Grado en Ingeniería en Tecnologías de Telecomunicación

Trabajo fin de grado

IA GENERATIVA PARA LA PREDICCIÓN Y GESTIÓN DE LA DEMANDA ENERGÉTICA

Autor
Álvaro González Tabernero

Directores
Francisco Martín Martínez
Jaime Boal Martín-Larrauri

Madrid
Junio 2025

Resumen

Abstract content

Agradecimientos

Índice general

Índice de figuras

Índice de cuadros

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 cotidianaLa 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 vehículos eléctricos ha aumentado considerablemente, con una creciente infraestructura de carga y un interés cada vez mayor por parte de los consumidores.

La carga de estos vehículos eléctricos 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 útlimos diez años

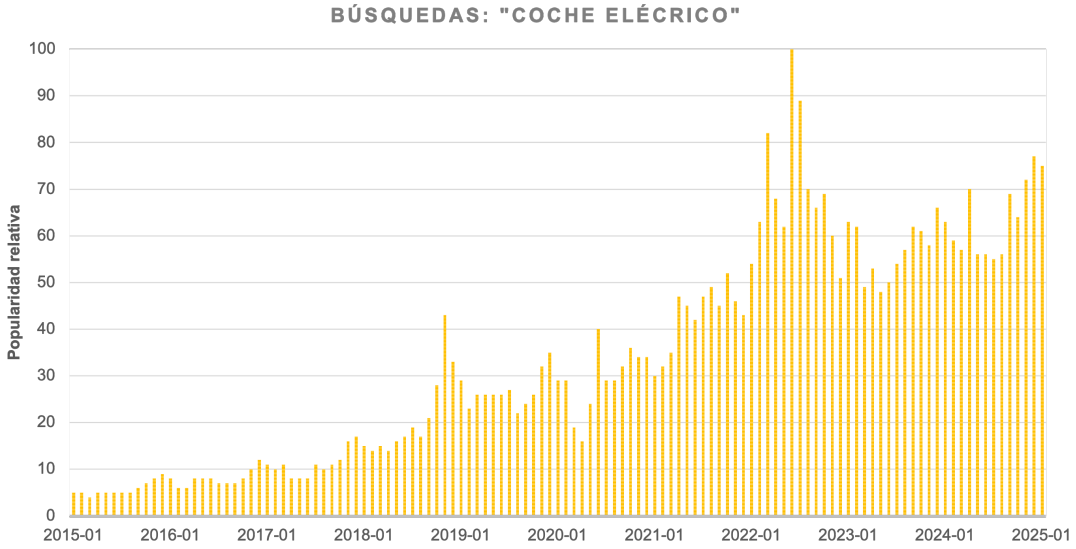

Figura 1.1: Evolución del interés por los vehículos eléctricos en España (2015-2025). Fuente: Google Trends [2]

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 vehículos eléctricos es un aspecto crítico para garantizar que la transición hacia un sistema de transporte sostenible no comprometa la estabilidad de la red eléctrica.

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 más eficiente y eficaz que los métodos tradicionales.

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 para aprender y adaptarse a nuevas condiciones y entornos, la convierte en una herramienta valiosa 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 vehículos eléctricos.

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.

1.2. Objetivos

El objetivo del proyecto es desarrollar in sistema gestor inteligente para la carga de EVs en un contexto dómestico. Para ello, se hará uso de técnicas de IA Generativa, RLNN (Reinforcement Learning Neural Network) 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.

Para lograr este objetivo, se han definido unos objetivos secundarios específicos. El primero tiene que ver con toda la parte generativa del proyecto: la creación de perfiles sintéticos de demanda no gestionable, disponibilidad y requisitos del EV. En segundo lugar,se realizará una optimización clásica, tratando el problema de la carga como un pograma lineal (Linear Programme, LP), y se resolverá como tal. Con esta optimización clásica, que garantiza un resultado óptimo según las restricciones que se le impongan en forma de ecuación, se compararán los resultados de ambos métodos.

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

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 son las dos siguientes:

- **Electric Vehicle Charging Patterns Dataset**
- **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

Cuadro 2.1: Pequeña 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. De manera que los datos que se han generado y se usan en este proyecto, tienen una resolución interna de un minuto, y una resolución externa de 15 minutos.

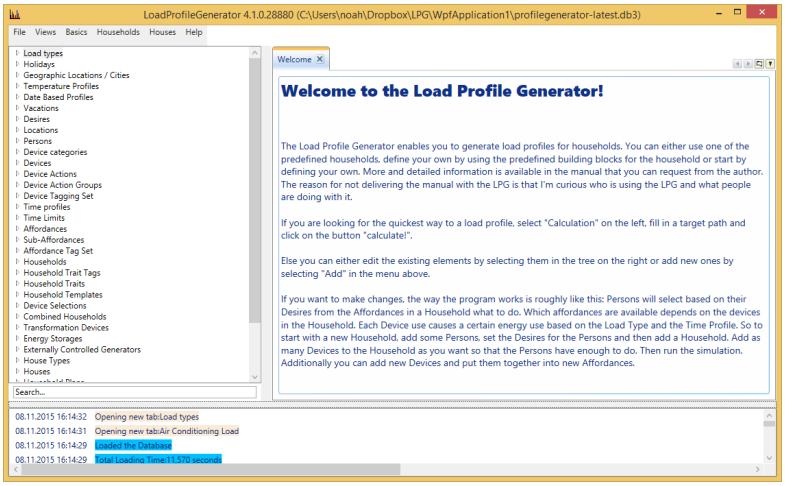

Figura 2.2: Capturas de pantalla de LPG, página principal. Fuente: *LoadProfileGenerator* [5].

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**
- **Programación No Lineal**

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

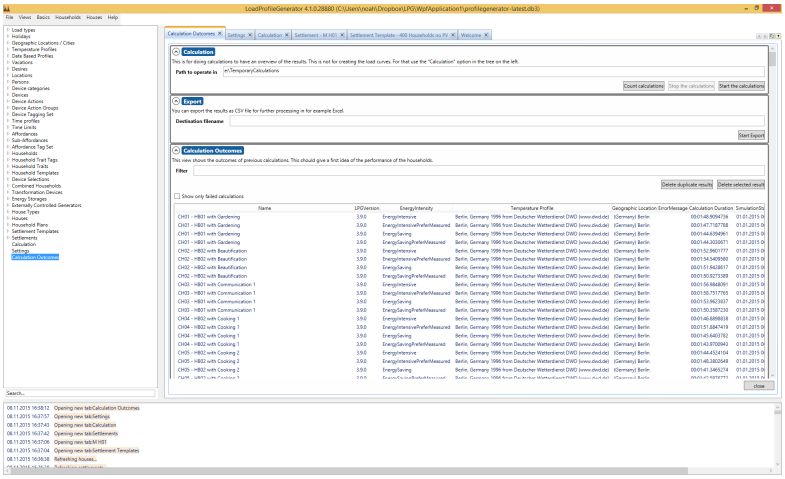

Figura 2.3: Capturas de pantalla de LPG, obtención de resultados. Fuente: *LoadProfileGenerator* [5].

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 fuerza que, al menos una de las variables, esté forzada a ser 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].

2.2.3. Programación No Lineal

Como dice su nombre, la programación no lineal es el caso en el que la función objetivo y/o las restricciones no son lineales [7]. Este tipo de programación es más complejo que la programación lineal, ya que las funciones no lineales pueden tener múltiples mínimos o máximos locales, lo que dificulta la búsqueda de la solución óptima. Además, la programación no lineal puede involucrar todo tipo de funciones: cuadráticas, exponenciales, logarítmicas o trigonométricas, entre otras [8].

2.3. Aprendizaje por Refuerzo (RL)

Stutton y Barto definen el Aprendizaje por Refuerzo (Reinforcement Learning, RL) como un .[en]foque 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 çuriosidad"del agente por encontrar nuevas y diferentes estrategias, mientras que explotación se refiere a la tendencia del agente a explotar las estrategias que ya concoe, y sabe que otorgan resultados positivos. Encontrar un equilibrio balanceado entre estas dos características es tremendamente importante para conseguir el aprendizaje posible del agente [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.
- **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.
- **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.
- **Estado (S)**: El contexto instantáneo en que se encuentra el agente en cualquier momento.
- **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.
- **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.

- **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.

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 y/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 2.4: Logo de DeepMind. Fuente: DeepMind [11].

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 [12].

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 omtimizadas en entornos muy complejos. [13].

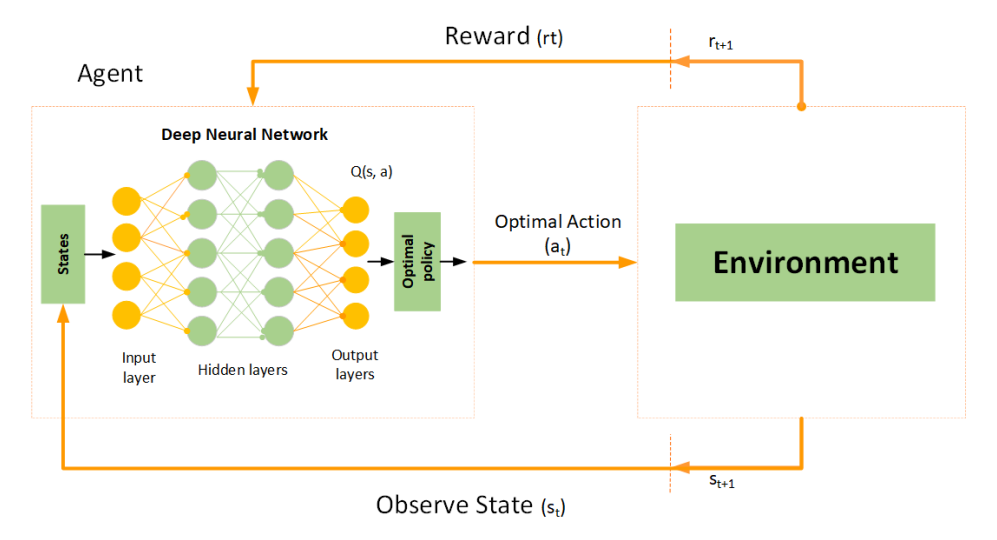

Figura 2.5: Arquitectura general de una Deep Q-Network (DQN). Fuente: Deep Q-Learning (DQN) [14].

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 [15].

Funcionamiento de DQN [16]

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.

- **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 de 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 [13]

(+) 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 [13]

(-) 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 [17].

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.

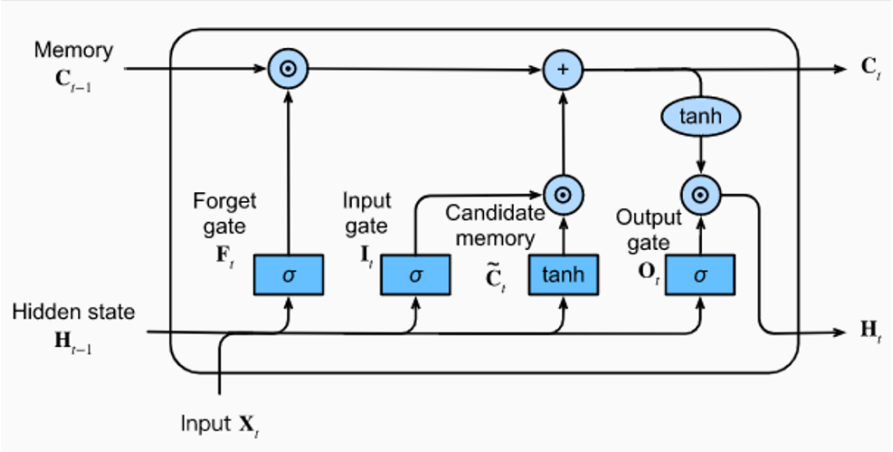

Figura 2.6: Diagrama LSTM. Fuente: *An Intuitive Explanation of LSTM* [18].

La capacidad de decidir dinámicamente qué recordar y qué olvidar convierte a las LSTM en una herramienta eficaz para trabajar con dependencias temporales prolongadas [18]. 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 contrapuestas. Éstas, 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 el output de la red A sea tan similar al original, que dificulte la diferenciación. [19].

Fortalezas	**Debilidades**
Generación de imágenes	Entrenamiento complejo
Reescalado	Inestabilidad y "mode collapse"
Multimodalidad	
Gran versatilidad	

Cuadro 2.2: Fortalezas y debilidades de las GANs

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. [20].

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

Cuadro 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. [21].

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

Cuadro 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. [22].

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

Cuadro 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:

Cuadro 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.

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		✓			✓

El 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:

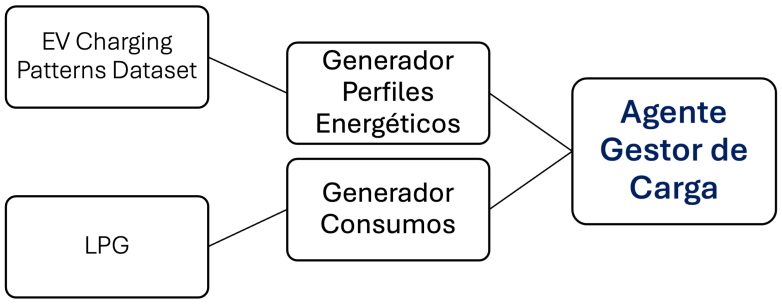

Figura 3.1: Metodología utilizada en el trabajo. Fuente: Elaboración propia.

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, 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.

La implementación del código relevante para el proyecto se ha realizado en *Python*[23], 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*[24], con *GurobiPy*[25] como solver. Para el desarrollo de todo lo relacionado con aprendizaje profundo, se han empleado las librerías de *Torch*[26].

La metodología utilizada en este trabajo se ha diseñado para abordar de manera sistemática y efectiva los objetivos planteados. La Figura 3.1 ilustra las etapas clave del proceso, que incluyen la recopilación de datos, el análisis de la información, el desarrollo de modelos predictivos y la implementación de estrategias de gestión de la demanda.

Capítulo 4

Caso de Estudio

4.1. Generador de Perfiles Energéticos

El uso de los vehículos es particular al estilo de vida de cada usuario, tanto para vehículos de combustión como para vehículos eléctricos. Por ello, es necesario generar perfiles energéticos que sinteticen y agrupen los diferentes usos energéticos —y por tanto la necesidad de carga de dichos vehículos— de los usuarios.

En el primer bloque que compone el trabajo, se ha desarrollado un algoritmo generador de perfiles de carga de EVs, partiendo del *Electric Vehicle Charging Dataset* [3]. A partir de este dataset, se han generado nuevas muestras sintéticas de perfiles de carga y uso típico de las estaciones de carga. Para ello, se ha utilizado una de las herramientas de IA Generativa, presentada en el Capítulo 2, concretamente se ha hecho uso de una GAN.

Los parámetros de la GAN se han ajustado para que los perfiles generados representen adecuadamente la variabilidad de la demanda energética para la carga de un EV, teniendo en cuenta las restricciones, requisitos y hábitos del usuario. El resultado es un conjunto de perfiles energéticos que pueden ser utilizados para entrenar y evaluar el gestor de carga de EVs, así como para simular diferentes escenarios de carga y consumo energético.

4.2. 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 vehículo eléctrico, hijos y hábitos laborales estándar. La configuración empleada 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	▪ Mujer (trabaja fuera, jornada completa) ▪ Hombre (trabaja en casa, jornada parcial) ▪ Niños: uno en edad escolar, otro en edad preescolar
Vehículo eléctrico	▪ 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

Cuadro 4.1: Parámetros de configuración de la vivienda en LPG

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

- **Adulto 1 (mujer):** trabaja fuera del hogar a jornada completa (8h–17h), desplazándose en vehículo eléctrico cinco días a la semana. Contribuye a la carga del EV en horario nocturno.
- **Adulto 2 (hombre):** 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.

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

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

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 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 vehículo eléctrico	Carga según trayectos diarios; horarios preferentes nocturnos
Consumo en espera (stand-by)	Dispositivos conectados permanentemente; carga latente (TV, router, cargadores)

Cuadro 4.2: 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.

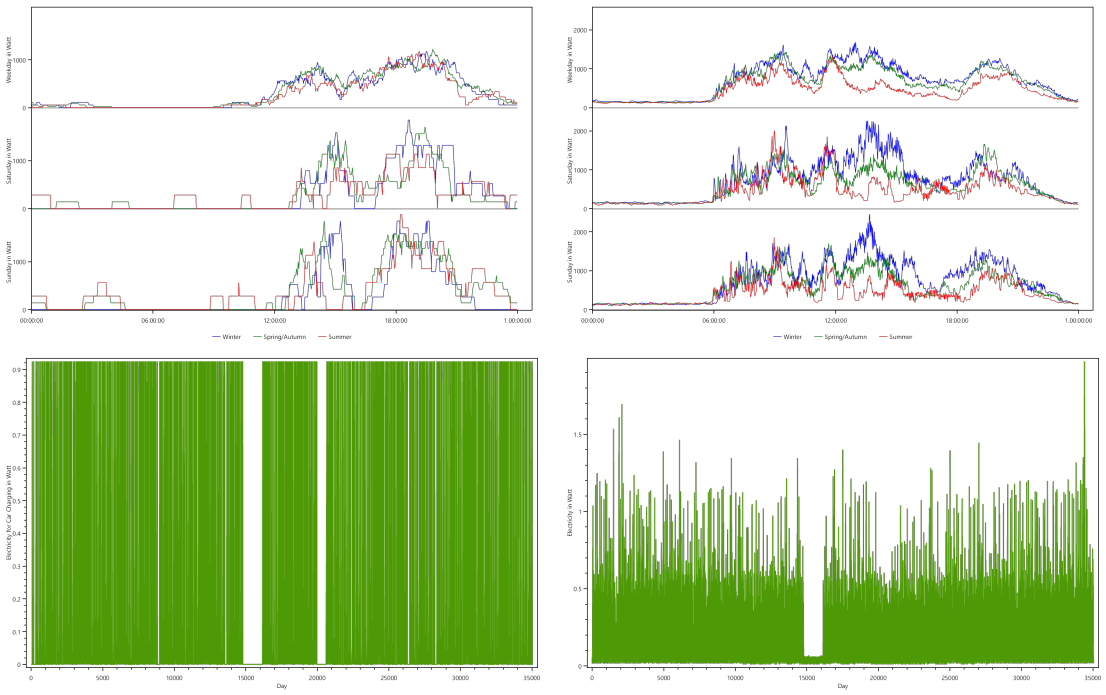

Figura 4.1: Gráficas generadas por el LPG.

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 vehículo eléctrico, 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.

Las gráficas en la fila inferior muestran el consumo total de energía para la carga del EV, y en general a lo largo del año, respectivamente. Se puede observar que el consumo energético varía a lo largo del año, con picos de consumo en invierno y verano, y una tendencia general de aumento del consumo a lo largo del año.

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 vehículo eléctrico. Además,

permiten identificar los momentos de mayor consumo energético, lo que es esencial para la gestión eficiente de la carga del EV.

Más imágenes de las gráficas generadas por el LPG, y análisis propios, se pueden encontrar en el anexo **??**.

4.3. Gestor 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, 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 vehículo eléctrico.

El gestor de carga tiene como objetivo optimizar la carga del vehículo eléctrico, 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.

4.3.1. Optimización Clásica

Primeramente, se han de definir el conjunto de variables que después formarán las restricciones y objetivos del problema de optimización. En este caso, las variables son las siguientes:

- $P(t)$: Potencia en el instante t [kW]
- $E(t)$: Energía en el instante t [kWh]
- P_{Max}: Potencia máxima en el cargador [kW]
- $SOC(t)$: Estado de carga en el instante t [kWh]
- $P_{NG}(t)$: Potencia del No Gestionable en t [kW]

- η: Eficiencia del sistema
- $A(t)$: Disponibilidad en el instante t [1: disponible, 0: no]
- $p(t)$: Precio de la electricidad en el instante t [€/kWh]
- $C(t)$: Coste de carga en el instante t [€]

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

$$\min_{P(t)} \quad \sum_t C(t) = \sum_t E(t) \cdot p(t) \tag{4.1}$$

$$\text{s.a.} \quad 0 \leq P(t) \leq P_{\max} \cdot A(t) \tag{4.2}$$

$$P(t) + P_{\text{NG}}(t) \leq P_{\text{red}} \tag{4.3}$$

$$SOC(t+1) = SOC(t) + \eta \cdot \frac{P(t) \cdot \Delta t}{\text{capacidad}} \tag{4.4}$$

$$\sum_t P(t) \cdot \Delta t \geq (SOC_f - SOC_i) \cdot \text{capacidad} \tag{4.5}$$

Una vez ya se ha planteado el problema, se implementa el Python [23] usando la librería *Pyomo* [24] para definir el modelo de optimización, y *GurobiPy* [25] 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.

Los resultados obtenidos con este algoritmo de optimización clásico sirven como base para comparar el rendimiento del gestor de carga basado en aprendizaje por refuerzo, y para evaluar la mejora en la eficiencia y reducción de costes que se puede lograr con el uso de técnicas del RL.

Los resultados obtenidos con el algoritmo de optimización clásico se muestran a continuación:

4.3.2. 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 vehículo eléctrico. 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`) 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.

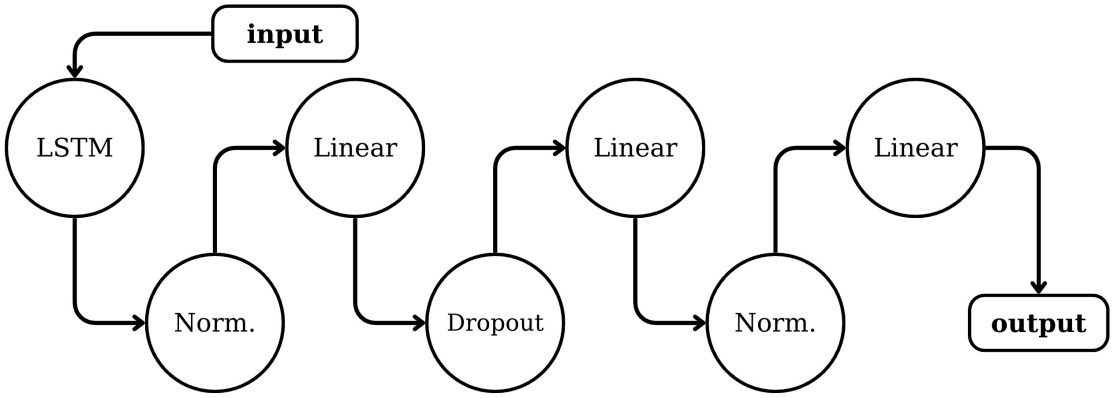

Figura 4.2: Arquitectura de la red neuronal DQN utilizada en el gestor de carga. Fuente: Elaboración propia.

Entorno de la Red DQN-LSTM

Para entrenar y evaluar el comportamiento del agente gestor, se ha desarrollado un entorno personalizado en Python [23]. 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 $\mathbf{s}_t \in \mathbb{R}^5$ en cada instante t incluye las siguientes variables:

- **State of Charge (SOC):** estado de carga actual del vehículo eléctrico.
- **Hora normalizada:** instante del día en formato continuo $[0, 1]$.
- **Consumo no gestionable (P_{NG}):** 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.

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 accion. Esto significa que el entorno debe calcular:

- La potencia solicitada $P = a_t \cdot P_{\max}$.
- 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 recompensa está diseñada para reflejar los objetivos del sistema: minimizar el coste energético, evitar violaciones de restricciones físicas, y alcanzar el SOC deseado al final del día. Se penaliza explícitamente el uso de energía cuando el vehículo no está disponible o cuando se supera el margen permitido por la potencia contratada. Por tanto, se penalizará al agente en los siguientes casos:

- Coste de la electricidad en ese instante.
- Cargar cuando el vehículo no está disponible ($A_t \approx 0$).
- Superar la potencia contratada ($P > P_{\text{red}} - P_{NG}$).
- Exceder el SOC máximo deseado, si la batería ya se ha cargado a ese nivel.

Adicionalmente, se recompensa positivamente al agente si el SOC final se encuentra en el rango deseado ($[0{,}75; 0{,}8]$, por ejemplo) al término del episodio. Pr tanto, la función de recompensa R_t se puede expresar maremáticamente como:

$$r_t = -\left(C_t + \lambda_1 \cdot A_t + \lambda_2 \cdot (P - P_{\text{red}} + P_{NG})^2 + \lambda_3 \cdot (SOC - SOC_{\max})^2\right) \quad (4.6)$$

donde λ_1, λ_2 y λ_3 son coeficientes de ponderación que ajustan la importancia de cada componente de la recompensa. El coste C_t se calcula como el producto de la potencia entregada P y el precio de la electricidad $p(t)$ en el instante t.

Formato del episodio. Cada episodio simula un día completo dividido en intervalos de 15 minutos (96 pasos). El SOC se inicializa aleatoriamente entre 3 % y 25 % al inicio del día, y se va actualizando conforme a las decisiones del agente.

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 $T = 4$ pasos (1 hora), 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**: 4 pasos
- **Tamaño del buffer de experiencia**: 10000

Para la elección de los valores de estos hiperparámetros, se ha realizado una algoritmo de *Grid Search* sobre un rango de valores, y se ha seleccionado el conjunto que mejor rendimiento ha obtenido en términos de recompensa media acumulada y estabilidad del aprendizaje.

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.

Capítulo 5

Resultados

Capítulo 6

Conclusiones y Trabajo Futuro

En este trabajo se ha presentado un enfoque innovador para la gestión de la demanda energética en el contexto de la creciente adopción de vehículos eléctricos (EV) y la integración de energías renovables. A través del uso de Inteligencia Artificial Generativa, se han desarrollado modelos capaces de predecir y optimizar el consumo energético, contribuyendo a una gestión más eficiente y sostenible de los recursos energéticos.

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

[12] 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.

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

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

[15] 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.

[16] 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.

[17] 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/.

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

[19] 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.

[20] 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.

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

[22] 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/.

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

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

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

[26] 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.