Grado en Ingeniería en Tecnologías de Telecomunicación

Trabajo fin de grado

APRENDIZAJE POR REFUERZO E IA GENERATIVA PARA LA GESTIÓN DE LA DEMANDA ENERGÉTICA

Autor
Álvaro González Tabernero

Dirigido por
Francisco Martín Martínez
Jaime Boal Martín-Larrauri

Madrid
Junio 2025

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

APRENDIZAJE POR REFUERZO E IA GENERATIVA PARA LA GESTIÓN DE LA DEMANDA ENERGÉTICA

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.: Álvaro González Tabernero Fecha: 16/06/2025

Autorizada la entrega del proyecto

LOS DIRECTORES DEL PROYECTO

Fecha: 16/06/2025

Fdo.: Francisco Martín Martínez

MARTIN MARTINEZ FRANCISCO MARIA - ***7410**
16/06/2025
AC FNMT Usuarios

Fdo.: Jaime Boal Martín-Larrauri

Firmado digitalmente por BOAL MARTIN LARRAURI JAIME - 05304600H
Fecha: 2025.06.16 09:38:51 +02'00'

APRENDIZAJE POR REFUERZO E IA GENERATIVA PARA LA GESTIÓN DE LA DEMANDA ENERGÉTICA

Autor: Álvaro González Tabernero
Directores: Francisco Martín Martínez, Jaime Boal Martín-Larrauri
Entidad colaboradora: Universidad Pontificia Comillas, ETS de Ingeniería - ICAI

RESUMEN DEL PROYECTO
Este proyecto explora el uso de inteligencia artificial, combinando aprendizaje por refuerzo e IA generativa, para optimizar la carga de vehículos eléctricos. Se compara su rendimiento con un modelo clásico de programación lineal, utilizando datos sintéticos realistas. Los resultados muestran que la IA ofrece soluciones adaptativas y eficientes, con potencial para sustituir o complementar métodos tradicionales en la gestión energética.

Palabras clave:
Aprendizaje por refuerzo, inteligencia artificial generativa, vehículo eléctrico, optimización energética, programación lineal, energías renovables.

MOVITACIÓN
El crecimiento exponencial del parque de vehículos eléctricos (EV) y la integración de fuentes de energía renovable intermitente exigen soluciones avanzadas para la gestión de la demanda energética. La necesidad de mantener la estabilidad de la red y minimizar los costes de operación plantea un desafío técnico relevante. Además, las soluciones clásicas basadas en programación matemática requieren el uso de solvers comerciales con licencias costosas y tiempos de ejecución poco viables para contextos operativos. Este proyecto propone un enfoque alternativo basado en aprendizaje por refuerzo (RL) e inteligencia artificial generativa para optimizar la carga de EVs bajo restricciones realistas.

OBJETIVOS
Diseñar, entrenar y evaluar un sistema de IA capaz de gestionar la carga de un vehículo eléctrico, minimizando el coste energético y respetando restricciones técnicas: SOC inicial y final, disponibilidad horaria y potencia contratada.

METODOLOGÍA
La metodología se estructura en tres fases:

1. Generación de perfiles sintéticos de consumo energético doméstico, y simulación de ese consumo mediante *Load Profile Generator* e IA generativa.

2. Formulación de un modelo clásico de optimización basado en *Min Cost Flow*,

resolviendo mediante programación lineal entera mixta.

3. Diseño, entrenamiento y evaluación de un modelo de aprendizaje poro refuerzo (RL) basado en DQN-LSTM.

Los datos generados alimentan tanto el optimizador clásico como los modelos de IA. Se definen métricas de evaluación para comparar coste energético, cumplimiento de restricciones, eficiencia €/kWh y comportamiento.

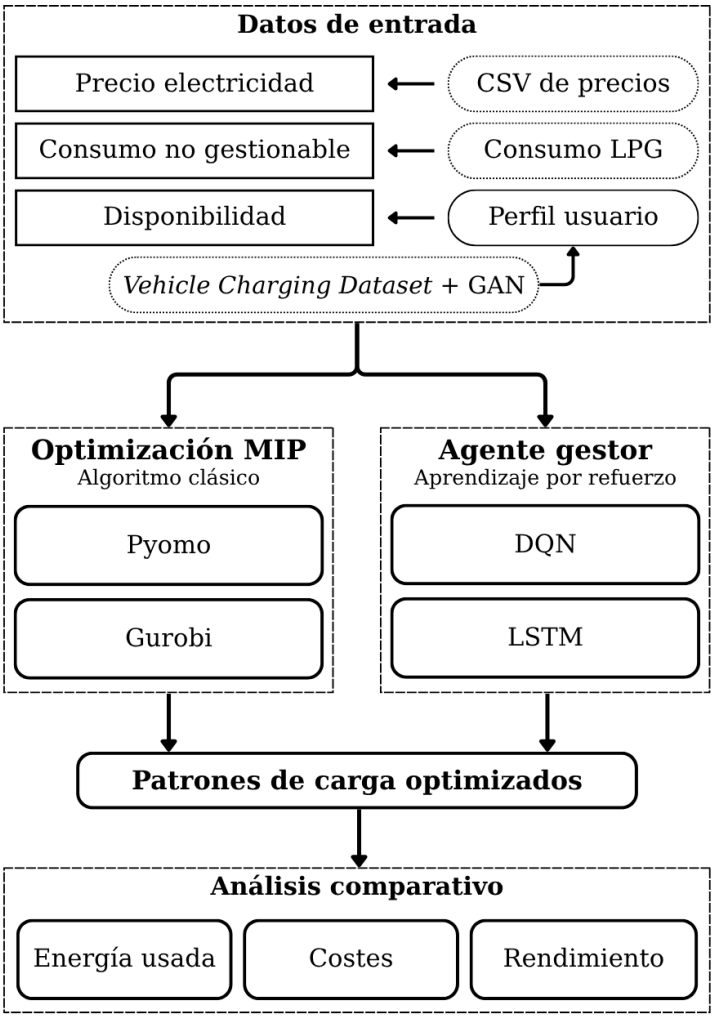

Figura 1: Metodología utilizada en el trabajo. Elaboración propia.

RESULTADOS

El modelo clásico ofrece soluciones óptimas en eficiencia energética (€/kWh), con

cumplimiento total de restricciones. El modelo RL es competitivo y muestra una mayor capacidad de adaptación ante eventos inesperados o entornos dinámicos. Los modelos generativos permiten una inferencia extremadamente rápida, aunque con menor precisión fuera del dominio entrenado.

CONCLUSIONES

Este trabajo demuestra que es viable sustituir solvers clásicos por modelos de IA en problemas energéticos realistas. La IA permite escalar a entornos más complejos y ofrece independencia tecnológica. La combinación de IA y optimización clásica puede dar lugar a soluciones híbridas más eficientes y adaptables para la carga inteligente de EVs, apoyando así la integración de renovables en la red.

REINFORCEMENT LEARNING AND GENERATIVE AI FOR ENERGY DEMAND MANAGEMENT

Author: Álvaro González Tabernero
Supervisors: Francisco Martín Martínez, Jaime Boal Martín-Larrauri
Collaborating Institution: Universidad Pontificia Comillas, ETS de Ingeniería - ICAI

PROJECT ABSTRACT
This project explores the use of artificial intelligence, combining reinforcement learning and generative AI, to optimize electric vehicle (EV) charging. Its performance is compared with that of a classical linear programming model, using realistic synthetic data. The results show that AI provides adaptive and efficient solutions, with potential to replace or complement traditional methods in energy management.

Keywords:
Reinforcement learning, generative artificial intelligence, electric vehicle, energy optimization, linear programming, renewable energy.

MOTIVATION
The exponential growth of the electric vehicle (EV) fleet and the integration of intermittent renewable energy sources require advanced solutions for energy demand management. The need to maintain grid stability and minimize operating costs poses a significant technical challenge. Moreover, classical optimization approaches rely on commercial solvers with costly licenses and impractical runtimes in operational environments. This project proposes an alternative approach based on reinforcement learning (RL) and generative artificial intelligence to optimize EV charging under realistic constraints.

OBJECTIVES
To design, train, and evaluate an AI-based system capable of managing EV charging, minimizing energy cost while meeting technical constraints: initial and final SOC, time availability, and contracted power.

METHODOLOGY
The methodology is structured in three phases:

1. Generation of synthetic domestic energy consumption profiles and simulation of such consumption using *Load Profile Generator* and generative AI.

2. Formulation of a classical optimization model based on *Min Cost Flow*, solved via mixed-integer linear programming (MILP).

3. Design, training, and evaluation of a reinforcement learning (RL) model based on DQN-LSTM.

The generated data feed both the classical optimizer and the AI models. Evaluation metrics are defined to compare energy cost, constraint fulfillment, €/kWh efficiency, and behavioral performance.

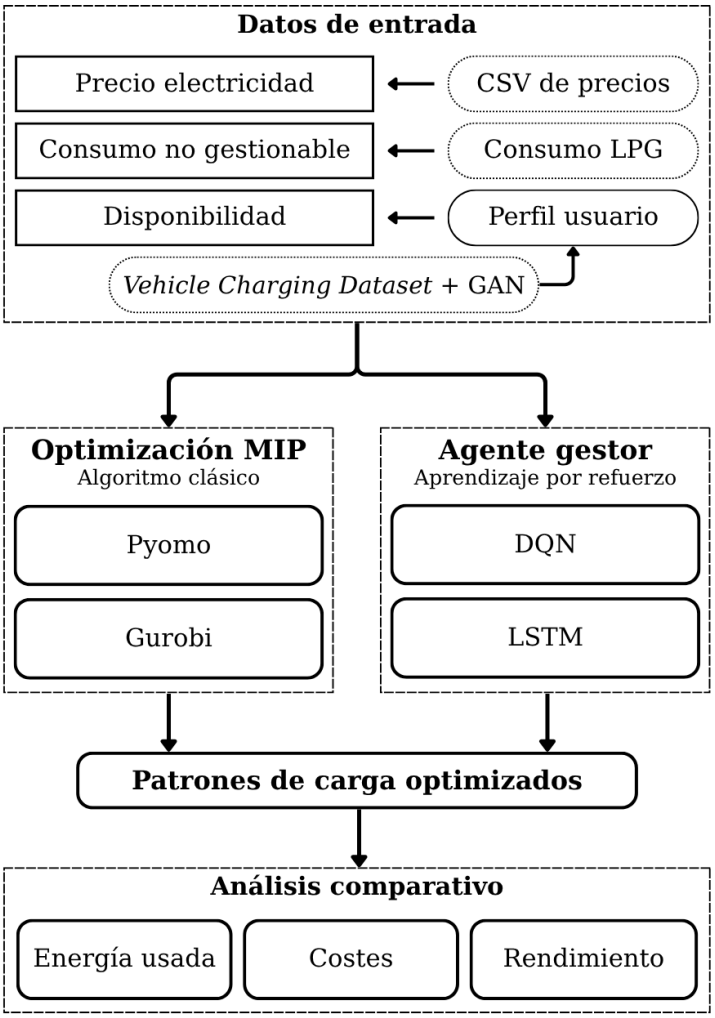

Figura 2: Methodology used in this project. Own elaboration.

RESULTS

The classical model provides optimal solutions in terms of energy efficiency (€/kWh), fully complying with all constraints. The RL model is competitive and shows greater adaptability to unexpected events or dynamic environments. Generative models

allow for extremely fast inference, although with lower accuracy outside the trained domain.

CONCLUSIONS

This work demonstrates that replacing classical solvers with AI models in realistic energy problems is feasible. AI enables scalability to more complex environments and reduces technological dependence. The combination of AI and classical optimization can yield hybrid solutions that are more efficient and adaptable for smart EV charging, supporting the integration of renewable energy into the grid.

Agradecimientos

Primero, quiero dar las gracias a mis tutores, Francisco y Jaime, por su ayuda, disponibilidad, y sobre todo por la paciencia que han tenido conmigo durante todo el proceso de desarrollo del proyecto.

Quiero agradecer a mi familia, empezando por mis padres Susana y Yago, por su apoyo y amor incondicional en todo lo que hago y mis ideas de bombero. A mi hermana Sofía, por enseñarme su infinita fuente de energía. A mis abuelos: Regina, Rosa e Isidoro, por ser la mayor suerte que puede tener un nieto.

A mis amigos, los de siempre y los más recientes, además de a todos los *Allmandrings* y *Straussis*. Especialmente quiero agradecer a mi mejor amigo, Miguel, por las ayudas y pequeños sabotajes mutuos a lo largo de estos años, y por hacer mejores a quienes le rodean.

Por úlitmo, quiero agradecer a mi tío, Javier, y mi abuelo, Pedro. Jamás estaría aquí sin su amor, inmenso cariño y aún más grande sabiduría. Por ser mis ejemplos a seguir, por definir qué es ser una buena persona y enseñarme a hacer las cosas bien, siempre.

Índice general

1. Introducción **1**
1.1. Motivación 3
1.2. Objetivos 3
1.3. Estructura 4

2. Estado del arte **7**
2.1. Fuentes de datos 7
2.1.1. Electric Vehicle Charging Patterns Dataset 8
2.1.2. Load Profile Generator 8
2.2. Optimización Clásica 10
2.2.1. Programación Lineal 10
2.2.2. Programación Entera 11
2.3. Aprendizaje por Refuerzo (RL) 11
2.3.1. Redes Neuronales dentro del RL 13
2.3.2. Deep Q-Networks (DQN) 14
2.3.3. Long Short-Term Memory (LSTM) 16
2.4. Técnicas de IA Generativa 17
2.4.1. Generative Adversarial Networks (GANs) 18
2.4.2. Variational Autoencoders (VAEs) 19
2.4.3. Modelos de difusión 20
2.4.4. Modelos Fundacionales 20
2.4.5. Aplicaciones en el sector eléctrico 21

3. Metodología **23**

4. Caso de estudio **25**
4.1. Generador de Perfiles Energéticos 25
4.1.1. Definición de perfil energético y categorías 25
4.1.2. Variabilidad entre categorías 26
4.1.3. Ejemplo de perfil energético generado 28
4.2. Precio de la Electricidad 28

4.3. Generador de Consumo 29
4.4. Gestión de Carga de EVs 33
4.4.1. Optimización Clásica 33
4.4.2. Red Neuronal de Aprendizaje por Refuerzo 39

5. Resultados **49**
5.1. Comparativa de Costes 49
5.2. Comparativa de Energía 51
5.3. Comparativa de Eficiencia 52
5.4. Comparativa de Rendimiento 52
5.5. Conclusiones de la Comparativa 54

6. Conclusiones y trabajo futuro **55**
6.1. Conclusiones 55
6.2. Próximos Pasos 56

Bibliografía **57**

Anexo I. Alineación con los Objetivos de Desarrollo Sostenible **61**

Índice de figuras

1. Metodología utilizada en el trabajo. Elaboración propia. VI
2. Methodology used in this project. Own elaboration. X

1.1. Evolución del interés por los EVs en España (2015-2025). Fuente: Google Trends [2] 2

2.1. Logo de LPG. Fuente: *LoadProfileGenerator* [4]. 8
2.2. Capturas de pantalla de LPG, página principal. Fuente: *LoadProfileGenerator* [5]. 9
2.3. Capturas de pantalla de LPG, obtención de resultados. Fuente: *LoadProfileGenerator* [5]. 9
2.4. Diagrama de interacción agente-entorno en Aprendizaje por Refuerzo. Fuente: *A brief explanation of state-action value function (Q) in RL* [11]. 13
2.5. Logo de DeepMind. Fuente: DeepMind [12]. 14
2.6. Arquitectura general de una Deep Q-Network (DQN). Fuente: Deep Q-Learning (DQN) [15]. 15
2.7. Diagrama LSTM. Fuente: *An Intuitive Explanation of LSTM* [19]. . 17
2.8. Diagrama de funcionamiento de una GAN. Fuente: Elaboración propia. 19

3.1. Metodología utilizada en el trabajo. Fuente: Elaboración propia. . . 24

4.1. Perfiles de disponibilidad generados para las distintas categorías de usuarios. Fuente: Elaboración propia 27
4.2. Precios de la electricidad en España durante 2021. Fuente: Elaboración propia a partir de datos de REE. 29
4.3. Gráficas generadas por el LPG. 32
4.4. Resultados del algoritmo de optimización clásico para el perfil `worker`: potencia aplicada y evolución del SOC en un día. Fuente: Elaboración propia. 36

4.5. Resultados del algoritmo de optimización clásico para el perfil `worker`: potencia aplicada y evolución del SOC a lo largo de un mes. Fuente: Elaboración propia. 37
4.6. Arquitectura de la red neuronal DQN utilizada en el gestor de carga. Fuente: Elaboración propia. 40
4.7. Diagrama general del sistema de gestión de carga de EVs desarrollado. Fuente: Elaboración propia. 43
4.8. Resultados del agente DQN-LSTM para el perfil `worker`: potencia aplicada y evolución del SOC en un día. Fuente: Elaboración propia. 45
4.9. Resultados del agente DQN-LSTM para el perfil `worker`: potencia aplicada y evolución del SOC a lo largo de un mes. Fuente: Elaboración propia. 45

Índice de tablas

2.1. Muestra de los datos del dataset *Electric Vehicle Charging Patterns*. 8
2.2. Fortalezas y debilidades de las GANs 18
2.3. Fortalezas y debilidades de los VAEs 20
2.4. Fortalezas y debilidades de los Modelos de Difusión 20
2.5. Fortalezas y debilidades de los Modelos Fundacionales 21
2.6. Clasificación de las distintas aplicaciones. Siendo los números en la segunda columna cada una de las técnicas por orden: (1) GANs, (2) VAEs, (3) Modelos de difusión, (4) Modelos fundacionales. 21

4.1. Perfiles energéticos generados para usuarios de EV, horas y probabilidades de disponibilidad 27
4.2. Ejemplo de perfil energético generado para la categoría `worker`. . . 28
4.3. Parámetros de configuración de la vivienda en LPG 30
4.4. Categorías de consumo energético simuladas 31
4.5. Resumen de métricas diarias por perfil para la optimización clásica. 38
4.6. Resumen de métricas diarias por perfil para el agente DQN-LSTM. 47
4.7. Tiempos de entrenamiento e inferencia del agente DQN-LSTM por perfil. 48

5.1. Comparativa de costes medios diarios por perfil entre la optimización clásica y el agente DQN-LSTM, junto con el porcentaje de reducción de coste. 50
5.2. Comparativa de energía media diaria por perfil entre la optimización clásica y el agente DQN-LSTM, junto con el porcentaje de reducción de consumo energético. 51
5.3. Comparativa de la eficiencia diaria (kWh/EUR) por perfil entre la optimización clásica y el agente DQN-LSTM, junto con el porcentaje de mejora en eficiencia. 52

Capítulo 1

Introducción

El acceso seguro y fiable a la electricidad es clave para el desarrollo económico y social de cualquier sociedad. Con el aumento de la demanda energética, se está acelerando la adopción de energías renovables para combatir el cambio climático y reducir la dependencia de combustibles fósiles.

La descarbonización del sistema energético y la mejora de la eficiencia en el uso de los recursos presentan retos en la optimización de la producción, la gestión de la demanda y la estabilidad de la red. Las infraestructuras deben adaptarse para integrar de forma eficiente las energías renovables, sin aumentar los costes ni comprometer la calidad del suministro.

El consumo energético en los hogares ha evolucionado significativamente en los últimos años, con un aumento en el uso de dispositivos electrónicos, electrodomésticos con mayor capacidad computacional y el auge del vehículo eléctrico (EV). Éste se ha convertido en una parte esencial de la transición hacia una movilidad más sostenible, pero su integración plantea desafíos significativos en términos de gestión de la demanda y estabilidad de la red eléctrica.

Los EV y vehículos híbridos enchufables (PHEV, por sus siglas en inglés) lejos están de ser tecnología del futuro, son el presente. Los EV y vehículos electrificados han cambiado por completo el paradigma energético doméstico, convirtiéndose en una parte integral de la vida cotidiana. La adopción de estos vehículos ha crecido exponencialmente en los últimos años, impulsada por la necesidad de reducir las emisiones de gases de efecto invernadero y la dependencia de los combustibles fósiles. En España, el número de EVs ha aumentado considerablemente, con una creciente infraestructura de carga y un interés cada vez mayor por parte de los consumidores.

La carga de estos EVs puede generar picos de demanda que, de no ser gestionados adecuadamente, podrían llegar a comprometer la fiabilidad de la red y aumentar significativamente los costes operativos [1], repercutiendo finalmente en los consumidores. La Figura 1.1 muestra la evolución del interés por los EV en España durante los últimos diez años

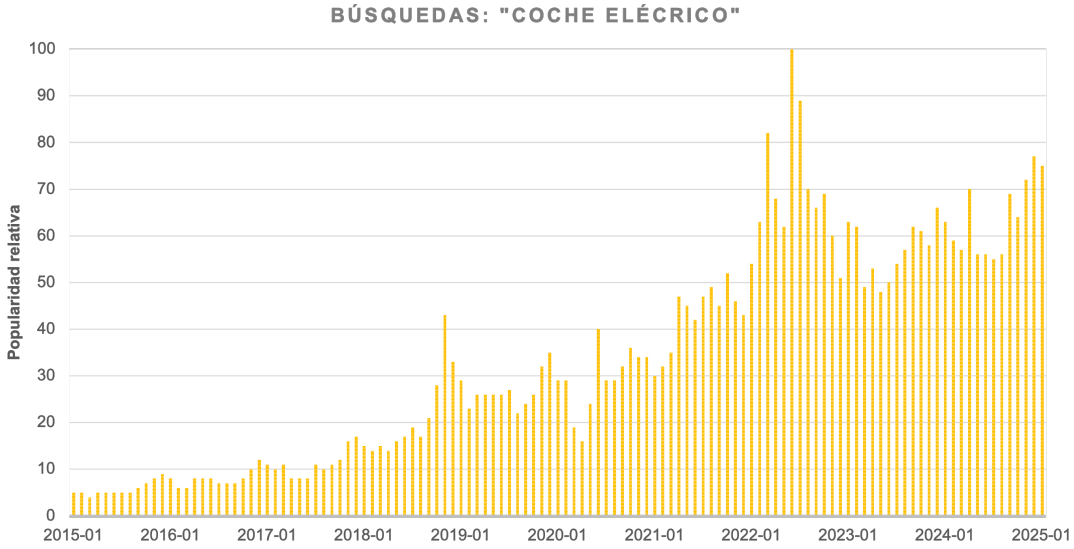

Figura 1.1: Evolución del interés por los EVs en España (2015-2025). Fuente: Google Trends [2]

Además, la variabilidad en la disponibilidad de energía renovable, como la solar y la eólica, las fuentes renovables más populares en nuestro país, introduce incertidumbre en la generación de energía. Esto, a su vez, requiere una gestión más dinámica y flexible de la demanda para garantizar un suministro estable y eficiente. La gestión de la carga de los EVs es un aspecto crítico para garantizar que la transición hacia un sistema de transporte sostenible no comprometa la estabilidad de la red eléctrica.

Por otro lado, la Inteligencia Artificial Generativa (IA Generativa) surge como una herramienta prometedora para enfrentar estos desafíos. Gracias a su capacidad de analizar grandes volúmenes de datos en tiempo real, la IA Generativa puede predecir patrones de consumo, optimizar la generación y distribución de energía, anticipar fallos en infraestructuras y ajustar el consumo a las condiciones de la red de manera eficiente y eficaz.

La IA Generativa permite la creación de modelos predictivos que pueden simular diferentes perfiles de carga, facilitando la toma de decisiones informadas y

la implementación de estrategias de gestión de la demanda más efectivas. Además, su capacidad natural para aprender y ajustarse progresivamente a nuevas condiciones y entornos la convierte en una herramienta realmente útil para la planificación y operación de sistemas energéticos cada vez más complejos.

1.1. Motivación

La gestión responsable y eficiente de la energía es una tarea de vital importancia para una sociedad cada día más consciente y proactiva en tareas de sostenibilidad. Añadiendo a este panorama el crecimiento experimentado por la movilidad eléctrica en el siglo XXI, dicha gestión eficiente y responsable del consumo energético aplica necesariamente a la carga de EVs.

La carga de estos vehículos puede generar picos de demanda que, si no se gestionan adecuadamente, pueden comprometer la fiabilidad de la red eléctrica y aumentar los costes operativos, repercutiendo finalmente en los consumidores. Además, la variabilidad en la disponibilidad de energía renovable, como la solar y la eólica, introduce incertidumbre en la generación de energía, lo que requiere una gestión más dinámica y flexible de la demanda para garantizar un suministro estable y eficiente.

Las innovadoras herramientas que presenta la IA Generativa, junto con las redes neuronales (Neural Networks, NN) basadas en aprendizaje por refuerzo (Reinforcement Learning, RL), exhiben un gran potencial para abordar estos retos de manera novedosa, y con resultados del más alto rendimiento.

Adicionalmente, los métodos clásicos de optimización suelen requerir el uso de solvers comerciales, cuya ejecución en tiempos razonables depende de licencias muy costosas para el usuario. Sustituir estos sistemas por una solución basada en inteligencia artificial—que solo incurre en un coste de entrenamiento—puede reducir drásticamente los costes operativos asociados, facilitando su adopción en entornos reales donde la escalabilidad y la eficiencia económica son clave.

1.2. Objetivos

El objetivo del proyecto es desarrollar un sistema gestor inteligente para la carga de EVs en un contexto dómestico. Para ello, se hará uso de técnicas de IA Generativa, aprendizaje por refuerzo y algoritmos clásicos de optimización.

El gestor deberá decidir, en base a una serie de restricciones y requisitos, tanto físicos como lógicos, si cargar o no el EV en un momento dado. Idealmente, el gestor conseguirá cargar el EV de la forma más eficiente posible, minimizando el coste de la carga y maximizando el uso de energía disponible, al tiempo que se cumplen las restricciones impuestas por el usuario y las características del sistema eléctrico.

Los objetivos secundarios específicos del proyecto son los siguientes:

1. Generar perfiles sintéticos de demanda no gestionable, disponibilidad y requisitos del EV mediante técnicas generativas.

2. Realizar una optimización clásica del problema de la carga, modelándolo como un programa lineal (Linear Programming, LP).

3. Comparar los resultados obtenidos mediante la optimización clásica con los alcanzados por el sistema basado en IA Generativa y aprendizaje por refuerzo.

Se evaluarán tres métricas principales: coste incurrido en la carga, cuantía de energía empleada y rendimiento computacional.

1.3. Estructura

El trabajo está estructurado en seis capítulos, incluyendo este introductorio. Tras esta breve introducción, exposición de la motivación para el proyecto y los objetivos que se pretenden alcanzar, el segundo capítulo se centra en la revisión del estado del arte, donde se analizarán las técnicas de IA Generativa, las redes neuronales con aprendizaje por refuerzo y su aplicación en la gestión de la demanda energética.

El tercer capítulo presenta el diseño y desarrollo del sistema propuesto, incluyendo la arquitectura de la red neuronal y el enfoque de aprendizaje por refuerzo utilizado, es decir, la metodología llevada a cabo durante el proyecto. El cuarto capítulo ofrece una explicación detallada de la implementación del sistema de acuerdo a lo explicado en el estado del arte, y contenido en el marco de la metodología. En este capítulo se detallan los algoritmos implementados, las técnicas de IA Generativa utilizadas y la forma en que se integran para lograr los objetivos del proyecto.

El quinto capítulo muestra una discusión de la comparación de los resultados obtenidos con los diferentes enfoques — la optimización clásica y el sistema de

RLNN + IAGen — y una evaluación de su rendimiento en términos de coste, eficiencia y rendimiento computacional. Finalmente, en el sexto capítulo se presentan las conclusiones del proyecto, así como las posibles líneas de investigación futura. Se reflexiona sobre los logros alcanzados, las lecciones aprendidas y las implicaciones de los resultados obtenidos. Además, se discuten las posibles aplicaciones prácticas del sistema desarrollado, dando fin al proyecto.

El trabajo se complementa con una serie de apéndices que incluyen detalles técnicos adicionales, enlace al código fuente y otros materiales relevantes que respaldan la investigación y el desarrollo. Estos apéndices proporcionan una visión más profunda de los aspectos técnicos del proyecto y permiten una comprensión más completa de los métodos y resultados presentados.

Capítulo 2

Estado del arte

A lo largo de este capítulo, se expondrá la base y el marco teórico sobre los cuales se desarrolla el proyecto, presentando los conceptos fundamentales y las tecnologías más relevantes en los ámbitos de la inteligencia artificial, el aprendizaje por refuerzo y la optimización clásica.

Se describirán las principales fuentes de datos empleadas, así como las metodologías y modelos que constituyen el estado del arte en la resolución de problemas complejos dentro del sector energético. El objetivo es proporcionar una visión global que sirva de base para comprender las técnicas y herramientas utilizadas en el del proyecto, contextualizando su aplicación y relevancia en el marco actual de la investigación.

2.1. Fuentes de datos

Tal y como se ha explicado en la intriducción, durante el desarrollo del proyecto se van a combinar técnicas de IA Generativa, Redes Neuronales con Aprendizaje por Refuerzo y algoritmos clásicos de optimización. Para ello, es necesario tener acceso a distintas fuentes de datos, que se adapten a cada una de las técnicas que se van a utilizar.

De acuerdo con los objetivos propuestos, y el planteamiento general del proyecto, las fuentes de datos para este proyecto, que serán explicadas a continuación, son las siguientes:

1. Electric Vehicle Charging Patterns Dataset

2. Load Profile Generator

2.1.1. Electric Vehicle Charging Patterns Dataset

Electric Vehicle Charging Patterns es un conjunto de datos obtenidoo desde la plataforma *Kaggle* [3], que contiene información sobre los patrones de carga de vehículos eléctricos (EVs) en un entorno urbano. Tal y como lo describe su autor, "otorga un análisis exhaustivo de patrones de carga de EVs y comportamiento de los usuarios". El dataset incluye 1320 muestras sobre el consumo energético durante la carga, la duración de la carga y detalles de cada uno de los vehículos registrados, de entre sus 20 campos.

	Modelo	Capacidad (kWh)	SoC Inicial (%)	SoC Final (%)
0	BMW i3	108.46	29.37	86.12
1	Hyundai Kona	100.00	10.12	84.66
2	Chevy Bolt	75.00	6.85	69.92
3	Hyundai Kona	50.00	83.12	99.62
4	Hyundai Kona	50.00	54.26	63.74

Tabla 2.1: Muestra de los datos del dataset *Electric Vehicle Charging Patterns*.

2.1.2. Load Profile Generator

Load Profile Generator es una aplicación desarrollada por Noah Pflugradt, de la *Bern University of Applied Sciences* [4] , que genera perfiles de carga sintéticos para un hogar, incluyendo la demanda de energía no gestionable, la disponibilidad de carga y los requisitos del vehículo eléctrico. Este generador es capaz de crear perfiles de carga sintéticos que simulan el comportamiento real de un hogar, lo que permite entrenar y evaluar modelos de IA sin necesidad de datos reales.

Figura 2.1: Logo de LPG. Fuente: *LoadProfileGenerator* [4].

Esta herramienta ha sido instrumental para el desarrollo del proyecto, permitiendo la generación de consumos completamente realistas, en una amplia variedad de escenarios predefinidos. La personalización para la generación de los datos dentro del progama permite generar consumos para diferentes configuraciones de hogares, variando el número de hijos, su actividad física, la distancia al trabajo y muchas más opciones.

Además de la variabilidad, en cuanto al análisis de datos se refiere, no sólo facilita el procesado de los datos, devolviendo los resultados en CSVs (Comma Separated Values) agrupados con el consumo global, y también separando por tipos de consumo; sino que el propio programa permite el ajuste granulado de la resolución temporal interna y externa.

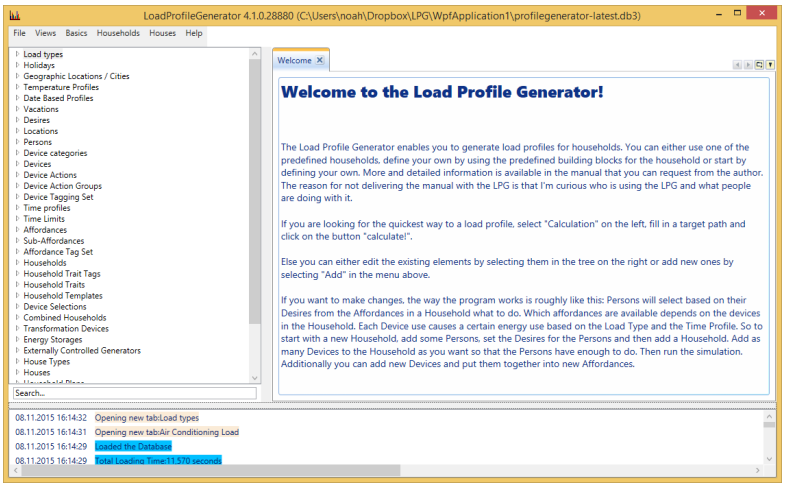

Figura 2.2: Capturas de pantalla de LPG, página principal. Fuente: *LoadProfileGenerator* [5].

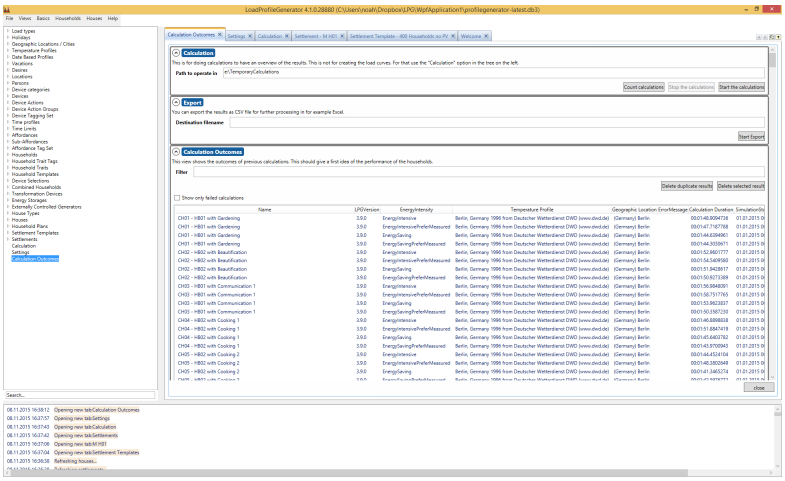

Figura 2.3: Capturas de pantalla de LPG, obtención de resultados. Fuente: *LoadProfileGenerator* [5].

2.2. Optimización Clásica

La optimización clásica, o programación matemática, es el conjunto de métodos, principios y técnicas que se utilizan para encontrar la mejor solución a un problema cuantitativo concreto, sujeto a un conjunto de restricciones o condiciones, que limitan el conjunto de soluciones posibles [6]. En su forma más básica, la optimización clásica busca maximizar o minimizar una función real, referida como función objetivo, resolviendo una serie de ecuaciones y/o desigualdades. Por tanto, se puede definir la optimización como la elección del mejor elemento dentro de una selección de elementos disponibles, sujetos a algún criterio [7].

La optimización clásica se divide en varias categorías, dependiendo de la naturaleza de la función objetivo y las restricciones. Las categorías más comunes son:

- Programación Lineal
- Programación Entera

2.2.1. Programación Lineal

La programación lineal es un caso concreto del espacio de métodos recogido bajo el término *Programación Convexa*. Este concepto se refiere a los programas que resuelven problemas en los que la función objetivo es cóncava (maximización) o convexa (minimización). Dentro de este marco, la programación lineal es el caso particular en el que la función objetivo y las restricciones son ecuaciones o inecuaciones estrictamente lineales [7].

Geométricamente, puede ser interpretado como si dichas restricciones definieran una figura en el espacio euclídeo, y la función objetivo es la dirección en la que se busca maximizar o minimizar el valor de la función [8]. Un ejemplo de un problema de programación lineal es el siguiente:

$$\begin{aligned} \text{Max.} \quad & z = 3x + 2y \\ \text{Sujeto a} \quad & x + y \leq 4 \\ & 2x + y \leq 6 \\ & x \geq 0, \quad y \geq 0 \end{aligned}$$

Aquí, la función objetivo es $z = 3x_1 + 2x_2$, y las restricciones son claramente lineales. El objetivo es encontrar los valores de x_1 y x_2 tal que z sea máxima mientras se cumplen las restricciones. La solución óptima se puede encontrar

utilizando el método simplex o el método de interior-puntos, entre otros algoritmos de optimización lineal [7].

2.2.2. Programación Entera

Este tipo de programación analiza programas lineales en los que se impone que, al menos una de las variables, sea un número entero. En este caso particular, la programación entera se aleja de la programación convexa, ya que la función objetivo y las restricciones no son estrictamente lineales, y su resolución suele resultar más compleja que la de un programa lineal al uso [7]. La programación entera se divide en dos categorías:

- **Programación Entera *Mixta* (MIP)**: En este caso, algunas variables son enteras y otras pueden tomar valores continuos. Este tipo de programación es muy común en problemas de optimización combinatoria, donde se busca una solución óptima entre un conjunto discreto de opciones, como la asignación de recursos o la planificación de rutas. Hay casos en los que las variables pueden tomar valores continuos dentro de un número de opciones discreto, por ejemplo: [0, 0.5, 1] [8].
- **Programación Entera *Pura* (IP)**: En este caso, todas las variables son enteras. Este tipo de programación se utiliza en problemas donde todas las decisiones deben ser discretas, como la selección de proyectos o la asignación de tareas [8].

2.3. Aprendizaje por Refuerzo (RL)

Stutton y Barto definen el Aprendizaje por Refuerzo (Reinforcement Learning, RL) como un "enfoque de aprendizaje automático en el que un agente aprende a tomar decisiones mediante la interacción con un entorno, recibiendo recompensas o penalizaciones en función de sus acciones". En otras palabras, RL es aprender qué hacer, asociar situaciones con acciones, para maximizar una recompensa numérica [9].

Es importante resaltar la diferencia entre el Aprendizaje Supervisado y el Aprendizaje por Refuerzo. En el primer caso, el modelo aprende a partir de un conjunto de datos etiquetados, donde cada entrada tiene una salida conocida. En el caso del Aprendizaje por Refuerzo, el modelo aprende a partir de su propia experiencia, obtenida tras una serie de interaccioens con el entorno. Por lo mismo, el RL es también diferente a lo comúnmente conocido como *Aprendizaje no Supervisado*, ya que este grupo de métodos típicamente buscan patrones o estructuras en una

colección de datos sin etiquetar [9].

Un problema único para los algoritmos de aprendizaje por refuerzo es el equilibrio entre exploración y explotación —o *exploration-exploitation trade-off*—. El término exploración suele hacer referencia a la "curiosidad" del agente por encontrar nuevas y diferentes estrategias, mientras que explotación se refiere a la tendencia del agente a explotar las estrategias que ya conoce, y sabe que otorgan resultados positivos. Encontrar un equilibrio entre estas dos características es tremendamente importante para conseguir que el agente aprenda [9].

Glosario de términos clave en Aprendizaje por Refuerzo

Tras una breve introducción al Aprendizaje por Refuerzo, es importante definir algunos de los términos clave que se utilizan en este campo. Estos términos son fundamentales para entender los conceptos y algoritmos que se desarrollan en el ámbito del RL. A continuación, se presentan los términos más relevantes y que serán utilizados a lo largo del trabajo: [10]

- **Agente**: Es la parte que toma decisiones y ejecuta acciones dentro del entorno, con el objetivo de maximizar su recompensa acumulada.
- **Estado (S)**: El contexto instantáneo en que se encuentra el agente en cualquier momento.
- **Acción (A)**: Conjunto de todas las posibles acciones que el agente puede realizar en un estado dado. Una acción concreta se denota habitualmente como a.
- **Recompensa (r)**: Retroalimentación que recibe el agente tras ejecutar una acción en un estado determinado. Indica el grado de éxito o fracaso de la acción tomada.
- **Entorno**: Es el "mundo."[en] el que opera el agente. Recibe como entrada el estado actual y la acción seleccionada, y devuelve la recompensa obtenida y el nuevo estado resultante.
- **Política (π)**: Estrategia que sigue el agente para decidir qué acción tomar en cada estado.
- **Valor o Utilidad (V)**: Valor esperado de la recompensa acumulada (a largo plazo y con descuento) que puede obtener el agente desde un estado concreto, siguiendo una política determinada.

- **Valor de acción (Q)**: Valor esperado de la recompensa acumulada (con descuento) que puede obtener el agente al tomar una acción específica en un estado dado, y siguiendo posteriormente una política determinada. Es la función que asigna pares estado-acción a recompensas esperadas.

- **Factor de descuento (γ)**: Parámetro que determina la importancia de las recompensas futuras frente a las inmediatas. Un valor bajo de γ resulta en un agente cortoplacista. En caso contrario, el agente tendrá una visión más global y a largo plazo.

Añadido, se muestra un diagrama que expone el flujo de entradas y salidas del entorno:

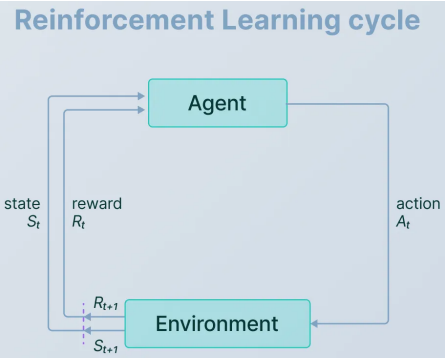

Figura 2.4: Diagrama de interacción agente-entorno en Aprendizaje por Refuerzo. Fuente: *A brief explanation of state-action value function (Q) in RL* [11].

2.3.1. Redes Neuronales dentro del RL

Al final, en un algoritmo de RL, un entorno es una función que transforma una acción en el siguiente estado y recompensa, y los agentes son funciones que transforman la función de recompensa y el estado en la siguiente acción. Es aquí donde las redes neuronales entran en juego, ya que, en esencia, no son más que aproximadores de funciones.

Por esta condición de aproximadores de funciones que presentan las redes neuronales, son una herramienta excelente cuando el conjunto de acciones o de estados es demasiado grande, o no del todo conocido, siendo este el caso de la inmensa mayoría de los casos de uso en el mundo real. En estos casos, las redes neuronales pueden aprender a aproximar la función de valor o la política del agente, permitiendo que el agente tome decisiones informadas basadas en la información disponible [10].

Una RLNN, por ejemplo, puede ser la solución idónea para una IA que aprenda a jugar al ajedrez. El ajedrez es el juego de estrategia por excelencia, y es tan complejo que no es posible enumerar todas las posibles jugadas y resultados. Por tanto, una RLNN puede aprender a jugar al ajedrez a través de la experiencia, jugando millones de partidas y ajustando su política de juego en función de las recompensas obtenidas al término de cada movimiento o cada partida.

Figura 2.5: Logo de DeepMind. Fuente: DeepMind [12].

Alpha Zero, la IA desarrollado por Google DeepMind, utiliza una red neuronal con aprendizaje por refuerzo, y ha aprendido a jugar al ajedrez, al Go y al shogi con un novel tal, que ha sido capaz de superar a los mejores jugadores humanos y por supuesto a otros programas de ajedrez, gracias a su capacidad para aprender, después de millones de partidas, y adaptarse a diferentes estilos de juego [13].

2.3.2. Deep Q-Networks (DQN)

Una *Deep Q-Network* (DQN) es uno de los algoritmos más potentes y populares en RL, ya que hace uso de la combinación de redes neuronales profundas y *Q-Learning*, permitiendo al agente aprender políticas tremendamente optimizadas en entornos muy complejos. [14].

Q-Learning es un algoritmo que enseña a los agentes a comportarse de manera óptima en un entorno Markoviano controlado, esto es un entorno en el que la probabilidad de transición entre estados depende únicamente del estado actual y la acción tomada, y no de estados anteriores. Funciona mejorando iterativamente la evaluación de una acción específica en un estado determinado [16].

Funcionamiento de DQN [17]

El algoritmo Deep Q-Network (DQN) es una técnica de aprendizaje por refuerzo que utiliza redes neuronales para aprender a tomar decisiones. En lugar de trabajar con tablas de valores como el Q-learning tradicional, DQN usa una red para aproximar los valores $Q(s, a)$, que indican lo buena que es una acción a en un estado s.

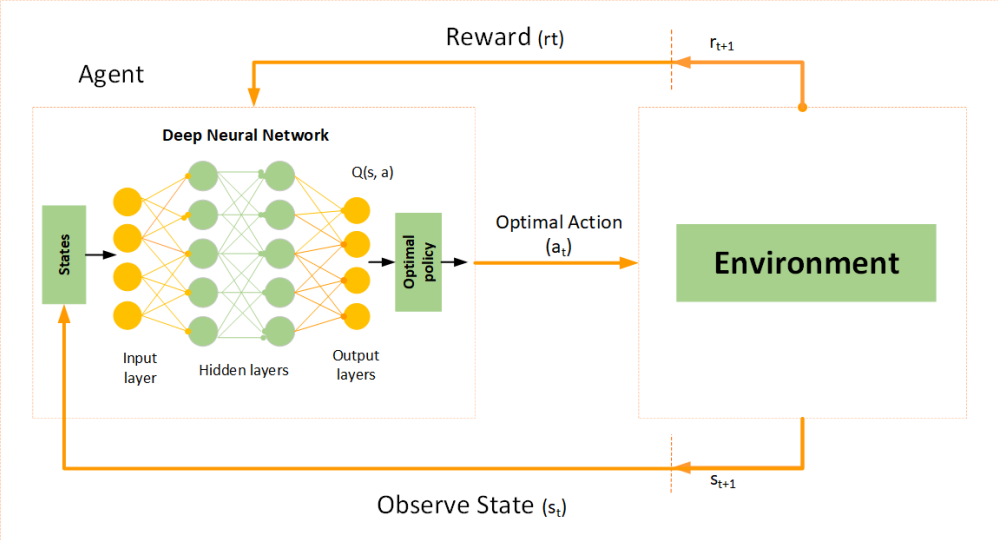

Figura 2.6: Arquitectura general de una Deep Q-Network (DQN). Fuente: Deep Q-Learning (DQN) [15].

- **Valor objetivo (Ecuación de Bellman):** Para aprender, DQN compara su predicción con un valor objetivo, calculado usando la ecuación de Bellman:

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^-)$$

donde:

- r es la recompensa obtenida tras hacer una acción,
- s' es el siguiente estado,
- γ es un número entre 0 y 1 que da más o menos importancia a las recompensas futuras,
- θ^- son los parámetros de una copia de la red (llamada red objetivo).

Esta actualización sigue el principio de optimalidad de Bellman, que define cómo se calcula el valor óptimo esperado de una acción en un estado.

- **Función de pérdida:** La red aprende minimizando la diferencia entre lo que predice y el valor objetivo:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim D}\left[(y - Q(s, a; \theta))^2\right]$$

Aquí, D es un conjunto de experiencias almacenadas (*replay buffer*), que permite entrenar de forma más estable y variada.

- **Entrenamiento:** Los parámetros θ de la red se actualizan usando un algoritmo de optimización (como descenso del gradiente) que intenta reducir el error en las predicciones.
- **Red objetivo:** Para evitar que el entrenamiento sea inestable, DQN usa una segunda red (la red objetivo) que se mantiene fija durante varios pasos y se actualiza solo cada cierto tiempo copiando los valores de la red principal.

Ventajas de DQN [14]

(+) El uso de redes neuronales profundas otorga a DQN la capacidad de aprender representaciones abstractas y de alta dimensionalidad, facilitando el aprendizaje en entornos complejos.

(+) La utilización de un *replay buffer* permite un entrenamiento más estable y eficiente, ya que el agente puede revisar experiencias pasadas y no depende únicamente de las más recientes.

(+) DQN tiene una gran capacidad de generalización, permitiéndole desenvolverse en situaciones no vistas previamente, lo que lo hace muy adecuado para problemas de RL complejos.

Inconvenientes de DQN [14]

(-) DQN es sensible a la elección de hiperparámetros, como la tasa de aprendizaje, el factor de descuento, el ratio de exploración y el tamaño del *replay buffer*, lo que puede afectar significativamente su rendimiento.

(-) No está diseñado para su uso en línea, siendo más adecuado para entornos *offline*.

(-) La actualización de *Q-learning* implementada en DQN puede ser propensa a la sobreestimación de los valores Q.

2.3.3. Long Short-Term Memory (LSTM)

Las redes *Long Short-Term Memory* (LSTM) son una arquitectura de redes neuronales recurrentes (RNNs) diseñada para gestionar información secuencial—como las series temporales—de forma más eficaz. A diferencia de las RNN simples, que utilizan un único vector oculto para propagar información, las LSTM incorporan una estructura interna más compleja que les permite mantener información relevante durante más tiempo y decidir qué conservar y qué olvidar [18].

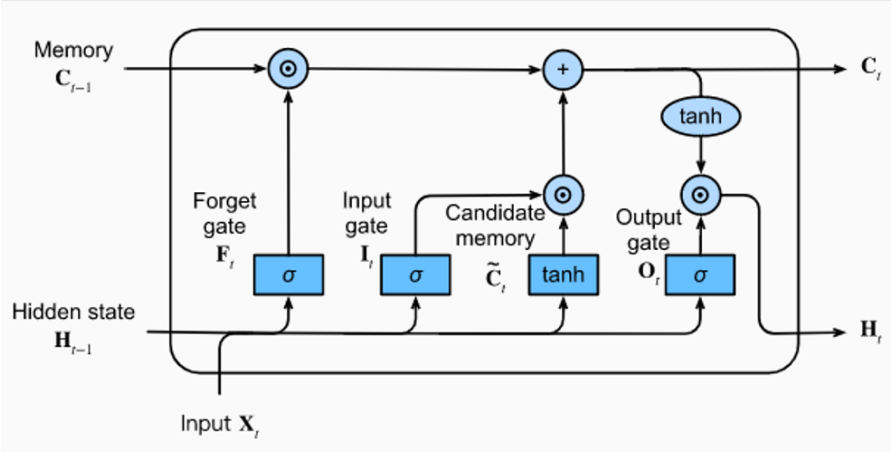

Figura 2.7: Diagrama LSTM. Fuente: *An Intuitive Explanation of LSTM* [19].

Internamente, una LSTM introduce tres puertas: la *puerta de entrada*, la *puerta de olvido* y la *puerta de salida*. Estas puertas actúan como filtros que controlan el flujo de información a lo largo del tiempo:

- La puerta de entrada determina qué parte de la información nueva debe almacenarse.
- La puerta de olvido decide qué parte del contenido anterior debe descartarse.
- La puerta de salida regula qué información del estado interno se transmite a la siguiente capa o paso de tiempo.

Gracias a esta estructura, las LSTM son especialmente útiles para tareas donde es importante tener en cuenta el contexto anterior, como el procesamiento de lenguaje natural, la clasificación de secuencias o la predicción de series temporales. La capacidad de decidir dinámicamente qué recordar y qué olvidar convierte a las LSTM en una herramienta eficaz para trabajar con dependencias temporales prolongadas [19]. Estas propiedades hacen que las LSTM sean más robustas para modelar dependencias a largo plazo en secuencias, en comparación con las RNN simples.

2.4. Técnicas de IA Generativa

A lo largo de esta sección, se tratará de explicar y exponer las principales técnicas usadas en el mundo de la inteligencia artificial generativa, que pueden trasladarse de manera práctica al sector eléctrico. Las cuatro técnicas que tratar son:

- Generative Adversarial Networks (GANs)
- Variational Autoencoders (VAEs)
- Modelos de difusión
- Modelos fundacionales

2.4.1. Generative Adversarial Networks (GANs)

Esta primera técnica, emplea aprendizaje no supervisado con dos redes neuronales antagónicas que son entrenadas de manera que compiten entre sí, en un proceso similar a un juego de "suma cero".

Durante el entrenamiento, una de las dos redes, la red A, genera candidatos para ser evaluados por la red B. Normalmente, la red A se entrena para que genere candidatos y estímulos siguiendo una determinada distribución o reglas, mientras que la red B pretende discriminar y diferenciar casos originales de casos generados artificialmente.

Con este planteamiento, el objetivo es complicarle la tarea a la red B, consiguiendo que la salida de la red A sea tan similar al original, que dificulte la diferenciación. [20].

Fortalezas	**Debilidades**
Generación de imágenes	Entrenamiento complejo
Reescalado	Inestabilidad y "colapso del modo"
Multimodalidad	
Gran versatilidad	

Tabla 2.2: Fortalezas y debilidades de las GANs

También se incluye un gráfico que muestra visualmente el funcionamiento y la organización en alto nivel de una GAN:

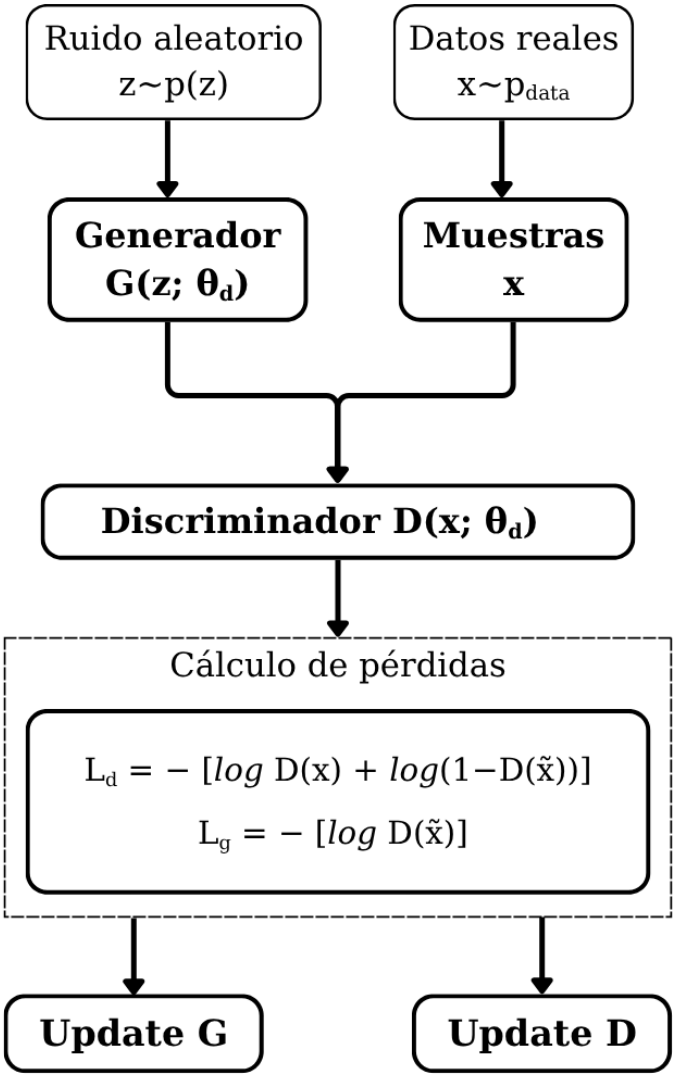

Figura 2.8: Diagrama de funcionamiento de una GAN. Fuente: Elaboración propia.

2.4.2. Variational Autoencoders (VAEs)

El segundo tipo de modelo generativo, los Autocodificadores Variacionales, son modelos que aprenden de sus datos como una representación comprimida de los mismos, asignados a una distribución probabilística, que luego se usará para generar variaciones de estos. Es decir, los VAE aprenden a identificar y extraer las características más relevantes de los datos con los que son entrenados. [21].

Fortalezas	**Debilidades**
Extracción componentes principales	Entrenamiento delicado
Compresión de datos	Ejemplos borrosos
Generar variaciones de datos entrenamiento	Interpretación de expertos

Tabla 2.3: Fortalezas y debilidades de los VAEs

2.4.3. Modelos de difusión

En tercer lugar, los modelos de difusión son una técnica de generación de datos muy utilizada para imágenes, que funciona de forma diferente a otros enfoques expuestos anteriormente, como las GANs.

En lugar de un sistema entre dos redes, los modelos de difusión utilizan un proceso gradual de adición y eliminación de ruido. El proceso de entrenamiento consiste en añadir ruido de forma progresiva a los datos originales, hasta que se vuelven irreconocibles para luego revertir este proceso, eliminando el ruido paso a paso, hasta recuperar una versión clara de los datos. Esto permite que el modelo pueda generar nuevas imágenes, u otro formato de datos, comenzando desde ruido aleatorio, porque ha aprendido a reconstruir datos realistas a partir de ruido. [22].

Fortalezas	**Debilidades**
Generación desde ruido aleatorio	Mayor tiempo de computación
Alto nivel de detalle	Mayor coste computacional
Multimodalidad	Poca versatilidad

Tabla 2.4: Fortalezas y debilidades de los Modelos de Difusión

2.4.4. Modelos Fundacionales

Finalmente, los modelos fundacionales son la punta de lanza de la IA Generativa. Son un tipo de modelo de Deep Learning, que ha sido entrenado con una vastísima y muy diversa cantidad de datos desestructurados y sin etiquetar. Lo que los hace destacar es su capacidad de aprendizaje generalizado, que les permite transferir sus conocimientos a diversas tareas sin necesidad de ser específicamente entrenados para ellas. Estos modelos, que pueden generar o comprender texto, imágenes, audio, entre otros formatos, resultan muy útiles en aplicaciones que requieran una entendimiento profundo del contexto o la creación de nuevos contenidos en distintos formatos. [23].

Entre ellos, los Modelos de Lenguaje Extensos (LLMs), como GPT o BERT, permiten generar y comprender texto de manera eficiente. Por su parte, los Modelos de Lenguaje Pequeños (SLMs) ofrecen soluciones más ligeras y específicas.

Fortalezas	**Debilidades**
Versatilidad sin conocimiento específico	Necesidad de gran cantidad de datos
Multimodalidad	Posibilidad de sesgos
Eficiencia	Posibilidad información desactualizada
Prolificidad	Conocimiento limitado

Tabla 2.5: Fortalezas y debilidades de los Modelos Fundacionales

2.4.5. Aplicaciones en el sector eléctrico

A continuación, se muestra una comparativa con las distintas aplicaciones y servicios de las anteriores técnicas en el sector eléctrico, así como qué técnicas serían necesarias para cada servicio, el riesgo, y qué mejorarían:

Aplicación	**Técnicas**	**Riesgo**	**Mejora**			
			Temporal	*Económica*	*Mantenimiento*	*Cualitativa*
Detección de anomalías	1,3		✓	✓	✓	✓
Creación de imágenes	1,3,4		✓	✓		✓
Análisis de imágenes	1,2,4		✓	✓	✓	✓
Generación escenarios consumo	1,2		✓	✓		✓
Predicciones (Demanda y Prod.)	1,2,4		✓	✓		✓
Análisis de mercado	2,4		✓	✓		✓
Estimación de estados	1,2		✓	✓	✓	✓
Interfaz interactiva	4		✓			✓

Tabla 2.6: Clasificación de las distintas aplicaciones. Siendo los números en la segunda columna cada una de las técnicas por orden: (1) GANs, (2) VAEs, (3) Modelos de difusión, (4) Modelos fundacionales.

El riesgo, representado por una escala de colores—cuanto más oscuro mayor riesgo—, se refiere a la seguridad de las aplicaciones, y cómo de seguro es realizar una inversión en cada una de ellas. Con ello, se tiene en cuenta el nivel y oportunidades de desarrollo en cada uno de los campos, la volatilidad de estos. Por tanto, un proyecto de sencilla aplicación con oportunidad de desarrollo va a ser más "seguro" que otra alternativa que encuentre mayor dificultad de implementación o

desarrollo.

Los tipos de mejora a los que se hace referencia describen los campos en los que cada aplicación puede ofrecer beneficios. De esta manera, la mejora temporal se refiere a que la aplicación ofrece mayor eficiencia en el tiempo que la herramienta o proceso que estaría reemplazando. Del mismo modo, la mejora económica indica que dicha aplicación incurre en menor coste, bien sea coste fijo, variable o de instalación, que las posibles alternativas.

Las aplicaciones marcadas positivamente en la casilla de mantenimiento quieren decir que ofrecen soluciones con mejores resultados para procesos de mantenimiento. Finalmente, la mejora cualitativa indica que las ventajas que plantea la aplicación son transversales, y se ven reflejadas directamente en la calidad de experiencia para el usuario.

Capítulo 3

Metodología

El desarrollo de este trabajo se ha dividido en tres partes bien diferenciadas: los dos generadores, tanto de perfiles energéticos como de consumo, y el gestor inteligente de carga de EVs, como se muestra en la Figura 3.1

Para la creación de dichos perfiles de consumo, se han empleado técnicas de IA Generativa, con el objetivo de obtener una gran variedad de perfiles de consumo, partiendo del *dataset* ya obtenido (se explica en la Sección 2.1.1), con limitado número de muestras. Estos perfiles se han generado de forma que representen adecuadamente la variabilidad de la demanda energética para la carga de un EV, teniendo en cuenta las restricciones, requisitos y hábitos del usuario. En segundo lugar, la generación de consumos energéticos se ha llevado a cabo usando LPG, cuyo *dataset* se describe en la Sección correspondiente.

La implementación del código relevante para el proyecto se ha realizado en *Python* [24], utilizando una serie de librerías y herramientas a disposición de estas tareas. Para la resolución del problema de optimización clásico, se ha hecho uso de la librería *Pyomo* [25], con *GurobiPy* [26] como *solver*. Para el desarrollo de todo lo relacionado con aprendizaje profundo, se han empleado las librerías de *Torch* [27].

La metodología seguida en este trabajo, representada en la Figura 3.1, se estructura en varias fases interconectadas. En primer lugar, se realiza la recopilación y preprocesamiento de los datos, tanto de perfiles de consumo como de generación energética. Posteriormente, se lleva a cabo la generación sintética de perfiles mediante IA generativa, lo que permite ampliar la base de datos y mejorar la representatividad de los escenarios analizados. A continuación, se desarrolla el gestor inteligente de carga de vehículos eléctricos, integrando los perfiles generados y aplicando técnicas de optimización y aprendizaje profundo para la toma de decisiones. Finalmente, se evalúan los resultados obtenidos, analizando el impacto

de las estrategias propuestas sobre la gestión de la demanda y la eficiencia energética.

Cada uno de estos bloques, tal y como se observa en la figura, está conectado de manera que la salida de una etapa sirve como entrada para la siguiente, asegurando así un flujo de trabajo coherente y alineado con los objetivos del proyecto. Esta estructura modular facilita la validación de cada componente y permite realizar ajustes de manera independiente para optimizar el rendimiento global del sistema.

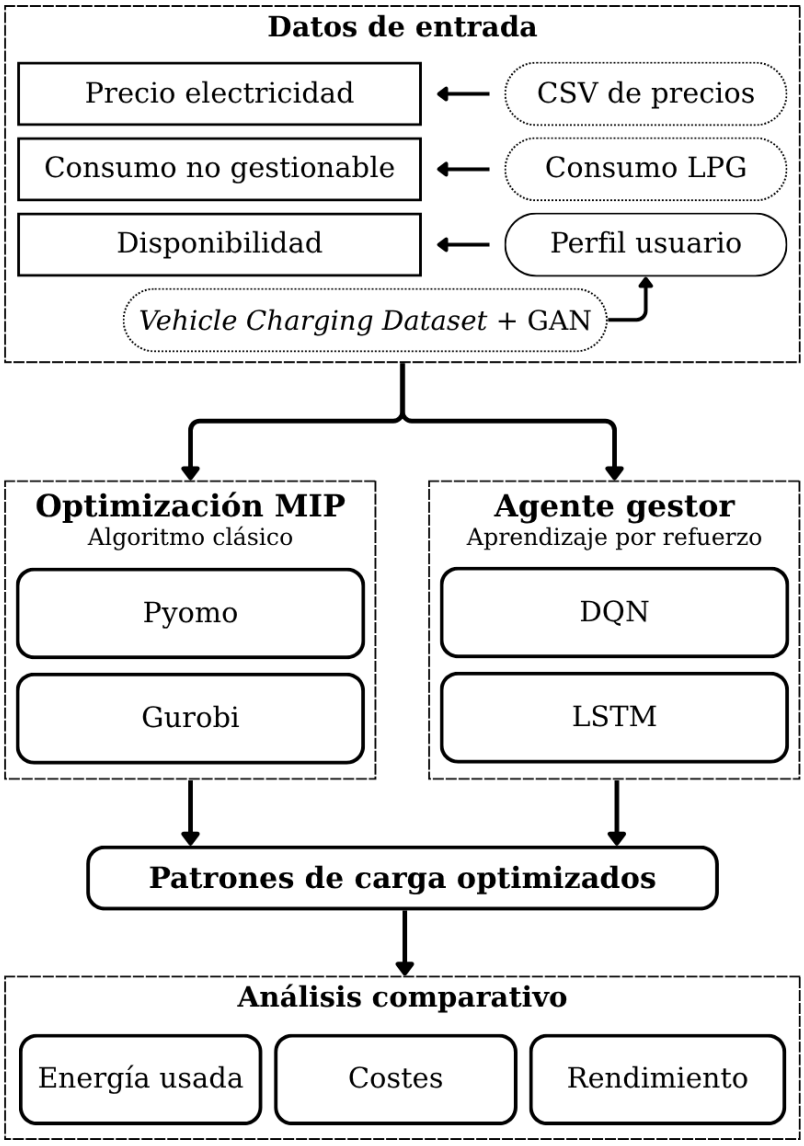

Figura 3.1: Metodología utilizada en el trabajo. Fuente: Elaboración propia.

Capítulo 4

Caso de estudio

En este capítulo se presenta el caso de estudio desarrollado para validar y comparar distintas estrategias de gestión de la carga de vehículos eléctricos (EVs) en el entorno doméstico. El objetivo es analizar el comportamiento de un sistema inteligente de gestión de carga, basado en técnicas de optimización clásica y aprendizaje por refuerzo, utilizando perfiles energéticos realistas y datos de consumo generados específicamente para este propósito.

El capítulo se estructura en tres bloques principales: en primer lugar, se describe el proceso de generación de perfiles energéticos de usuarios de EV, con ayuda de técnicas de IA generativa; en segundo lugar, se detalla la simulación del consumo energético doméstico mediante herramientas especializadas; y, finalmente, se expone el diseño, implementación y evaluación del gestor de carga inteligente, comparando su rendimiento frente a un enfoque clásico de optimización. Todo ello permite obtener una visión integral de los retos y oportunidades asociados a la integración eficiente de los EV en el hogar.

4.1. Generador de Perfiles Energéticos

4.1.1. Definición de perfil energético y categorías

En el contexto de este trabajo, un **perfil energético** de usuario de EV se define como una función que describe, para cada intervalo horario de un día típico, la *probabilidad de disponibilidad* del vehículo para la carga y, opcionalmente, la *demanda energética esperada* (kWh) asociada a la recarga en ese intervalo. Formalmente, un perfil energético es una secuencia de 24 valores (o 96 si la resolución es de 15 minutos), donde cada valor representa la probabilidad de que el EV esté disponible para cargar y, en algunos casos, la energía media demandada en ese intervalo. Así,

un perfil energético sintetiza los hábitos de uso y recarga de un usuario concreto a lo largo de la jornada.

Por otro lado, una **categoría** es un grupo de usuarios que comparten patrones similares de uso y disponibilidad del EV, como por ejemplo "trabajador", "jubilado", "viajero", etc. Cada categoría se caracteriza por un conjunto típico de perfiles energéticos, que reflejan la variabilidad dentro de ese grupo. En la tabla siguiente, el término "Perfil" hace referencia a la *categoría*, mientras que los valores horarios y probabilidades representan un perfil energético típico de esa categoría.

4.1.2. Variabilidad entre categorías

Cada categoría presenta perfiles energéticos característicos, tanto en la distribución horaria de la disponibilidad como en la energía total demandada. Por ejemplo, la categoría `worker` (trabajador) muestra una alta disponibilidad para la carga durante la noche (00:00–06:00 y 21:00–00:00), con probabilidades de 0.95 y 0.85 respectivamente, y una baja disponibilidad en horario laboral (06:00–21:00, probabilidad 0.10). La categoría `retired` (jubilado) presenta alta disponibilidad durante casi todo el día, con valores de 0.95 en la madrugada y mañana (00:00–12:00), 0.70 en mediodía (12:00–16:00) y 0.90 en la tarde-noche (16:00–00:00), reflejando una mayor flexibilidad en los hábitos de carga.

Por su parte, la categoría `traveller` (viajero) alterna periodos de alta disponibilidad nocturna (00:00–08:00, 0.90) con franjas de baja disponibilidad diurna (08:00–18:00, 0.40) y una recuperación parcial por la tarde (18:00–00:00, 0.80). El perfil `night_owl` (nocturno) concentra la carga casi exclusivamente en horario nocturno (00:00–06:00, 0.98; 20:00–00:00, 0.90), con muy baja disponibilidad durante el día (06:00–20:00, 0.15).

Finalmente, el perfil `flexible` mantiene una disponibilidad media y estable a lo largo del día, con valores de 0.80 en la madrugada (00:00–08:00), 0.60 en horario laboral (08:00–18:00) y 0.75 en la tarde-noche (18:00–00:00).

En resumen, los perfiles energéticos generados reflejan tanto la probabilidad de carga en cada franja horaria como la adaptación a los hábitos y necesidades de cada tipo de usuario, permitiendo al gestor de carga ajustar su estrategia de forma personalizada.

Los perfiles generados se han agrupado en las siguientes categorías, que representan los distintos tipos de usuarios y sus hábitos de carga:

Perfil	Hábitos de carga	Horas	Prob.
Trabajador (`worker`)	Carga principalmente de noche; baja disponibilidad en horario laboral.	00:00–06:00 06:00–21:00 21:00–00:00	0.95 0.10 0.85
Flexible	Disponibilidad media y estable, con ligera bajada en horario laboral.	00:00–08:00 08:00–18:00 18:00–00:00	0.80 0.60 0.75
Jubilado (`retired`)	Alta disponibilidad todo el día, salvo ausencias puntuales a mediodía y tarde.	00:00–12:00 12:00–16:00 16:00–00:00	0.95 0.70 0.90
Viajero (`traveller`)	Alta disponibilidad salvo ausencias frecuentes en horario diurno y algunas tardes.	00:00–08:00 08:00–18:00 18:00–00:00	0.90 0.40 0.80
Nocturno (`night_owl`)	Carga casi exclusiva por la noche; baja disponibilidad diurna.	00:00–06:00 06:00–20:00 20:00–00:00	0.98 0.15 0.90

Tabla 4.1: Perfiles energéticos generados para usuarios de EV, horas y probabilidades de disponibilidad

Visualmente, los perfiles energéticos generados para cada categoría de usuario se pueden observar en la Figura 4.1. Aquí se muestran las probabilidades de disponibilidad de carga presentados en la tabla anterior.

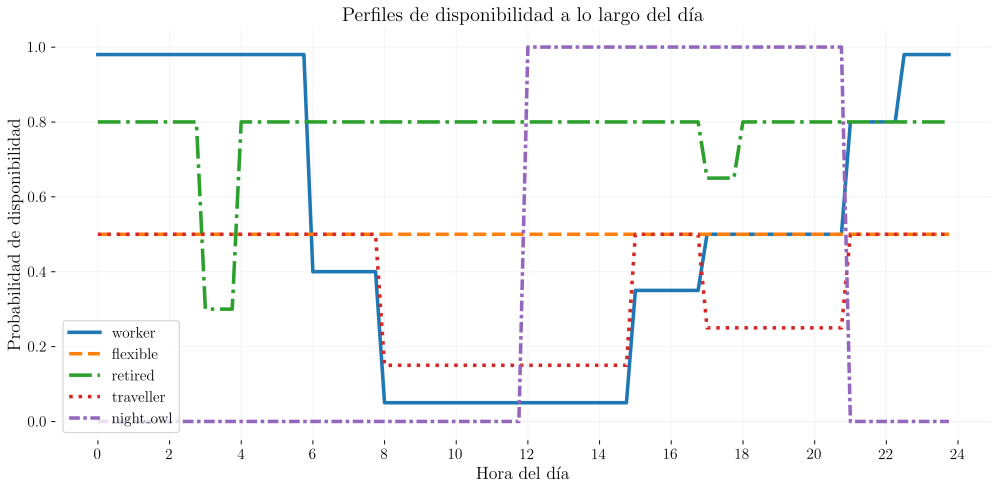

Figura 4.1: Perfiles de disponibilidad generados para las distintas categorías de usuarios. Fuente: Elaboración propia

4.1.3. Ejemplo de perfil energético generado

A continuación se muestra un ejemplo concreto de perfil energético generado para la categoría `worker` (trabajador), con resolución horaria. Para cada hora del día se indica la probabilidad de disponibilidad y la energía media demandada si se produce carga:

Hora	Prob. disponibilidad	Energía media (kWh)
00:00–01:00	0.95	1.8
01:00–02:00	0.95	1.7
02:00–03:00	0.95	1.5
03:00–04:00	0.95	1.2
04:00–05:00	0.95	1.0
05:00–06:00	0.90	0.8
06:00–07:00	0.10	0.2
07:00–08:00	0.10	0.1
08:00–17:00	0.05	0.0
17:00–21:00	0.10	0.3
21:00–00:00	0.85	1.5

Tabla 4.2: Ejemplo de perfil energético generado para la categoría `worker`.

En este ejemplo, se observa que la mayor probabilidad de disponibilidad y demanda energética se concentra en horario nocturno, reflejando el hábito de cargar el vehículo tras la jornada laboral. La energía media por franja se ha estimado a partir de los datos originales y los generados por la GAN, considerando la necesidad diaria de recarga para cubrir los desplazamientos típicos de un trabajador.

4.2. Precio de la Electricidad

Para el desarrollo del trabajo son esenciales los precios de la elcectricidad, que se han obtenido directamente de Red Eléctrica de España (REE), que publica estos precios cada hora. Se han escogido los precios para la península (PVPC) durante el año 2021, ya que es reciente y es un año completo. Además, encaja con las fechas del resto de datos. A continuación se muestra una representación visual descriptiva de los precios obtenidos:

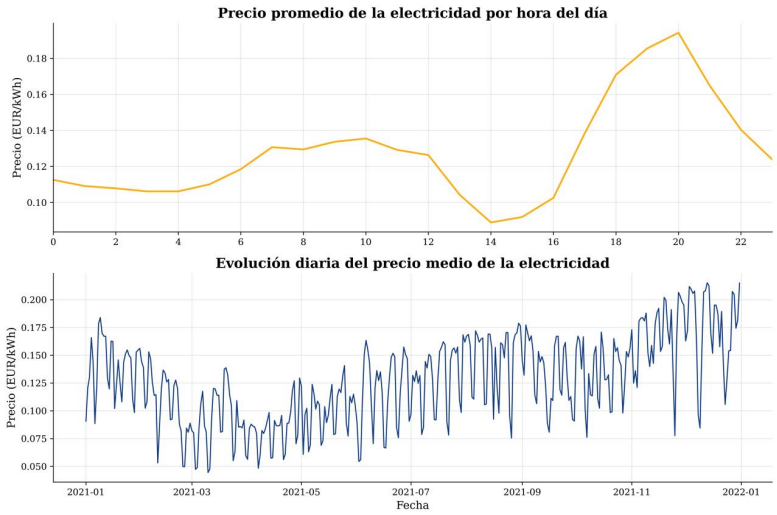

Figura 4.2: Precios de la electricidad en España durante 2021. Fuente: Elaboración propia a partir de datos de REE.

Estas variables no solo definen el estado y las restricciones físicas del sistema, sino que también son fundamentales en la formulación de la función objetivo de la optimización clásica y en la función de recompensa del agente de aprendizaje por refuerzo. En la optimización, se utilizan para minimizar el coste total de la carga y garantizar el cumplimiento de las restricciones operativas. En el caso del aprendizaje por refuerzo, estas mismas variables se emplean para construir el vector de estado del entorno y calcular la recompensa en cada paso, guiando así el aprendizaje del agente hacia políticas que optimicen tanto el coste como la comodidad y la viabilidad técnica de la carga del EV.

4.3. Generador de Consumo

Este segundo bloque se centra en la tarea de generación de consumos energéticos domésticos, utilizando el software *Load Profile Generator (LPG)*. Se ha configurado una vivienda de tipo unifamiliar en el entorno urbano de Madrid, modelada para representar una situación típica de un hogar con presencia de EV, hijos y hábitos laborales estándar.

La vivienda, como se muestra en la tabla 4.3, está ocupada por cuatro personas, cuyas rutinas diarias han sido configuradas para representar un hogar realista con uso intensivo de energía:

1. **Adulto 1:** trabaja fuera del hogar a jornada completa (8h–17h), desplazándose en EV cinco días a la semana. Contribuye a la carga del EV en horario

nocturno.

2. **Adulto 2:** trabaja parcialmente desde casa, con jornada reducida. Su presencia en el hogar durante el día impacta en el uso de climatización, cocina y dispositivos electrónicos.

3. **Niño 1:** escolarizado, con jornada matinal completa. Genera picos de consumo por la mañana y por la tarde (actividades, ducha, iluminación).

4. **Niño 2:** edad preescolar, cuidado en casa. Su presencia constante condiciona el uso continuo de climatización, iluminación y aparatos electrónicos de ocio.

La configuración empleada, entonces, es la siguiente:

Parámetro	**Descripción**
Tiempo simulado	365 días (1 año completo)
Resolución temporal interna	1 minuto
Resolución temporal externa	15 minutos
Ubicación geográfica	Madrid, España
Tipo de vivienda	Unifamiliar, tamaño medio (100–150 m^2), con garaje y trastero
Aislamiento térmico	Estándar
Equipamiento	Completo en electrodomésticos
Composición familiar	▪ Adulto 1 (trabaja fuera, jornada completa) ▪ Adulto 2 (trabaja en casa, jornada parcial) ▪ Niños: uno en edad escolar, otro en edad preescolar
EV	▪ Un coche compartido ▪ Uso diario laboral y escolar ▪ Carga preferente nocturna
Perfil de temperatura	Datos realistas para 2025 en Madrid
Calendario	Incluye fines de semana y festivos nacionales

Tabla 4.3: Parámetros de configuración de la vivienda en LPG

Con estas características cargadas en la configuración del programa generador, se ha procedido a la generación de los siguientes datos de consumo energético:

Categoría	**Descripción**
Electrodomésticos de cocina	Horno, vitrocerámica, microondas; lavavajillas; frigorífico y congelador (consumo continuo)
Electrodomésticos de limpieza	Lavadora, secadora y aspiradora
Entretenimiento y uso de oficina	Televisores, ordenadores, router; consolas, altavoces, impresoras
Iluminación	Luz general por estancias, activación dependiente de presencia y hora del día
Climatización	Calefacción eléctrica (invierno); aire acondicionado o ventiladores (verano)
Agua caliente sanitaria (ACS)	Consumo eléctrico en ausencia de caldera de gas
Carga del EV	Carga según trayectos diarios; horarios preferentes nocturnos
Consumo en espera (stand-by)	Dispositivos conectados permanentemente; carga latente (TV, router, cargadores)

Tabla 4.4: Categorías de consumo energético simuladas

Con esta configuración se han generado los consumos energéticos horarios agregados para los distintos elementos. Este perfil de consumo no gestionable sirve como input para el gestor inteligente de carga, presentado en la siguiente sección.

Descripción de los Datos Generados

A continuación se muestran algunas de las gráficas generadas por la aplicación, que describen los datos generados por el LPG. Estas gráficas muestran el consumo energético de la vivienda a lo largo de un año, con una resolución de 15 minutos.

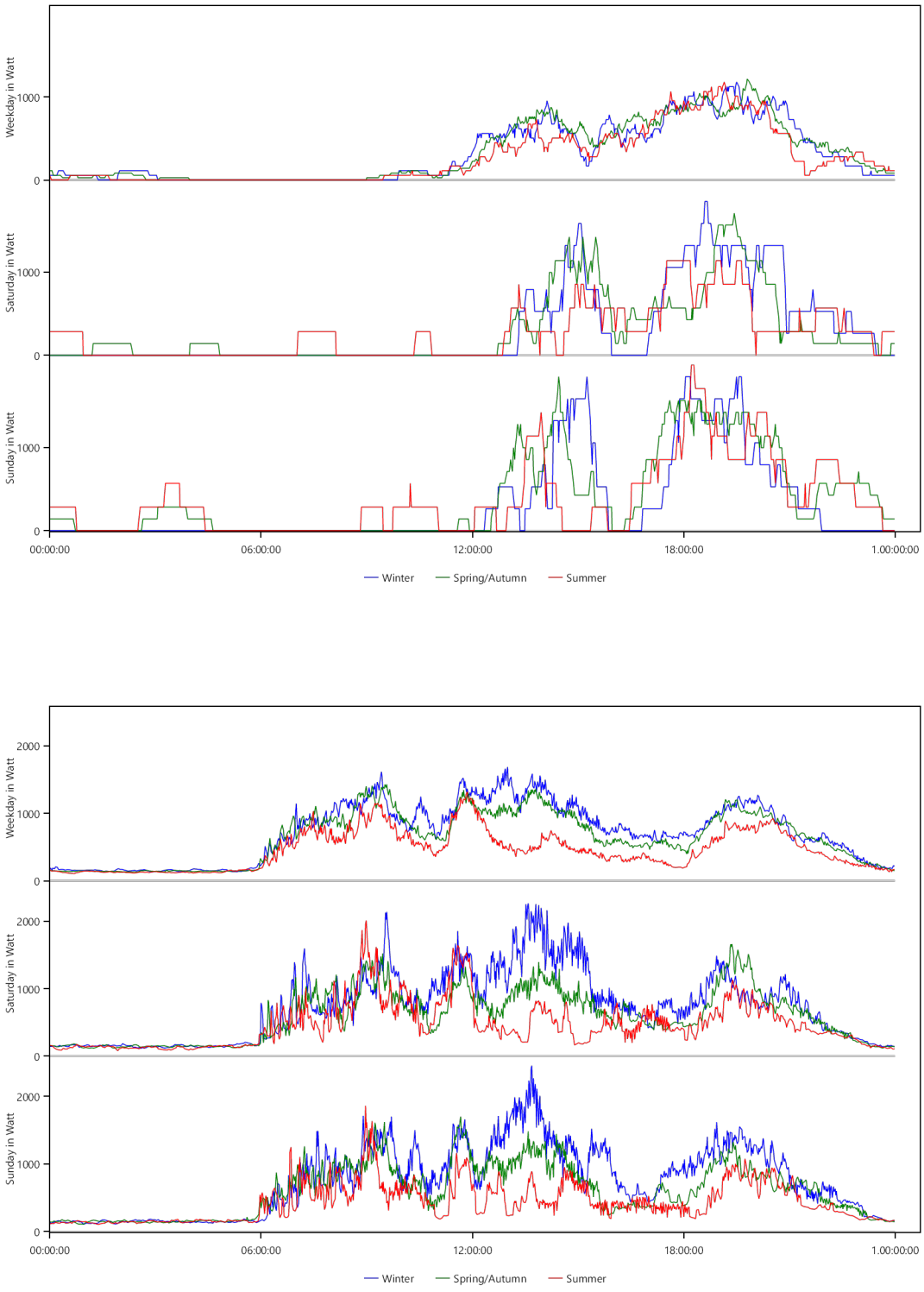

Figura 4.3: Gráficas generadas por el LPG.

Las dos primeras gráficas muestran el consumo energético de la vivienda, promediado a lo largo de un año, durante los días laborables de la semana y el fin de semana. La gráfica de la izquierda muestra el consumo energético para la carga del EV, mientras que la gráfica de la derecha muestra el consumo total de energía de la vivienda. Además, se han representado las distintas estaciones del año con líneas de distinto color, para un análisis más detallado.

Estas gráficas, siendo una muestra de todas las disponibles para el análisis, proporcionan una visión general del consumo energético de la vivienda, y son útiles para entender los patrones de consumo y carga del EV. Además, permiten identificar los momentos de mayor consumo energético, lo que es esencial para la gestión eficiente de la carga del EV.

4.4. Gestión de Carga de EVs

El gestor de carga es el tercer y último bloque del trabajo, y es realmente el foco del mismo. Para su desarrollo, se ha implementado una red neuronal de aprendizaje por refuerzo DQN añadiendo unas capas de LSTM, que se ha entrenado con los perfiles de consumo energético, obtenidos tras la generación de datos adicionales; y toda la información de la vivienda y del EV.

El gestor de carga tiene como objetivo optimizar la carga del EV, minimizando por supuesto el coste incurrido en la carga, teniendo en cuenta las restricciones y requisitos del usuario. Para ello, se ha implementado un algoritmo de aprendizaje por refuerzo, que permite al gestor de carga aprender de la experiencia y mejorar su rendimiento a lo largo del tiempo.

Antes de usar redes neuronales, se ha implementado un algoritmo de optimización clásico, que sirve como base para comparar el rendimiento del gestor de carga. Este algoritmo utiliza las técnicas de optimización clásica, tal y como se ha presentado en el Capítulo 2, para obtener una solución óptima al problema que presenta la carga del EV.

4.4.1. Optimización Clásica

Primeramente, se han de definir los índices, parámetros y variables que formarán las restricciones y objetivos del problema de optimización.

Índices:

- t: Instante de tiempo, $t = 1, \ldots, T$ (por ejemplo, $T = 24$ para resolución horaria o $T = 96$ para 15 minutos)

Parámetros:

- Δt: Duración del intervalo de tiempo [h], en este caso $\Delta t = 0{,}25$ (15 minutos)
- $precio(t)$: Precio de la electricidad en el instante t [€/kWh], obtenido del perfil de REE (véase sección de precios de la electricidad)
- $P_{\max}$: Potencia máxima del cargador [kW], valor fijo según el modelo de cargador (por ejemplo, $P_{\max} = 7{,}4$ kW)
- P_{red}: Potencia máxima contratada con la red [kW], valor fijo (por ejemplo, $P_{\text{red}} = 10$ kW)
- $P_{NG}(t)$: Potencia no gestionable en t [kW], obtenida del perfil de consumo generado por LPG (véase sección de generador de consumo)
- $A(t)$: Disponibilidad del vehículo en t (variable continua entre 1, si disponible, y 0, no disponible), según el perfil energético generado (véase sección de generador de perfiles energéticos)
- η: Eficiencia del sistema de carga (por ejemplo, $\eta = 0{,}95$)
- capacidad: Capacidad total de la batería del EV [kWh] (por ejemplo, 40 kWh)
- SOC_i: Estado inicial de carga al comienzo del día [kWh]
- SOC_f: Estado objetivo de carga al final del día [kWh]

Variables de decisión:

- $P(t)$: Potencia de carga aplicada en el instante t [kW]
- $SOC(t)$: Estado de carga del EV en el instante t [kWh]
- $C(t)$: Coste de carga en el instante t [€], $C(t) = E(t) \cdot p(t)$

Una vez se tienen claras las variables, tras un análisis del problema, se definen las siguientes restricciones y función objetivo:

$$\min_{P(t)} \quad \sum_{t=1}^{T} P(t) \cdot precio(t) \cdot \Delta t \tag{4.1}$$

$$\text{s.a.} \quad 0 \leq P(t) \leq P_{\max} \cdot A(t) \qquad \forall\, t = 1, \ldots, T \tag{4.2}$$

$$P(t) + P_{NG}(t) \leq P_{\text{red}} \qquad \forall\, t = 1, \ldots, T \tag{4.3}$$

$$SOC(t+1) = SOC(t) + \eta \cdot P(t) \cdot \Delta t \qquad \forall\, t = 1, \ldots, T \tag{4.4}$$

$$SOC(1) = SOC_i \tag{4.5}$$

$$SOC(T+1) \geq SOC_f \tag{4.6}$$

$$0 \leq SOC(t) \leq \text{capacidad} \qquad \forall\, t = 1, \ldots, T \tag{4.7}$$

Una vez ya se ha planteado el problema, se implementa el Python [24] usando la librería *Pyomo* [25] para definir el modelo de optimización, y *GurobiPy* [26] como solver. El código se ha implementado de forma que se pueda cargar el perfil de consumo energético generado por el LPG, y se pueda ejecutar el algoritmo de optimización para obtener la solución óptima al problema de carga del EV.

En la Figura 4.4 se muestra el comportamiento diario del algoritmo de optimización clásico para el perfil `worker`, donde se observa la evolución de la potencia entregada al cargador y la evolución del estado de carga (SOC) del EV a lo largo del día. Por otro lado, en la Figura 4.5 se muestran las mismas variables pero a lo largo de un mes entero, donde se puede observar las tendencuas de carga y consumo del EV, así como la variabilidad en la potencia aplicada y el SOC a lo largo del mes.

En el análisis diario, se puede observar cómo el algoritmo de optimización clásico se adapta a las costumbres impuestas por el perfil de usuario y, sobre todo minimizando el coste, cargando mayoritariamente el EV durante la noche, y en momentos del día en los que el consumo energético es menor, y hay mayor probabilidad de que el vehículo esté disponible para la carga. Además, se observa que el SOC del vehículo se mantiene dentro de los límites establecidos, y se carga de forma eficiente para minimizar el coste total de la carga.

En la Figura 4.5 se observa la evolución de la potencia de carga y el estado de carga (SOC) del vehículo eléctrico a lo largo de un mes completo para el perfil `worker`. Esta visualización permite analizar el comportamiento del algoritmo de optimización clásica en un horizonte temporal amplio, mostrando cómo se distribuyen las sesiones de carga y cómo se adapta el sistema a las restricciones de disponibilidad y precio de la electricidad.

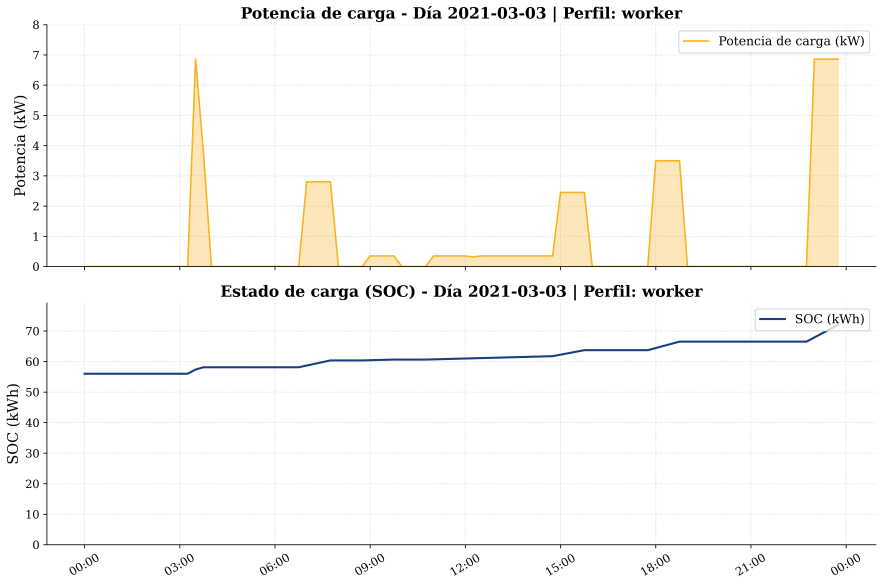

Figura 4.4: Resultados del algoritmo de optimización clásico para el perfil `worker`: potencia aplicada y evolución del SOC en un día. Fuente: Elaboración propia.

A lo largo del mes, se aprecia que la mayor parte de la carga se concentra en las horas nocturnas, coincidiendo con los periodos de mayor disponibilidad del vehículo y precios más bajos, lo que confirma que el optimizador es capaz de aprovechar las ventanas más favorables para minimizar el coste. Además, el SOC se mantiene dentro de los límites establecidos y alcanza el objetivo diario de carga en la mayoría de los días, reflejando una gestión eficiente bajo condiciones ideales y perfiles de consumo predecibles.

En resumen, la Figura 4.5 ilustra tanto la capacidad del algoritmo clásico para optimizar la carga bajo condiciones conocidas como sus limitaciones ante escenarios dinámicos, justificando así la necesidad de enfoques más flexibles y adaptativos como los basados en aprendizaje por refuerzo.

Se muestra únicamente el perfil `worker` como ejemplo más representativo, ya que sería el perfil de consumo energético más habitual.

Siguiendo con los resultados de la optimización clásica, a continuación se presentan una serie de métricas y resultados adicionales obtenidos por el algoritmo de optimización clásico, que permiten evaluar su rendimiento y eficiencia en la gestión de la carga del EV.

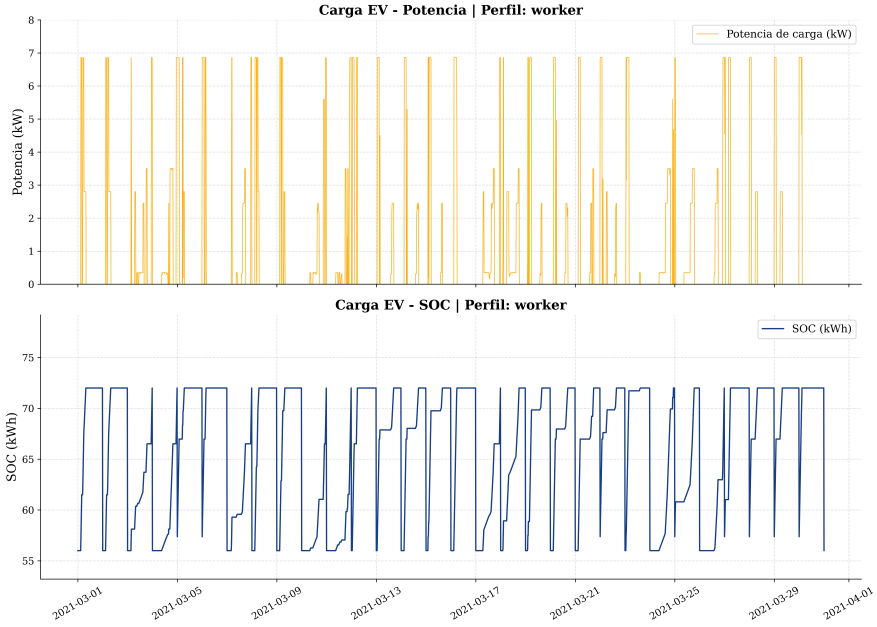

Figura 4.5: Resultados del algoritmo de optimización clásico para el perfil `worker`: potencia aplicada y evolución del SOC a lo largo de un mes. Fuente: Elaboración propia.

Entorno de ejecución. Las pruebas se han realizado en un MacBook Pro 14"(2024) con Apple M4 Pro (12 núcleos CPU a 4.05 GHz, 18 núcleos GPU), 24 GB de memoria RAM unificada LPDDR5X a 7500 MT/s y SSD NVMe PCIe 4.0 de 1 TB. El sistema operativo es *macOS Sequoia 15.5*. El entorno Python utilizado es 3.11.8, con las siguientes versiones de librerías principales: PyTorch 2.3.0, NumPy 1.26.4, Pyomo 6.7.1 y Gurobi 11.0.1. El entrenamiento y la inferencia se han realizado en CPU.

Analizando los resultados de la Tabla 4.5, se observa que el coste medio diario de la carga es muy similar entre los distintos perfiles, situándose en torno a 1,65–1,90 EUR, con pequeñas variaciones según los hábitos de disponibilidad y carga de cada usuario. El perfil `retired` presenta el menor coste medio (1,28 EUR), seguido de `flexible` (1,44 EUR), mientras que `worker` y `traveller` alcanzan valores ligeramente superiores (1,60 EUR y 1,49 EUR, respectivamente). Estas diferencias se explican principalmente por la mayor flexibilidad horaria de los perfiles `retired` y `flexible`, que les permite aprovechar mejor los periodos de menor precio de la electricidad.

En cuanto a la energía media diaria cargada, todos los perfiles presentan los

Perfiles	Coste medio (EUR)	Energía media (kWh)	Eficiencia media (EUR/kWh)
worker	1,60	16,00	0,10
flexible	1,44	16,00	0,09
retired	1,28	16,00	0,08
night_owl	1,49	16,00	0,09
traveller	1,45	16,00	0,09

Tabla 4.5: Resumen de métricas diarias por perfil para la optimización clásica.

mismos valores (16 kWh), ya que el algoritmo de optimización clásica está diseñado para satisfacer completamente la demanda energética prevista para cada usuario, cumpliendo siempre el objetivo de carga fijado. Esto garantiza que el vehículo esté disponible con el nivel de carga requerido para cuando lo quiera utilizar el usuario, independientemente del perfil.

Respecto a la eficiencia (EUR/kWh), los valores se mantienen en un rango estrecho, entre 0,08 y 0,10 EUR/kWh, redondeados. El perfil `retired` consigue la mayor eficiencia (0,08 EUR/kWh), gracias a su alta disponibilidad durante casi todo el día, lo que le permite cargar en los tramos más económicos. El resto de perfiles presentan eficiencias muy similares, sin diferencias significativas.

En resumen, el algoritmo de optimización clásica es capaz de adaptarse a los distintos perfiles de usuario, optimizando el coste y la energía cargada según las restricciones y hábitos de cada uno, cumpliendo siempre con los objetivos de carga establecidos, de una manera rígida y predefinida.

Por último, se han registrado los tiempos de ejecución del algoritmo de optimización clásica, tanto para la resolución de cada perfil de usuario de forma individual como para el conjunto completo de perfiles. Los resultados obtenidos son los siguientes:

- **Tiempo acumulado resolviendo modelos individuales:** 0,5x2 segundos
- **Tiempo total de optimización:** 10,20 segundos

Estos valores reflejan la eficiencia computacional del enfoque clásico, permitiendo resolver el problema de optimización en tiempos muy reducidos incluso para múltiples perfiles.

4.4.2. Red Neuronal de Aprendizaje por Refuerzo

Para la implementación del gestor, se ha usado una red neuronal de aprendizaje por refuerzo DQN-LSTM, entrenada con los mismos datos que el algoritmo de optimización clásico, es decir, los perfiles de consumo energético generados por el LPG, y la información de la vivienda y del EV. En cierta forma, tal y como se ha explicado en el Capítulo 2, las redes neuronales pueden ser concebidas, en esencia, como un sistema aproximador de funciones. Por lo tanto, el diseño de la red neuronal irá muy en línea con la estructura del problema de optimización clásico, y las variables que se han definido anteriormente.

El modelo propuesto se estructura en varias capas secuenciales, diseñadas para procesar información temporal y generar una salida final. La arquitectura se compone de los siguientes bloques:

- **Capa LSTM**: Una capa tipo LSTM (`nn.LSTM`) que permite procesar secuencias de datos temporales, aprendiendo relaciones a lo largo del tiempo. Toma como entrada el tamaño de los vectores (`input_dim`), el tamaño del estado oculto (`hidden_dim`, por defecto 128) y el número de capas apiladas (`lstm_layers`, por defecto serán 2).

- **Normalización de capa**: Después de la LSTM, se aplica una normalización (`nn.LayerNorm`) para estabilizar el entrenamiento y mejorar la convergencia del modelo.

- **Primera capa totalmente conectada**: Una capa lineal (`nn.Linear`) transforma la salida de la LSTM, manteniendo la dimensión oculta. Actúa como parte del bloque de decisión del modelo.

- **Dropout**: Se introduce una capa de regularización (`nn.Dropout`) con probabilidad 0.2 para evitar sobreajuste, desactivando aleatoriamente algunas neuronas durante el entrenamiento.

- **Segunda capa totalmente conectada**: Otra capa lineal que sigue procesando la representación interna, manteniendo la dimensión.

- **Segunda normalización**: Se aplica una nueva normalización de capa para estabilizar las activaciones antes de la salida final.

- **Capa de salida**: Finalmente, una capa lineal proyecta la representación latente al tamaño deseado de la salida (`output_dim`), adaptada al problema.

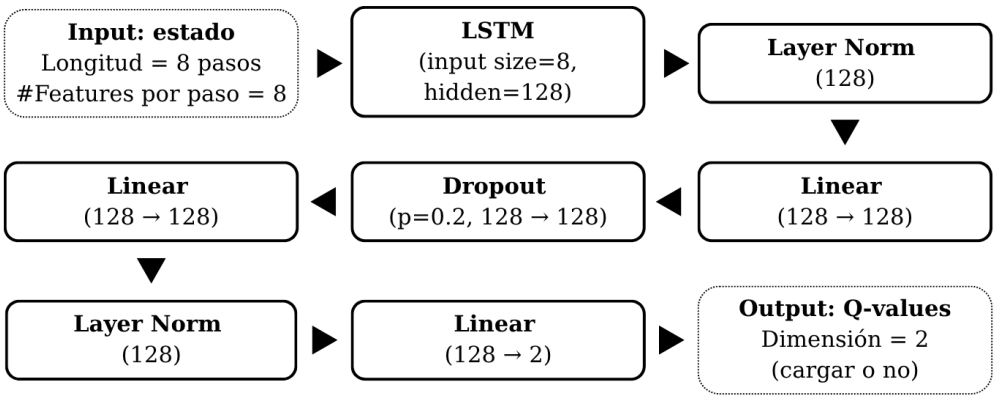

Figura 4.6: Arquitectura de la red neuronal DQN utilizada en el gestor de carga. Fuente: Elaboración propia.

Entorno de la Red DQN-LSTM

Para entrenar y evaluar el comportamiento del agente gestor, se ha desarrollado un entorno personalizado en Python [24]. Este entorno representa un hogar con un vehículo eléctrico, y tiene en cuenta tanto el consumo energético no gestionable como la disponibilidad del EV y el precio de la electricidad en cada intervalo de tiempo.

Estado del entorno. El vector de estado $\mathbf{s}_t \in \mathbb{R}^8$ en cada instante t incluye las siguientes variables:

- **State of charge (SOC):** estado de carga actual del EV, expresado en kWh.
- **SOC objetivo:** nivel de carga deseado, también en kWh.
- **Hora normalizada:** instante del día en formato continuo $[-1, 1]$.
- **Consumo no gestionable ($P_{NG}(t)$):** potencia usada por la vivienda que no se puede desplazar.
- **Disponibilidad ($A(t)$):** probabilidad de que el vehículo esté conectado en ese instante.
- **Precio de la electricidad:** coste por kWh en el instante actual.
- **Media del precio de la electricidad:** media del precio de la electricidad en el pasado reciente.

- **Desviación del precio de la electricidad:** diferencia entre el precio actual y su media reciente, indicando si está por encima o por debajo del precio habitual.

Acciones. La acción es binaria $a(t) \in \{0, 1\}$, indicando si se decide cargar (1) o no cargar (0) durante el intervalo actual de 15 minutos.

Dinámica. Dado un estado y una acción, se debe calcular el nuevo estado del entorno, que desencadena dicha acción. Esto significa que el entorno debe calcular:

- La potencia solicitada $P(t) = a(t) \cdot P_{\max}(t)$.
- La energía cargada, ajustada a las restricciones físicas y de disponibilidad.
- La actualización del SOC: $\Delta SOC = P \cdot \Delta t$, con $\Delta t = 0{,}25$ h.
- El coste de la carga en ese instante: $C(t) = P \cdot p(t)$.
- La recompensa obtenida, que depende de la acción tomada y del estado del entorno.

Función de recompensa. La función de recompensa implementada tiene como objetivo guiar al agente hacia un comportamiento eficiente, seguro y económicamente óptimo. Para ello, combina penalizaciones por violar restricciones físicas o económicas, y recompensas por decisiones adecuadas en función del contexto del entorno. Esta función es una traducción literal de las restricciones impuestas en el optimizador clásico visto anteriormente.

En primer lugar, se penaliza el coste económico de la carga en cada instante de tiempo, calculado como $C(t) = P \cdot p(t)$, donde P es la potencia aplicada y $p(t)$ el precio de la electricidad en el instante t. Esta penalización incentiva al agente a cargar en momentos más baratos.

Además, se penaliza al agente en tres casos clave:

- Cuando intenta cargar sin disponibilidad del vehículo ($A(t) = 0$), se reduce la recompensa con la energía desaprovechada.
- Cuando la potencia solicitada supera el margen disponible contratado con la red: $P > P_{\text{red}} - P_{NG}(t)$, en cuyo caso se corrige la acción y se aplica de nuevo una penalización con el exceso.

- Cuando el SOC supera el valor objetivo fijado, o si se excede la capacidad de la batería. En este caso, se penaliza con lo que habría sido el coste incurrido en esta carga superflua.

Por otro lado, se añaden incentivos positivos en los siguientes casos:

- Si el vehículo alcanza el SOC objetivo al final del día, se otorga una recompensa adicional muy significativa (+50) para fomentar la planificación de largo plazo.

- Si se carga cuando la disponibilidad es alta y el SOC aún no ha alcanzado el objetivo, se otorga una pequeña recompensa positiva para favorecer la acción oportuna.

- Toda energía cargada que se acerque al SOC deseado se premia con una recompensa proporcional.

Formato del episodio. Cada episodio simula un día completo dividido en intervalos de 15 minutos (96 pasos). El SOC se inicializa a un 70 % de la capacidad de la baterían al inicio del día, para representar un uso habitual diario de un EV, y se va actualizando conforme a las decisiones del agente.

En la figura 4.7 se muestra el diagrama general resumido del sistema de gestión de carga de EVs desarrollado.

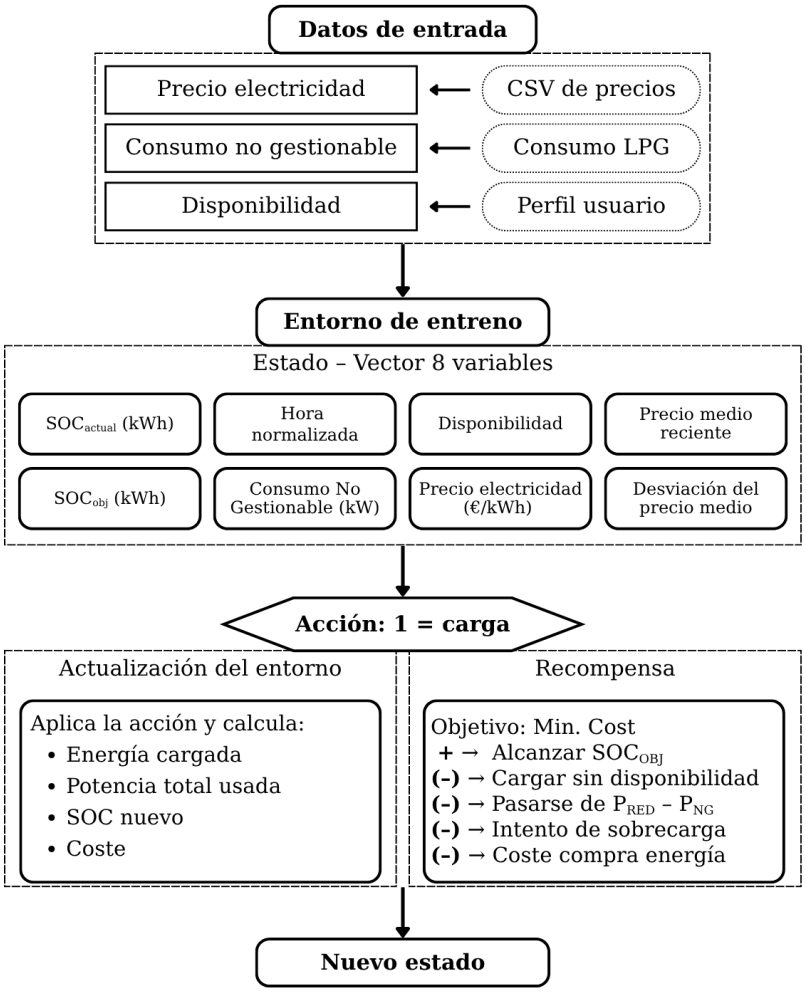

Figura 4.7: Diagrama general del sistema de gestión de carga de EVs desarrollado. Fuente: Elaboración propia.

Entrenamiento del Agente DQN-LSTM.

Durante el entrenamiento, el agente observa secuencias de estados $s(t-T), \ldots, s(t)$ para estimar la utilidad esperada de ejecutar cada acción $a \in \{0, 1\}$. La red neuronal devuelve dos valores, que serían la acción de cargar o no cargar el vehículo en el intervalo actual.

Representación temporal. Para aportar contexto al momento de la carga, la disponibilidad y precio, de manera que no sea un análisis puntual de la situación, se utiliza una secuencia de entrada temporal de longitud $T = 8$ pasos (2 horas) hacia el pasado, alimentada a través de la capa LSTM.

Parámetros del entrenamiento. El entrenamiento se realiza durante $N = 365$ episodios completos (días), con los siguientes hiperparámetros base:

- **Tasa de aprendizaje (α)**: 0.001
- **Factor de descuento (γ)**: 0.95
- **Exploración inicial (ϵ)**: 0.1, decreciendo progresivamente
- **Tamaño del batch**: 64
- **Tamaño de la secuencia temporal**: 8 pasos
- **Tamaño del buffer de experiencia**: 10000

Para la elección de los valores de estos hiperparámetros, se ha realizado un algoritmo de búsqueda en rejilla sobre un rango de valores, y se ha seleccionado el conjunto que mejor rendimiento ha obtenido en términos de recompensa media acumulada.

Además, se emplea una política ϵ-greedy, en la que el agente escoge acciones aleatorias con probabilidad ϵ y sigue la política aprendida con probabilidad $1 - \epsilon$. El valor inicial de ϵ se fija en 0.1, y se reduce progresivamente a lo largo de los episodios mediante una estrategia de decaimiento exponencial:

$$\epsilon_t = \epsilon_{\min} + (\epsilon_{\text{start}} - \epsilon_{\min}) \cdot e^{-\kappa t} \tag{4.8}$$

con $\epsilon_{\text{start}} = 0{,}1$, $\epsilon_{\min} = 0{,}05$, $\kappa = 0{,}995$ una constante de decaimiento, y t el número de episodios transcurridos. Con esta estrategia, se busca que el agente sea mucho más "curioso" al comenzar el entrenamiento, explorando más posibilidades en el espacio de acciones, y vaya convergiendo progresivamente hacia las mejores políticas que ha encontrado.

Algoritmo. Se emplea el algoritmo clásico de *Deep Q-Learning* con *replay buffer*, que incluye actualizaciones periódicas de la red objetivo y política ϵ-greedy. El error se calcula mediante pérdida MSE (*Mean Squared Error*, Error Cuadrático Medio) entre la estimación del valor actual y el objetivo de Bellman, tal y como se ha presentado en el Capítulo 2.

Monitorización y métricas. Durante el entrenamiento, se muestra una barra de progreso en terminal, y se registran métricas como recompensa promedio por episodio. También se exportan ficheros `.csv` de depuración con los valores de potencia aplicada, SOC, precio y recompensas en cada paso, permitiendo inspeccionar el comportamiento del agente en detalle.

Resultados del Agente DQN-LSTM.

Los resultados obtenidos por el agente gestor, después del entrenamiento, se muestran a continuación.

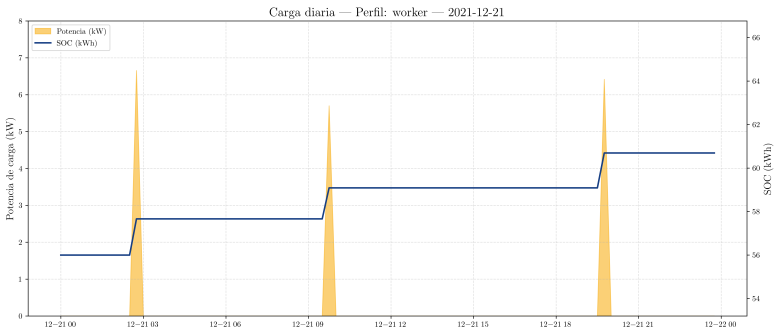

Figura 4.8: Resultados del agente DQN-LSTM para el perfil `worker`: potencia aplicada y evolución del SOC en un día. Fuente: Elaboración propia.

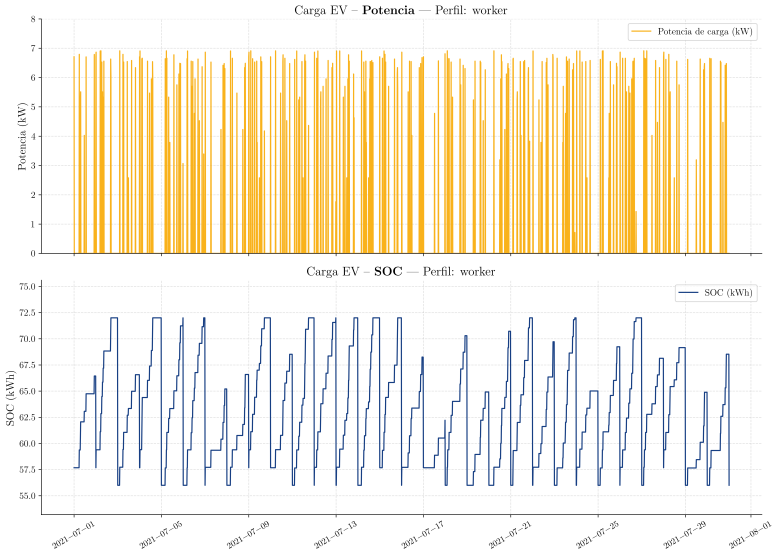

Figura 4.9: Resultados del agente DQN-LSTM para el perfil `worker`: potencia aplicada y evolución del SOC a lo largo de un mes. Fuente: Elaboración propia.

Al igual que en el caso del optimizador clásico, la Figura 4.8 muestra el comportamiento diario del agente DQN-LSTM para el perfil `worker`, donde se observa la evolución de la potencia aplicada al cargador y el estado de carga (SOC) del vehículo eléctrico a lo largo de un día. Por otro lado, la Figura 4.9 presenta estas mismas variables a lo largo de un mes completo, permitiendo analizar el comportamiento

del agente en un horizonte temporal más amplio.

En el análisis diario, se aprecia cómo el agente DQN-LSTM es capaz de identificar los periodos más favorables para la carga, concentrando la mayor parte de la energía suministrada durante la noche, cuando la disponibilidad del vehículo es alta y el precio de la electricidad es más bajo. Además, el agente ajusta la potencia de carga en función de las restricciones del entorno, como la potencia máxima contratada o el consumo no gestionable de la vivienda, moderando la potencia entregada en los momentos de mayor demanda doméstica.

A lo largo del mes, el comportamiento del agente se mantiene consistente, mostrando una tendencia a cargar el vehículo en los intervalos más económicos y asegurando que el SOC alcance el objetivo diario en la mayoría de los días. El agente demuestra capacidad de adaptación ante la variabilidad de la disponibilidad y el precio de la electricidad, optimizando el coste de la carga y respetando las restricciones impuestas.

Sin embargo, aunque se hayan impuesto incentivos y restricciones para ello, la red no consigue cargar la batería del EV al SOC objetivo todos los días. EN estos casos, relativamente poco comunes, en los que el agente no consigue este objetivo, la diferencia suele ser razonablemente pequeña. El SOC objetivo definido para el problema es el 90 % de una batería de 80kWh, es decir, 72 kWh. En la Figura 4.9 se observa que, en la mayoría de los días, el SOC del vehículo suele alcanzar dicha cifra, y los días que no lo consigue suele ser por 6 o 7 kWh.

De nuevo, como con el optimizador clásico, se muestra únicamente el perfil `worker` como ejemplo más representativo, ya que sería el perfil de consumo energético más habitual.

Finalmente, se han extraído métricas y resultados adicionales obtenidos tras el entrenamiento y pruebas del agente gestor, para evaluar su rendimiento en la gestión de la carga del vehículo léctrico. Estos resultados se muestran a continuación:

Perfiles	Coste medio (EUR)	Energía media (kWh)	Eficiencia media (EUR/kWh)
worker	1,21	12,90	0,09
flexible	1,17	12,68	0,09
retired	1,22	13,46	0,09
traveller	1,17	12,84	0,09
night_owl	1,16	12,66	0,09

Tabla 4.6: Resumen de métricas diarias por perfil para el agente DQN-LSTM.

A continuación se presenta un análisis diferenciado por perfil. Para el perfil **worker**, el coste medio diario es de 1,21 EUR, con una energía media cargada de 12,90 kWh. El agente tiende a priorizar la carga en los periodos nocturnos, adaptándose a la menor disponibilidad diurna y optimizando el coste. La eficiencia se mantiene en 0,09 EUR/kWh, mostrando un buen equilibrio entre coste y energía suministrada.

En el caso del perfil **flexible**, se observa el menor coste medio (1,17 EUR) y una energía media de 12,68 kWh. La mayor flexibilidad horaria permite al agente aprovechar mejor los tramos de menor precio, lo que se traduce en un coste reducido y una eficiencia óptima.

Para el perfil **retired**, es destacable que presenta la mayor energía media cargada (13,46 kWh) y un coste medio de 1,22 EUR. La alta disponibilidad durante todo el día permite al agente distribuir la carga de forma más uniforme, aunque el coste es ligeramente superior debido a la mayor cantidad de energía suministrada.

En cuanto al perfil **traveller**, con un coste medio de 1,17 EUR y una energía media de 12,84 kWh, el agente adapta la estrategia a los periodos de menor disponibilidad diurna, concentrando la carga en los intervalos más favorables y manteniendo la eficiencia.

Finalmente, el perfil **night_owl** registra el menor coste medio (1,16 EUR) y una energía media de 12,66 kWh. El agente aprovecha la alta disponibilidad nocturna para cargar el vehículo en los momentos de menor precio, lo que se refleja en la eficiencia y el bajo coste.

Por otro lado, la eficiencia constante en todos los perfiles indica que el agente es capaz de identificar y aprovechar los periodos de menor precio de la electricidad, optimizando la relación entre coste y energía suministrada. En conjunto, estos

resultados evidencian el potencial de los métodos basados en aprendizaje por refuerzo para la gestión inteligente y personalizada de la carga de EVs en entornos domésticos.

El rendimiento computacional de la red también ha sido registrado durante el entrenamiento y las pruebas, y los tiempos obtenidos son los siguientes:

Entorno de ejecución. Las pruebas se han realizado en un MacBook Pro 14"(2024) con Apple M4 Pro (12 núcleos CPU a 4.05 GHz, 18 núcleos GPU), 24 GB de memoria RAM unificada LPDDR5X a 7500 MT/s y SSD NVMe PCIe 4.0 de 1 TB. El sistema operativo es *macOS Sequoia 15.5*. El entorno Python utilizado es 3.11.8, con las siguientes versiones de librerías principales: PyTorch 2.3.0, NumPy 1.26.4, Pyomo 6.7.1 y Gurobi 11.0.1. El entrenamiento y la inferencia se han realizado en CPU.

A continuación se presentan los tiempos de entrenamiento e inferencia obtenidos para cada perfil durante el entrenamiento y pruebas del agente DQN-LSTM:

Perfil	**Tiempo de entrenamiento (s)**	**Tiempo de inferencia (s)**
worker	0.12	0.0001
flexible	0.11	0.0001
retired	0.10	0.0001
traveller	0.13	0.0001
night_owl	0.12	0.0001

Tabla 4.7: Tiempos de entrenamiento e inferencia del agente DQN-LSTM por perfil.

El tiempo medio de entrenamiento por perfil es relativamente bajo, 4.17 minutos por perfil, lo que indica que el agente DQN-LSTM es capaz de aprender de forma eficiente a partir de los datos recibidos. Además, y más importante aún, el tiempo medio de inferencia es muy bajo, con una media de 0.252 milisegundos por perfil, alcanzando un mínimo de tan solo 174 milisegundos de media si el perfil es `worker`. Esto indica que el agente es capaz de tomar decisiones en tiempo real, lo que es esencial para la gestión eficiente de la carga del EV.

Además, con unos tiempos de inferencia tan reducidos, una posible implementación en hardware de este gestor no requeriría de mucha potencia computacional, sino tan solo para el entrenamiento.

Capítulo 5

Resultados

En el capítulo anterior se ha visto cómo se desenvolvían los distintos algoritmos de optimización intentando resolver el problema planteado en el trabajo. A lo largo de este capítulo, se va a hacer un análisis conjunto de los resultados obtenidos por separado para cada uno de los algoritmos. Se hará especial hincapié en tres comparaciones fundamentales para el proyecto: coste, energía y rendimiento. Adicionalmente, como derivado natural de la evaluación de los costes y el consumo energético, se analizará también la eficiencia de cada uno de los algoritmos.

Para facilitar la comparación entre la optimización clásica y el agente DQN-LSTM, en cada sección se presentarán los resultados de ambos métodos en paralelo, utilizando tablas comparativas. De este modo, se puede observar de forma directa las diferencias y ventajas de cada enfoque para cada perfil analizado. Además, se puede calcular el porcentaje de diferencia (mejora o empeoramiento) de DQN-LSTM respecto a la optimización clásica para cada métrica relevante, lo que permite cuantificar el impacto de la inteligencia artificial frente a los métodos tradicionales.

5.1. Comparativa de Costes

La primera comparativa que se va a realizar es la de los costes medios diarios por perfil, tanto para la optimización clásica como para el agente DQN-LSTM. Para ello, se muestran en la Tabla 5.1 los costes medios diarios para cada perfil:

A partir de los resultados presentados en la Tabla 5.1, se observa de manera clara que el agente DQN-LSTM supera consistentemente a la optimización clásica en todos los perfiles analizados. En términos de coste medio diario, la reducción obtenida por el agente DQN-LSTM oscila entre el 4,92 % (fuera de lo normal) y el 32,23 %, con una media del 22,52 %.

Perfil	**Clásica** (EUR)	**DQN-LSTM** (EUR)	**Cambio(%)**
worker	1,60	1,21	32,23
flexible	1,44	1,17	23,08
retired	1,28	1,22	4,92
night_owl	1,49	1,16	28,45
traveller	1,45	1,17	23,93
Media	**1,452**	**1,19**	**22,52**

Tabla 5.1: Comparativa de costes medios diarios por perfil entre la optimización clásica y el agente DQN-LSTM, junto con el porcentaje de reducción de coste.

Sin embargo, es importante matizar estos resultados desde una perspectiva crítica. La reducción de coste y energía por parte del agente DQN-LSTM no implica necesariamente un mejor comportamiento en la carga de EVs. De hecho, el agente DQN-LSTM tiende a no satisfacer completamente la demanda de carga del vehículo, lo que puede llevar a una menor satisfacción del usuario. Por tanto, aunque el agente logra un coste total más bajo, esto se debe en parte a que no cubre la carga requerida, lo que limita la efectividad real de la mejora observada. Adicionalmente, como se verá en los siguientes apartados de la comparación, esta tendencia que presenta el agente se traduce matemáticamente en unos resultados numéricos que pueden llevar a error.

En contraparte, la optimización clásica siempre consigue satisfacer la idea del usuario para la carga de su EV, lo que implica que, aunque el coste sea mayor en principio, la satisfacción del usuario es más alta. Por tanto, la reducción de coste y energía no debe interpretarse automáticamente como una mejora en la eficiencia de carga, sino como una consecuencia de no cubrir completamente la demanda de carga.

De todas maneras, mirando por perfiles, la reducción de coste es más pronunciada en los perfiles *worker*, *flexible* y *night_owl*, donde se superan reducciones del 20 %. En el perfil *retired*, la diferencia es menor, con una reducción del 4,92 %, lo que indica que el agente DQN-LSTM no logra una mejora significativa en este caso. Esto puede deberse a que el perfil *retired* tiene una demanda de carga más baja y menos restricciones, lo que limita el potencial de optimización del agente.

En resumen, aunque el agente DQN-LSTM consiga una reducción en los costes medios diarios incurridos por la carga de EVs, es crucial considerar que esta mejora se produce a costa de no necesariamente satisfacer siempre el objetivo ideal de

carga del vehículo.

5.2. Comparativa de Energía

De la misma manera que con el coste, se va a realizar una comparativa de la energía consumida por cada uno de los algoritmos. Para ello, se muestran en la Tabla 5.2 los consumos medios diarios para cada perfil, diferenciando entre la optimización clásica y el agente DQN-LSTM.

Perfil	**Clásica** (kWh)	**DQN-LSTM** (kWh)	**Cambio(%)**
retired	16,00	13,46	18,87
night_owl	16,00	12,66	26,38
flexible	16,00	12,68	26,18
traveller	16,00	12,84	24,61
worker	16,00	12,90	24,03
Media	**16**	**12,91**	**23,93**

Tabla 5.2: Comparativa de energía media diaria por perfil entre la optimización clásica y el agente DQN-LSTM, junto con el porcentaje de reducción de consumo energético.

En la comparativa de energía (Tabla 5.2) se observa que el agente DQN-LSTM reduce notablemente el consumo energético medio diario respecto a la optimización clásica en todos los perfiles, con una media de reducción cercana al 24 %. Por perfiles, la reducción es bastante homogénea: oscila entre el 18,87 % en el perfil *retired* y el 26,38 % en *night_owl*.

Como se introdujo en la comparativa de costes, es importante recordar el comportamiento del agente a lo largo de estas comparativas. Mientras que el optimizador clásico está obligado a cargar la cantidad de energía suficiente para alcanzar todos los días el SOC objetivo, el agente no. Es por ello que usa una menor cantidad de energía en todos los perfiles. Con un comportamiento ideal de un agente perfecto, el diferencial de energía usada por los dos métodos debería ser infinetesimal, ya que, tal y como se explicó en el marco teórico del trabajo, el agente no es más que un algoritmo que trata de aproximar la misma funcion que el optimizador, con otras características en el *approach* .

5.3. Comparativa de Eficiencia

Siguiendo la misma línea que en las comparativas anteriores, se va a realizar una comparativa de la eficiencia de cada uno de los algoritmos. Para ello, se muestran en la Tabla 5.3 las eficiencias medias diarias para cada perfil, diferenciando entre la optimización clásica y el agente DQN-LSTM. La eficiencia se calcula como el cociente entre la energía consumida y el coste, es decir, Eficiencia = $\frac{\text{Energía}}{\text{Coste}}$.

Perfil	**Clásica** (EUR/kWh)	**DQN-LSTM** (EUR/kWh)	**Cambio**
worker	0,10	0,09	0,00
flexible	0,09	0,09	0,00
retired	0,08	0,09	0.01
night_owl	0,09	0,09	0,00
traveller	0,09	0,09	0,00
Media	**0,09**	**0,09**	**0,00**

Tabla 5.3: Comparativa de la eficiencia diaria (kWh/EUR) por perfil entre la optimización clásica y el agente DQN-LSTM, junto con el porcentaje de mejora en eficiencia.

Finalmente, en la comparativa de eficiencia (Tabla 5.3) se observa que la eficiencia media diaria es prácticamente idéntica entre la optimización clásica y el agente DQN-LSTM, con una media de 0,09 EUR/kWh para ambos métodos. Esto indica que, a pesar de las diferencias en coste y energía consumida, ambos enfoques logran una eficiencia similar en términos de costo por unidad de energía.

En caso de que el agente se comportara en la misma línea que el optimizador, es muy probable que experienciara una ligera modificación a estos parámetros de eficiencia, ya que, tal y como se comporta ahora, prefiere dejarse un pequeño margen sin cargar a incurrir en más coste. En resumen, la eficiencia de ambos métodos es comparable, lo que sugiere que, aunque el agente DQN-LSTM logra reducir los costes y el consumo energético, no lo hace a expensas de una menor eficiencia en términos de coste por unidad de energía.

5.4. Comparativa de Rendimiento

En esta sección se analiza el rendimiento computacional de ambos enfoques, considerando tanto el tiempo de optimización global como la velocidad de respuesta

en la toma de decisiones.

- **Optimización clásica:** El tiempo total de optimización para todos los perfiles es de 10.2 segundos en total. Cada instancia individual se resuelve en unos 9.1 ms por perfil, lo que permite obtener soluciones prácticamente instantáneas. Este comportamiento es característico de algoritmos clásicos bien ajustados y con problemas de tamaño moderado, donde la estructura matemática permite una resolución eficiente y determinista.
- **Agente DQN-LSTM:** El tiempo de entrenamiento inicial es considerablemente mayor, situándose en torno a los 4 minutos por perfil. Sin embargo, una vez completado el entrenamiento, el tiempo de inferencia para resolver una instancia individual es extremadamente bajo, del orden de 0.16 ms por perfil. Esto supone una mejora de más de 50 veces respecto a la optimización clásica en la fase de inferencia, permitiendo respuestas prácticamente instantáneas incluso en escenarios de alta demanda o con restricciones de tiempo real.

Estos resultados ponen de manifiesto varias diferencias clave:

- **Coste de entrada vs. coste de operación:** La optimización clásica requiere poco tiempo de cálculo inicial y es ideal para escenarios donde las condiciones del problema no cambian frecuentemente. Sin embargo, si el entorno es dinámico y se generan nuevas instancias de manera continua, el coste de resolver cada una puede acumularse rápidamente.
- **Escalabilidad y adaptabilidad:** El agente DQN-LSTM, aunque requiere un entrenamiento inicial costoso, ofrece una escalabilidad superior en la fase de operación. Una vez entrenado, puede adaptarse a nuevas instancias en tiempo real con un coste computacional mínimo. Esto es especialmente ventajoso en aplicaciones donde la rapidez de respuesta es crítica, como sistemas de control en tiempo real o gestión dinámica de recursos.
- **Flexibilidad ante cambios:** Si el entorno o los parámetros del problema cambian significativamente, la optimización clásica puede requerir una reoptimización completa, mientras que el agente DQN-LSTM puede, en muchos casos, adaptarse mediante un reentrenamiento incremental o incluso transferir conocimiento aprendido a nuevas situaciones, reduciendo el tiempo de adaptación.
- **Consumo de recursos:** El entrenamiento del agente DQN-LSTM puede requerir hardware especializado (GPU), mientras que la optimización clásica suele ser menos exigente en este aspecto. Sin embargo, en la fase de inferencia, ambos métodos pueden operar eficientemente en hardware convencional.

5.5. Conclusiones de la Comparativa

Cerrando este apartado, se puede conluir que la optimización clásica consigue resolver el problema planteado en el trabajo de forma óptima y con gran rapidez, cumpliendo siempre con las restricciones y requisitos impuestos por el usuario o cliente.

Por otro lado, el gestor inteligente de carga deja algo que desear en cuanto a su comportamiento general se refiere, dejando de satisfacer la demanda ideal de carga en algunos casos. Sin embargo, aunque se deban tomar estos resultados con cautela, el agente DQN-LSTM consigue—en principio— reducir los costes todos los perfiles analizados. El punto fuerte del agente es su rapidez computacional y el mínimo tiemp de inferencia que presenta, permitiendo una respuesta casi instantánea a las decisiones de carga. Esto lo convierte en una herramienta que, de ser finamente refinada, podría obtener resultados finales muy competitivos con la optimización clásica, además de ser mucho más veloz y, en la ejecución, más eficiente económicamente.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este trabajo se ha presentado un sistema de predicción y gestión de la demanda energética para la carga de vehículos eléctricos, utilizando un enfoque de modelado híbrido que combina técnicas de aprendizaje automático y aprendizaje por refuerzo con memoria, basado en IA generativa.

Tanto el optimizador clásico como el sistema de IA generativa han resultado eficaces para gestionar la carga de vehículos eléctricos bajo las restricciones planteadas. El enfoque clásico, mediante programación lineal, ha ofrecido soluciones óptimas en coste y eficiencia, respetando tanto los límites técnicos como las preferencias del usuario.

También se ha logrado generar perfiles sintéticos de demanda y disponibilidad de energía con la ayuda de técnicas de IA Generativa, lo que ha permitido simular escenarios realistas para la carga de vehículos eléctricos.

Finalmente, tras un análisis comparativo de los resultados obtenidos por ambos enfoques, se ha concluido que, el optimizador clásico sigue presentando las mejores soluciones para la carga de vehículos eléctricos, en el marco que explora este trabajo, pero presentan mucho margen de mejora computacional, y de adaptación al usuario. Aquí es donde el agente inteligente se desmarca, aunque su comportamiento es subóptimo, obviando alguna preferencia importante del usuario en pro de ahorrar costes.

6.2. Próximos Pasos

Aunque el sistema desarrollado ha demostrado ser eficaz y cumple con los objetivos planteados, existen varias áreas de mejora y expansión que podrían explorarse en el futuro:

- **Introducción de algoritmos de clustering:** Incorporar técnicas como *k-means* [28] o *k-nearest neighbors* [29] para identificar y clasificar perfiles de demanda y patrones de carga, permitiendo una gestión más personalizada y eficiente de los recursos energéticos.
- **Mejora de la política de control mediante métodos Actor-Critic:** Implementar enfoques avanzados de aprendizaje por refuerzo, como métodos Actor-Critic o políticas de gradiente, que permitan asignar recompensas suaves (*soft rewards*) en función de la dirección y calidad de las acciones tomadas por el sistema, facilitando un aprendizaje más fino y adaptativo.
- **Capacidades predictivas avanzadas:** Integrar modelos de predicción para anticipar la demanda energética y la disponibilidad de recursos, mejorando la planificación y la toma de decisiones en tiempo real.
- **Ampliación de la memoria y el *replay buffer*:** Aumentar la capacidad de memoria del sistema, permitiendo el acceso a experiencias pasadas relevantes, como datos de fechas o situaciones similares de años anteriores, para enriquecer el aprendizaje y la toma de decisiones.
- **Integración de energías renovables en el consumo no gestionable:** Considerar la aportación de fuentes renovables en la parte de la demanda no gestionable, incrementando la sostenibilidad y la eficiencia global del sistema.
- **Automatización del aprendizaje de los últimos elementos del vector de estado:** Permitir que la red neuronal aprenda de manera autónoma la relevancia y el uso de los últimos elementos del vector de estado, en lugar de definirlos manualmente, proporcionando si acaso únicamente la predicción de los precios del mercado.
- **Flexibilización de la dimensión de salida del modelo:** Adaptar la arquitectura de la red para que pueda gestionar dinámicamente diferentes números de acciones, permitiendo así la futura incorporación de nuevas acciones sin necesidad de rediseñar la salida.

Bibliografía

[1] R. Yao y K. Steemers. «A method of formulating energy load profile for domestic buildings in the UK». En: *Energy and Buildings* 37.6 (2005), págs. 663-671. DOI: 10.1016/j.enbuild.2004.09.007.

[2] Google Trends. *Interés de búsqueda por "vehículo eléctrico" en España (2015-2025)*. Accedido el 3 de junio de 2025. 2025. URL: https://trends.google.com.

[3] Vala Khorasani. *Electric Vehicle Charging Patterns*. https://www.kaggle.com/datasets/valakhorasani/electric-vehicle-charging-patterns. Accessed: 2025-06-04. 2023.

[4] Noah Pflugradt. «Load Profile Generator: Modeling of Synthetical Load Profiles». Tesis doct. Bern University of Applied Sciences, 2020. URL: https://www.loadprofilegenerator.de/.

[5] Noah Pflugradt. *Load Profile Generator – Screenshots*. https://www.loadprofilegenerator.de/screenshots/. Capturas de pantalla accedidas el 4 de junio de 2025. 2025.

[6] Stephen J. Wright. *optimization*. Accessed: 2025-06-04. Encyclopedia Britannica. Abr. de 2025. URL: https://www.britannica.com/science/optimization.

[7] Wikipedia contributors. *Optimización (matemática)*. Wikipedia, la enciclopedia libre. Consultado el 4 de junio de 2025. 2025. URL: https://es.wikipedia.org/wiki/Optimizacion_(matematica).

[8] Stefan Funke. «Discrete Optimization – Lecture Notes». Course material, University of Stuttgart. Stuttgart, Germany, 2024.

[9] Richard S. Sutton y Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2.ª ed. MIT Press, 2018. URL: https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf.

[10] Chloe E. Wang. *A Look into Neural Networks and Deep Reinforcement Learning*. Accessed: 2025-06-04. 2022. URL: https://chloeewang.medium.com/a-look-into-neural-networks-and-deep-reinforcement-learning-2d5a9baef3e3.

[11] Satyaki Saha. *A Brief Explanation of State-Action Value Function (Q) in RL*. Accessed: 2025-06-04. 2020. URL: https://medium.com/intro-to-artificial-intelligence/a-brief-explanation-of-state-action-value-function-q-in-rl-ed010ca7d69b.

[12] DeepMind. *DeepMind – Solving intelligence to advance science and benefit humanity*. Accessed: 2025-06-04. 2025. URL: https://deepmind.google/.

[13] DeepMind. *AlphaZero: Shedding new light on chess, shogi, and Go*. DeepMind Blog, accessed 2025-06-04. 2017. URL: https://deepmind.google/discover/blog/alphazero-shedding-new-light-on-chess-shogi-and-go.

[14] Shruti Dhumne. *Deep Q-Network (DQN)*. Accessed: 2025-06-04. 2019. URL: https://medium.com/@shruti.dhumne/deep-q-network-dqn-90e1a8799871.

[15] Samina Amin. *Deep Q-Learning (DQN)*. Accessed: 2025-06-04. 2020. URL: https://medium.com/@samina.amin/deep-q-learning-dqn-71c109586bae.

[16] Christopher J. C. H. Watkins y Peter Dayan. «Q-learning». En: *Machine Learning* 8.3–4 (1992), págs. 279-292. DOI: 10.1007/BF00992698. URL: https://link.springer.com/article/10.1007/BF00992698.

[17] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. Inf. téc. arXiv:1312.5602. Accessed: 2025-06-04. DeepMind Technologies, 2013. URL: https://arxiv.org/abs/1312.5602.

[18] Andreas Bulling. *Machine Perception and Learning – Lecture 4: Recurrent Neural Networks (RNNs)*. Lecture slides, University of Stuttgart, Winter Semester 2024/2025. 2024. URL: https://www.collaborative-ai.org/.

[19] Ottavio Calzone. *An Intuitive Explanation of LSTM*. Accessed: 2025-06-04. 2020. URL: https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c.

[20] Z. Xiao. «Generative Adversarial Network and Its Application in Energy Internet». En: *Mathematical Problems in Engineering* 2022 (2022), págs. 1-7. DOI: 10.1155/2022/9985522.

[21] A. Moradzadeh et al. «Short-term electricity demand forecasting via variational autoencoders and batch training-based bidirectional long short-term memory». En: *Sustainable Energy Technologies and Assessments* 52 (2022), pág. 102209. DOI: 10.1016/j.seta.2022.102209.

[22] Lilian Weng. *What are Diffusion Models?* Accessed: 2024-10-16. 2021. URL: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/.

[23] Kadri Umay. *Use Cases for Foundation Models (aka LLMs) in the Energy Industry*. Accessed: 2024-10-14. 2024. URL: https://www.linkedin.com/pulse/use-cases-foundation-models-aka-llms-energy-industry-kadri-umay/.

[24] Python Software Foundation. *Lenguaje de Programación Python*. 2024. URL: https://www.python.org.

[25] Pyomo Developers. *Pyomo Optimization Modeling in Python*. Version 6.7.0, accessed 2024-06-04. 2024. URL: https://pyomo.org/.

[26] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2024. URL: https://www.gurobi.com.

[27] A. Paszke, S. Gross, F. Massa et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». En: *Advances in Neural Information Processing Systems*. 2019, págs. 8024-8035.

[28] Stuart P. Lloyd. «Least squares quantization in PCM». En: *IEEE Transactions on Information Theory* 28.2 (1982), págs. 129-137. DOI: 10.1109/TIT.1982.1056489.

[29] Naomi S. Altman. «An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression». En: *The American Statistician* 46.3 (1992), págs. 175-185. DOI: 10.1080/00031305.1992.10475879.

[30] Naciones Unidas. *Objetivos de Desarrollo Sostenible (ODS)*. https://sdgs.un.org/goals. Consultado en junio de 2025. 2015. URL: https://sdgs.un.org/goals.

Anexo I. Alineación con los Objetivos de Desarrollo Sostenible

Introducción

Este proyecto se sitúa en el marco de la transición energética y la búsqueda de soluciones más eficientes para la gestión de la demanda doméstica. Desde esa perspectiva, se identifican varios Objetivos de Desarrollo Sostenible (ODS) con los que guarda alineación directa o indirecta.

Los ODS fueron establecidos por la Asamblea General de las Naciones Unidas como parte de la Agenda 2030 para el Desarrollo Sostenible, y constituyen un marco de referencia internacional para abordar los principales desafíos sociales, económicos y ambientales del planeta [30].

El presente trabajo contribuye principalmente a los objetivos relacionados con energía limpia, consumo responsable y acción climática, integrando herramientas de inteligencia artificial para una toma de decisiones más eficiente y sostenible en el ámbito energético.

ODS principales

ODS 1. ODS 7 – Energía asequible y no contaminante
El sistema de gestión inteligente de carga desarrollado en este trabajo contribuye a un uso más eficiente de la energía eléctrica, reduciendo el coste y mejorando la integración de fuentes renovables a través de decisiones optimizadas.

ODS 2. ODS 12 – Producción y consumo responsables
El uso de perfiles personalizados de disponibilidad y demanda, junto con

técnicas de optimización y aprendizaje automático, favorece patrones de consumo más ajustados a las necesidades reales y al contexto energético, contribuyendo a un consumo más racional y eficiente.

ODS complementario

ODS 13 – Acción por el clima
De forma indirecta, la reducción del consumo eléctrico innecesario y la mejora de la eficiencia en la carga de vehículos eléctricos ayudan a disminuir las emisiones asociadas a la generación energética, especialmente en horarios de alta demanda.

Conclusión

Aunque el proyecto tiene un enfoque técnico y experimental, sus aplicaciones están alineadas con los esfuerzos globales por hacer un uso más sostenible de los recursos energéticos, fomentar la eficiencia y reducir el impacto ambiental de las nuevas tecnologías de movilidad.