la calidad del modelo inverso y la suposición de dinámicas compartidas entre demostrador e imitador. BCO entrena el modelo inverso $\mathcal{M}_\theta$ usando $\mathcal{D}_{pre}$. Este enfoque asume **dinámicas deterministas y compartidas** entre el demostrador y el imitador. Si el entorno cambia, el modelo inverso $\mathcal{M}_\theta$ deja de reflejar las nuevas dinámicas y pierde validez. El método se degrada significativamente en entornos estocásticos, con oclusiones [20].

Han aparecido varias extensiones, entre ellas GAIfO que sustituye el modelo inverso por un discriminador adversarial que opera sobre pares $(\boldsymbol{s}, \boldsymbol{s'})$ [21]. FORM usa modelos generativos de efectos de política, exhibiendo mayor robustez ante distractores visuales y ruido en comparación con métodos adversariales [33].

3.7. GENERATIVE ADVERSARIAL IMITATION FROM OBSERVATION (GAIFO)

Generative Adversarial Imitation from Observation (GAIfO) es un gran avance en el Imitation Learning, ya que permite aprender políticas solo con la **observación de transiciones de estado** (s, s'), es decir, no necesita conocer las acciones expertas [21]. Está **inspirado en GAIL**, pero usando solo transiciones de estado, resuelve limitaciones críticas de los métodos clásicos al hacer posible el entrenamiento mediante videos o demostraciones visuales en las que no se conocen las acciones del experto [34].

La formulación matemática central de GAIfO se basa en un juego **min-max** que minimiza la divergencia entre las medidas de ocupación $\rho_\pi(s, s')$ del agente y $\rho_{\pi_E}(s, s')$ del experto:

$$\min_{\pi} \max_{D} E_\pi[\log D\,(s, s')] + E_{\pi_E}\big[\log\big(1 - D(s, s')\big)\big] - \lambda \mathcal{H}(\pi)\,,$$

donde el **discriminador D estima la probabilidad de que una transición (s, s') venga del experto** [21]. La recompensa adversarial $r(s, s') = \log D\,(s, s')$ usa *RL* para guiar el aprendizaje y así incentiva al agente a generar trayectorias indistinguibles de las demostraciones [34].

El algoritmo sigue un esquema iterativo de 3 etapas (véase Algoritmo 6): **generación de trayectorias** con la política actual, **actualización del discriminador** usando clasificación binaria mediante la minimización de la pérdida de entropía cruzada, que penaliza los errores de clasificación entre transiciones expertas y generadas [34], y **optimización de la política** usando TRPO con la recompensa adversarial. Se usa TRPO porque garantiza actualizaciones estables mediante regiones de confianza y así evita cambios bruscos en políticas estocásticas de alta dimensionalidad [35].

La arquitectura típica emplea una política estocástica (MLP con salida gaussiana) y un discriminador (MLP o CNN para estados visuales) que procesa pares de estados consecutivos [21].

Algoritmo 6 GAIfO

Require: Trayectorias expertas solo-estado $\tau_E = (s, s')$, hiperparámetros de entrenamiento

Ensure: Política entrenada π_φ

1 Inicializar política paramétrica π_φ con parámetros aleatorios φ

2 Inicializar discriminador paramétrico D_θ con parámetros aleatorios θ

3 **Mientras** la política mejora:

4 Ejecutar π_φ y almacenar transiciones de estado resultantes $\tau = (s, s')$ // Muestreo interactivo

5 Actualizar D_θ usando función de pérdida: $-\left(E_\tau[log\, D_\theta(s,s')] + E_{\tau_E}[log\,(1 - D_\theta(s,s'))]\right)$

6 Calcular función de recompensa adversarial: $r(s,s') \leftarrow -log\,(D_\theta(s,s'))$ // Transformación adversarial

7 Actualizar π_φ realizando actualizaciones TRPO con $r(s,s')$ // Actualización de política

8 **Fin Mientras**

9 **Retornar** π_φ

Algoritmo 6. Generative Adversarial IfO

A pesar de su potencial, GAIfO tiene varias limitaciones. La **inestabilidad** típica del entrenamiento adversarial, que requiere ajuste fino de sus hiperparámetros como la frecuencia de actualización del discriminador [34]. Su **eficiencia muestral limitada** dificulta aplicaciones en robots físicos [36], y si un entorno tiene dinámicas diferentes al experto su rendimiento decae [37].

Las extensiones más recientes corrigen estas limitaciones, se ha descubierto que las señales propioceptivas aceleran la convergencia en tareas visuales complejas [38], y usar técnicas de aprendizaje auto-supervisado reduce las demostraciones a unas 100 muestras [36]. Para transferencia entre dominios, GARAT combina GAIfO con transformaciones de acción adaptativas, demostrando éxito en entornos con desajustes dinámicos [37].

4. EXPERIMENTOS

Los experimentos se realizaron en dos entornos de Gymnasium. Para validar que los modelos funcionen correctamente se usó ***CartPole-v1***, ya que es de acción discreta, baja dimensionalidad y episodios cortos en el que todos los algoritmos convergían a la recompensa máxima rápidamente. Una vez validados, se pasó a ***HalfCheetah-v4*** para comparar y evaluar los algoritmos en un entorno de acción continua y alta dimensión, para el cual se necesita mayor capacidad de red, ajustar precisamente los hiperparámetros y entrenamientos más largos.

4.1. GENERACIÓN DE TRAYECTORIAS

Para generar políticas expertas, se utilizó **SAC (*Soft Actor-Critic*)** de *Stable*-Baselines3, se entrenó durante **2M de pasos**, alcanzando una recompensa media de **6036.72 ±43.80** en 100 episodios de evaluación en *HalfCheetah* y se monitorizó que convergiese de forma estable con *Tensorboard*. Para *CartPole-v1* se usó **PPO (Proximal Policy Optimization)**, ya que SAC no es compatible con entornos discretos, y se obtuvo una recompensa de 500 estable.

Las trayectorias se obtienen ejecutando la política entrenada sin exploración (determinista). Cada episodio se registra como un objeto *TrajectoryWithRew* que incluye observaciones, acciones y recompensas originales del entorno.

Para asegurar datos de calidad y variados, se hizo un análisis PCA de los estados para comprobar la cobertura del espacio y la distribución de acciones. También se normalizaron las observaciones automáticamente con *RunningMeanStd*, y así se generaron unos conjuntos de datos reproducibles y fáciles de auditar, que nos sirvieron como base para los experimentos de *Imitation Learning*.

4.2. DISEÑO EXPERIMENTAL

Inicialmente, se decidió comparar todos los algoritmos con 100 trayectorias expertas, pero revisando la literatura, se vio que la mayoría de artículos usan entre 50 y 5 [1]. Con 100 trayectorias se cubre gran parte de la distribución de estados y la

reducción progresiva de trayectorias (100, 50, 20, 10, 5) permite medir la robustez frente a la escasez de datos, comparar la eficiencia muestral y ver si más trayectorias siempre es mejor o no.

También, se comenzó con la idea de compararlos con 1M de pasos, pero como se verá en resultados, los métodos adversariales necesitan más pasos para converger. Se escogió la medida de 2M de pasos porque la mayoría de los métodos alcanzan cerca del 80% del rendimiento del experto y el que menos ≈60%. También esta decisión se debe a que se ha ejecutado todo el entrenamiento en un ordenador con una RTX 3070, y si quería hacer muchas pruebas, no podía subir el número de pasos ya que con 2M algún método ya tardaba 3h en una sola ejecución y había que hacer como poco una ejecución por combinación de trayectorias, eso sin contar la búsqueda de hiperparámetros.

4.3. IMPLEMENTACIÓN DE ALGORITMOS

En la fase de experimentación se emplearon, cuando existían, implementaciones consolidadas de la comunidad. Los algoritmos de **BC**, **GAIL** y **AIRL** se importaron directamente de la librería ***imitation***, que es una librería *open-source* que implementa algoritmos de *Imitation Learning* en *backends como PyTorch* y *StableBaselines3* [39]. Aunque el repositorio incluye una implementación de **SQIL**, no es compatible con entornos continuos, por lo que no servía para HalfCheetah-v4. Se decidió implementarlo desde cero, la literatura para dominios continuos era escasa y su conocida inestabilidad obligaron a ajustar cuidadosamente cada hiperparámetro. El cambio de paradigma entre acciones discretas y continuas fue el reto principal.

BCO y GAIfO, al ser algoritmos novedosos en el campo del *Imitation from Observation* no tenían ninguna implementación pública en librerías conocidas. Para **BCO**, se partió de una implementación no oficial en GitHub y del artículo original, centrándose en programar el bucle iterativo que reentrena tanto el modelo inverso como la política [40], [38]. Para **GAIfO**, el *paper* original especifica cómo implementarlo para entornos de baja dimensionalidad [21], pero no para entornos más complejos como *HalfCheetah-v4*, pero al seguir una lógica parecida a GAIL, solo que sin tener acceso a las acciones, se puedo adaptar esa lógica y escalarla con éxito a HalfCheetah-v4.

4.4. OPTIMIZACIÓN DE HIPERPARÁMETROS

Para ajustar los métodos implementados de cero (**BCO, GAIfO y SQIL**) se empleó **Optuna**, un *framework* de optimización bayesiana (TPE) que explora secuencialmente el espacio de búsqueda y descarta de forma temprana ensayos con peor proyección [3]. En **BCO** se exploraron el número de interacciones previas (50k-300k), el factor α que pondera iteraciones adicionales (0-0.5), el número de ciclos de refinado y las épocas de entrenamiento de la política y del modelo de dinámica inversa. Se fijó un máximo de 50 pruebas y se descartaba cualquier configuración que, según la estimación previa, superase los 2M de pasos totales. Aunque a la hora de la práctica, para poder compararlos en igualdad de condiciones todos los algoritmos, se fijó α a 0, y solo se investigó el *learning rate* y el número de interacciones previas.

Para **GAIfO** se optimizaron la tasa de aprendizaje del discriminador (10^{-5}–10^{-3}, escala log), el tamaño de lote (256–1024), el número de épocas del discriminador y el coeficiente de *gradient penalty* λ_{GP} (0.3–10), con 2M de pasos por ensayo.

Para **SQIL** se buscaron tasas de aprendizaje de actor, crítico y α-entropía (10^{-6}–10^{-3}), el *batch size* (1024 o 2048) y la frecuencia de actualización. Cada estudio tuvo 50 pruebas. En los tres casos la métrica objetivo fue el retorno medio en 20 episodios de evaluación, Optuna maximizó ese valor.

Los algoritmos ya disponibles en la librería *imitation* (**BC**, **GAIL** y **AIRL**) no se reoptimizaron exhaustivamente. Se usaron los hiperparámetros que se recomendaban en la literatura y se verificó que su rendimiento alcanzaba las cifras que decían, introduciendo ajustes menores como el *batch size* 128 en AIRL o probar manualmente a cambiar los

tamaños de las capas y quedarnos con el que mejor resultado se obtenga.

4.5. HIPERPARÁMETROS Y ARQUITECTURA

Las 3 tablas siguientes recogen, la arquitectura de redes y los principales hiperparámetros empleados en cada algoritmo. La versión íntegra de la configuración, está incluida en el repositorio como un archivo yaml permite replicar los experimentos al detalle: aquí sólo se muestran los ajustes de mayor impacto. Todas las ejecuciones se realizaron con seed=44 y 2M de pasos de interacción con el entorno. Las excepciones son los métodos offline como BC, SQIL que usan transiciones al no interactuar con el entorno. BCO son 300k interacciones previas (steps) y 1.7M transiciones.

Alg.	Back-end	*Actor*	Crítico	Activ.
BC	— (BC)	64-64	—	Tanh
BCO	— (BC)	64-64	—	ReLU
GAIL	TRPO	64-64	64-64	Tanh
GAIfO	TRPO	64-64	64-64	Tanh
AIRL	TRPO	64-64	64-64	Tanh
SQIL	SAC-like	256-256	2×(256-256)	ReLU

Tabla 1. Back-end RL & red Actor/Crítico

La Tabla 1. Back-end RL & red Actor/Crítico resume el núcleo del aprendizaje por refuerzo y la topología actor-crítico. BC/BCO no usan RL sino aprendizaje supervisado, por ello ambos conservan la arquitectura mínima de *Stable-Baselines3* (dos capas de 64 neuronas). GAIL, GAIfO y AIRL se entrenan con TRPO por coherencia con el *paper* original que busca la estabilidad [15], [17], [21]. SQIL se apoya en un esquema parecido a SAC, y siguiendo lo habitual de este algoritmo, usa redes más anchas (256-256) y un doble crítico (*Double-Q)* para mitigar la sobreestimación [8].

El uso de *tanh* como activación facilita comprimir la acción a -1,1 sin capas adicionales, pero se usa ReLU con BCO y SQIL porque ofrece gradientes más fuertes para la política extraída de las observaciones o al optimizar con SAC.

Alg.	Tipo	Capas	Activ.	Especiales
BC	—	—	—	—
BCO	InvDyn	256-256	ReLU + LN	Entrada ⟨s, s′⟩
GAIL	Disc.	256-256-128	ReLU	RunningNorm
GAIfO	State-only Disc.	256-256	ReLU	$\lambda_{GP} = 5.17$ y Entrada ⟨s, s′⟩
AIRL	RewardNet	128-128	ReLU	+Potential Φ 128-128-64
SQIL	—	—	—	Rew. +1/0

Tabla 2. Redes auxiliares

La Tabla 2. Redes auxiliares recoge las redes que respaldan la política principal. BC y SQIL no las usan: BC clona las acciones directamente y SQIL asigna recompensas fijas +1/0. BCO añade un modelo de dinámica inversa (MLP 256-256 con ReLU y *LayerNorm*) que, a partir de los estados consecutivos (s, s') infiere la acción correspondiente.

En GAIL, un discriminador de tres capas (256-256-128) con ReLU normalizado mediante RunningNorm diferencia trayectorias expertas y generadas. GAIfO usa un discriminador *state-only* de dos capas (256-256) con ReLU y penalización de gradiente λ_{GP} para estabilizar el entrenamiento adversarial. AIRL incorpora un Reward Net con dos partes: una recompensa densa $r_\psi(s, a)$ (128-128) y

un potencial $\Phi_\psi(s)$ (128-128-64) que juntos permiten *reward shaping*. Estas redes suelen ser más profundas que la política porque necesitan separar distribuciones.

Alg.	Batch π / Disc.	*LR π /Critic/ Disc.*	Buffer(s)	Updates / round
BC	64	1e-3	—	—
BCO	64	1e-3 / —	demo β =100 k	—
GAIL	2048 /2048	TRPO def. / 3e-4	2k	16
GAIfO	2048 / 512	TRPO def. / 5.5e-5	2k	16
AIRL	128 / 2048	1e-3 / 3e-4	1k	24
SQIL	256	3e-4 / 3e-4 y lr_α=1e-4	100k/100k demo/agent	every 10 steps

Tabla 3. Hiperparámetros de entrenamiento

La Tabla 3. Hiperparámetros de entrenamiento, muestra como el *batch size*, *learning rate* y la gestión de memoria se adaptan a la lógica de cada algoritmo. BC y BCO, al hacer clonación supervisada, usan lotes pequeños, mientras que los adversariales necesitan 2048 muestras para estimar con baja varianza la divergencia entre experto y agente. SQIL usa 256 porque sigue la práctica de SAC. Los *learning rate* también reflejan este equilibrio: el del discriminador es más bajo que el de la política para evitar que domine la dinámica, y SQIL añade un lr_α que ajusta de forma automática la entropía. En los *replay buffers* adversariales, el tamaño equilibra varianza y diversidad. SQIL usa dos memorias separadas, una para demostraciones y otra para experiencias propias, para mantener la proporción adecuada. El número de actualizaciones se ajusta para evitar sobreajuste; en AIRL se aumenta para compensar el *reward shaping* complejo.

Estos parámetros siguen las configuraciones originales y aseguran comparabilidad y replicabilidad.

4.6. VISUALIZACIÓN

Para la Figura 4 se ha empleado un *smoothing* de 0.6 para todos los algoritmos menos SQIL (0.9) debido a su gran inestabilidad para hacerlo más fácil de seguir.

5. RESULTADOS

Para la validación de modelos en CartPole-v1, todos los modelos llegaron a los 500 puntos de recompensa promedio, y en la Tabla 4 se ve cuanto tardó cada uno en hacerlo y estabilizarse ahí.

Alg	BC	*BCO*	GAIL	GAIfO	AIRL	SQIL
Steps	32k	36k*	230k	26k	200k	15k**

Tabla 4. Steps hasta estabilizar recompensa max CartPole-v1

* En BCO fueron 20k interacciones previas y 16k transiciones
** SQIL lo alcanza en 15k y se mantiene hasta los 70k en 500, pero ahí decae hasta casi 0 rápidamente y se mantiene hasta el final.

BC y GAIfO convergen rápido (menos de 35k steps) porque la tarea, al ser lineal y de baja dimensión, se ajusta bien a sus redes ligeras y la señal de error es densa desde el principio. BCO necesita algo más de tiempo al dedicar las 20k primeras transiciones a inferir acciones, pero una vez el modelo inverso está calibrado, la política imitada converge con eficacia. GAIL y AIRL tardan más porque sus discriminadores requieren muchas iteraciones adversariales para estabilizar la estimación de recompensa en un entorno tan limitado. SQIL aprende muy rápido debido a su recompensa binaria +1/0, pero esa misma señal, al ser tan escasa provoca el sobreajuste. Cuando el buffer del agente domina sobre las muestras expertas, la política se aleja de la región recompensada y la puntuación cae abruptamente.

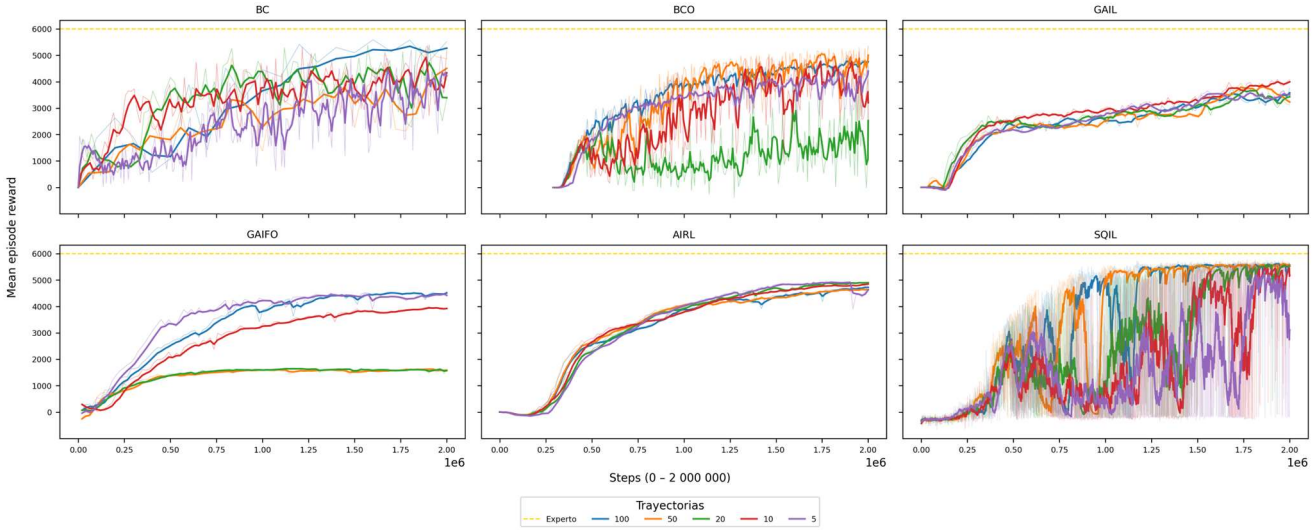

Figura 4. Entrenamiento Algoritmos

Una vez validados los modelos se pasó al entrenamiento de los algoritmos variando trayectorias para HalfCheetah-v4.

La Figura 4 refleja la evolución de la recompensa promedio durante el proceso de entrenamiento de los algoritmos por 2M de pasos en HalfCheetah-v4.

En BC el agente aprende sólo por regresión supervisada, imita la acción que ve en cada estado [4]. Con pocas trayectorias la red recibe pocos ejemplos de ciertas fases del movimiento y tiene alta varianza, de ahí las oscilaciones y un techo de rendimiento cercano a 3k. Al añadir demostraciones se aumenta la cobertura y densidad del espacio de estados y acciones. El gradiente deja de cambiar de signo tan bruscamente, la pérdida converge con menos rebotes y la política generaliza mejor. Por eso con 50-100 trayectorias la curva es más lisa, sube hasta el rango 4.5k-5k y se mantiene estable.

En BCO la curva empieza en el paso 300k porque el agente primero acumula 300k interacciones previas aleatorias para entrenar el modelo de dinámica inversa. Aunque no vea las acciones expertas, este modelo consigue inferirlas con precisión y, tras es ese arranque, la política alcanza recompensas muy cercanas a las de BC, mostrando que la inferencia está bien hecha. El caso raro es con 20 trayectorias, ya que el rendimiento es bastante peor que con las demás. Probablemente ese lote justo no cubre bien estados clave y el modelo sobreajusta transiciones frecuentes y deja huecos sin recompensa, empeorando la política final.

En la gráfica podemos ver como en *HalfCheetah-v4,* GAIL y AIRL casi no ganan rendimiento al pasar de 5 a 100 trayectorias porque con pocas demostraciones ya alcanzan saturación de información.

Primero, 5-10 episodios cubren casi todo el ciclo de zancada, por lo que las trayectorias adicionales son muy redundantes. Segundo, ambos algoritmos remuestrean sin parar esos datos: el discriminador de GAIL y la red de recompensa + potencial de AIRL actualizan miles de veces con el mismo *buffer* experto, por lo que una muestra pequeña vale para converger [15], [17]. Tercero, sus back-ends son conservadores: TRPO limita el tamaño del paso de la política, una vez la señal de recompensa es coherente, el avance está regido por esa restricción y no porque falten datos [35]. Por último, la recompensa densa que generan ambos generaliza muy bien el espacio continuo, mientras la normalización y el *shaping* penalizan estados fuera de lo enseñado por el experto, y así estabilizan la política. Así que, si tus muestras son mínimamente diversas, dar aún más muestras no da información al algoritmo relevante y el rendimiento es prácticamente el mismo.

GAIfO sólo observa secuencias de estados, no las acciones. Su discriminador tiene que inferir la dinámica implícita a partir de la coherencia temporal de las demostraciones [21]. Con 5 y 10 trayectorias el lote es pequeño pero muy consistente, todos los episodios provienen de la misma política experta y describen curvas parecidas, de modo que el discriminador aprende una señal clara y el agente llega a superar el rendimiento de GAIL. Con 100 trayectorias hay una mayor diversidad que es compensada estadísticamente y hace que la señal vuelva a ser fiable.

En cambio, con 20 y 50 trayectorias pasa algo que está entre medias de los dos casos y es indeseable, el conjunto tiene suficiente variación para confundir al discriminador, pero no tanta como para que esa variación se promedie. El *grad penalty* y el lote pequeño usado en cada actualización refuerza la subrepresentación de posiciones raras, produciendo recompensas ruidosas que hacen que la política se bloquee cerca de 1k. Por eso hay tal bajón en esas dos.

SQIL premia con +1 cada transición que coincide con la demostración y 0 el resto. Esa recompensa binaria es extremadamente *sparse* y provoca gradientes "todo o nada": con que la política se desvíe un poco la señal se desploma y el crítico SAC oscila al intentar corregirlo [8]. Además, el *replay buffer* se mantiene en una mezcla fija 50% experto y 50% agente. Con solo 5-20 demostraciones ese bloque experto es pequeño, las mismas transiciones se sobre-muestrean miles de veces, el agente se sobre ajusta, luego explora zonas sin recompensa y vuelve a caer en la fase +1/0, generando ondulaciones hasta el final de los 2M.

Con 50 o 100 trayectorias la parte +1 cubre mucho mejor el espacio de estados. El crítico recibe objetivos más variados y la política ya no rebota entre zonas recompensadas y neutras. Alrededor de 1M de pasos entra en régimen casi estacionario, acercándose al rendimiento experto, aunque sigue manteniendo cierta variabilidad inherente a esta señal binaria.

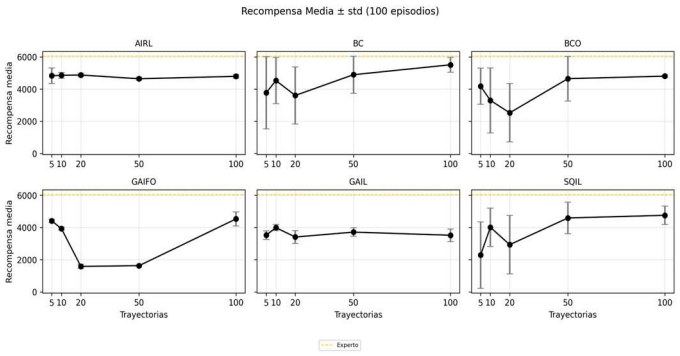

Figura 5. Recompensa media +- std variando trayectorias

La Figura 5 muestra la recompensa media alcanzada por cada algoritmo según su número de trayectorias y, además, cuanto varía su política en 100 episodios de evaluación, eso nos revela disparidades marcadas en la estabilidad y consistencia de los algoritmos. **AIRL es el más estable** con sus coeficientes de variación cercanos al 2% (excepto con 5 trayectorias que es un 10%) y el que mejores resultados consigue en promedio. Esta estabilidad refleja que el mecanismo de aprendizaje usando recompensas adversariales es muy robusto y generaliza bien con pocas demostraciones. A GAIL le pasa más o menos lo mismo, es muy estable y consigue recompensas parecidas con cada número de trayectorias, pero son peores que las de AIRL.

BC es el que presenta el patrón más claro de mejora con datos adicionales, escalando desde 3772 ± 2241 puntos con 5 trayectorias hasta 5509 ± 462 puntos con 100 trayectorias. La reducción tan grande de la desviación estándar indica que más datos no solo mejoran el rendimiento promedio, también hacen más estable al algoritmo y predecible. Esto es lógico teniendo en cuenta la naturaleza supervisada de BC, en la que más ejemplos dan mejor rendimiento.

BCO muestra una variabilidad extrema, sobre todo con configuraciones intermedias. Con 20 trayectorias, tiene una *std* de 1811 puntos, que es el coeficiente de variación más alto del experimento (71.7%). Esta inestabilidad sugiere que el modelo inverso es muy sensible a la distribución específica de estados en las demostraciones, y que si es

subóptimo puede llevar a fallos catastróficos en la inferencia.

SQIL es el que más dependencia de datos muestra, con una reducción del coeficiente de variación desde 90% con 5 trayectorias hasta el 12.1% con 100 trayectorias. La cobertura insuficiente del espacio de estados con pocas trayectorias hace que se generen grandes gradientes muy esporádicos e inestabilidad.

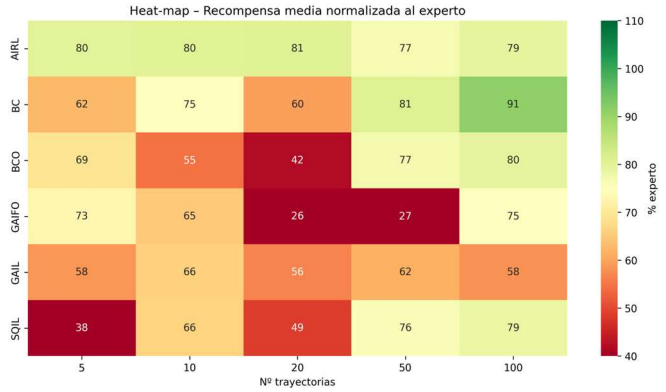

Figura 6. Recompensa media normalizada

El *heatmap* de la Figura 6 refleja la recompensa media normalizada con respecto a la obtenida por el experto y nos revela diferencias claras en la eficiencia de los algoritmos. **BC con 100 trayectorias alcanza el 91.2% del rendimiento experto**, así establece el benchmark más alto entre todos los modelos que se han evaluado. Este resultado es muy notable ya que BC es muy simple conceptualmente, sugiriendo que, para entornos con muchas demostraciones, imitar directamente puede superar a métodos más sofisticados.

AIRL mantiene consistentemente ≈ 80% del rendimiento experto, variando muy poco entre configuraciones. Esta consistencia es algo que lo diferencia mucho de los otros métodos, que son altamente sensibles, haciendo que sea la opción más confiable cuando buscamos estabilidad.

GAIfO presenta la mayor disparidad interna, cae al 27% con 25-50 trayectorias y se recupera sólo cuando el set es muy pequeño (coherencia) o muy grande (promedio estadístico). Esta variabilidad indica que el algoritmo es frágil ante cambios en la cantidad de datos.

GAIL refuerza la idea de saturación temprana de información, ya que mejora poco al añadir datos (de un 60% a un 66%).

BCO se acerca a BC con 100 demos, demostrando que su modelo inverso compensa el no tener acceso a acciones, pero sufre mucho con 20. Parece que tiene un patrón bimodal, ya que varía bastante el resultado obtenido según las trayectorias.

SQIL pasa de 38% a 79% al ampliar muestras, eso indica que la recompensa binaria necesita gran cobertura del espacio de estados para no provocar oscilaciones y entrar en régimen permanente.

Se puede observar que, con **20 trayectorias**, todos los modelos menos AIRL empeoran su rendimiento. Esto quiere decir que con 20 demostraciones el conjunto es demasiado grande como para ser totalmente homogéneo, pero demasiado pequeño como para cubrir bien el espacio de estados, es decir, introduce variabilidad sin contrapeso estadístico. Esa "zona gris" rompe la coherencia interna que necesitan los métodos adversariales para poder identificar la distribución experta y, a la vez, deja huecos que perjudican a los métodos basados solo en supervisión. El resultado es una señal de recompensa con más ruido, aprendizaje con errores y un peor rendimiento promedio.

Los experimentos exponen un compromiso claro entre **la estabilidad y el techo de rendimiento**. AIRL es el más predecible, pero no se acerca tanto al experto como BC (80% contra 90%), aunque BC tiene mayor varianza. GAIfO y BCO muestran fragilidad, con 5 trayectorias GAIfO alcanza el 73% pero con 20 cae al 26%, señal de sobreajuste y sensibilidad a la diversidad de las demostraciones. En escenarios con muchas demostraciones la simplicidad de BC supera a métodos adversariales más complejos, cuestionando si de verdad se necesitan arquitecturas tan sofisticadas. La elección de algoritmo, por tanto, tiene que guiarse por la disponibilidad de demostraciones y la tolerancia a la inestabilidad del dominio objetivo.

6. CONCLUSIONES Y TRABAJOS FUTUROS

Este trabajo ha abordado el estudio comparativo de algoritmos de Imitation Learning (IL), pasando primero por una revisión del estado del arte de los algoritmos, entornos y métricas de evaluación, pero sobre todo de los algoritmos, y se ha centrado en su eficiencia muestral y capacidad de generalización entre entornos de complejidad diferente. La implementación y evaluación de los seis algoritmos escogidos por ser los más representativos de las principales familias metodológicas (BC, GAIL, AIRL, SQIL, BCO y GAIfO), de los cuales los tres últimos han sido implementados desde cero, ha permitido poder compararlos rigurosamente y descubrir patrones significativos en su comportamiento.

Los resultados obtenidos son evidencia de que no existe un algoritmo universalmente superior, sino que cada método tiene una serie de fortalezas específicas según el contexto en el que se aplique. BC ha demostrado que depende claramente del volumen de datos, alcanzando un sorprendente 91.2% del rendimiento experto con 100 trayectorias en HalfCheetah-v4, lo que va en contra de la creencia de que los métodos más complejos siempre son mejores que los más simples. Por otro lado, AIRL ha destacado por ser muy consistente. Ha mantenido aproximadamente un 80% del rendimiento experto independientemente del número de trayectorias del que disponga, lo que hace que sea la opción más robusta cuando los datos que se tienen son variables o limitados.

Un hallazgo particularmente relevante es el haber identificado una "zona crítica" alrededor de las 20 trayectorias, donde todos los algoritmos menos AIRL empeoran su rendimiento. Este suceso nos muestra un umbral crítico donde el conjunto de demostraciones es demasiado heterogéneo como para poder mantener la coherencia interna, pero insuficiente para poder cubrir de forma adecuada el espacio de estados.

Los métodos basados solo en observaciones (BCO y GAIfO) han mostrado que son muy sensibles a la calidad y distribución de las demostraciones, con patrones bimodales que sugieren que hay una interacción compleja entre la cantidad de demostraciones y la capacidad que tienen para inferir acciones correctamente. SQIL, ha sido la que más ha mejorado al aumentar el número de demostraciones, pasando de un 38% a un 79%. Esto confirma que su señal binaria necesita que el espacio de estados sea cubierto de forma más amplia para poder estabilizarse.

El estudio ha cumplido de manera satisfactoria sus dos objetivos iniciales: 1) analizar los algoritmos y su eficiencia muestral para poder determinar cuántas demostraciones necesita cada uno para tener un rendimiento aceptable y, 2) evaluar su capacidad para generalizar, comparando cómo se comportan en entornos de distinta complejidad (discretos y continuos) bajo unas condiciones controladas.

Como **trabajos futuros**, gracias a este estudio se abren varias líneas de investigación prometedoras. Primero, explorar métodos híbridos que unan AIRL por su estabilidad y BC por su rendimiento y sencillez, especialmente para aplicaciones donde se busque tanto la robustez como un gran rendimiento. La "zona crítica" identificada a las 20 trayectorias merece un análisis en mayor profundidad, en el que se investiguen técnicas de aumento de datos o regularización específicas que puedan mitigar esta degradación del rendimiento que ocurre. Además, extender el análisis a entornos con distribuciones que cambien haría que se pueda evaluar la robustez de estos algoritmos ante perturbaciones o cambios en la dinámica del sistema, que es algo que no se ha hecho en este trabajo. Agregar métricas cualitativas y de seguridad puede ser valioso, ya que según la literatura no se usan casi a la hora de evaluar algoritmos de IL. Finalmente, una extensión de este trabajo valiosa y natural sería abordar el problema de la brecha existente entre simulación y realidad, y así comprobar si los patrones que han sido observados en este trabajo se mantienen si se transfieren estas políticas obtenidas en simulación al mundo real con sistemas físicos.

7. BIBLIOGRAFÍA

[1] M. Zare, P. M. Kebria, A. Khosravi, y S. Nahavandi, «A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges», *IEEE Trans. Cybern.*, vol. 54, n.º 12, pp. 7173-7186, dic. 2024, doi: 10.1109/TCYB.2024.3395626.

[2] N. Gavenski, F. Meneguzzi, M. Luck, y O. Rodrigues, «A Survey of Imitation Learning Methods, Environments and Metrics», 30 de julio de 2024, *arXiv*: arXiv:2404.19456. doi: 10.48550/arXiv.2404.19456.

[3] T. Akiba, S. Sano, T. Yanase, T. Ohta, y M. Koyama, «Optuna: A Next-generation Hyperparameter Optimization Framework», en *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Anchorage AK USA: ACM, jul. 2019, pp. 2623-2631. doi: 10.1145/3292500.3330701.

[4] C.-C. Chuang, D. Yang, C. Wen, y Y. Gao, «Resolving Copycat Problems in Visual Imitation Learning via Residual Action Prediction», en *Computer Vision – ECCV 2022*, vol. 13699. doi: 10.1007/978-3-031-19842-7_23.

[5] S. Ross, G. Gordon, y D. Bagnell, «A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning», en *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, JMLR Workshop and Conference Proceedings, jun. 2011, pp. 627-635. [En línea]. Disponible en: https://proceedings.mlr.press/v15/ross11a.html

[6] J. Zhang y K. Cho, «Query-Efficient Imitation Learning for End-to-End Simulated Driving», *Proc. AAAI Conf. Artif. Intell.*, vol. 31, n.º 1, Art. n.º 1, feb. 2017, doi: 10.1609/aaai.v31i1.10857.

[7] R. Hoque *et al.*, «LazyDAgger: Reducing Context Switching in Interactive Imitation Learning», en *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, ago. 2021, pp. 502-509. doi: 10.1109/CASE49439.2021.9551469.

[8] S. Reddy, A. D. Dragan, y S. Levine, «SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards», 25 de septiembre de 2019, *arXiv*: arXiv:1905.11108. doi: 10.48550/arXiv.1905.11108.

[9] M. Kalakrishnan, P. Pastor, L. Righetti, y S. Schaal, «Learning objective functions for manipulation», en *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany: IEEE, may 2013, pp. 1331-1336. doi: 10.1109/ICRA.2013.6630743.

[10] N. Ratliff, D. Bradley, J. A. Bagnell, y J. Chestnutt, «Boosting Structured Prediction for Imitation Learning», en *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, y T. Hofmann, Eds., The MIT Press, 2007, pp. 1153-1160. doi: 10.7551/mitpress/7503.003.0149.

[11] A. J. Snoswell, S. P. N. Singh, y N. Ye, «Revisiting Maximum Entropy Inverse Reinforcement Learning: New Perspectives and Algorithms», en *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, dic. 2020, pp. 241-249. doi: 10.1109/SSCI47803.2020.9308391.

[12] C. Finn, S. Levine, y P. Abbeel, «Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization», en *Proceedings of The 33rd International Conference on Machine Learning*, PMLR, jun. 2016, pp. 49-58. [En línea]. Disponible en: https://proceedings.mlr.press/v48/finn16.html

[13] D. S. Brown, R. Coleman, R. Srinivasan, y S. Niekum, «Safe Imitation Learning via Fast Bayesian Reward Inference from Preferences».

[14] A. J. Chan y M. van der Schaar, «Scalable Bayesian Inverse Reinforcement Learning», 11 de marzo de 2021, *arXiv*: arXiv:2102.06483. doi: 10.48550/arXiv.2102.06483.

[15] J. Ho y S. Ermon, «Generative Adversarial Imitation Learning», en *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2016. [En línea]. Disponible en: https://proceedings.neurips.cc/paper_files/paper/2016/file/cc7e2b878868cbae992d1fb743995d8f-Paper.pdf

[16] Y. Li, J. Song, y S. Ermon, «InfoGAIL: Interpretable Imitation Learning from Visual Demonstrations», en *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. [En línea]. Disponible en: https://proceedings.neurips.cc/paper_files/paper/2017/file/2cd4e8a2ce081c3d7c32c3cde4312ef7-Paper.pdf

[17] J. Fu, K. Luo, y S. Levine, «Learning Robust Rewards with Adversarial Inverse Reinforcement Learning», 13 de agosto de 2018, *arXiv*: arXiv:1710.11248. doi: 10.48550/arXiv.1710.11248.

[18] I. Kostrikov, K. K. Agrawal, D. Dwibedi, S. Levine, y J. Tompson, «Discriminator-Actor-Critic: Addressing Sample Inefficiency and Reward Bias in Adversarial Imitation Learning», 15 de octubre de 2018, *arXiv*: arXiv:1809.02925. doi: 10.48550/arXiv.1809.02925.

[19] R. Dadashi, L. Hussenot, M. Geist, y O. Pietquin, «Primal Wasserstein Imitation Learning», 17 de marzo de 2021, *arXiv*: arXiv:2006.04678. doi: 10.48550/arXiv.2006.04678.

[20] F. Torabi, G. Warnell, y P. Stone, «Behavioral Cloning from Observation», en *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Stockholm, Sweden: International Joint Conferences on Artificial Intelligence Organization, jul. 2018, pp. 4950-4957. doi: 10.24963/ijcai.2018/687.

[21] F. Torabi, G. Warnell, y P. Stone, «Generative Adversarial Imitation from Observation», 18 de junio de 2019, *arXiv*: arXiv:1807.06158. doi: 10.48550/arXiv.1807.06158.

[22] P. Sermanet *et al.*, «Time-Contrastive Networks: Self-Supervised Learning from Video», en *2018 IEEE International Conference on Robotics and Automation (ICRA)*, may 2018, pp. 1134-1141. doi: 10.1109/ICRA.2018.8462891.

[23] «Gymnasium Documentation». Accedido: 5 de junio de 2025. [En línea]. Disponible en: https://gymnasium.farama.org/environments/classic_control/cart_pole.html

[24] «Gymnasium Documentation». Accedido: 5 de junio de 2025. [En línea]. Disponible en: https://gymnasium.farama.org/environments/mujoco/half_cheetah.html

[25] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, y J. Peters, «An Algorithmic Perspective on Imitation Learning», *Found. Trends® Robot.*, vol. 7, n.º 1-2, pp. 1-179, 2018, doi: 10.1561/2300000053.

[26] Z. Liu *et al.*, «Dynamics Adapted Imitation Learning».

[27] T. Luo, T. Pearce, H. Chen, J. Chen, y J. Zhu, «C-GAIL: Stabilizing Generative Adversarial Imitation Learning with Control Theory», *Adv. Neural Inf. Process. Syst.*, vol. 37, pp. 29464-29488, dic. 2024.

[28] C.-M. Lai, H.-C. Wang, P.-C. Hsieh, Y.-C. F. Wang, M.-H. Chen, y S.-H. Sun, «Diffusion-Reward Adversarial Imitation Learning», en *Advances in Neural Information Processing Systems*, A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, y C. Zhang, Eds., Curran Associates, Inc., 2024, pp. 95456-95487. [En línea]. Disponible en: https://proceedings.neurips.cc/paper_files/paper/2024/file/ad47b1801557e4be37d30baf623de426-Paper-Conference.pdf

[29] P. Wang, D. Liu, J. Chen, H. Li, y C.-Y. Chan, «Decision Making for Autonomous Driving via Augmented Adversarial Inverse Reinforcement Learning», en *2021 IEEE International Conference on Robotics and Automation (ICRA)*, may 2021, pp. 1036-1042. doi: 10.1109/ICRA48506.2021.9560907.

[30] S. Y. Arnob, «Off-Policy Adversarial Inverse Reinforcement Learning», 3 de mayo de 2020, *arXiv*: arXiv:2005.01138. doi: 10.48550/arXiv.2005.01138.

[31] D. Garg, S. Chakraborty, C. Cundy, J. Song, y S. Ermon, «IQ-Learn: Inverse soft-Q Learning for Imitation», en *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2021, pp. 4028-4039. [En línea]. Disponible en: https://proceedings.neurips.cc/paper/2021/hash/210f760a89db30aa72ca258a3483cc7f-Abstract.html

[32] R. Furuyama, D. Kuyoshi, y S. Yamane, «Extrinsicaly Rewarded Soft Q Imitation Learning with Discriminator», 30 de enero de 2024, *arXiv*: arXiv:2401.16772. doi: 10.48550/arXiv.2401.16772.

[33] A. Jaegle, Y. Sulsky, A. Ahuja, J. Bruce, R. Fergus, y G. Wayne, «Imitation by Predicting Observations», en *Proceedings of the 38th International Conference on Machine Learning*, PMLR, jul. 2021, pp. 4665-4676. [En línea]. Disponible en: https://proceedings.mlr.press/v139/jaegle21b.html

[34] F. Torabi, G. Warnell, y P. Stone, «Adversarial Imitation Learning from State-only Demonstrations», en

Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, en AAMAS '19. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, may 2019, pp. 2229-2231.

[35] J. Schulman, S. Levine, P. Abbeel, M. Jordan, y P. Moritz, «Trust Region Policy Optimization», en *Proceedings of the 32nd International Conference on Machine Learning*, PMLR, jun. 2015, pp. 1889-1897. [En línea]. Disponible en: https://proceedings.mlr.press/v37/schulman15.html

[36] D. Jung, H. Lee, y S. Yoon, «Sample-efficient adversarial imitation learning», *J Mach Learn Res*, vol. 25, n.º 1, p. 31:1545-31:1576, ene. 2024.

[37] S. Desai, I. Durugkar, H. Karnan, G. Warnell, J. Hanna, y P. Stone, «An Imitation from Observation Approach to Transfer Learning with Dynamics Mismatch», en *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, y H. Lin, Eds., Curran Associates, Inc., 2020, pp. 3917-3929. [En línea]. Disponible en: https://proceedings.neurips.cc/paper_files/paper/2020/file/28f248e9279ac845995c4e9f8af35c2b-Paper.pdf

[38] F. Torabi, G. Warnell, y P. Stone, «Imitation Learning from Video by Leveraging Proprioception», en *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, International Joint Conferences on Artificial Intelligence Organization, jul. 2019, pp. 3585-3591. doi: 10.24963/ijcai.2019/497.

[39] «HumanCompatibleAI/imitation: Clean PyTorch implementations of imitation and reward learning algorithms». Accedido: 3 de junio de 2025. [En línea]. Disponible en: https://github.com/HumanCompatibleAI/imitation/tree/master

[40] «tsujuifu/pytorch_bco: A PyTorch implementation of BCO». Accedido: 3 de junio de 2025. [En línea]. Disponible en: https://github.com/tsujuifu/pytorch_bco

ANEXO I

Enlace al repositorio de GitHub con todo el código usado para el proyecto: https://github.com/AntonioLDM17/Imitation-Learning-Analysis-and-Comparison.git

La Tabla 5 contiene todas las métricas relevantes para la comparación de algoritmos.

alg	*Tray*	*env*	*media*	*std*	*cv*	*%expert*
AIRL	5	HalfCheetah-v4	4829,98	487,64	0,1010	79,93
AIRL	10	HalfCheetah-v4	4856,45	184,78	0,0380	80,37
AIRL	20	HalfCheetah-v4	4874,92	108,96	0,0224	80,68
AIRL	50	HalfCheetah-v4	4641,42	90,73	0,0195	76,81
AIRL	100	HalfCheetah-v4	4792,12	114,72	0,0239	79,31
BC	5	HalfCheetah-v4	3772,47	2240,65	0,5939	62,43
BC	10	HalfCheetah-v4	4531,91	1434,74	0,3166	75,00
BC	20	HalfCheetah-v4	3605,62	1774,76	0,4922	59,67
BC	50	HalfCheetah-v4	4893,42	1156,24	0,2363	80,98
BC	100	HalfCheetah-v4	5508,99	462,39	0,0839	91,17
BCO	5	HalfCheetah-v4	4176,61	1118,61	0,2678	69,12
BCO	10	HalfCheetah-v4	3298,21	2016,71	0,6115	54,58
BCO	20	HalfCheetah-v4	2527,41	1811,27	0,7167	41,83
BCO	50	HalfCheetah-v4	4648,00	1386,71	0,2983	76,92
BCO	100	HalfCheetah-v4	4803,88	66,04	0,0137	79,50

GAIFO	5	HalfCheetah -v4	4412,21	96,38	0,0218	73,02
GAIFO	10	HalfCheetah -v4	3928,71	96,26	0,0245	65,02
GAIFO	20	HalfCheetah -v4	1588,94	127,79	0,0804	26,30
GAIFO	50	HalfCheetah -v4	1631,15	33,38	0,0205	26,99
GAIFO	100	HalfCheetah -v4	4533,58	441,07	0,0973	75,03
GAIL	5	HalfCheetah -v4	3530,46	269,83	0,0764	58,43
GAIL	10	HalfCheetah -v4	3993,42	188,94	0,0473	66,09
GAIL	20	HalfCheetah -v4	3411,02	393,14	0,1153	56,45
GAIL	50	HalfCheetah -v4	3716,53	262,52	0,0706	61,51
GAIL	100	HalfCheetah -v4	3522,43	391,95	0,1113	58,29
SQIL	5	HalfCheetah -v4	2292,48	2063,15	0,9000	37,94
SQIL	10	HalfCheetah -v4	4015,44	1191,32	0,2967	66,45
SQIL	20	HalfCheetah -v4	2936,26	1812,92	0,6174	48,59
SQIL	50	HalfCheetah -v4	4594,82	981,17	0,2135	76,04
SQIL	100	HalfCheetah -v4	4760,05	578,09	0,1214	78,78
EXPERT (SAC)	–	HalfCheetah -v4	6042,55	47,74	0,0079	100

Tabla 5. Métricas de todos los algoritmos.