Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Estimación y predicción de la radiación solar mediante el uso de redes neuronales convolucionales, cámaras all sky y datos anemométricos

Autor: Juan Miguel López Panal

Tutores: José Ramón Domínguez Frejo y Javier García Martín

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías Industriales

Estimación y predicción de la radiación solar mediante el uso de redes neuronales convolucionales, cámaras all sky y datos anemométricos

Autor:
Juan Miguel López Panal

Tutores:
José Ramón Domínguez Frejo y Javier García Martín
Investigador postdoctoral TU Delft (Países Bajos)
Profesor Titular de Universidad

Dpto. Ingeniería de Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2024

Trabajo Fin de Grado: Estimación y predicción de la radiación solar mediante el uso de redes neuronales convolucionales, cámaras all sky y datos anemométricos

Autor: Juan Miguel López Panal
Tutores: José Ramón Domínguez Frejo y Javier García Martín

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Quisiera expresar mi más sincero agradecimiento a todas aquellas personas que han contribuido de alguna manera a la realización de este trabajo. En primer lugar, agradecer tanto a José Ramón como a Javier por su orientación a lo largo del proyecto y sus comentarios que me han servido de guía durante el proceso de investigación para mejorar y alcanzar el resultado deseado.

No puedo dejar de mencionar a mi familia y amigos, en especial a mi madre, mi padre y mi hermano que han sabido estar a mi lado y aconsejarme sabiamente durante este proceso; y a Alejandro, cuyo apoyo incondicional y palabras de aliento fueron fundamentales en los momentos de dificultad y duda. Finalmente, agradezco a mi amiga María que haya estado ahí para escucharme y darme fuerzas cuando más complicado fue este viaje.

Por último, pero no menos importante, quiero agradecer a todas las personas cuyas obras y contribuciones previas han servido de inspiración y fundamento para este trabajo.

Su generosidad y apoyo han sido fundamentales en el desarrollo de este proyecto, y por ello les estoy profundamente agradecido.

Os estaré eternamente agradecido por haberme acompañado durante todo este camino.

Resumen

En este proyecto de fin de grado se han generado múltiples modelos de redes neuornales convolucionales capaces de analizar imágenes y datos escalares para obtener como resultado la radiación solar que reciben un conjunto de placas solares.

Las bases de datos empleadas están basadas en imágenes de ojo de pez captadas desde la estación meteorológica situada en el laboratorio L1 de la ETSI. Esta información matricial se mapea con datos escalares como la radiación directa, tomada por el pirheliómetro, y se combina con datos relativos a las condiciones atmosféricas. En el caso de estudio, se ha tomado únicamente la información relativa al viento (dirección y velocidad), si bien la estación está diseñada para obtener otro tipo de datos de interés.

El objetivo del proyecto ha sido la estimación de radiación solar, enfocada tanto al tiempo real como a la predicción de la misma. Así, se han ejecutado un total de 3 bases de datos distintas: una inicial para la estimación actual de la radiación dado un conjunto de imágenes, una segunda empleada para el entrenamiento de modelos para predicción basada exclusivamente en imágenes; y la última análoga a la anterior pero con la conjunción de datos matriciales (imágenes) y escalares (parámetros del viento) como entrada.

El modelo inicial para estimar la radiación actual presenta una configuración simplificada basada en modelos previos como LeNet-5. Este mismo modelo sirve de base para aquéllos posteriores que se entrenarán con el afán de predecir la radiación solar directa. En los modelos asociados a las dos primeras bases de datos se sigue la misma estructura de red; no obstante, debido a la incorporación de datos escalares como inputs del entrenamiento, los modelos generados en el último caso poseen una composición de capas un poco más compleja.

Se ha valorado la eficacia que tienen estos modelos y la mejora de los mismos sobre métodos tradicionales empleados. Asimismo, se ha discutido sobre cómo de certeras son las estimaciones predichas conforme el horizonte de predicción aumenta. Como base de este estudio, se ha descrito la problemática actual que genera el uso de las energías renovables en cuanto a la gestión de la información que se extrae mediante la sensorización de dichas plantas.

De la misma forma, se han expuesto los resultados obtenidos mediante este trabajo y la utilización de los mismos en soluciones reales y tangibles que podrían ser empleadas en un futuro. En general, los resultados de los modelos han sido bastante satisfactorios, obteniéndose, en la estimación, RMSE de menos de 0,08 y, en relación a la predicción, valores de menos de 0,13. Se ha concluido que la reducción del error viene asociado a la introducción de los parámetros de viento como entrada, observándose que la unión de datos matriciales con escalares de diferente naturaleza favorece al aprendizaje del modelo en gran medida.

Para la realización de este proyecto, ha sido necesario obtener nuevas habilidades en relación a programación e investigación en la inteligencia artificial, así como conocimientos sobre la generación de energía eléctrica mediante esta tecnología y su gestión y retos.

Abstract

In this essay, multiple convolutional neural network models capable of analyzing images and scalar data have been developed to determine the direct solar irradiance stimation and prediction.

The databases used are based on "fisheye" images captured from the weather station located in the ETSI's L1 laboratory. This matrix information is mapped with scalar data such as direct radiation, taken by the pyrheliometer, and combined with data related to atmospheric conditions. In the case study, only information related to wind (direction and speed) has been used, although the station is designed to obtain other relevant data.

The aim of the project was to estimate solar radiation, focusing on both real-time estimation and prediction. Thus, a total of three different databases have been created: an initial one for the current estimation of radiation given a set of images; a second one used for training models for prediction based exclusively on images; and the last one, similar to the previous one but with the combination of matrix data (images) and scalar data (wind parameters) as inputs.

The initial model for estimating current radiation presents a simplified configuration based on previous models such as LeNet-5. This same model is the basis for those subsequent ones that will be trained with the aim of predicting direct solar radiation. The models associated with the first two databases follow the same network structure; however, due to the incorporation of scalar data as training inputs, the models generated in the latter case have a slightly more complex layer composition.

The effectiveness of these models and their improvement over traditional methods used has been evaluated. Furthermore, the accuracy of the predicted values as the prediction horizon increases has been discussed. As the basis of this study, the current issues generated by the use of renewable energies regarding the management of the information extracted through the sensorization of these plants have been described.

Likewise, the results obtained through this work and their use in real and tangible solutions that could be employed in the future have been presented. In general, the results of the models have been quite satisfactory, achieving RMSE values of less than 0,08 for estimation and less than 0,13 for prediction. In conclusion, the reduction of error is associated with the introduction of wind parameters as input, observing that the combination of matrix and varied scalar data greatly improves the model's learning.

In order to make this Bachelor's thesis possible, we have experimented a skill development related to programming and researching about artificial intelligence areas. Additionally, we have acquired knowledge from generating electricity using this technology, also its management and its challenges.

Índice

1 Introducción y Fundamentos Teóricos

1.1 Contextualización del Problema e Importancia de la Predicción Solar

La energía solar es una de las fuentes renovables con mayor disponibilidad en el mundo. En la actualidad, las llamadas centrales solares las emplean mediante diversos métodos para la producción de electricidad. Es un recurso útil debido a su bajo impacto mediambiental, su ubicuidad y, más importante, su bajo coste en la actualidad. Sin embargo, plantea una serie de retos desde el punto de vista tecnológico para estas plantas, pues la radiación solar está sujeta a múltiples factores tanto geográficos como climáticos, estacionales, ...

Dentro de las centrales solares, existen dos grandes familias que hacen uso de la radiación solar para la generación de electricidad: las centrales fotovoltaicas y las centrales termosolares, como las conocidas en inglés como "concentrating solar thermal systems".

La producción de energía en centrales fotovoltaicas [12] se basa en el efecto fotovoltaico [4], proceso a través del cual un fotón de luz, al interaccionar con un electrón, altera su estado energético. Esta reacción se genera en los paneles fotovoltaicos, que se extiende en campos de gran amplitud, aunque también se pueden observar en tejados, como en la Figura 1.1 cuando se trata de producción para consumo destinado a hogares, ya sea de forma individual como instalaciones comunitarias que abastece a un pequeño núcleo de personas.

Figura 1.1 Placas solares sobre el tejado de un edificio [24].

En las centrales termosolares, la luz se usa como medio para calentar un punto concreto de la planta, de esta forma, se genera una fuente de calor que generalmente se emplea para calentar un fluido que, al evaporarse, genera electricidad al mover una turbina, como en las centrales eléctricas convencionales. Estas últimas, a su vez, se clasifican según la geometría del espejo que refleja la luz; así, se tienen plantas cilindro parabólicas, como se aprecia en la Figura 1.2, de heliostatos con una torre central como receptor, de discos solares parabólicos, o con reflectores lineales [5].

Figura 1.2 Campo de cilindros parabólicos en una central termosolar [5].

El reto que supone trabajar con la radiación solar está en su variabilidad, de ahí que su predicción sea tan importante ya que en ella radica el beneficio económico de la empresa: la energía que se genera en la central depende directamente de la energía solar que recibe. La irradiancia que finalmente llega a una planta está sujeta a muchos parámetros: nubosidad, estación del año, latitud, clima, etc [5]. Aun teniendo constancia de todos ellos, su valor fluctuará, con lo que la previsión de energía producida en la central eléctrica depende de muchos factores, llegando a ser muy complejo obtener un modelo que calcule este dato.

El análisis de los datos climáticos alrededor de la planta presenta una problemática clara: puede llegar a ser muy compleja y, a su vez, la toma de medidas de los fluidos caloportadores de las plantas conlleva una alteración de la temperatura de los mismos, la cual debe restringirse a unos límites dados. Partiendo del escenario supuesto en el proyecto OCONTSOLAR, se plantea la dificultad de tratar con vehículos autoguiados (AGV) o vehículos aéreos no tripulados (UAV), como drones, en los alrededores de las plantas termosolares como captación de información atmosférica en tiempo real [22].

Este proyecto impulsado por ERC Advanced Grant, en colaboración con la Universidad de Sevilla, pretende desarrollar nuevas metodologías de control haciendo uso de sensores móviles incorporados a vehículos no tripulados. De esta forma, se hace uso de una flota de drones y vehículos autoguiados capaces de recabar una estimación distribuida de las plantas solares (tecnología que se estudia implantar).

Para poder obtener predicciones tangibles de manera que se puedan hacer este tipo de cálculos, tradicionalmente en la industria se han utilizado las bases de datos que se generan en las estaciones meteorológicas, localizada generalmente próximas a las centrales. En este caso, los modelos asumen que el comportamiento que se ha tenido hasta ese momento será el que se tendrá en un futuro, son modelos reactivos. También ha existido el uso de correlaciones y modelos que calculan la radiación en base a la mayor radiación solar que se tiene en un periodo de tiempo concreto. En ellos, sí existen elementos que modifican la evolución con respecto a los datos pasados. No obstante, ambas metodologías tienen en común la poca eficiencia a la hora de predecir la radiación a corto plazo. Asimismo, asumen un comportamiento sesgado y no captan de forma global la dinámica de la radiación solar [26].

Para obtener datos más realistas del clima en tiempo real, una posibilidad sería introducir vehículos que se muevan de forma automático en los alrededores de la planta, como la flota de drones [21] propuesto en el proyecto OCONTSOLAR, cuya implantación supondría una mejora en la frecuencia de actualización de datos así como en su certeza; permitiendo predecir irradiancias de forma más efectiva. No obstante, es una tecnología en fase de experimentación y a día de hoy no se ha implantado. En el presente, las

plantas solares poseen una estación solar anexa, en la mayoría de ocasiones, para la monitorización de los parámetros de interés.

En la actualidad, debido al calentamiento global, los modelos que se tienen pueden no funcionar ya que las épocas de sequía ocurren con mayor frecuencia y la lluvia cae de forma torrencial con mayor asiduidad. En aras de obtener resultados que se ajusten mejor repercutiendo de forma directa sobre el beneficio económico de las centrales solares, se ve necesario el uso de modelos más complejos que sean capaces de generar unos resultados realistas basados en los múltiples parámetros de entrada que se relacionan con la radiación solar. Debido a una mayor simplicidad, muchas plantas captan información climática mediante imágenes del cielo o sistemas de cámaras de sombras, cuya función es tomar fotos del suelo desde una posición elevada para captar las sombras generadas por las nubes [15]. De esta forma, se tiene una visión global de los objetos que interceden entre los elementos de captación solar de la planta y el sol, generando un mapa de irradiancia con unas tasas de actualización de 0 a 30 minutos [15]. Se observa que las frecuencias que se tienen mediante este método son efectivas para el cálculo de la radiación a largo plazo; sin embargo, no son eficientes para el corto plazo. El objetivo de este estudio se centra en este método de captación de información, incorporando datos escalares y una etapa posterior de tratamiento mediante modelos que garanticen una predicción adecuada en plazos de tiempo más reducidos.

A pesar de que una mejor metodología para la captación de información pudiera ser la expuesta en OCONSOLAR, las plantas actuales presentan estaciones meteorológicas donde aúnan todos los sensores. La información que se obtiene es numerosa y de gran variedad, con lo que es lógico plantear modelos que puedan manejar tal cantidad de variables para obtener resultados útiles. De esta forma, parece lógico el uso de modelos basados en basado en machine learning, como las redes neuronales. Siguiendo el estudio realizado por José Barrientos [1], el cual se referenciará a lo largo de la memoria, se empleará para comparar nuestros resultados y hacer símiles ya que la tesis que analizamos parte de la que él investigó en dicho estudio preliminar.

1.2 Conceptos Básicos sobre la Radiación Solar

La radiación solar total que incide sobre la atmósfera es absorbida en gran medida por la misma; sin embargo, parte de ella consigue traspasar llegando a la corteza terrestre. Esta fracción, a su vez, se divide en dos tipos:

- Aquélla que se conoce como radiación solar dispersa. Este tipo de radiación se genera mediante emisores de longitudes de onda cortas, provenientes de todas las zonas del cielo [19].
- Y la radiación solar directa que, a diferencia de la anterior, no proviene de diferentes zonas de la atmósfera. Es la radiación que no presenta grandes modificaciones en su dirección hasta llegar a la corteza terrestre [27].

La conjunción de ambas es lo que se conoce como radiación global [19], este parámetro genera una gran repercusión dentro de las plantas con campos fotovoltaicos pues el GHI, *global horizontal irradiance*, determina la cantidad de energía que puede generar una celda solar mediante efecto fotoeléctrico. El GHI es la cantidad total de radiación global, tanto directa como difusa, en una en una superficie concreta de la corteza terrestre [8]. Asimismo, en las centrales solares que usan tecnologías relacionadas con la concentración de radiación solar es muy importante el DNI, *direct normal irradiance*,

que es la radiación directa recibida en un plano normal a la dirección de los haces de luz provenientes del Sol [3].

Para obtener estas mediciones en las centrales solares, se suelen localizar cerca de los campos de placas fotovoltaicas o de espejos termosolares unas estaciones meteorológicas que presentan varios dispositivos para poder conocer el estado del clima:

- El piranómetro es un dispositivo que se usa para medir la cantidad de irradiancia solar. En general, mide la radiación solar global que llega a una zona plana del terreno, de esta forma, genera medidas de vatios por metro cuadrado (W/m^2). Aportan información necesaria para analizar y predecir condiciones medioambientales [23].
- El pirheliómetro, a diferencia del anterior, mide únicamente la radiación solar directa proveniente de una dirección concreta del Sol [3]. Al igual que el anterior aparato, da una señal de radiación en unidades de vatio por metro cuadrado (W/m^2). Este dispositivo se muestra en la Figura 1.3, donde se encuentran varios dispositivos orientados en una dirección concreta.

Estos dos instrumentos son los más comunes dentro de una estación meteorológica para tener una estimación de la radiación solar que llega al entorno de la central eléctrica. Sin embargo, no son los únicos, puesto que este parámetro es muy dependiente de otras condiciones climáticas. Por ello, también existen sensores de temperatura ambiente, ya que la eficiencia de una placa solar es inversamente proporcional a esta variable; sensores de humedad relativa, puesto que grandes concentraciones favorecen la existencia de niebla que entorpece la llegada de radiación solar directa; y sensores de dirección y velocidad del viento, para tener un conocimiento sobre las nubes que atraviesan el campo de la central.

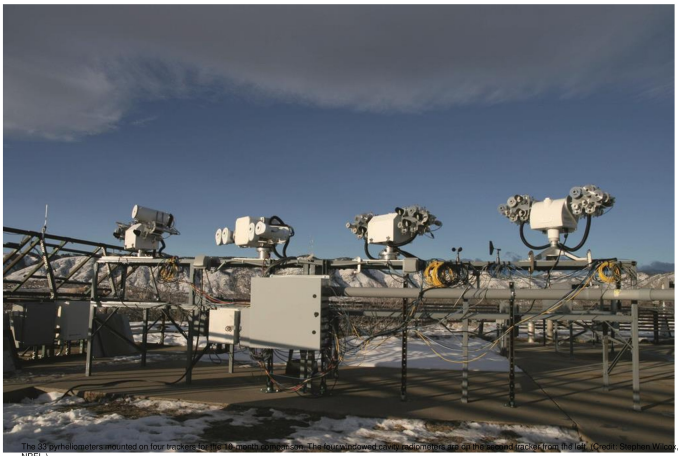

Figura 1.3 Pirheliómetro usado en una estación meteorológica [25].

1.3 Problemática asociada a la Fotovoltaica en el Paradigma Actual

En la actualidad, existe una gran inserción de energías renovables en nuestro entorno, tanto para autoconsumo como de manera industrial. Centrándonos en las plantas de generación fotovoltaica, dan lugar a un reto sin precedentes desde muchos puntos de vista. Eléctricamente proporcionan múltiples desafíos en la red ya que generan comportamientos que hasta ahora no se habían tenido antes. Por ejemplo, existen retos dentro del balance eléctrico basado en la curva de demanda-generación. A continuación, se presenta una

situación paradigmática de la introducción de fuentes renovables basadas en energía solar:

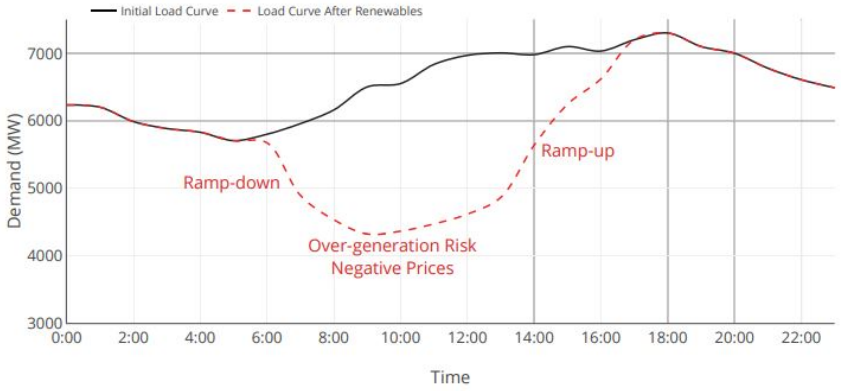

Figura 1.4 Curva de Pato [14].

Para observar el fenómeno que se quiere explicar, debemos centrarnos en la curva roja discontinua de la Figura 1.4. El uso de renovables basadas en solar propicia un comportamiento anómalo de la red puesto que, no sólo las plantas de generación industriales vierten a la red, sino que el autoconsumo de los hogares, que cada vez toma más peso, da lugar a un aporte de la energía mayor en los momentos de mayor irradiación solar. De esta forma, hay puntos de la curva, centrados en las horas en el rango de las 10:00 hasta las 14:00 donde la demanda se ve compensada con la generación de estas tecnologías. Esto implica que otras plantas de generación deban bajar el ritmo o incluso parar para no saturar la red, hecho que en muchas ocasiones es imposible (energía nuclear) o muy caro (ciclos combinados). La previsión en la generación de energía eléctrica mediante renovables puede mitigar parte de estos efectos adversos, pues con la introducción de dispositivos de almacenamiento, se podría absorber parte de la energía que previamente se ha predicho que se va a generar, y verterla en momentos posteriores (hacia el final del día) donde la demanda crezca de forma abrupta y las renovables basadas en solar no generen como se tenía en horas previas.

No sólo desde el punto de vista de la generación suponen un reto. Se debe hacer especial hincapié en la naturaleza de la generación de fotovoltaica. Esta energía proviene, como ya se explicó, del efecto fotoeléctrico y da lugar a una generación eléctrica en continua. Para verter a la red ésta debe pasar por un inversor en primera instancia para transformarse a alterna. Este artefacto en su frecuencia tiene muchas repercusiones porque desde el punto de vista de la red, la energía que se genera no tiene *inercia*, lo que implica que una desconexión de una planta o la falta de generación de la misma puede aumentar mucho la tensión de la red, y no sólo de manera local.

En la Figura 1.5, proporcionada por la empresa e-distribución y, a su vez, facilitada por la empresa eléctrica ENTSO-e, se muestra un mapa de tensiones. Para entenderlo, debemos explicar lo que ocurrió el día 24 de julio de 2021. Acaeció un incendió al sur de Francia dentro de una planta de generación, lo que propició un aumento de la demanda en la interconexión entre España y el país vecino. Para satisfacer esta necesidad, la red española se vio forzada a aumentar la demanda de las plantas más próximas a dicho punto. Entre estas existían las plantas de fotovoltaica que, al carecer de inercia, no pudieron adaptarse a cambios bruscos en la demanda como el que estaba ocurriendo, ya que la energía que generan depende de forma total de las condiciones atmosféricas

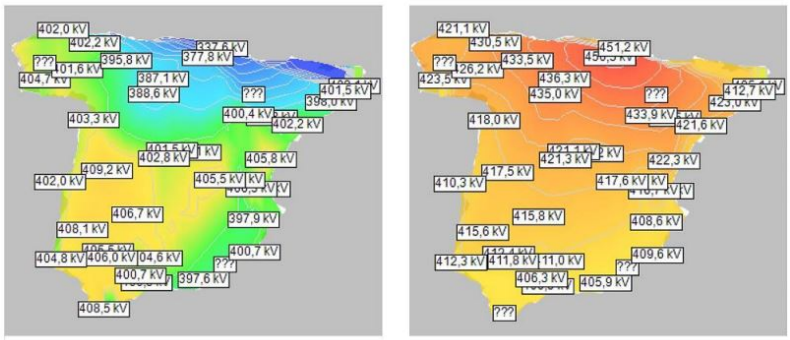

Figura 1.5 Aumento de la tensión provocado por el incidente del 24/07/2021.

que se den en ese momento. Al no poder satisfacer este pico de potencia, se fueron desconectando una a una, en un efecto en cadena, lo que aumentó la tensión de nuestra red como se observa en la subfigura derecha, en la imagen izquierda se tienen las tensiones previas al accidente. Aunque ocurrió por un hecho fortuito, esta circunstancia da lugar a muchas consideraciones como es la aparamenta y protección de una planta de fotovoltaica para que no se desconectase de forma tan brusca o cómo proceder cuando una planta de estas características, cuya dependencia solar es tan alta, no genera. Las plantas fotovoltaicas en días de muchas nubes pueden llegar a no generar electricidad, hecho que como se ha observado, aumenta la tensión de los puntos donde se conecta. Es importante tener una previsión de la potencia que puede llegar a generar un campo fotovoltaico ya que, en caso de no producir, deben existir una serie de medidas que garanticen el buen funcionamiento de la red eléctrica en la que operen.

La predicción de la energía eléctrica generada por una fuente renovable de estas características es clave en la operación de la red donde vierte, ya que permite adelantarse a fenómenos que den lugar a efectos muy adverso que, por desgracia, no son localizados. Teniendo un conocimiento previo del comportamiento más probable que tenga la planta de generación, se pueden solventar contratiempos, minimizando las repercusiones sobre la red eléctrica.

1.4 Introducción a la Inteligencia Artificial y el Aprendizaje Automático

Debido al aumento de aplicaciones relacionadas con la generación de energía eléctrica mediante energía solar, se ve necesario el uso de series temporales cada vez más extensas y que contienen un mayor número de parámetros cuyo estudio pormenorizado implica modelos matemáticos más precisos [24], como se mencionaba en el subcapítulo anterior.

La predicción de radiación solar se ha convertido, hoy en día, en un asunto de interés para plantas de energía renovable, como las fotovoltaicas o las termosolares; que ven necesario un cambio de paradigma, ya que la mayoría de las técnicas actuales están basadas en un horizonte de predicción de como mínimo 12 horas, lo cual no permite plantear un control óptimo de la planta ni una optimización del sistema de energía solar en su conjunto [24].

Por todo ello, cada vez se promueve más el uso de la inteligencia artificial basada en redes neuronales dentro de la gestión de las plantas. Éstas son capaces de analizar

las series temporales de datos que tienen como entrada. Estos datos numéricos acompañados de imágenes del cielo, permiten al modelo generado mediante la red, inferir características que, a priori, son difíciles de hallar de forma manual. Obteniéndose, de esta forma, modelos capaces de predecir radiación optimizando los recursos y con horizontes temporales más reducidos, produciendo un beneficio directo sobre el control de la planta.

1.5 Objetivos del Trabajo

El estudio que se va a realizar en este informe tiene como objetivo principal determinar la eficacia de las redes neuronales a la hora de estimar la radiación solar. Para ello:

1. Se mencionarán modelos tradicionales haciendo observaciones sobre la simplificación que suponen el uso de técnicas de machine learning sobre conjuntos de información tan extensos.
2. Se emplearán diferentes bases de datos para analizar cómo este rendimiento se ve afectado por parámetros íntimamente relacionados a la radiación tales como la velocidad o la dirección del viento, ya que el movimiento de las nubes en el entorno del sensor puede generar un impacto considerable sobre la estimación de la irradiancia tanto presente como futura.
3. Se discutirá el empeoramiento de la eficacia cuando se amplía el horizonte temporal de estimación.

1.6 Principales Enfoques y Técnicas Utilizadas

Este proyecto se centra en el análisis de imágenes mediante inteligencia artificial para la deducción y el cálculo de radiación solar. Las imágenes se han tomado mediante la estación meteorógica del departamento de termodinámica situado en el techo del laboratorio L1 de la Escuela Técnica Superior de Ingeniería.

Esta estación provee de una base de datos de imágenes all sky, en formato de ojo de pez. En ellas, aparece el cielo en su totalidad sin ningún obstáculo visual entre el objetivo y el sol más que las nubes que se crucen y generen sombra. Asimismo, la estación está dotada con varios con pirheliómetros, para captar datos de la DNI, y piranómetros, para los datos correspondientes a la GHI. Además, se encuentran otros objetos para la captación de información climatológica como la velocidad y la dirección del viento, mediante un anemómetro y una veleta, respectivamente.

El trabajo se enfoca en la extracción de información mediante las imágenes que la cámara efectúa y guarda en su base de datos, en la Figura 1.6 se muestra un ejemplo. Así, junto con la información de irradiancia captadas por los diferentes instrumentos, se empleará una red neuronal convolucional que extraiga la información de las imágenes para poder calcular la radiación en los instantes pedidos. Elongando los horizontes de predicción cada vez más, para evaluar el comportamiento de la red. En un segundo apartado, se empleará como datos de entrada la velocidad y la dirección del viento, además de las imágenes cuyas etiquetas son la radiación solar. Para evaluar el desempeño de la red al aportar una base de datos mayor con parámetros de distinta naturaleza.

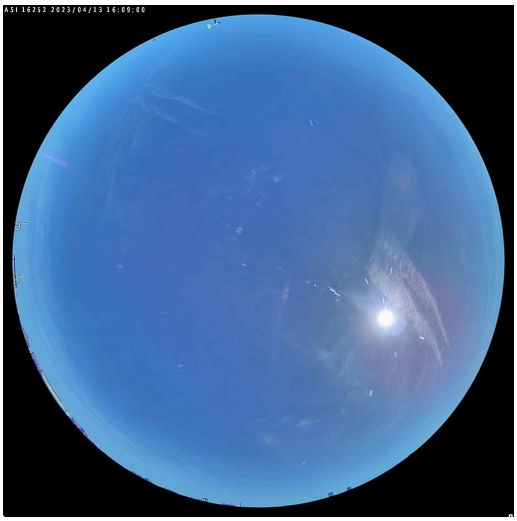

Figura 1.6 Ejemplo de imagen captada por la estación meteorológica de la ETSI.

2 Estado del Arte

2.1 Métodos Tradicionales de Predicción de Radiación Solar

Dentro de los métodos tradicionales de predicción de radiación solar, se encuentran dos grandes grupos:

- La aproximación dinámica, cuya utilidad se restringe a predicciones basadas en periodos de tiempo amplios [26]. Así, no es posible generar un modelo operativo que pueda estimar la radiación en plazos de tiempo cortos, para esta situación, el uso de redes neuronales es preferible debido a su versatilidad, al mapear, de forma no lineal, un gran conjunto de entradas con varias salidas.
- Los modelos empíricos se basan en datos y generan su predicción mediante distintos modelos, a través del uso de ecuaciones cuyas entradas son parámetros climáticos generalmente [26].

Profundizando en estos últimos, existen diversas ecuaciones que generan correlaciones entre la radiación global horizontal en una superficie, la incógnita a calcular, y variables que van desde las horas de sol presentes en esa localización hasta aspectos íntimamente relacionados con el clima como la humedad relativa o la temperatura.

Entre las ecuaciones que encontramos en la literatura, Angstrom propone el primer modelo basado en las horas de luz solar, Ecuación 2.1; donde H es la media mensual de GHI, S es la media mensual de horas diarias de luz solar, S_0 es la media mensual máxima de horas diarias de luz solar o la duración del día, H_0 es la media mensual de radiación extraterrestre diaria y a y b son coeficientes de regresión [26].

$$\frac{H}{H_0} = a + b \cdot (\frac{S}{S_0}) \tag{2.1}$$

Posteriormente, se completó este modelo mediante la introducción de un miembro cuadrático y un nuevo coeficiente de regresión, Ecuación 2.1.

$$\frac{H}{H_0} = a + b \cdot (\frac{S}{S_0}) + c \cdot (\frac{S}{S_0})^2 \tag{2.2}$$

Otros autores estudiaron la relación entre la radiación y factores climáticos calculando modelos que se basan en estos. Algunas de las numerosas ecuaciones halladas en la literatura se plasman en la Tabla 2.1; donde T es la temperatura media en el ambiente, RH es la humedad relativa y S_{max} es la duración máxima de horas de luz solar.

Tabla 2.1 Modelos empíricos basados en factores climáticos [26].

Ecuación
$\frac{H}{H_0} = 0.97877 + 0.05722 \cdot T$
$\frac{H}{H_0} = 1.1973 - 0.00829 \cdot RH$
$\frac{H}{H_0} = 0.5475 + 0.5987 \cdot (\frac{S}{S_{max}}) - 0.0035 \cdot RH$
$\frac{H}{H_0} = 1.309 + 0.601 \cdot (\frac{S}{S_{max}}) - 0.9990 - 0.01287 \cdot T$
$\frac{H}{H_0} = 1.3467 + 0.5305 \cdot (\frac{S}{S_{max}}) - 1.567 + 0.0033 \cdot RH + 0.00806$

Como se observa hacia el final de la tabla, existen varias ecuaciones cuyos parámetros de entrada no son sólo factores medioambientales sino también variables relacionadas con la radiación solar, como las horas de luz. Así, las correlaciones más simples se combinan para generar nuevos modelos que estiman valores basados en un conjunto de entrada mayor. De esta forma, y debido a la naturaleza del problema que estudiamos, cuanto mayor sea el número de variables de entrada mejor será la predicción que haga el modelo. Sin embargo, para llevar a cabo esta idea sería necesaria la introducción de modelos más avanzados que ecuaciones lineales o cuadráticas, para obtener resultados útiles mediante una mejora del rendimiento en la gestión de toda esta información. De ahí que, en la actualidad, se esté estudiando modelos basados en inteligencia artificial para procesar este flujo de información, generando modelos matemáticos más avanzados que den resultados óptimos.

2.2 Estado del Arte sobre Predicción de Radiación Solar utilizando Inteligencia Artificial

Se ha destacado la complejidad del problema en el que se basa el estudio: el conjunto de entrada puede ser tan amplio que, manualmente, sea complicado o hasta imposible manejarlo. Para generar modelos de forma eficiente en la gestión de toda esta información y alcanzar soluciones óptimas, en la actualidad se estudian las redes neuronales. Estos algoritmos presentan diversas formas de aprendizaje automático, los cuales determinan la estructura de la propia red.

Así, se pueden encontrar métodos de aprendizaje automático no supervisado, cuyo objetivo es hacer *clusters* con la información de entrada, es decir, tienen la capacidad de encontrar cómo los datos proporcionados se relacionan entre sí, sin necesidad de aportar un resultado numérico a la salida [13]. Son muy usados en estudios de las ramas sociales o de las médicas, como la psiquiatría. El aprendizaje automático no supervisado se caracteriza por la falta de una variable que analice la respuesta del modelo en su entrenamiento [13].

Otro método muy usado debido a su manera intuitiva de aprendizaje es el *reinforcement learning*, aprendizaje por refuerzo, el cual genera un agente artificial que aprende el comportamiento óptimo mediante las experiencias que obtiene de la interacción con el ambiente [11]. Este ambiente, genéricamente, es una simulación del mundo real, de

manera que el modelo comprende cómo relacionarse gracias a las consecuencias de sus actos sobre el mismo, que se traducen en unas señales de recompensa para la red [11]. En ocasiones, se ha llevado a cabo el aprendizaje de los agentes mediante videojuegos o juegos de mesa. Son muy útiles siempre que los ambientes no sean parcialmente observables, esto es, que la información que se obtiene al ejercer la acción no esté completa, en cuyo caso se necesitaría de una memoria para el agente artificial [11].

Por otro lado, existe el aprendizaje automático supervisado que es utilizado para tareas que involucran la regresión (devolver un valor) o clasificación. Pretende encontrar la correlación entre los parámetros de entrada y las salidas con las que se entrena, generando un patrón para la obtención de resultados [28]. De esta forma, el set de entrenamiento que conforma la entrada son un conjunto de valores con sus correspondientes salidas y, mediante el ajuste de los parámetros internos del modelo, se minimiza el error producido entre las salidas, ya conocidas, y los resultados que se predicen.

Dentro de este último método, se enmarcan las redes neuronales, modelos que se utilizarán en este estudio. Presentan gran eficacia a la hora de hallar características mediante el análisis de imágenes, pudiendo manejar una gran cantidad de información para llegar al resultado final. En concreto, nos centraremos en las redes neuronales convolucionales (CNN), que se detallarán en una sección posterior.

2.2.1 Modelo del Perceptrón y Redes Neuronales

Cuando el conjunto de datos que se trata aumenta de tamaño, la complejidad del problema crece a su vez, entonces la comprensión que se tiene sobre el mismo empieza a reducirse de manera que su tratamiento manual se hace más complicado. Tradicionalmente, llegados a este punto se han solucionado mediante aproximaciones estadísticas. Sin embargo, en la actualidad, el uso de redes neuronales como el perceptrón multicapa o *MultiLayer Perceptron* (MLP), ha demostrado ser más efectivo que estas aproximaciones, pues tienen la capacidad de aproximar cualquier función continua [7].

El MLP está conformado por un sistema de neuronas o nodos, como se aprecia en la Figura 2.1, las cuales representan un modelo no lineal que relaciona un vector de entradas con un vector de salidas. Los nodos se encuentran conectado mediante pesos y señales de salida, que son funciones de la suma de entradas a la neurona modificadas mediante una función de activación. La superposición de muchas funciones no lineales simples permite aproximar a cualquier función continua. La salida de un perceptrón depende de unos pesos y esta, a su vez, conforma la entrada de las neuronas de la siguiente capa, lo que determina un flujo de información [7].

La arquitectura del MLP es variable pero, en general, consiste en varias neuronas que conforman diferentes capas:

- La capa de entrada, *input layer*, no juega ningún papel más que el de introducir la información a la red mediante un vector [7].
- La capa oculta, *hidden layer*, están entre la entrada y la salida. Su función es extraer las características de los vectores de entrada.
- La capa de salida, *output layer*, se encarga de dar el resultado de la red. Este puede ser numérico, en cuyo caso hablaríamos de una capa de regresión, o una clase, clasificación.

El modelo del perceptrón multicapa presenta, generalmente, varias capas ocultas antes de tener una capa final. Estas capas se encuentran completamente conectadas entre sí, *fullyconnected*, es decir, todas las neuronas de una capa están relacionadas con la

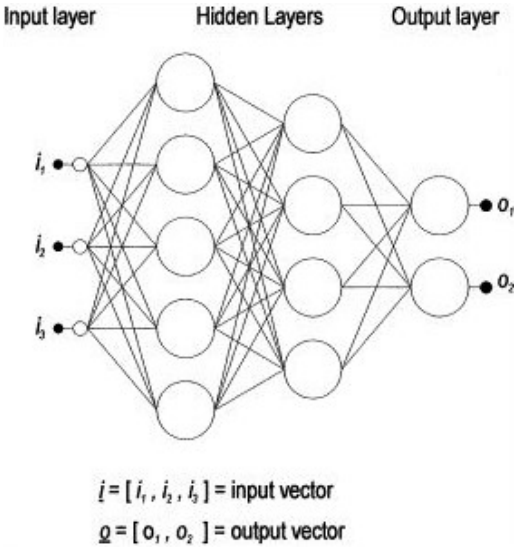

Figura 2.1 Modelo del Perceptrón Multicapa [7].

capa siguiente [7]. Dentro de una misma capa no se deben conectar las neuronas entre sí, para que la dirección del flujo de información esté garantizado y no se creen bucles.

La neurona o perceptrón genera una salida ponderando, de manera lineal, las componentes del vector a la entrada con unos pesos (w) y añadiéndole un sesgo (b), siguiendo la Ecuación 2.3. De esta forma, este modelo se parece a su análogo biológico, Figura 2.2, siendo la sinapsis los pesos, el núcleo es donde se lleva a cabo la operación lineal y el axón es donde se genera la salida, mediante la función de activación.

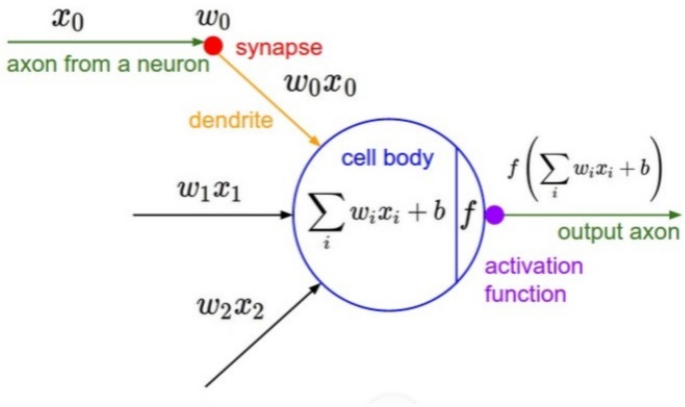

Figura 2.2 Modelo de un Perceptrón o Neurona.

$$result = \sum_{i=1}^{n} x_i \cdot \sum_{i=1}^{n} w_i + b = x_1 \cdot w_1 + x_2 \cdot w_2 + ... + x_n \cdot w_n + b \tag{2.3}$$

Una vez que se ha calculado el resultado de la neurona, éste es transferido mediante la función de activación a la siguiente. Hay muchos tipos de funciones de activación y, teóricamente, cualquiera puede ser usada obteniendo salidas equivalentes. Sin embargo, debido al método de entrenamiento de la red, del cual luego se hará mención, el gradiente juega un papel esencial en el cálculo de los pesos, con lo que la función de activación debe ser diferenciable [7]. Tanto la sigmoide como la tangente hiperbólica comprimen la salida entre dos valores, mientras que la ReLU, *Rectified Linear Unit*, permite un aprendizaje

acelerado siendo una función muy sencilla, Ecuación 2.4. Todas las neuronas de una capa presentan la misma función de activación, de manera que diferentes capas dentro de una misma red pueden tener funciones de activación distintas [7].

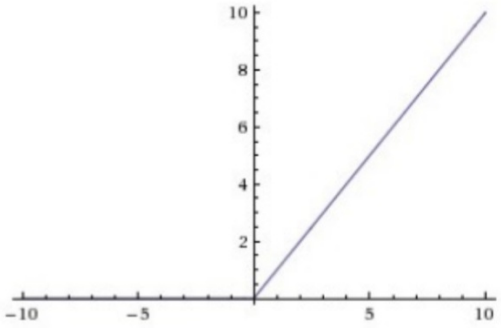

Figura 2.3 Representación de la función ReLU.

$$f(x) = max(0,x) \tag{2.4}$$

Para comprender el entrenamiento de un MLP, se visualiza la Figura 2.4, donde se ha considerado una red neuronal simple con dos pesos. Para cualquier combinación de los dos pesos, la red generará una salida en función del valor de entrada, la cual tendrá asociado un error. Variando los pesos en todos los valores posibles, se genera una "superficie de error". El objetivo del entrenamiento es hallar la combinación de pesos cuyo resultado presente el menor error [7]. En general, las redes neuronales presentan más de dos pesos, con lo que no es posible generar una superficie bidimensional; no obstante, el concepto es extrapolable a dimensiones mayores.

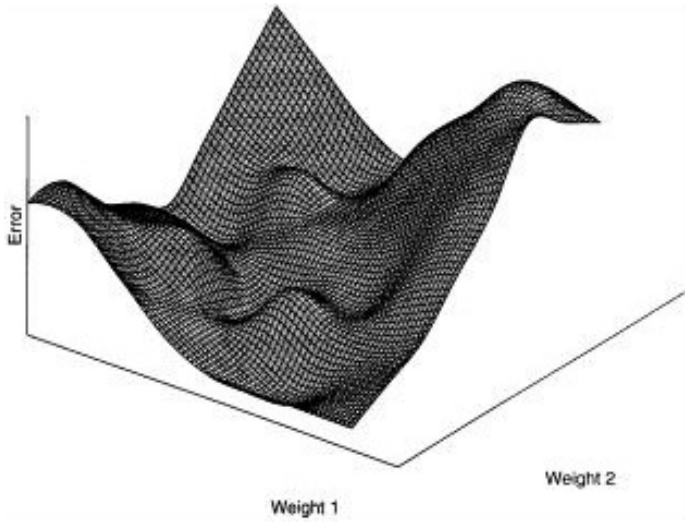

Figura 2.4 Representación de la superficie del error del backpropagation [7].

El algoritmo de entrenamiento de retroproyección, o simplemente backpropagation, usa el descenso del gradiente para localizar el mínimo absoluto dentro del superficie de error. Este procedimiento es el algoritmo menos costoso en términos de computación. Para llevar a cabo el algoritmo, en primer lugar se inicializan los pesos (y el sesgo) a un valor aleatorio, esto es, se situaría en un lugar cualquiera dentro de la superficie. En

este momento, se calcula el gradiente local del punto en el que se encuentra y modifica los pesos de forma que siga la dirección del gradiente local más "inclinado" [7]. Como se aprecia, existe un posible impedimento: que el algoritmo se quede estancado en un mínimo local en lugar del global. Para ello, se implementan dos parámetros dentro del mismo:

- Learning rate: Determina el tamaño de paso tomado en el proceso iterativo del descenso del gradiente [7]. Físicamente, un learning rate mayor permite avanzar más velozmente mientra que uno pequeño permite aproximarse a una zona concreta de forma más pausada dentro de la superficie. En general, es muy recomendable usar un leraning rate alto al principio, para avanzar por la superficie descartando los mínimos locales; y pasada un número de epochs, reducirlo para centrarnos en la zona donde se encuentre el mínimo global.
- Momentum term: Es utilizado para ayudar si el proceso de descenso del gradiente se estanca en un mínimo local, añadiendo una fracción del cambio de los pesos anteriormente realizado al que se está actualizando ahora. Esto se efectúa porque la actualización de los pesos dentro de un mínimo local es poco significativa, al aportar valores de la iteración anterior el cambio permite a los pesos salir de dicho mínimo [7].

Anteriormente, se mencionó el término *epoch*. Una epoch, o época, es cada iteración que se realiza dentro del algoritmo donde se actualizan los pesos [1]. En cada una de ellas, los pesos se van modificando de forma que el error que genera la red se minimice.

No existen reglas para decidir sobre el número de capas que conforman el MLP o el número de neuronas de cada una. Se determinan manualmente y, en general, dependen de la complejidad de la entrada y el mapeo de las salidas, la cantidad de ruido en los datos de entrada y la cantidad de datos de entrada disponibles [7]. De esta forma, tras realizar el entrenamiento, se pueden tener varios escenarios posibles. Si el número de neuronas y de capas es demasiado bajo, el algoritmo de backpropagation no será capaz de converger hasta llegar al óptimo durante el entrenamiento, esto se conoce como *underfitting*. Si, por el contrario, existen muchas neuronas y muchas capas habrá un *overfitting*. Esto implica que la red no aprende las relaciones entre entradas y salidas, ya que no es capaz de discernir el ruido que presentan los datos de entrenamiento. En estos casos, asimilará dicho ruido de manera que en errará en las predicciones que haga. Se dice que las redes neuronales con overfitting no son generalistas [7]. Dado la analogía del perceptrón y la neurona, cuando una red presenta overfitting se asimila a cuando un cerebro memoriza sin comprensión sobre lo que aprende, de manera que carece de la capacidad filtrar los datos erróneos.

El número de parámetros que presenta un perceptrón se corresponde con el número de componentes que tenga el vector de entrada más una unidad, correspondiente al sesgo. Para una red completa, se debe calcular este número y multiplicar por el número de neuronas. Asimismo, como se ha mencionado anteriormente, el algoritmo en sí posee dos parámetros que son elegidos de forma manual. De igual modo, las capas que conforman el MLP así como el número de perceptrones en su interior se deciden de forma manual.

2.2.2 Redes Neuronales Convolucionales

Del apartado anterior, se ha expuesto el conjunto de parámetros que se necesitan entrenar. Sea la entrada de una neurona una imagen a color de 200x200 píxeles, los valores que necesitan ser calculados en cada epoch sería: $200 \cdot 200 \cdot 3 + 1 = 120001$; nótese que el número total de píxeles viene multiplicado por 3 ya que las imágenes se subdividen en 3

campos rojo (R), verde (G) y azul (B), por lo general. Se observa que si extrapolamos esta mecánica a una red completa con múltiples perceptrones en cada capa, los valores que deben calcularse en cada iteración son muy elevados. No es factible que los parámetros que deben entrenarse dentro de una red dependan de la dimensión de la entrada.

Las redes neuronales convolucionales son una familia de redes neuronales artificiales (ANN) adaptadas al tratamiento de imágenes, presentando rasgos heredados de la anterior así como mejoras y avances para obtener una mayor versatilidad. Las CNN tienen un flujo de información directo, y son capaces de extraer características de un set de datos de entrada mediante el uso de estructuras de convolución [18]. Estas redes presentan como elemento esencial el perceptrón como ocurría con las ANN; sin embargo, en las capas convolucionales, los pesos son los elementos que conforman los filtros kernels. Estos kernels, al aplicárselo a las imágenes, obtienen los distintos mapas de características [18]. Con esta nueva estructura se consiguen múltiples mejoras respecto de las ANN:

- Conexión local: Cada neurona ya no está conectada con todas las neuronas de la capa anterior, sino con un grupo reducido de ella. Esto permite una mejora de la eficiencia al reducir los parámetros a calcular y, de estar forma, aumentar la velocidad de convergencia [18].
- Pesos compartidos: Un grupo de conexiones, pueden compartir los mismos pesos, lo que reduce aún más el conjunto de parámetros [18].
- Reducción de la dimensión de los datos muestreados: La capa de pooling permite reducir el tamaño de los datos de entrada mientras retiene la información útil de la misma. Asimismo, puede reducir el número de parámetros desechando información trivial contenida en ellos [18].

Las redes convoluciones presentan 4 parámetros esenciales. Las kernels convolucionales son muy importantes en la extracción de información. Las salidas de cada uno se conocen como mapa de características, *feature map*. Cuando se establece el kernel de un determinado tamaño, se pierde información en los bordes. Por ello, se genera el *padding*, que consiste en rellenar con ceros el borde de los datos de entrada para extenderlos, de forma que se ajusten directamente sobre los datos de entrada.

Para determinar la convolución de los filtros, se utiliza el *stride*, que indica el nivel de solapamiento de las máscaras convolucionales consecutivas (a mayor stride, menor solape, con lo que menor densidad de convolución). Hacia el final del proceso, tras haber pasado por todos los kernels de una capa convolucional, el mapa de características presenta un número elevado de valores que puede ocasionar overfitting. Para evitarlo, se tiene la capa de pooling que permite filtrar las características, obviando la información redundante. De esta forma, existe el max pooling, el average pooling [18] y el min pooling. Con el average pooling, el kernel aplicado sobre una zona calcula la media de los valores a los que afecte, con el primero toma el valor máximo mientras que con el último, el mínimo. Se usan dependiendo de las características a buscar.

Después de una capa convolucional, se tienen las capas de activación cuya labor es analizar el feature map de la capa convolucional anterior, generando una activación si es necesaria. Para las CNN, las capas de activación presentan funciones como la ReLU que se vio en la sección anterior; siendo ésta generalmente la que se usa.

Como se observa en la Figura 2.5, la matriz amarilla a la entrada conformaría la entrada. En primer lugar, se le añade un borde de ceros (padding) para no perder toda la información contenida en los límites. Tras ello, se aplica el kernel de 3x3 (matriz azul), con un stride de 2. De esta forma, se visualiza que a mayor stride, menor es el proceso

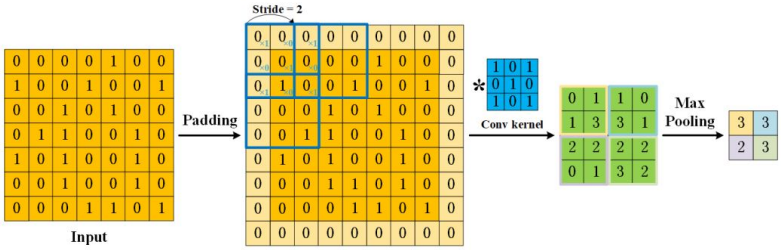

Figura 2.5 Procedimiento de un filtro bidimensional de una CNN [18].

de convolución. El mapa de características corresponde con la matriz verde, a la cual se le aplica un max pooling para que permanezcan los valores mayores. Nótese que cuando se aplica un filtro, el tamaño de la matriz de salida respecto al de entrada se reduce, lo mismo ocurre cuando se aplica una capa de pooling. En general, dentro de una CNN, conforme se apliquen capas convolucionales y de pooling, el tamaño de la matriz que se obtendrá en cada caso irá reduciéndose en largo y en ancho; sin embargo, su profundidad irá aumentando ya que esta dimensión se relaciona al número de kernels que trata la entrada de la capa convolucional. De forma genérica, se va aumentando el número de filtros conforme se profundiza en la CNN, de forma tal que las primeras capas extraigan información genérica de la imagen mientras que los últimos sean unos filtros para datos más exhaustivos. Cuando el proceso de convolución ha finalizado, se tiene una última capa, la fullyconnected, cuya entrada es toda la información extraída en la convolución que, al pasar por una capa de neuronas completamente conectadas, aportan la salida de la CNN.

La visión computarizada basada en inteligencia artificial aúna aplicaciones como la conducción autónoma y el IoT, *Internet of Things*; para las cuales se hace uso de las redes convolucionales. Esto se debe a que las CNN exhiben un mayor rendimiento a la hora de clasificar imágenes o reconocer patrones, en comparación con las ANN [9].

3 Metodología

Dentro de este capítulo, se expondrá la base de datos que se ha usado, comentando la naturaleza de las entradas, el tratamiento que se ha realizado sobre ellas, la selección del algoritmo que estructura nuestra red convolucional y su desempeño.

3.1 Descripción del Conjunto de Datos Utilizado y su Preprocesamiento

Ya se expuso de forma escueta el conjunto de datos que se utilizaría para este estudio en la Sección 1.6. Ahora, se detallará la base de datos empleada, justificando las fechas entre las cuales se verificaron los datos.

La base de datos engloba un conjunto de imágenes all sky a color, por tanto, las imágenes están conformadas por tres matrices RGB. Estas matrices presentan unas dimensiones de 156x156 píxeles. En estas imágenes se visualiza el cielo abierto sin ningún obstáculo dentro de su campo de visión que le impida tomar una captura de la posición del sol. Estas fotografías se realizan con una frecuencia de 1 minuto.

Las fotografías conforman los datos de entrada cuyas etiquetas asociadas son la radiación solar. Así, en un primer experimento se utilizó como conjunto de entrada estas imágenes junto a los datos de irradiancia asociados a ellas. Como se observa en las Figura 1.6 o Figura 3.1 el cielo cambia no sólo en la posición del sol sino también en la nubosidad presente en el ambiente. Ambos parámetros influyen en la radiación solar captada por los aparatos de medida de la estación meteorológica, presentando la primera imagen un valor mayor que la segunda.

Las frecuencias de la cámara y el piranómetro son diferentes: mientras la primera hace una fotografía cada minuto, el otro dispositivo es capaz de dar información de la radiación solar global cada 5 segundos. Por ello, cuando se estructuró la base de datos para posteriormente pasársela a la red se mapeó la radiación a la imagen de forma tal que este valor sea justo el del minuto en el que se hace la imagen, es decir, si una captura es tomada a las 10:09, la radiación asociada a dicha imagen será la de las 10:09:00 descartando los demás samples. No se optó por hacer una media de todo el minuto porque, observando los datos que manejábamos, no se cometía un gran error en el valor de irradiancia, ya que ésta no fluctúa de forma rápida en periodos de tiempo tan cortos.

Si bien es cierto que la base de datos de donde extraemos toda esta información presenta un pequeño desfase temporal entre los diferentes datos captados por la estación. Así, en algunos casos la irradiancia asociada a la imagen es en realidad la correspondiente a la misma un pequeño lapso de tiempo anterior. Como se ha comentado, en general la evolución de la radiación es muy progresiva y no se comete un gran error al despreciar

Figura 3.1 Imagen all sky con el cielo parcialmente nublado.

este desfase. Aunque no hayamos hecho un estudio en profundidad de este fenómeno, se debería tener en cuenta en un futuro proyecto que implemente estos modelos.

Para preprocesar la información contenida en la base de datos, se realizó este mapeado que se ha descrito entre imágenes y radiación. Asimismo, antes de que la CNN manejara los datos, todos los valores de radiación solar se dividieron entre 1.000. Esto se hizo porque, si se observa el conjunto de datos, las radiaciones mayores rondan los 900 W/m^2 pero no superan el millar. De esta forma, todos los valores quedan divididos por una cota superior obteniéndose valores adimensionales y, más importante, en el en torno del cero. Esto tiene sus repercusiones positivas puesto que si observamos la Ecuación 2.3, los valores de entrada multiplican a los pesos directamente; teniéndose valores de entrada más pequeños los valores de los propios pesos tenderán a ser, a su vez, pequeños. Computacionalmente, es más favorable manejar números cercanos al cero ya que se garantiza una estabilidad en el algoritmo.

Los datos a los que se tiene acceso desde la estación meteorológica son muy extensos. Sin embargo, tomaremos una muestra significativa de ellos que datará desde el 7 de marzo hasta el 19 de abril de 2023. Dentro de ese mes, se tiene un total de 18.900 samples de información sobre radiación diferentes habiendo sólo 67 datos que se consideran "dañados", esto es, que al realizar la base de datos, se considera si la imagen presentaba algún error al cargar o si la propia información sobre la radiación presentaba algún fallo. Como se observa el porcentaje de información que se pierde es ínfimo en comparación con las muestras que se van a analizar.

En total dentro de este estudio se realizaron 3 experimentos distintos para los cuales se emplearon tres diferentes bases de datos de entrada. La primera base de datos ya ha sido descrita, la cual consistió en un mapeo de las imágenes con la radiación actual que se relacionaba con las mismas. Para poder llevar a cabo unas redes capaces de hacer predicciones a corto plazo sobre la radiación, se generó una segunda base de datos en la que se relacionaron las fotografías a cielo abierto junto con los valores de radiación actual, como en el caso anterior, y 5 valores de radiación más, correspondientes a la radiación relativa a 1, 2, 5, 10 y 20 minutos en el futuro. En cada entrenamiento, se empleó únicamente un valor de radiación fijando el horizonte temporal en el que se centra la red. De esta forma, se mapeó las fotografías con valores de radiación en distintos momentos previos a que se obtuviera la foto. Por último, se creó la base de datos que incorpora la información del viento. Al segundo set de entrada se le añadieron los datos

de viento actuales: velocidad, medido en m/s, y dirección, en grados, actuales. De igual forma, estos números se normalizaron para que estuvieran en el entorno del 0, con lo que se dividió por 100 los valores de velocidad (en los datos ningún valor superaba la centena), y por 10 para el caso de la dirección (en la práctica, no superaban la centena).

En la realización de las otras dos bases de datos se tiene un total de 18.881 muestras totales de información favorable y 76 datos que estaban dañados por las razones que anteriormente se exponían. Es lógico que al generar una base de datos donde la dependencia es mayor se vea reducida la cantidad de información manejable. Sin embargo, la reducción de la base de datos es prácticamente inexistente.

3.2 Selección de Algoritmos de Inteligencia Artificial

Partiendo de estudios de reconocimiento de características mediante el análisis de imágenes como [20], el autor emplea el primer modelo de red convolucional operativo para un uso práctico, LeNet-5. Este modelo está compuesto de un total de 7 capas de diferente naturaleza: convolucionales, *subsampling*, lo que se corresponderían con las que denotamos como capas de pooling; y fully connected [16]. Este modelo se utilizó en primera instancia para reconocimiento de letras mayúsculas y símbolos de escritura comunes. El modelo consta de las siguientes capas:

- Primera Capa Convolucional y Segunda Capa Subsampling: La entrada son imágenes de 32x32 píxeles. Los kernels que presenta son de 28x28, previniendo de esta forma sesgar la información proveniente de los bordes. La primera capa presenta un total de 6 filtros. Tras ello, en la capa segunda se pasa a un mapa de características de la mitad de filas y columnas, esto es, de 14x14 [16].
- Tercera Capa Convolucional y Cuarta Capa Subsampling: La capa convolucional presenta 16 filtros. Este aumento en el número de kernels que conforman la capa se implementa para que cada conjunto de filtros extraiga parte de la información contenida en la capa de pooling anterior. En consecuencia, se tiene una especialización de los perceptrones según el kernel al que pertenezcan. De la misma forma que antes, la capa de pooling divide por dos las filas y columnas del mapa de características [16].
- Quinta Capa Convolucional: La última capa convolucional presenta un total de 120 kernels, donde cada uno tiene como salida un mapa de características escalar (de dimensión 1x1) [16].
- Sexta Capa Fullyconnected: Esta capa está compuesta por 84 perceptrones (uno por cada clase) y presenta como función de activación la tangente hiperbólica [16].
- Capa de Salida (Output): Con una clase por cada signo ortográfico a identificar, inicialmente fue entrenada para 84 símbolos ASCII diferentes. Esta capa se compone de un función RBF (*radial basis function*) que emplea una clasificación mediante distancia euclídea [16].

Como se observa se tiene al inicio de la red dos bloques de convolución y pooling. Esta estrategia permite a LeNet-5 extraer información de las imágenes de entrada y sintetizar las características que hay en ellas. En la quinta capa, el bloque omite el submuestreo, ya que el mapa de características de cada filtro es escalar, en su lugar se introduce una fullyconnected. De esta forma, la información que ha ido extrayéndose en los dos primeros bloques llegando a un gran detalle se pasa, como un vector, a la fullyconnected donde

se lleva a cabo el análisis de todas las características obtenidas. Finalmente, mediante distancia euclídea se consigue asignar la clase correspondiente a la imagen. Asimismo, los valores de las imágenes de entrada se encuentran normalizadas entre los valores -0.1, correspondiente al blanco de fondo, y a 1.175, que sería negro [16]. Como se mencionó en el Capítulo 3.1, en nuestro caso normalizaremos los valores para que se encuentren en el entorno del cero, ya que es más óptimo y agiliza el entrenamiento, por tanto no se tendrá datos de entrada que sean negativos. De la misma forma, se menciona que la función de activación de los perceptrones es una tangente hiperbólica; como se propuso en el Capítulo 3.1, se empleará una ReLU ya que potencia la velocidad de aprendizaje de la red y aporta mayor consistencia al entrenamiento [20].

No sólo el modelo importa sino también las opciones de entrenamiento con las que se configura el aprendizaje de la red. Por ejemplo, cuando se lleva a cabo una época los datos se encuentran en unas posiciones concretas, si estos permanecen en los mismos lugares en las iteraciones sucesivas, se asume el riesgo de que la red pierda generalidad. Por ello, es conveniente que, para que la mayor cantidad de información pase por el mayor número de conexiones posible, se dé un *shuffle* o reordenación de los datos en cada época. Asimismo, para la LeNet-5 no se tiene en cuenta añadir *dropout*, pero es un método muy efectivo para garantizar una mayor generalización de la red. El *dropout* consiste en la inhabilitación de un porcentaje aleatorio de neuronas en cada época, de manera que las conexiones se recombinan de formas aleatorias, teniendo caminos de información que se *apagan* o *encienden* según la neurona se habilite. En nuestro caso, se han implementado ambas metodologías a la hora de diseñar el entrenamiento del modelo.

Los optimizadores generan el algoritmo que gobierna el comportamiento de la red. Típicamente, define una regla de actualización que viene dada por unos hiperparámetros a elegir [6]. De entre los diferentes optimizadores que nos permite selccionar Matlab en su toolbox de machine learning, escogimos el método de gradiente descente estocástico con momento (SGDM), una modificación del SGD aunando la acción del *momentum*. Este aspecto es un término que recoge una fracción acumulativa de los gradientes calculados en épocas anteriores, lo cual permite una aceleración en el descenso del gradiente cuando la pendiente de la función de pérdida es más abrupta [1], es decir, aumenta la sensibilidad con respecto a la pendiente de la función de pérdidas. Se escogió este método debido a su simplicidad y, sobre todo, a lo intuitivo de sus hiperparámetros para escoger sus valores. Igualmente, en el trabajo [1] se emplea el optimizador ADAMAX, de forma que, tras realizar las experimentaciones, se tendrá una comparación interesante para describir la influencia del optimizador usado sobre la función de pérdidas. También, se tiene indicios de que, en general, el optimizador ADAM no obtiene resultados peores que el SGD [6], con lo que el cambio de optimizador propone una hoja de ruta a la hora de sintonizar los hiperparámetros en función de los resultados comparados de este estudio y el trabajo [1].

En un inicio, se probó con una leve modificación de la LeNet-5, simplemente cambiando la capa final de clasificación por una de regresión. Nótese que nuestro problema se encuadra en el marco propio de las redes de regresión, es decir, tras el aprendizaje de la red, se pretende que ésta tenga la capacidad de devolver un resultado numérico. Por contra, LeNet-5 fue diseñada para reconocimiento de símbolos ortográficos sencillos [16], proceso eminentemente de clasificación. De ahí, que llevásemos a cabo el cambio. A pesar de ello, fue sólo una primera aproximación puesto que los resultados fueron muy pobres.

Las diferencias esenciales entre redes neuronales se asientan en cómo las capas fundamentales están definidas y unidas así como el método de entrenamiento empleado en su aprendizaje [20]. Por tanto, sintetizamos la información observada en trabajos como [1], [16], [17] o [9], y se obtuvo una idea genérica para diseñar nuestro modelo

de red convolucional: las funciones de activación debían ser ReLU para permitir un mayor rendimiento durante el entrenamiento; los kernels, desde la entrada hacia la capa fullyconnected, deben ir aumentando en número y, a su vez, disminuyendo sus mapas de características, permitiendo que la información se vaya compartimentando de forma gradual, adquiriendo con cada convolución un mayor nivel de abstracción. Asimismo, se introducirán el método de droput en la última capa (fullyconnected) para garantizar una mayor generalización, de forma que la capa se adapte a conjuntos de entrada más diversos. De igual forma, se optará por usar capas de maxpooling, para que la red se centre en aquellos resultados mayores, es decir, salidas más energéticas.

El resultado de emplear toda esta información fue un proceso iterativo que nos llevó desde redes con estructura más repetitiva (uso de varios bloques de convolución y pooling seguidos) a más condensados conformen se iban añadiendo los conceptos anteriormente expuestos. La red que finalmente se empleó se presenta en la Figura 3.2. Aunque parezca una red un tanto simple, el modelo fue evolucionando hacia esto ya que el aumentar el número de capas convolucionales y de pooling concluía en un error mayor. Se optó, por tanto, por una capa convolucional, *conv*, que extrae toda la información de la red. Presenta 8 kernels en total con un mapa de características de tamaño 3x3 a la salida, teniendo un stride de 1 y aplicándose padding para que la información de los bordes estuviese contenida. Tras ésta, y antes de llegar a la capa ReLU, se introduce una capa de normalización de un subconjunto o, también llamada *batchnormalization layer*, (*batchnorm*), una técnica que permite normalizar las activaciones de las capas ocultas de una red, ayuda a mejorar la precisión y la velocidad del entrenamiento [2]. Esta capa no afecta sobre la dimensión de los datos pues cambia su valor para que las capas aguas abajo de ella puedan sintetizarlos de forma más fácil y veloz. Tras ello, se presenta la capa de activación correspondiente a un maxpooling, con mapas de características de 2x2. Finalmente, toda esa información se lleva a la capa completamente conectada (*fc*), la cual genera las conexiones necesarias de la información para llevar a cabo los cálculos correspondientes que finalmente se obtendrán como resultado en la capa final, *regressionoutput*. Esta capa final tiene como única función devolver un número real tras el análisis de las imágenes. Toda la información que se ha expresado se puede visualizar en la Figura 3.2, donde se muestra la estructura de la red.

Como se mencionó en la Sección 3.1, las bases de datos se subdividieron atendiendo a 3 casuísticas: imágenes con información de radiación actual, imágenes mapeadas con información de radiación actual y futura; e imágenes mapeadas con información de radiación actual y futura aunado con la información del viento. Para las dos primeras situaciones se utilizó un modelo análogo al de la Figura 3.2, mientras que para el último caso, nos vimos obligados a modificar el modelo. Esto ocurrió debido a la presencia de datos escalares en la entrada, en los casos anteriores las entradas eran imágenes únicamente. Para llevar a cabo esta implementación, se debió introducir una serie de capas de forma que la red se convirtiera en una *gráfica de capas de red*. Esta transición generó una gran problemática pues el manejo de los tipos de datos como entrada era complejo. A pesar de ello, no hay grandes variaciones respecto a cómo se comporta una red neuronal convolucional; sin embargo, Matlab designa de esta forma este tipo de estructuras, dotándolas de un tratamiento especial a la hora de aplicar sobre ella cambios, estructuras de datos a su entrada y añadir o eliminar capas. El modelo completo se observa en la Figura 3.3.

Como se aprecia en dicha figura, en la rama de la izquierda se presenta el modelo que antes discutíamos casi en su totalidad. En este caso, se ha prescindido de la capa fully connected que se tenía en el modelo original; pasando a tener tras de sí una capa de *flatten*. Esta capa tiene como función poder unir datos que presentan diferentes dimensiones

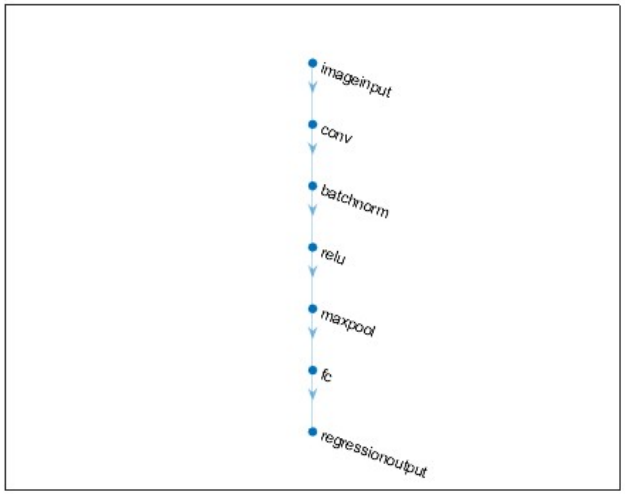

Figura 3.2 Red convolucional empleada para el cálculo de la radiación solar en base a imágenes.

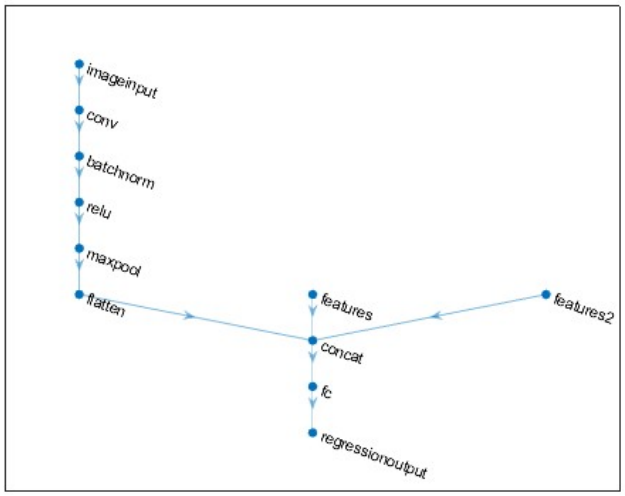

Figura 3.3 Red convolucional empleada para el cálculo de la radiación solar en base a imágenes y datos del viento.

para que de esta forma, la red pueda procesar todos los datos de entrada sin que exista error. Existen a su vez otras dos ramas (*features* y *features2*), que generan completas los datos de entrada para este nuevo modelo, siendo, respectivamente, velocidad y ángulo del viento. Estas tres ramas convergen en una capa de unión (*concat*), que permite aunar toda la información de entrada para que sea llevada a la capa fullyconnected del final. En particular, en esta capa no se emplea un método de inicialización de pesos y sesgo predeterminado, como sí hacíamos en los casos anteriores, sino que se parte de los valores que cada modelo obtuvo para cada experimento en concreto. Se profundizará más en este tema en el Capítulo 4.

3.3 Evaluación de Modelos

Aunque en el Capítulo 4 se haga una discusión más detallada de lo que se expondrá, se va a introducir en este apartado el conjunto empleado para la validación de los modelos, así como la función de pérdidas empleada para evaluar el rendimiento de las redes neuronales implementadas.

De los conjuntos de datos que se mencionaron en la Sección 3.1, el 90% de los datos se utilizaron para el entrenamiento del modelo mientras que el 10% restante se empleó en el testeo de las CNN. Dentro del set de entrenamiento, este conjunto se dividió, a su vez, en datos dirigidos a entrenamiento y datos usados en la validación interna del modelo, que el propio entrenamiento utiliza para mejorar su rendimiento y permitir una mejora del modelo durante esta etapa. Estos conjuntos no son observables desde el punto de vista del usuario, a pesar de haber buscado documentación sobre cómo Matlab lleva a cabo este proceso, ha sido imposible encontrar de forma exacta el porcentaje con el que se reparte estos subconjuntos.

Como función de pérdidas, se utilizó la raíz del error cuadrático medio, cuya expresión se encuentra en la Ecuación 3.1, donde $\hat{y_i}$ representa los valores predichos por la red e y_i los valores reales de los samples empleados en el testeo. Se empleó esta función puesto que da un valor realista sobre el error cometido entre el modelo y la realidad, el uso del MSE (equivalente al RMSE pero sin raíz cuadrada), puede llevarnos a equívoco pues si se manejan valores pequeños, éste reflejará un error mucho más bajo al estar al cuadrado. Además, el RMSE presenta buenas propiedades matemáticas a la hora de ser usado en el cálculo de la pendiente para la actualización de los pesos y el sesgo en cada epoch.

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y_i} - y_i)^2}{n}} \tag{3.1}$$

4 Implementación y Experimentos

4.1 Detalles de la Implementación del Modelo

Como ya se ha visualizado en las Figuras 3.2 y 3.3, el modelo que se implementó originalmente fue una simplificación de modelos anteriores extraídos de documentación y ensayos como la LeNet-5. La estructura del modelo quedó fijada, con las diferentes capas que se muestran en dichas figuras; siendo la CNN empleada para la ingesta de datos matriciales y escalares conjuntamente, la evolución lógica del modelo original anterior. Para llevar a cabo los entrenamientos se tuvo en cuenta diferentes aspectos como el tiempo de realización y la configuración de los hiperparámetros. En la Sección 4.2, se detalla expresamente las decisiones tomadas sobre los valores asociados a cada parámetro.

La experimentación empleada para generar los modelos fue llevada a cabo de la misma forma:

- En primer lugar, se generaba la base de datos con la que se fuera a trabajar durante el entrenamiento del modelo. Así, se utilizó la base de datos que únicamente contenía información sobre la radiación solar asociada a cada imagen para un instante concreto según la frecuencia que se describió en la Sección 3.1. Para la siguiente etapa del estudio, se conformó la base de datos en la que se mapeó cada imagen no sólo con la información de radiación en el instante actual de tiempo sino también con los valores de irradiancia de 1, 2, 5, 10 y 20 minutos atrás en el tiempo. Finalmente, se construyó el último conjunto a la que a cada imagen, a su vez, se le asociaba los datos escalares de dirección y velocidad del viento.
- Cuando la base de datos quedaba construida, se generaba el modelo. A partir de ahora, se hablará de *modelo en una rama* cuando se haga referencia a la red neuronal de la Figura 3.2; y el de la Figura 3.3 se le llamará *modelo en tres ramas*. Para las dos primeras bases de datos se utilizaron modelos en una rama mientras que para el último conjunto se empleó un modelo en tres ramas.
- Para el entrenamiento se hacía la diferenciación dentro del conjunto basado en los porcentajes comentados en la Sección 3.3. Esta segmentación entre conjunto de entrenamiento y conjunto de validación se mantendrá en la segunda y tercera parte del estudio, en el que se llevan a cabo la labor de predicción, para que, de esta forma, los de resultados de las distintas redes sean extrapolables y, sobre todo, comparables.
- Una vez extraído los diferentes conjuntos se pasa a configurar los parámetros propios del entrenamiento, los cuales se detallarán en la Sección 4.2.

- Una vez terminado el entrenamiento, se realizaba la fase de testeo y se evaluaba el RMSE de cada CNN.

4.2 Configuración de Parámetros

En este apartado se ahondará sobre las decisiones tomadas sobre ciertos parámetros tanto del modelo de las CNN implementadas como de los propios experimentos que se han realizado. En primer lugar, haremos unos breves comentarios sobre las redes neuronales. Se ha comentado en varias secciones que las funciones de activación empleadas en este estudio son ReLU debido a su mejor desempeño a la hora de calcular la actualización de pesos y sesgo así como por su contribución a carga computacional que eliminan del proceso. Igualmente, en las capas de pooling se optó por un maxpooling para descartar aquellas salidas menos energéticas, centrándonos, de esta forma, en aquellos valores más elevados.

En cuanto al experimentos se refiere, ya comentamos diferentes aspectos de ellos en la Sección 3.2. El optimizador seleccionado fue el SGDM debido al conocimiento previo que teníamos de éste. Sabiendo que partimos de un optimizador que puede tener un rendimiento un tanto peor respecto al ADAMAX usado en [1], se optó por utilizar un learning rate exponencial. Ya se explicó que este parámetro ajusta el tamaño de paso en el proceso para el cálculo del gradiente [7], el tipo exponencial permite tener una evolución exponencial descendente con el número de épocas. De esta forma, comenzamos con un learning rate de valor elevado y conforme el entrenamiento avanza va decrementando un cierto exponente; con ello, se consigue que la red avance por la superficie del error (Figura 2.4) saltando los mínimos locales y avanzando hacia el mínimo global. En este caso, se impuso un learning rate inicial de 0,00003, cuyo valor fue encontrado mediante prueba y error, la fórmula que sigue este parámetro es la siguiente:

$$learning\ \ rate = 0.00003 \cdot 0.75^{\alpha} \tag{4.1}$$

Donde α es un factor relacionado con la frecuencia de las épocas realizadas. El learning rate se actualiza a un 75% de su valor anterior cada 12 epochs. Este valor se calculó en función del número de epochs totales que se realiza por experimento. De esta forma, con esta tasa de actualización el parámetro se actualizaba 25 veces en total, asegurándonos de esta forma que el entrenamiento no se estanque en mínimos locales. Se debe prestar mucha atención a este parámetro pues influye mucho sobre la estabilidad del experimento. Mediante diferentes pruebas se descubrió el valor de partida pues si lo aminorábamos un orden de magnitud la curva del RMSE tendía a inestabilizarse y llegar a valores muy altos, mientras que si lo hacíamos mayor la tasa de actualización era tan pobre que el error tendía a estancarse y apenas se veía modificado.

El número de iteraciones máximas es de 300 epochs en total. Se probó con números redondos, aumentando de 50 en 50, en las primeras experimentaciones hasta llegar a un valor en el que el RMSE tendía hacia un régimen permanente. Se comprobó empíricamente que con un máximo de 300 iteraciones el error conseguía evolucionar hasta estabilizarse con bastante holgura antes de terminar el entrenamiento con lo que se decidió fijar dicho valor. Además, matemáticamente, es más favorable un número divisible entre 2, 3, 5, 6 y 10 para llevar a cabo diversas operaciones con él, como el factor exponencial calculado anteriormente.

Como se describió en la Sección 3.3, se tomó un 90% de los datos totales de la base de datos para la fase de entrenamiento. Como en la base de datos original se tienen 18.900 samples totales, el conjunto llevado a entrenamiento es de 17.010 muestras

totales. Con este valor, hicimos el cálculo para dar valor al parámetro *minibatchsize* de Matlab. Este factor determina el tamaño del minilote que se usa en cada epoch, donde un minilote es un subconjunto de muestras que se utilizan en el cálculo para la actualización de los pesos y el sesgo. De esta manera, se emplea un grupo reducido de muestras pertenecientes al conjunto de entrenamiento para evaluar el gradiente de la función de pérdidas, ahorrándose efectuar dicho cálculo para un número significativo de muestras. A pesar de que este número es aconsejable que sea grande puesto que así se toman más muestras para realizar los cálculos y, con ello, se obtiene una visión más global de cómo evoluciona la red, por motivos de carga computacional se decidió un valor de 1.134. Para llegar al conjunto completo de entrenamiento, sería necesario multiplicar por 15; éste es el número de samples que efectivamente se utiliza para realizar los cálculos de la actualización. En un principio, se tomaron número más reducidos pero el entrenamiento podía llegar a alargarse más de dos y hasta 3 días con lo que se optó por llegar a esta solución de compromiso.

Asimismo, se impuso una tasa de validación interna del modelo de 30 épocas. Por lo que, durante un experimento, se llevarán a cabo 10 validaciones para que la red ratifique que la evolución que se está produciendo con las actualizaciones garantiza cumple con los samples reservados para tal fin. Igualmente, como se mencionó en la Sección 3.2, en cada iteración se procede a un shuffle del conjunto de muestras antes de emplearlas en el experimento, para garantizar la generalidad de la red y que ésta no "aprenda de memoria" los resultados.

4.3 Experimentos Realizados y Resultados Obtenidos

El procedimiento detallado en la Sección 4.1 se sigue cada vez que se va a realizar un entrenamiento. Como también se ha comentado, el estudio se dividió en tres etapas. En este apartado, se va a exponer cómo se realizó cada experimento y qué parámetros se extrajeron de cada red para que sirviera como base para la red siguiente.

En primer lugar, se diseñó un experimento con la base de datos inicial en la que únicamente existía el mapeo de una serie de imágenes, que abarcan desde inicios de marzo hasta mediados de abril de 2023, con la radiación que el instante actual leían los sensores. En este conjunto se tratan un total de 18.900 samples diferentes, de los cuales 17.010 se utilizarán para generar el entrenamiento y las muestras restantes se emplearán en la verificación del modelo. Se generó la estructura de la red y, tras ellos, se introdujeron los valores de los diferentes parámetros según se ha explicado en la Sección 4.2. Un aspecto importante que no se ha comentado hasta ahora es la inicialización de los pesos y el sesgo de las capas de CNN que vamos a generar. Este paso será distinto en cada una de las iteraciones realizadas y se insistirá en ello cada vez que se exponga un experimento.

Para el primer experimento se utilizó una inicialización de valores del tipo *HE initializer*, método muy extendido cuando se utilizan funciones de activación de tipo ReLU. El procedimiento consiste en asignar a los pesos unos valores según una función normal cuya varianza se calcula en base al número de entradas de cada capa, por lo tanto, cada capa presentará diferentes pesos. Es un procedimiento muy útil que se utilizó por primera vez en el estudio [10] y, con él, se garantiza que el gradiente quede acotado y no tome ni valores muy elevados ni muy reducidos, con el fin de que el entrenamiento converja.

El experimento, cuyo tiempo de entrenamiento abarcó entorno a 2 días, obtuvo un resultado de RMSE de 0,0777. Este fue el resultado más ajustado que se pudo obtener con los parámetros implementados como se ha especificado anteriormente. Si observamos

una semana en particular, como se ilustra en la Figura 4.1, tenemos una visión global de cómo la red es capaz de estimar el valor de radiación en función de las imágenes de entrada. Se ha escogido una semana con diferentes tipos de perfiles de radiación (con y sin nubes) para poder observar mejor el comportamiento de la predicción (en azul). La red, por lo general, se ajusta bien a la tendencia que presenta la curva de radiación real (en rojo). De hecho, centrándonos en el día 1 y 3 de abril, se puede observar una bajada abrupta de la radiación, provocada por la entrada de nubes, que sigue de manera favorablemente la predicción de la red. Este hecho es más notorio en el primer día de abril pues la radiación genera unos picos, quizá por nubes pequeñas, que es capaz de seguir la red. Si bien, nuestra CNN genera bastantes picos cuando la radiación se encuentra en sus valores máximos, como se observa en el día 2 de abril. Sin embargo, es curioso que cuando el valor de radiación máximo alcanzado es menor, como ocurre el día 5, la red tiene valores más estables y que se aproximan mucho mejor a la curva real. Asimismo, el error cometido por la noche no se anula, hecho que sería óptimo, porque prestando atención en los *valles* que se generan en la curva real, la predicción en dichos momentos es pequeña pero no nula.

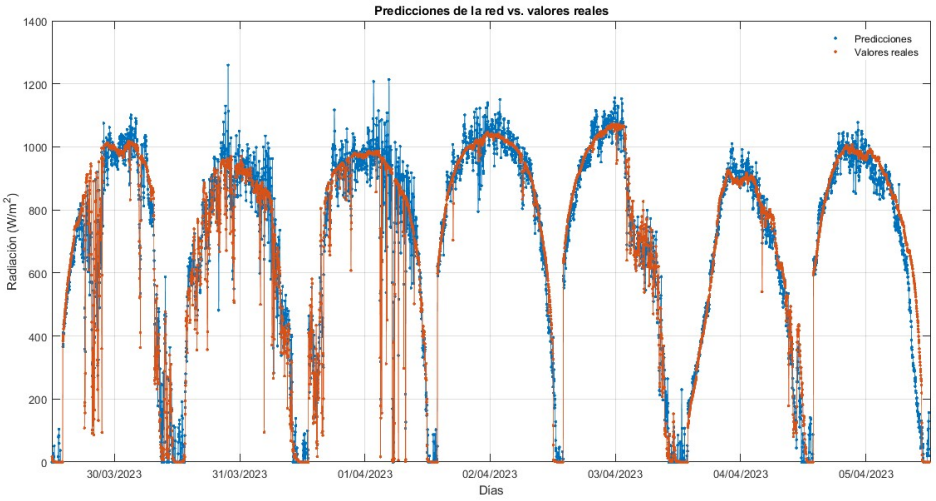

Figura 4.1 Estimación de la radiación actual generada por la red en contraposición con los valores reales de radiación.

Para la segunda etapa del estudio, dedicado a la predicción, se procedió de la siguiente manera: en lugar de inicializar los parámetros según un método como el HE, utilizamos los resultados de la etapa anterior. Así, para la red convolucional capaz de predecir la radiación dentro de un minuto se toma la red resultado del experimento anterior y se entrena con la nueva base de datos. Nótese que los nuevos datos de entrada son imágenes mapeadas a un vector de radiaciones que va desde el valor de la irradiancia actual hasta dentro de 20 mnt, con lo que en cada experimento se debe concretar cual es el valor que debe tener mapeado. Para el siguiente ensayo, se pretende obtener una red que prediga en un plazo de 2 minutos para lo cual se toma el modelo de predicción de un minuto y se entrena con las entradas correspondientes. Se procede con esta metodología en cada iteración construyendo el nuevo modelo sobre el resultado de la experiencia previa, con este proceso, nos ahorramos carga computacional derivado de una inicialización más pobre, como con el método HE, ya que de este modo la red tiene cierto conocimiento y es capaz de adaptarse de mejor forma a una modificación en las

muestras de entrada. En las figuras que siguen se visualizan el funcionamiento de cada modelo:

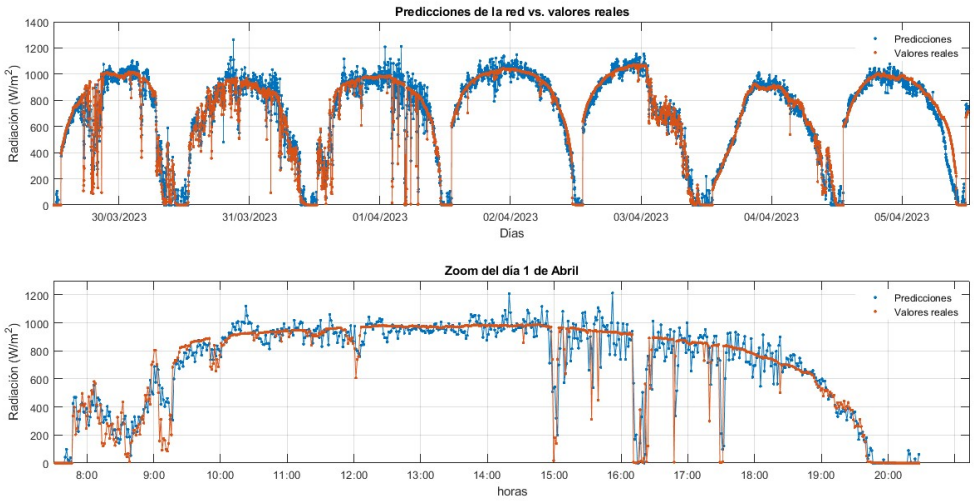

Figura 4.2 Predicción generada por la red en contraposición con los valores reales de radiación (predicción en un 1 minuto).

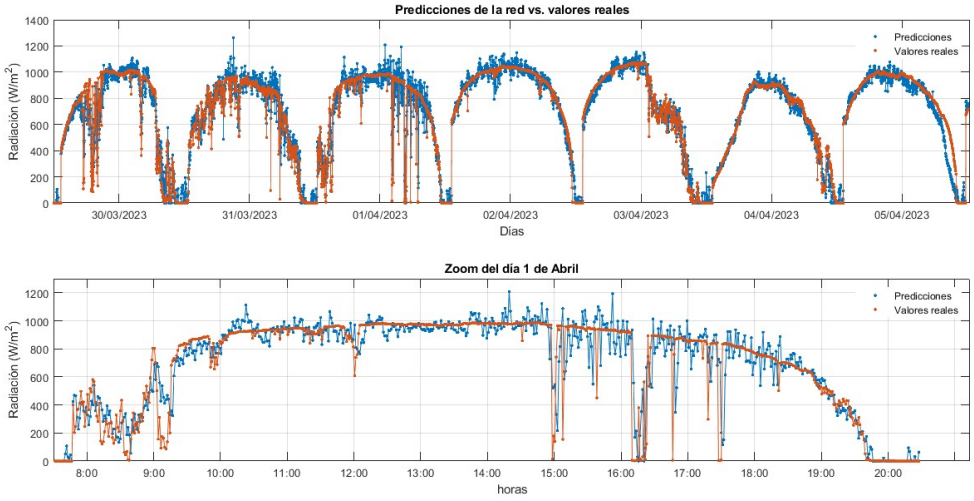

Figura 4.3 Predicción generada por la red en contraposición con los valores reales de radiación (predicción de 2 minutos).

Por lo general, no hay cambios significativos a grandes rasgos. Ocurre como en el experimento anterior con los valores fluctuantes cuando se alcanza la radiación máxima en el último día de marzo y en los dos primeros de abril. Igualmente, se tienen unos valores que no llegan anularse cuando la irradiancia es cero, en los periodos nocturnos. Este hecho ocurre debido a que en realidad la red no era capaz de procesar que en los periodos nocturnos la irradiancia se hacía cero y por tanto la predicción debía ser tal durante esas horas. Para compensar este defecto, en el código que calcula el error se introdujo una cota en el resultado de la red, lo que implicó una mejora sustancial del RMSE al no permitir que las predicciones puedan alcanzar valores negativos. La red, durante el entrenamiento, genera predicciones tanto positivas como negativas en

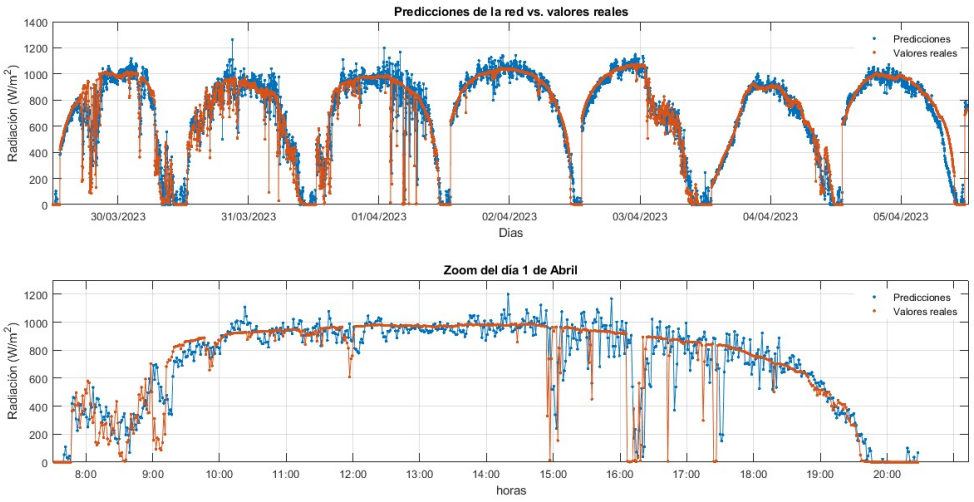

Figura 4.4 Predicción generada por la red en contraposición con los valores reales de radiación (predicción de 5 minutos).

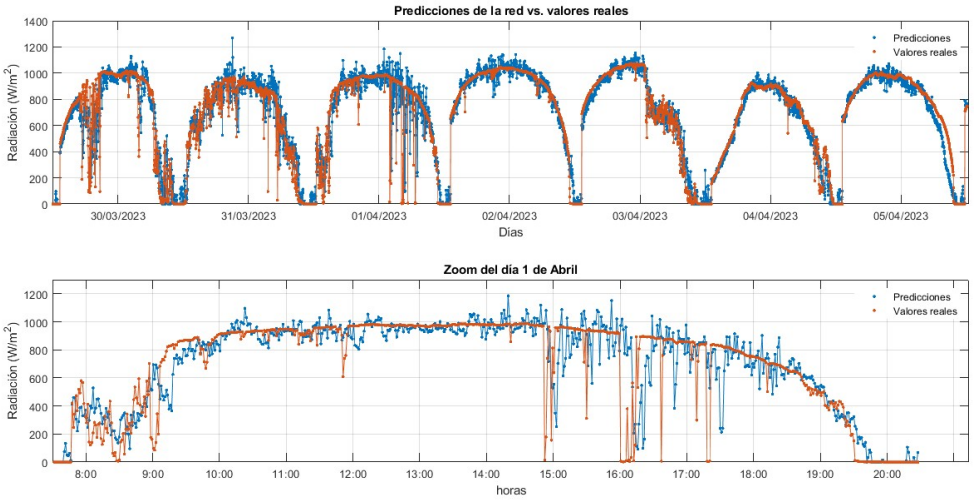

Figura 4.5 Predicción generada por la red en contraposición con los valores reales de radiación (predicción de 10 minutos).

esos periodos de manera que la media tiende a 0, al restringir los valores a resultados positivos, las predicciones positivas que la red genera en compensación con las positivas permanecen.

Asimismo, en esta etapa de estudio, se introduce un zoom de cómo evoluciona la radiación en el día 1 de abril. Se ha elegido esta fecha al ser un caso paradigmático en el que existe una gran fluctuación de la irradiancia debido a la nubosidad. Se puede observar, sobre todo en la zona entre las 9:00 y las 10:00, que es dónde se ve claramente que el aumento del horizonte predicción provoca una predicción más pobre. Así, en la Figura 4.2, la gráfica azul refleja muy bien la evolución de la curva naranja; no obstante, hacia el final de la secuencia estudiada, en la Figura 4.6, se observa que la tendencia de la irradiancia real no es captada de forma tan exacta. De hecho, es en los momentos de mayor cambio, amanecer y atardecer, donde se dan las mayores discrepancias. Igualmente, en los huecos que ocurren en las horas centrales del día son predichas casi fielmente hasta

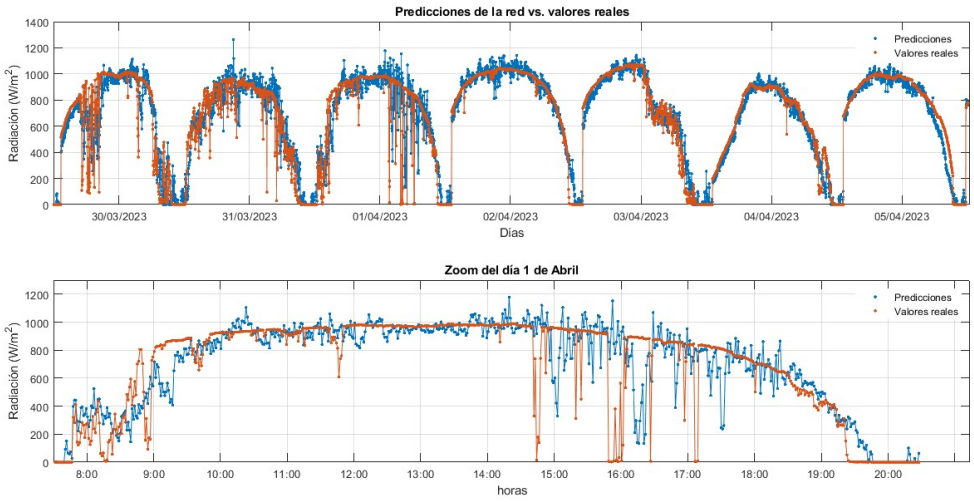

Figura 4.6 Predicción generada por la red en contraposición con los valores reales de radiación (predicción de 20 minutos).

que el horizonte es de 5 minutos, cuando éste se extiende, la red neuronal presenta dificultades para seguirlo, teniendo una respuesta casi retardada cuando horizonte es máximo.

Aunque se ahonde más en el tema en la Sección 5.1, cabe destacar que el error va aumentando conforme el horizonte temporal de predicción agranda. Esto es lógico pensarlo puesto que la predicción en sí presenta una incertidumbre mayor conforme el plazo de predicción se alarga. En la Figura 4.7 se visualiza la evolución del error en función del horizonte de predicción para la semana del 30 de marzo al 5 de abril mostrada en las figuras anteriores. En ella, se observa que el RMSE tiene un aumento progresivo conforme el horizonte temporal se hace más grande; incluso, se asemeja a una trayectoria lineal, salvando el cambio entre los 2 y los 5 minutos. A pesar de este aumento, los errores ciertamente no son elevados conforme el horizonte se amplía, lo que implica que son modelos competentes para la predicción. Las diferencias entre errores oscilan entre 0,078 y 0,0120, teniéndose valores relativamente aceptables para las primeras redes convolucionales cuyos horizontes son pequeños, y valores de error mayores que tienen cierto margen de mejora, como es el caso de los últimos modelos.

En última instancia, se llevó a cabo el experimento en el que los inputs de las redes convolucionales eran una combinación de información matricial y escalar. Se entiende que, al introducir información de distinta naturaleza, la red puede tener una visión más general de la situación y, de esta forma, generar una predicción más certera que las que se han tenido en los casos anteriores. Antes de exponer detalles sobre los experimentos realizados, vamos a comentar cómo se generaron los modelos. Al igual que en la etapa anterior, partiremos de una inicialización de los parámetros de las neuronas que basándonos que los modelos que llevamos entrenados hasta ahora. En este caso, como se observó en la Figura 3.3, la modificación que se hace con respecto al modelo que generaba el entrenamiento a partir de imágenes es la introducción de dos ramas que permiten la inclusión de los datos escalares, además de una parte de posprocesado cuando estos datos se han unido. Por tanto, parece lógico usar como punto de partida los pesos de los respectivos modelos que ya se han entrenado en cada una de las nuevas redes a generar. De esta manera, la forma de proceder fue la siguiente: para el modelo de predicción de 1 minuto cuyas entradas son imágenes y datos escalares, se puso como

inicialización los pesos extraídos de la red que predice en un horizonte temporal de 1 minuto cuando la entrada son únicamente imágenes. Se aplicó para cada uno de los modelos obteniéndose el funcionamiento que se visualiza en las Figura 4.8, Figura 4.9, Figura 4.10, Figura 4.11 y Figura 4.12.

Vemos, de igual forma, cómo evoluciona el valor del error conforme el horizonte temporal de predicción aumenta; pero, en este caso, haremos una comparación con el RMSE que se tenía anteriormente:

Aunque parezca que las predicciones apenas cambian en un inicio, se tiene una mejora sustancial en el error que se comete cuando se tiene como parámetros de entrada los datos relacionados con el viento. Como se ve en la Figura 4.13, aunque con un horizonte de predicción de 1 minuto, el error es casi el mismo, conforme el horizonte de predicción aumenta se observa una reducción del RMSE. De hecho, esta disminución es tanto mayor cuanto mayor es el horizonte de predicción, con lo que la deriva del RMSE generada por una ampliación de este horizonte se ve mermada en gran medida.

Al igual que se decidió hacer en la experimentación anterior, debajo de cada gráfica se ha incluido un zoom correspondiente al día 1 de abril. A grandes rasgos, se observa lo que ocurría en la etapa anterior, en los primeros experimentos la predicción es capaz de seguir en gran parte la evolución que presenta la irradiancia real pero en lo últimos casos, cuando el horizonte es tan amplio, la predicción de los modelos se traducen en valores que parecen presentar un pequeño retardo y no se ajustan bien a la tendencia real.

Nótese que los errores que salen son muy pequeños, esto se debe a la utilización de valores normalizados, hecho que se empleó desde un primer momento en el entrenamiento y se siguió en la evaluación de los modelos. Es por ello que en el eje de ordenadas de las Figuras 4.7, 4.13 y 5.1 aparecen las unidades $10^{-3}W/m^2$ ya que en realidad dicho error hay que multiplicarlo por un factor de escala de mil para obtener el RMSE real que comete cada CNN. Se tendrá en cuenta cuando se comparen los modelos en secciones posteriores.

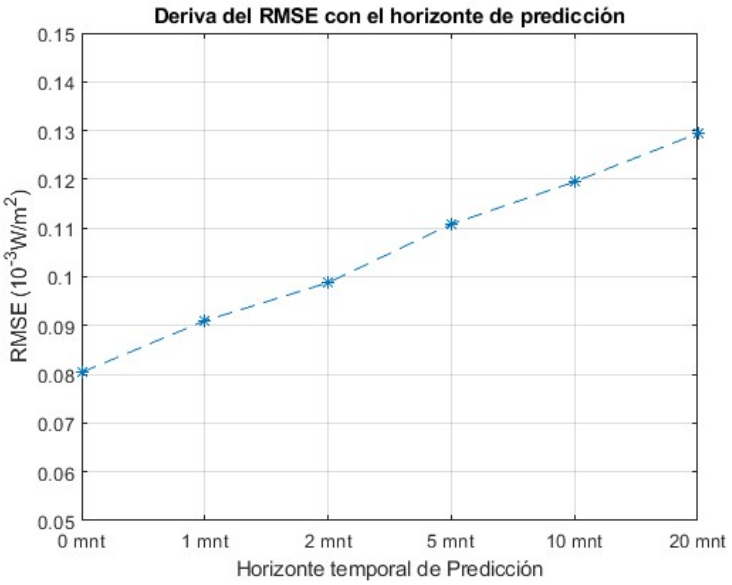

Figura 4.7 Evolución del RMSE en función del horizonte temporal de predicción (para redes cuya entrada son imágenes exclusivamente.

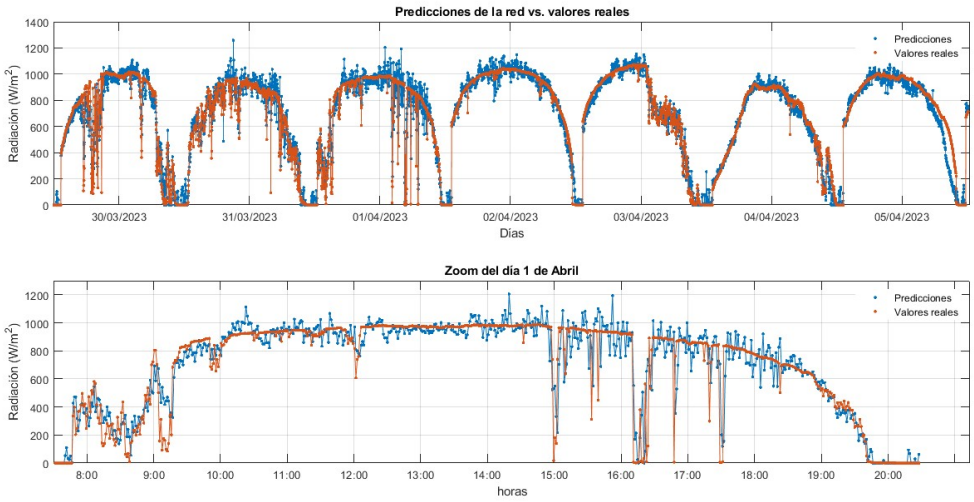

Figura 4.8 Predicción generada por la red en contraposición con los valores reales de radiación (predicción de 1 minuto).

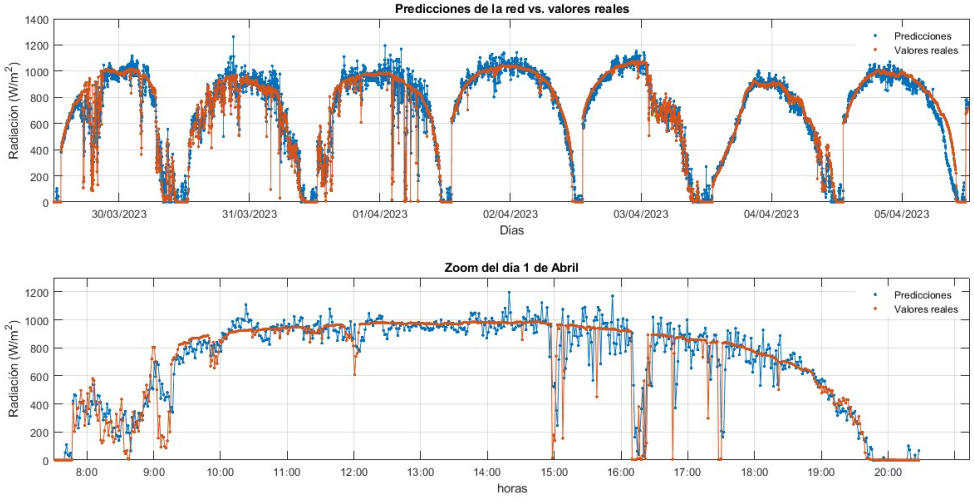

Figura 4.9 Predicción generada por la red en contraposición con los valores reales de radiación (predicción de 2 minutos).

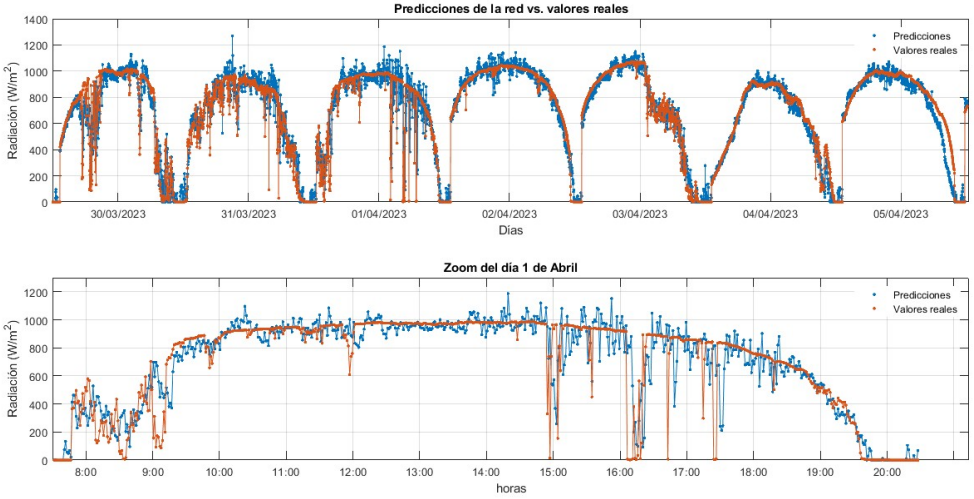

Figura 4.10 Predicción generada por la red en contraposición con los valores reales de radiación (predicción de 5 minutos).

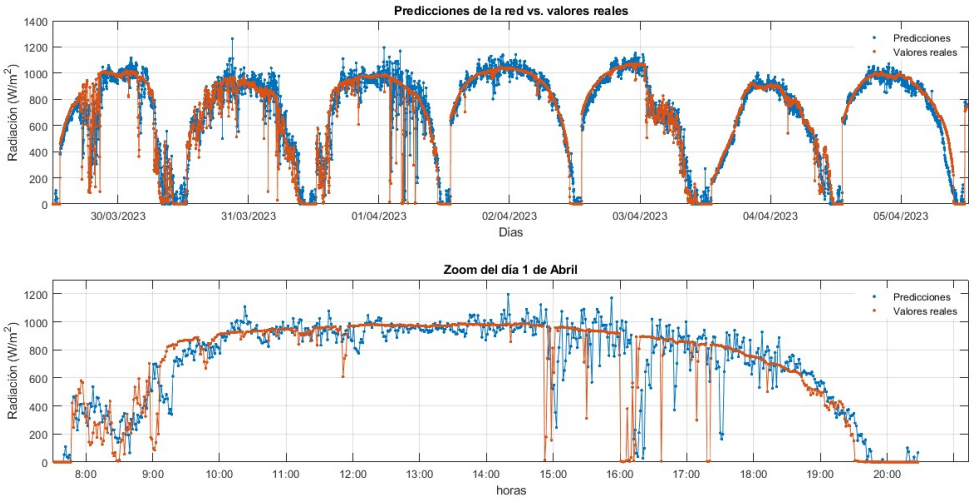

Figura 4.11 Predicción generada por la red en contraposición con los valores reales de radiación (predicción de 10 minutos).

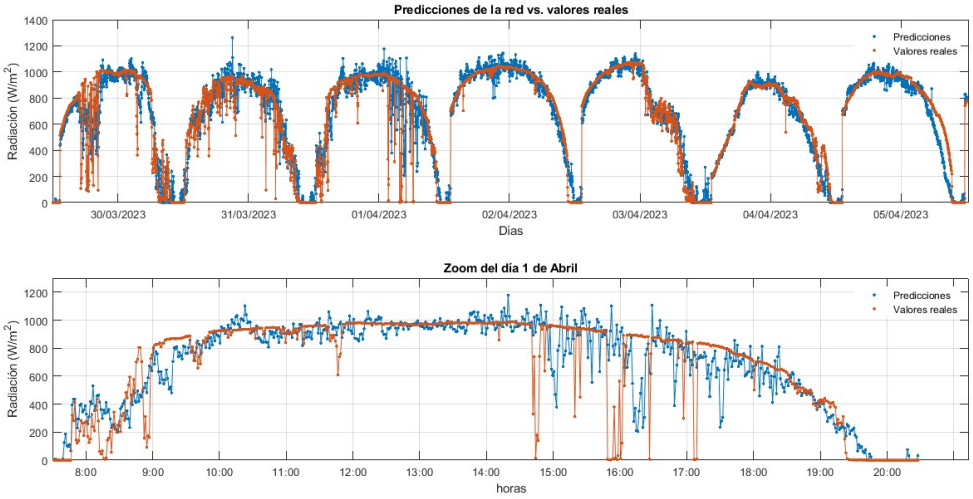

Figura 4.12 Predicción generada por la red en contraposición con los valores reales de radiación (predicción de 20 minutos).

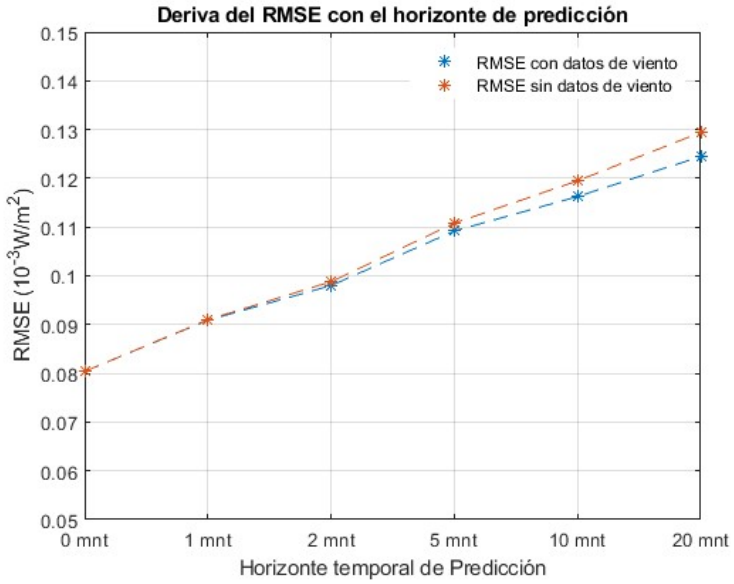

Figura 4.13 Comparación de la evolución del RMSE en función del horizonte temporal de predicción para la semana del 30 de marzo al 5 de abril.

5 Análisis de Resultados

5.1 Interpretación de los resultados obtenidos

Aunque en la sección 4.3 se comentó parte de los resultados obtenidos, en esta sección se hablarán de ellos de forma más detallada. En primer lugar, se debe aclarar que la Figura 4.13 presenta valores de errores semanales, esto es, el RMSE que los diferentes modelos experimentan para una muestra concreta dentro del conjunto de validación. Por ello, se va a exponer a continuación una gráfica en la que se observará el RMSE cometido por cada red según el horizonte de predicción que presente, así como si se tuvo como entradas los datos de viento o no.

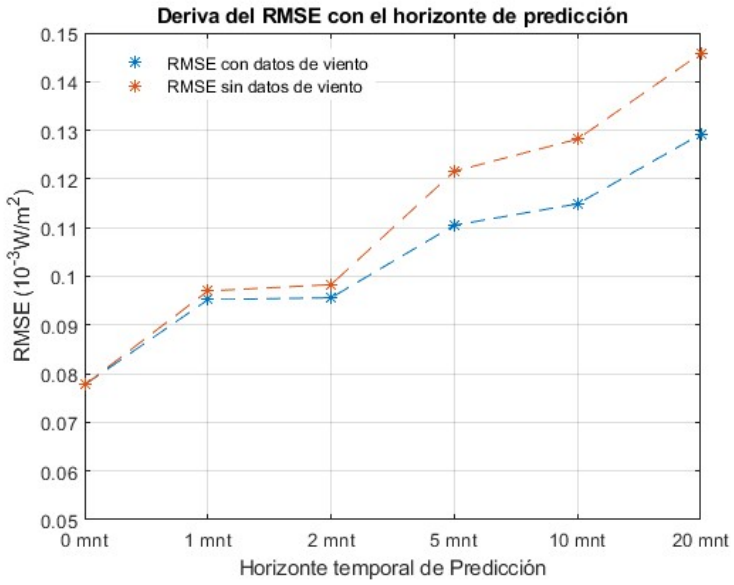

Figura 5.1 Comparación de la evolución del RMSE en función del horizonte temporal de predicción.

En general, los valores de esta gráfica aumentan con respecto a la de la sección anterior ya que se tomaba una muestra concreta en la que el error puede reducirse. En esta figura, los valores sí son del conjunto de validación completo y se aprecian mucho mejor los cambios significativos que aporta la introducción de los datos de viento. Conforme el horizonte de predicción aumenta, la curva azul, que representa el RMSE del conjunto de validación usando los datos de viento, alcanza valores que son mucho menores. Esto

implica que se tienen predicciones mucho más certeras cuando se usan datos como la dirección y la velocidad del viento en la fase de entramiento. Se debe destacar que si bien mejora para todos los experimentos, en cuanto al caso particular en el que el horizonte es nulo (estimación de radiación actual) se obtiene una muy leve mejora en el sexto decimal del error. Puesto que en este estudio se toman 4 decimales en los errores, esta mejora es despreciable y apenas visible respecto al resto de situaciones. En este caso no aporta mejora en la función de pérdidas puesto que para predecir la radiación actual no es necesario conocer la evolución de las nubes del entorno, si no la situación del cielo actual.

A pesar de este hecho singular, el conocimiento de dos magnitudes relacionadas con el viento tiene un gran impacto como se puede apreciar en la Figura 5.1. Teniéndose una reducción del error máxima del 88,6% para el caso más desfavorable estudiado, cuyo horizonte temporal asciende a 20 minutos. Partiendo de un modelo bien entrenado, las CNN que nacen de él obtienen resultados muy competentes capaces de establecer predicciones con un margen de error pequeño. De hecho, esto es extrapolable con cualquier magnitud que se relacione con el clima. Este hecho ya se puntualizó en la Sección 2.1 donde se vieron varios métodos tradicionales de predicción de radicación, de esta forma, aunando más información climática (temperatura ambiente, temperatura de la placa, humedad, horas de luz, ...) se pueden generar modelos con una precisión mayor que combinen la inteligencia artificial y el estudio sobre los modelos climáticos para poder dar valores útiles a las industrias basadas en generación de electricidad mediante tecnología fotovoltaica.

Cabe esperar que los modelos presenten errores similares con datos nuevos pues como ya se comentó, se ha abogado por el uso de métodos que garanticen la generalización de las redes creadas. Asimismo, se prevé que sean competentes y que generen una predicción no sólo precisa sino también rápida, pues durante las experimentaciones llevadas a cabo, las CNN eran capaces de procesar el conjunto de validación en cuestión de segundos.

5.2 Comparación con métodos tradicionales

Se ha destacado el hecho de la localización a la hora de predecir la radiación y el uso de los modelos; pero, salvando este concepto, en este apartado se va a hacer referencia a diferentes estudios que aparecen en el trabajo [26].

Lo que se aprecia grosso modo son dos hechos principales. En primer lugar los modelos que se emplean para generar predicciones son muy simples, obteniendo el valor a calcular en función de uno, dos, o a lo sumo tres parámetros relacionados con la climatología del lugar (temperatura ambiente, horas de sol, humedad, ...). Algunas de estas correlaciones se han visualizado en la Tabla 2.1 de la Sección 2.1, es obvio que necesitan ser simplistas pues están pensados para llevar a cabo los cálculos mediante computación básica o incluso manualmente. Esto da lugar al segundo hecho, presentan eficiencias muy pobres, si tomamos por eficiencias la evaluación de los errores de los diferentes modelos. En la siguiente tabla, se muestran algunos extractos de los estudios mencionados en el trabajo [26]:

En los tres primeros ejemplos se usó un modelo basado en la máxima y la mínima temperatura ambiente en un periodo de 6 años. Se empleó para la estimación de la radiación solar global en Ibadán, Nigeria. Los experimentos que le siguen son seis modelos basados en la correlación lineal Angstrom-Prescott y modificaciones cuadrática, cúbica, exponencial, logarítmica y de exponente para la estimación de la radiación global

Tabla 5.1 Extracto de Modelos empíricos basados en factores climáticos y sus correspondientes RMSE [26].

Modelo	RMSE
Original Hargreaves	4,55
Hargreaves models with linear regression	1,59
Hargreaves models with power regression	1,62
Linear	1,3073
Quadratic	1,1997
Cubic	1,1425
Exponential	1,4223
Logarithmic	1,6393
Power	1,2390

horizontal media durante un mes. Para ello, hacía uso de las horas de luz en Jaipur, La India [26]. Como se aprecia ninguno de los valores que se tiene en la Tabla 5.1 presenta un RMSE por debajo de la unidad. Estos datos dados en W/m^2 quedan muy lejos de los estudios en los que se han implantado modelos basados en inteligencia artificial como son [1] o éste mismo. La introducción de modelos de aprendizaje automático refuerza la idea que se comentó en la Sección 2.1 ya que da lugar a la unión de múltiples parámetros de entrada, además de distinta naturaleza, para proporcionar a la red una visión más completa del entorno garantizando una predicción mucho más acertada que métodos basados en correlaciones simples.

Asimismo, las correlaciones que se han construido presentan horizontes de predicción muy amplios, mensuales o trimestrales, por ejemplo. Otra ventaja de utilizar modelos basados en aprendizaje automático es la versatilidad en el horizonte de predicción que presentan. Si bien es cierto que para el caso de estudio que se ha llevado a cabo quizá un horizonte de predicción de 20 minutos genera poco interés en el ámbito industrial o comercial eléctrico, pero este se puede alterar a antojo del usuario ya que variará según los datos que se les introduzca a los modelos a entrenar en la fase de entrenamiento. En el caso de las correlaciones, al ser la mayoría deducidas de forma empírica, presentan coeficientes que deben respetar los horizontes para los cuales se calcularon. Dentro del entrenamiento, el propio proceso iterativo modifica estos parámetros sin necesidad de que el usuario lleve a cabo ningún cálculo más que la selección de los datos de entrada y los hiperparámetros del método de entrenamiento seleccionado y la estructura de la red, para lo cual este estudio ya ha profundizado en dichas ideas.

5.3 Discusión sobre la Efectividad y Eficiencia de los Modelos de Inteligencia Artificial

A pesar de que se ha hecho mención en algunas secciones de este trabajo, es importante tener en cuenta la particularización de los modelos. Las redes neuronales que se han creado a lo largo de este estudio se han hecho mediante unos datos particulares en un periodo de tiempo concreto. Por ello, los modelos generarán predicciones acertadas y serán útiles siempre que los conjuntos que evalúen vengan dados por imágenes como las que se han analizado hasta ahora. Como durante el proceso de entrenamiento, los

modelos sólo han sido entrenados con cámaras all-sky en una posición concreta, si se tuvieran como entrada otras imágenes incluso con la cámara girada, los resultados que proporcionaría presentarían errores. Por ello, es importante recalcar que los modelos que se han generado tienen un fuerte carácter situacional y en caso de cambios sustanciales en la los datos de entrada, se deberían llevar a cabo reprocesos y reentrenamientos de las CNN.

Igualmente, el conjunto de validación que se utilizó es un poco reducido pues no asciende ni a 2.000 muestras; sin embargo, todos los modelos operan con gran velocidad con ellos generando valores en cuestión de segundos. Se espera que el aumento de la evaluación de datos no tenga gran repercusión porque, en términos computacionales, la etapa más costosa viene dada por el entrenamiento que ha llegado a durar 2 días en los casos en los que tardó más.

Aún con las limitaciones que se han descrito, los modelos entrenados han conseguido superar las expectativas iniciales que se planteaban previas a la realización de este estudio. Si bien es cierto que no se puede comparar de forma directa los modelos cuyo horizonte temporal de predicción es distinto de 0, la red que sirve de base para todos estos experimentos, que se visualiza en la Figura 3.2, presenta un error cuadrático medio menor de la unidad, eficacia que supera los modelos que usamos de comparación del trabajo [1]. Se ha validado la efectividad de las redes de predicción para el conjunto de validación y se han obtenido RMSE que fluctúan entorno los valores recogidos en la Figura 5.1, ya que son ventanas del conjunto que presentan menos muestras y por tanto el error, que viene ponderado en función del número de muestras que se evalúan, tiende a ser mayor. Sin embargo, las desviaciones quedan acotadas y no hay valores que difieran mucho de las magnitudes reales.

Cabe destacar el hecho de las incongruencias puntuales que presentan ciertas muestras de la base de datos. Para ahondar en ello, vamos a llevar a cabo la observación de las 10 peores estimaciones que proporciona la CNN que predice la radiación actual:

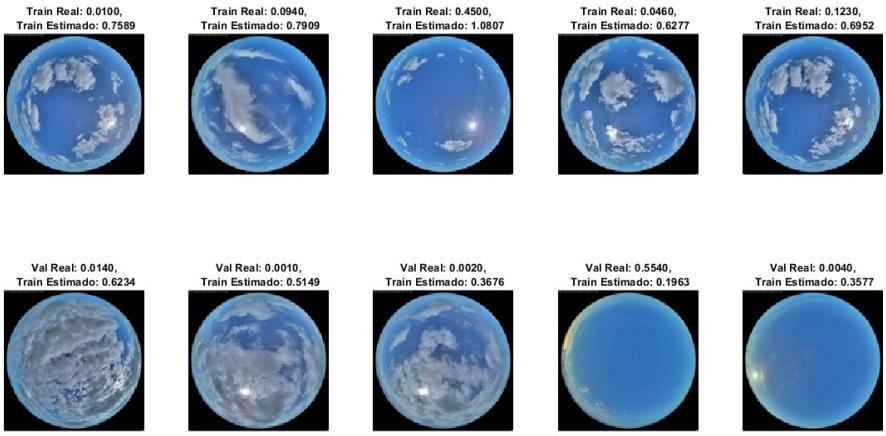

Figura 5.2 Diez peores estimaciones para la red que predice la radiación actual.

Si enumeramos del 1 al diez las imágenes empezando por la fila superior y recorriendo las imágenes de izquierda a derecha, las imágenes 1 y 5 son ciertamente parecidas, en ambas se ve que el cielo se encuentra muy nublado y además el sol se está poniendo. Aunque las estimaciones en ambos casos se encuentren alejadas del valor que corres-

ponde, debemos fijarnos en el valor real, lo que en la figura se denomina como *Train Real*. Este número debe ser multiplicado por mil como ya hemos comentado para obtener el valor de radiación real. Para el primer caso tendríamos 10 W/m^2 y para la imagen cinco sería de 123 W/m^2. Como se ha destacado ambas imágenes reflejan una situación muy parecida; no obstante, el valor de radiación que expresan es bastante dispar el uno del otro. Se observa que en fotogramas que se corresponden con el amanecer, como la última y la penúltima, la estimación empeora, como se ha comentado anteriormente en los momentos como el atardecer y el amanecer, los modelos presentan las mayores desviaciones. De hecho, para la imagen 10 el sol se encuentra más alto y la radiación asociada a ella es menor que en el caso de la novena foto. Con todo, se visualizó una muestra muy significativa del conjunto imagen-radiación que posteriormente se empleó para el entrenamiento de las redes y, aunque existen estos fenómenos adversos en los que algunas correspondencias parecen erróneas, no siguen una correspondencia o patrón claros, por lo que no fue posible de discriminarlas o apartarlas del conjunto. De hecho, es un suceso que afecta de forma directa y muy negativa al rendimiento de los diferentes modelos que entrenamos pues están aprendiendo datos de forma errónea; sin embargo, no son muy numerosos y su efecto se compensa con el resto de imágenes que sí generan un mapeado entre imágenes y valores de radiación directa coherente.

Por ello, se puede afirmar que los modelos que en este estudio se han presentado son competentes a la hora de generar una predicción, aunque también debe tenerse en cuenta que son redes que necesitan de varias horas de entrenamiento para procesar dicho conjunto. Por ejemplo, para generar la imagen que antes se mostraba, en la cual procesa el conjunto completo de entrenamiento consistente en 17.010 muestras, tardó aproximadamente 2 minutos y 20 segundos, con lo que es probable que con conjuntos de datos mayores tienda a dilatarse más en el tiempo durante su procesado.

6 Conclusiones

6.1 Resumen de los Hallazgos

Durante este proyecto se han generado múltiples modelos con diferentes intereses, obteniendo CNN cuyo principal objetivo ha sido obtener predicciones certeras sobre la radiación directa dado una imagen del cielo en varios horizontes de predicción. Como se ha analizado a lo largo del estudio, los modelos que se han entrenado presentan una gran eficacia a la hora de predecir dicho valor. Se ha estudiado cómo la introducción de parámetros de distinta naturaleza para enriquecer los datos de entrada durante la etapa de entrenamiento mejora el rendimiento de la red, permitiendo trabajar con errores menores y acotándolos conforme este horizonte aumenta.

Cabe destacar la simplicidad de los modelos generados, puesto que como se observan en las Figuras 3.2 y 3.3, las CNN no presentan una estructura de capas muy compleja. Al inicio del estudio, partiendo de los modelos ya explicados, se pensó que la solución óptima tendría una estructura mucho más compleja y basada en varios bloques conformados por capas de convolución y subsampling; sin embargo, la disminución en el valor de la función de pérdidas conforme avanzábamos en los entrenamientos, nos guio a una configuración muy simple que, aunándolo con las estrategias que se han comentado como con el método de entrenamiento escogido, nos han llevado a modelos competentes y útiles en el ámbito de la predicción. Ha sido una idea que durante del estudio nos ha sorprendido puesto que se pensaba que la extracción de características de imágenes necesitaba de metodologías más complejas. Esto implica que el tratamiento de imágenes y la extracción de características de ellas para un tratamiento posterior es un proceso más sencillo de lo que se esperaba, desde el punto de vista del aprendizaje automático.

Los resultados obtenidos son coherentes pues están acorde con lo que nos dicta la lógica: conforme el horizonte temporal de predicción aumenta, las predicciones presentan más incertidumbre y tienden a tener un mayor error; no obstante, introduciendo datos relacionados con el viento, los modelos son capaces de adaptar la predicción y obtener un resultado más ajustado si cabe.

6.2 Contribuciones del Trabajo

A lo largo de este trabajo, se han implementado modelos capaces de hacer predicciones a corto plazo sobre la radiación solar. Se ha estudiado la capacidad de mejora de dichas CNN mediante la ingesta de datos de diferente naturaleza: imágenes (matriciales), velocidad y dirección del viento (escalares). Con todo ello, se han implementado modelos

predictivos capaces de prever el comportamiento de la irradiancia solar ajustándose muy bien a las curvas reales. Estos modelos contribuyen tanto sobre el sector industrial, empresas dedicadas a la generación, como a nivel individual, en el autoconsumo. Haciendo uso de modelos que presentan horizontes de predicción más ajustado que los que actualmente se utilizan, propician un uso más eficiente de la electricidad, pudiendo solventar parte de las problemáticas mencionadas en el 1.3. Asimismo, impulsan un desarrollo más sostenible pues permiten almacenar aquella energía sobrante en horas punta y ayudan a tener un control más exhaustivo sobre el balance eléctrico de la red, creando un modelo más eficiente de generación de potencia, coordinada entre las diferentes plantas.

6.3 Limitaciones y Posibles Direcciones Futuras de Investigación

A continuación, se comentará aquellos avances que se podrían implantar en este proyecto para obtener resultados mejores o beneficios mayores. En lo referente a metodologías de entrenamiento, se podría ahondar en los resultados finales a los que se llega cuando se utilizan procedimientos como el ADAMAX, es decir, modificar el ya utilizado SGDM para obtener cambios relevantes en el entrenamiento que puedan mejorar la efectividad de la red.

Si bien ha sido interesante de forma experimental haber trabajado con horizontes de predicción tan reducidos, de cara a un tratamiento industrial sería más lógico tener predicciones diarias, o a lo sumo horarias en función del alcance de la empresa. En general se usan modelos basados en información a nivel mensual, con lo que este cambio propiciaría una mejora sustancial de la información a extraer de las plantas que exploten.

Asimismo, sería muy importante ampliar el conjunto de datos de entrenamiento, llevándolo a un total de un año para que se tengan modelos capaces de predecir de forma competente la radiación en cualquier momento del año. De igual forma, ayudaría a mejorar mucho la discriminación de los samples que se describieron en la Sección 5.3, donde la información de la radiación no casaba con lo que la imagen proyectaba. Asimismo, es muy recomendable ampliar los datos de entrada para en la fase de entrenamiento, ya que como se ha concluido, una mayor diversidad de valores sobre el entorno permite al modelo generar una estimación más acertada de la radiación solar. Entre los parámetros que están disponibles, se recomienda en gran medida la humedad, puesto que afecta en la conversión de radiación directa en radiación difusa, o la temperatura ambiente, puesto que el efecto fotoeléctrico se ve mermado conforme ésta aumenta y es un parámetro que debe tener un control muy exhaustivo.

Entre las futuras investigaciones, cabe destacar que el empleo de una consecución de fotogramas como inputs de la red neuronal se traduciría en una mejora sustancial del modelo. De esta forma, las CNN en realidad entrenarían con *vídeos* de cómo evoluciona el entorno, con lo que no sería necesario tener una estación de viento. Otra vía interesante que se contempla consiste en realizar estimaciones en puntos alejados de la cámara. Desplazando la cámara lejos del pirheliómetro, se podría entrenar el modelo para que aprendiera a estimar las medidas de este último. Así, una única cámara podría cubrir la DNI a lo largo de un espacio extenso, es decir, se generarían modelos que podría reducir la cantidad de sensores necesarios en una planta.

De igual forma, recomendaría el uso de ordenadores más potentes para la fase de entrenamiento de los modelos. En especial, esta etapa ha sido muy crítica debido al modelo que se ha usado ya que, no sólo tardaba debido a las claras limitaciones en cuanto a rendimiento, sino también al hecho de que en muchos experimentos se ha

recalentado llevando a abortar algunos o a ejecutarlos en periodos nocturnos cuando la temperatura era menor.

Una posible utilización de la información extraída en este estudio sería el empleo de los modelos de predicción para el control, la gestión, la operación y la unión de un campo fotovoltaico con la red eléctrica para solventar las problemáticas descritas en la Sección 1.3. Sería interesante diseñar un sistema autosostenible capaz de verter a la red cuando ésta lo demande y llevarlo a baterías de pilas en aquellos periodos donde la generación no cese, pero la demanda se encuentra cubierta debido a otras tecnologías.

7 Anexos

7.1 Códigos Fuente Utilizados

Código 7.1 Código para la generación de la base de datos empleada en el experimento de la red neuronal que calcula la radiación actual.

```
clear
% Ruta de las carpetas
ruta_base_imagenes ='C:\Users\juanm\OneDrive - UNIVERSIDAD DE SEVILLA\
    TFG\camara de cielo';
folder_radiacion = 'C:\Users\juanm\OneDrive - UNIVERSIDAD DE SEVILLA\TFG
    \camara de cielo\2023_meteo';

% Generar lista de fechas
fecha_inicio = datetime(2023, 03, 07);
fecha_fin = datetime(2023, 04, 19);
fechas = fecha_inicio:fecha_fin;

% Leer los nombres de los archivos y cargar datos
imagenesEst = {};
info_imagenes = [];
radiacion = [];
info_radiacion = [];
imagenes_danadas = [];
rad_danadas = [];

tic
    for fecha = fechas
        toc
        tic
        fecha
        fecha_str = datestr(fecha, 'yyyymmdd');
        folder_imagenes = fullfile(ruta_base_imagenes, fecha_str);


            % Cargar datos de radiación
```

```
            num_dia = days(fecha - datetime(2023, 01, 01)) + 1; %poner la
                 otra fecha si no
            archivo_radiacion = sprintf('meteo_2023_%03d.txt', num_dia);
            try
            opts = detectImportOptions(fullfile(folder_radiacion,
               archivo_radiacion), 'Delimiter', '\t'); % Asegúrate de
               usar el delimitador correcto
            opts.VariableNamesLine = 0; % Si hay encabezados en la
               primera línea
            opts.MissingRule = 'fill';
            opts = setvartype(opts, 'double');
            opts = setvartype(opts, 1, 'char'); % Establecer la primera
               columna como tipo 'char'
            datos_radiacion = readtable(fullfile(folder_radiacion,
               archivo_radiacion), opts);
            catch
            fprintf('Error al cargar los datos del día: %s\n',
               archivo_radiacion);
               % Almacenar fecha la rad dañada
               rad_danada = struct('fecha', fecha_str, 'hora', 25, '
                  minuto', 61, 'segundo', 61);
               rad_danadas = [rad_danadas; rad_danada];
               continue;
            end


            % Convertir la columna de tiempo a segundos
            tiempo = datos_radiacion.Var1;
            segundos = zeros(size(tiempo));
            for j = 1:numel(tiempo)
               t = datevec(tiempo{j}, 'HH:MM:SS');
               segundos(j) = t(4) * 3600 + t(5) * 60 + t(6);
            end
            datos_radiacion.Var1 = segundos;


        archivos_imagenes = dir(fullfile(folder_imagenes, '*.jpg')); %
            Ajusta la extensión de archivo según corresponda
        n = numel(archivos_imagenes)


        for i = 1:n
         if(mod(i,50)==0)
            i
         end

          % Intentar cargar imagen
            try
               img = imread(fullfile(folder_imagenes, archivos_imagenes(
                  i).name));
```

```
        catch
            fprintf('Error al cargar la imagen: %s\n',
                archivos_imagenes(i).name);
            % Almacenar fecha, hora y minuto de la imagen dañada
            info_danada = struct('fecha', fecha_str, 'hora', [], '
                minuto', []);
            tokens = regexp(archivos_imagenes(i).name, '(\d{2})', '
                tokens');
            if numel(tokens) >= 2
                info_danada.hora = str2double(tokens{1}{1});
                info_danada.minuto = str2double(tokens{2}{1});
            end
            imagenes_danadas = [imagenes_danadas; info_danada];
            continue;
        end
        % Redimensionar y normalizar imagen
        img = imresize(img, [128 128]); % Cambia el tamaño según lo
            que necesites
        img = double(img) / 255;


        % Almacenar fecha, hora y minuto de la imagen cargada
        info_img = struct('fecha', fecha_str, 'hora', [], 'minuto',
            []);
        tokens = regexp(archivos_imagenes(i).name, '(\d{2})', 'tokens
            ');
        if numel(tokens) >= 2
            info_img.hora = str2double(tokens{5}{1});
            info_img.minuto = str2double(tokens{6}{1});
        end
        if (info_img.hora<7 || info_img.hora>22)
            continue;
        end



        % Obtener el valor de radiación correspondiente a la imagen
            actual

        idx= info_img.hora * 3600 + info_img.minuto * 60;
        flag=0;
        for z=1:size(datos_radiacion.Var12)
            if(datos_radiacion.Var1(z)==idx)
                valor_radiacion = datos_radiacion.Var12(z);
                flag=1;
            end
        end
        if flag==0
```

```
                fprintf('Error al cargar los datos del día %s a la hora %
                    s, minuto %s y segundo %s\n', archivo_radiacion,t(4),t
                    (5),t(6));
                % Almacenar fecha, hora y minuto de la rad dañada
                  rad_danada = struct('fecha', fecha_str, 'hora', [], '
                      minuto', [],'segundo', []);
                    rad_danada.hora = info_img.hora;
                    rad_danada.minuto = info_img.minuto;
                    rad_danada.segundo = 0;

                rad_danadas = [rad_danadas; rad_danada];
                continue;
            end

            imagenesEst{end+1} = img;
            info_imagenes = [info_imagenes; info_img];
            radiacion = [radiacion; valor_radiacion];

            % Almacenar fecha, hora, minuto y segundo de los datos de
                radiación
            info_rad = struct('fecha', fecha_str, 'hora', [], 'minuto',
                [], 'segundo', []);
            info_rad.hora = info_img.hora;
            info_rad.minuto = info_img.minuto;
            info_rad.segundo = 0;
            info_radiacion = [info_radiacion; info_rad];

        end
    end

N = length(imagenesEst); % número de imágenes
image_size = size(imagenesEst{1}); % tamaño de la imagen

% Pre-allocation
imagenes = zeros([size(imagenesEst{1}), N]);

% Llenando el array
for i = 1:N
    imagenes(:, :, :, i) = imagenesEst{i};
end
```

Código 7.2 Código empleado para generar la red neuronal inicial que es capaz de estimar la radiación actual.

```
%% Esta parte se lanza solo una vez
radiacion=radiacion/1000; %hacerlo solo una vez

% Dividir los datos en conjuntos de entrenamiento y validación
tic
```

```
numImgs=size(imagenes,4);
idx = randperm(numImgs);
N = round(0.9 * numImgs); % Usa el 90% de los datos para el
    entrenamiento
idxTrain = idx(1:N);
idxVal = idx(N+1:end);

%ejecutar la primera vez nada mas para no cambiar los datos de
    validacion
XTrain = imagenes(:,:,:,idxTrain);
YTrain = radiacion(idxTrain);
XVal = imagenes(:,:,:,idxVal);
YVal = radiacion(idxVal);
toc

%% Esta segunda parte se ira modificando

% Definir la arquitectura de la red
tic
Arquitectura inicial de la que partimos
layers = [
    imageInputLayer([128 128 3])
    convolution2dLayer(3,8,'Padding','same', 'WeightsInitializer', 'he')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    fullyConnectedLayer(1, 'WeightsInitializer', 'he')
    regressionLayer];
toc

% Especificar las opciones de entrenamiento
tic

options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.00015, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod', 12, ...
    'LearnRateDropFactor', 0.75, ...
    'MaxEpochs',300, ...
    'MiniBatchSize', 1134, ... %para tener en torno a 15 iteraciones por
        epoca
    'Shuffle','every-epoch', ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
toc


% Continuar el entrenamiento anterior
% Extraer las capas de la red
```

```
% layers = net.Layers; %extrae los valores de las capas anteriores
tic
% Entrenamiento
[net,info] = trainNetwork(XTrain, YTrain, layers, options);
toc
% Predecir los valores de radiación en los datos de validación
YPred = predict(net, XVal);
[m,~] = size(YPred);

for h=1:m
    if YPred(h) < 0
        YPred(h) = 0;
    end
end

% Calcular el RMSE
rmse = sqrt(mean((YPred - YVal).^2));
fprintf('RMSE en los datos de validación: %.4f\n', rmse);

% Graficar la pérdida de entrenamiento y validación en escala logarí
    tmica
figure
semilogy(info.TrainingLoss)
hold on
semilogy(info.ValidationLoss)
hold off
legend('Pérdida de entrenamiento', 'Pérdida de validación')
xlabel('Epoch')
ylabel('Pérdida')

% Graficar las predicciones y los valores reales
figure
plot(YPred, '.-')
hold on
plot(YVal, '.-')
hold off
legend('Predicciones', 'Valores reales')
xlabel('Número de muestra')
ylabel('Radiación')
title('Predicciones de la red vs. valores reales')
```

Código 7.3 Código para la generación de la base de datos empleada en el experimento de la red neuronal que calcula la radiación en un horizonte temporal de 1, 2, 5, 10 y 20 minutos haciendo uso de imágenes.

```
clear
% Ruta de las carpetas
ruta_base_imagenes ='C:\Users\juanm\OneDrive - UNIVERSIDAD DE SEVILLA\
    TFG\camara de cielo';
```

```
folder_radiacion = 'C:\Users\juanm\OneDrive - UNIVERSIDAD DE SEVILLA\TFG
    \camara de cielo\2023_meteo';

% Generar lista de fechas
fecha_inicio = datetime(2023, 03, 07);
fecha_fin = datetime(2023, 04, 19);
fechas = fecha_inicio:fecha_fin;

% Leer los nombres de los archivos y cargar datos
imagenesEst = {};
info_imagenes = [];
radiacion1 = [];
radiacion2 = [];
radiacion3 = [];
radiacion4 = [];
radiacion5 = [];
radiacion6 = [];
info_radiacion = [];
imagenes_danadas = [];
rad_danadas = [];

tic
    for fecha = fechas
        toc
        tic
        fecha
        fecha_str = datestr(fecha, 'yyyymmdd');
        folder_imagenes = fullfile(ruta_base_imagenes, fecha_str);


            % Cargar datos de radiación
            num_dia = days(fecha - datetime(2023, 01, 01)) + 1; %poner la
                 otra fecha si no
            archivo_radiacion = sprintf('meteo_2023_%03d.txt', num_dia);
            try
            opts = detectImportOptions(fullfile(folder_radiacion,
                archivo_radiacion), 'Delimiter', '\t'); % Asegúrate de
                usar el delimitador correcto
            opts.VariableNamesLine = 0; % Si hay encabezados en la
                primera línea
            opts.MissingRule = 'fill';
            opts = setvartype(opts, 'double');
            opts = setvartype(opts, 1, 'char'); % Establecer la primera
                columna como tipo 'char'
            datos_radiacion = readtable(fullfile(folder_radiacion,
                archivo_radiacion), opts);
            catch
            fprintf('Error al cargar los datos del día: %s\n',
                archivo_radiacion);
                % Almacenar fecha la rad dañada
```

```
            rad_danada = struct('fecha', fecha_str, 'hora', 25, '
                minuto', 61, 'segundo', 61);
            rad_danadas = [rad_danadas; rad_danada];
            continue;
        end


        % Convertir la columna de tiempo a segundos
        tiempo = datos_radiacion.Var1;
        segundos = zeros(size(tiempo));
        for j = 1:numel(tiempo)
            t = datevec(tiempo{j}, 'HH:MM:SS');
            segundos(j) = t(4) * 3600 + t(5) * 60 + t(6);
        end
        datos_radiacion.Var1 = segundos;


    archivos_imagenes = dir(fullfile(folder_imagenes, '*.jpg')); %
        Ajusta la extensión de archivo según corresponda
    n = numel(archivos_imagenes)


    for i = 1:n
     if(mod(i,50)==0)
        i
     end

      % Intentar cargar imagen
        try
            img = imread(fullfile(folder_imagenes, archivos_imagenes(
                i).name));
        catch
            fprintf('Error al cargar la imagen: %s\n',
                archivos_imagenes(i).name);
            % Almacenar fecha, hora y minuto de la imagen dañada
            info_danada = struct('fecha', fecha_str, 'hora', [], '
                minuto', []);
            tokens = regexp(archivos_imagenes(i).name, '(\d{2})', '
                tokens');
            if numel(tokens) >= 2
                info_danada.hora = str2double(tokens{1}{1});
                info_danada.minuto = str2double(tokens{2}{1});
            end
            imagenes_danadas = [imagenes_danadas; info_danada];
            continue;
        end
        % Redimensionar y normalizar imagen
        img = imresize(img, [128 128]); % Cambia el tamaño según lo
            que necesites
        img = double(img) / 255;
```

```
% Almacenar fecha, hora y minuto de la imagen cargada
info_img = struct('fecha', fecha_str, 'hora', [], 'minuto',
    []);
tokens = regexp(archivos_imagenes(i).name, '(\d{2})', 'tokens
    ');
if numel(tokens) >= 2
    info_img.hora = str2double(tokens{5}{1});
    info_img.minuto = str2double(tokens{6}{1});
end
if (info_img.hora<7 || info_img.hora>22)
    continue;
end



% Obtener el valor de radiación correspondiente a la imagen
% actual, dentro de 1 mnt, 2 mnt, 5mnt, 10 mnt y 20 mnt

for l=1:6
    if(l<4) idx(l) = info_img.hora * 3600 + (info_img.minuto +
         (l-1)) * 60;
    elseif(l<6) idx(l) = info_img.hora * 3600 + (info_img.
        minuto + 5*(l-3)) * 60;
    else idx(l) = info_img.hora * 3600 + (info_img.minuto +
        20) * 60;
    end
end

%Vemos si no existe error en todas y cada una de las veces q
    se intenenten
%cargar una imagen

flag1=0; %para radiacion actual
for z=1:size(datos_radiacion.Var12)
    if(datos_radiacion.Var1(z)==idx(1))
        valor_radiacion1 = datos_radiacion.Var12(z);
        flag1=1;
    end
end
flag2=0; %para radiacion dentro de 1 mnt
for z=1:size(datos_radiacion.Var12)
    if(datos_radiacion.Var1(z)==idx(2))
        valor_radiacion2 = datos_radiacion.Var12(z);
        flag2=1;
    end
end
flag3=0; %para radiacion dentro de 2 mnt
for z=1:size(datos_radiacion.Var12)
```

```
            if(datos_radiacion.Var1(z)==idx(3))
                valor_radiacion3 = datos_radiacion.Var12(z);
                flag3=1;
            end
        end
        flag4=0; %para radiacion dentro de 5 mnt
        for z=1:size(datos_radiacion.Var12)
            if(datos_radiacion.Var1(z)==idx(4))
                valor_radiacion4 = datos_radiacion.Var12(z);
                flag4=1;
            end
        end
        flag5=0; %para radiacion dentro de 10 mnt
        for z=1:size(datos_radiacion.Var12)
            if(datos_radiacion.Var1(z)==idx(5))
                valor_radiacion5 = datos_radiacion.Var12(z);
                flag5=1;
            end
        end
        flag6=0; %para radiacion dentro de 20 mnt
        for z=1:size(datos_radiacion.Var12)
            if(datos_radiacion.Var1(z)==idx(6))
                valor_radiacion6 = datos_radiacion.Var12(z);
                flag6=1;
            end
        end

        if flag1*flag2*flag3*flag4*flag5*flag6 == 0
            fprintf('Error al cargar los datos del día %s a la hora %
                s, minuto %s y segundo %s\n', archivo_radiacion,t(4),t
                (5),t(6));
            % Almacenar fecha, hora y minuto de la rad dañada
              rad_danada = struct('fecha', fecha_str, 'hora', [], '
                  minuto', [],'segundo', []);
                rad_danada.hora = info_img.hora;
                rad_danada.minuto = info_img.minuto;
                rad_danada.segundo = 0;

            rad_danadas = [rad_danadas; rad_danada];
            continue;
        end

        imagenesEst{end+1} = img;
        info_imagenes = [info_imagenes; info_img];

        radiacion1 = [radiacion1; valor_radiacion1];
        radiacion2 = [radiacion2; valor_radiacion2];
        radiacion3 = [radiacion3; valor_radiacion3];
        radiacion4 = [radiacion4; valor_radiacion4];
        radiacion5 = [radiacion5; valor_radiacion5];
```

```
            radiacion6 = [radiacion6; valor_radiacion6];


            % Almacenar fecha, hora, minuto y segundo de los datos de
                radiación
            info_rad = struct('fecha', fecha_str, 'hora', [], 'minuto',
                [], 'segundo', []);
            info_rad.hora = info_img.hora;
            info_rad.minuto = info_img.minuto;
            info_rad.segundo = 0;
            info_radiacion = [info_radiacion; info_rad];

        end
    end

radiacion = [radiacion1, radiacion2, radiacion3, radiacion4, radiacion5,
     radiacion6];

N = length(imagenesEst); % número de imágenes
image_size = size(imagenesEst{1}); % tamaño de la imagen

% Pre-allocation
imagenes = zeros([size(imagenesEst{1}), N]);

% Llenando el array
for i = 1:N
    imagenes(:, :, :, i) = imagenesEst{i};
end
```

Código 7.4 Código empleado para generar la red neuronal que es capaz de estimar la predicción de la radiación en un horizonte temporal de 1, 2, 5, 10 y 20 minutos haciendo uso de imágenes.

```
%% Esta parte se lanza solo una vez
radiacion=radiacion/1000; %hacerlo solo una vez

% Dividir los datos en conjuntos de entrenamiento y validación
tic
numImgs=size(imagenes,4);
idx = randperm(numImgs);
N = round(0.9 * numImgs); % Usa el 90% de los datos para el
    entrenamiento
idxTrain = idx(1:N);
idxVal = idx(N+1:end);

%ejecutar la primera vez nada mas para no cambiar los datos de
    validacion
XTrain = imagenes(:,:,:,idxTrain); %columna 1 para la radiacion actual
YTrain = radiacion(idxTrain,1); %columna 1 para la radiacion actual
```

```
YTrain_1m = radiacion(idxTrain,2); %columna 2 para la radiacion dentro
    de 1 mnt
YTrain_2m = radiacion(idxTrain,3); %columna 3 para la radiacion dentro
    de 2 mnt
YTrain_5m = radiacion(idxTrain,4); %columna 4 para la radiacion dentro
    de 5 mnt
YTrain_10m = radiacion(idxTrain,5); %columna 5 para la radiacion dentro
    de 10 mnt
YTrain_20m = radiacion(idxTrain,6); %columna 6 para la radiacion dentro
    de 20 mnt
XVal = imagenes(:,:,:,idxVal);
YVal = radiacion(idxVal,1); %columna 1 para la radiacion actual
YVal_1m = radiacion(idxVal,2); %columna 2 para la radiacion dentro de 1
    mnt
YVal_2m = radiacion(idxVal,3); %columna 3 para la radiacion dentro de 2
    mnt
YVal_5m = radiacion(idxVal,4); %columna 4 para la radiacion dentro de 5
    mnt
YVal_10m = radiacion(idxVal,5); %columna 5 para la radiacion dentro de
    10 mnt
YVal_20m = radiacion(idxVal,6); %columna 6 para la radiacion dentro de
    20 mnt
toc

%% Hacer esta parte siempre cuando empecemos de 0
radiacion=radiacion/1000;

%% Esta segunda parte se ira modificando, para dentro de 1 mnt

% Definir la arquitectura de la red
tic
Arquitectura inicial de la que partimos
layers = [
    imageInputLayer([128 128 3])
    convolution2dLayer(3,8,'Padding','same', 'WeightsInitializer', 'he')
    batchNormalizationLayer
    reluLayer
    maxPooling2dLayer(2,'Stride',2)
    fullyConnectedLayer(1, 'WeightsInitializer', 'he')
    regressionLayer];
toc

% Especificar las opciones de entrenamiento
tic

options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.00003, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod', 12, ...
    'LearnRateDropFactor', 0.75, ...
```

```
    'MaxEpochs',300, ...
    'MiniBatchSize', 1134, ... %para tener en torno a 15 iteraciones por
        epoca
    'Shuffle','every-epoch', ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
toc


% Continuar el entrenamiento anterior
% Extraer las capas de la red
% layers = net.Layers; %extrae los valores de las capas anteriores
tic
% Entrenamiento
[net,info] = trainNetwork(XTrain, YTrain_5m, layers, options); %cambiar
    siempre
toc
% Predecir los valores de radiación en los datos de validación
YPred = predict(net, XVal); %esta línea nunca hay que cambiarla ya que
    XVal contiene las imágenes actuales
[m,~] = size(YPred);

for h=1:m
    if YPred(h) < 0
        YPred(h) = 0;
    end
end

% Calcular el RMSE
rmse = sqrt(mean((YPred - YVal_5m).^2)); %cambiar siempre
fprintf('RMSE en los datos de validación: %.4f\n', rmse);

% Graficar la pérdida de entrenamiento y validación en escala logarí
    tmica
figure
semilogy(info.TrainingLoss)
hold on
semilogy(info.ValidationLoss)
hold off
legend('Pérdida de entrenamiento', 'Pérdida de validación')
xlabel('Epoch')
ylabel('Pérdida')

% Graficar las predicciones y los valores reales
figure
plot(YPred, '.-')
hold on
plot(YVal_5m, '.-') %cambiar siempre
hold off
```

```
legend('Predicciones', 'Valores reales')
xlabel('Número de muestra')
ylabel('Radiación')
title('Predicciones de la red vs. valores reales')
```

Código 7.5 Código para la generación de la base de datos empleada en el experimento de la red neuronal que calcula la radiación en un horizonte temporal de 1, 2, 5, 10 y 20 minutos haciendo uso de imágenes y datos escalares.

```
clear
% Ruta de las carpetas
ruta_base_imagenes ='C:\Users\juanm\OneDrive - UNIVERSIDAD DE SEVILLA\
    TFG\camara de cielo';
folder_radiacion = 'C:\Users\juanm\OneDrive - UNIVERSIDAD DE SEVILLA\TFG
    \camara de cielo\2023_meteo';

% Generar lista de fechas
fecha_inicio = datetime(2023, 03, 07);
fecha_fin = datetime(2023, 04, 19);
fechas = fecha_inicio:fecha_fin;

% Leer los nombres de los archivos y cargar datos
imagenesEst = {};
info_imagenes = [];
radiacion1 = [];
radiacion2 = [];
radiacion3 = [];
radiacion4 = [];
radiacion5 = [];
radiacion6 = [];
vel_viento = [];
dir_viento = [];
info_radiacion = [];
imagenes_danadas = [];
rad_danadas = [];

tic
    for fecha = fechas
        toc
        tic
        fecha
        fecha_str = datestr(fecha, 'yyyymmdd');
        folder_imagenes = fullfile(ruta_base_imagenes, fecha_str);


            % Cargar datos de radiación
            num_dia = days(fecha - datetime(2023, 01, 01)) + 1; %poner la
                 otra fecha si no
            archivo_radiacion = sprintf('meteo_2023_%03d.txt', num_dia);
```

```
    try
    opts = detectImportOptions(fullfile(folder_radiacion, archivo
        _radiacion), 'Delimiter', '\t'); % Asegúrate de usar el
        delimitador correcto
    opts.VariableNamesLine = 0; % Si hay encabezados en la
        primera línea
    opts.MissingRule = 'fill';
    opts = setvartype(opts, 'double');
    opts = setvartype(opts, 1, 'char'); % Establecer la primera
        columna como tipo 'char'
    datos_radiacion = readtable(fullfile(folder_radiacion,
        archivo_radiacion), opts);
    catch
    fprintf('Error al cargar los datos del día: %s\n', archivo_
        radiacion);
        % Almacenar fecha la rad dañada
        rad_danada = struct('fecha', fecha_str, 'hora', 25, '
            minuto', 61, 'segundo', 61);
        rad_danadas = [rad_danadas; rad_danada];
        continue;
    end


    % Convertir la columna de tiempo a segundos
    tiempo = datos_radiacion.Var1;
    segundos = zeros(size(tiempo));
    for j = 1:numel(tiempo)
        t = datevec(tiempo{j}, 'HH:MM:SS');
        segundos(j) = t(4) * 3600 + t(5) * 60 + t(6);
    end
    datos_radiacion.Var1 = segundos;


archivos_imagenes = dir(fullfile(folder_imagenes, '*.jpg')); %
    Ajusta la extensión de archivo según corresponda
n = numel(archivos_imagenes)


for i = 1:n
 if(mod(i,50)==0)
    i
 end

  % Intentar cargar imagen
    try
        img = imread(fullfile(folder_imagenes, archivos_imagenes(
            i).name));
    catch
        fprintf('Error al cargar la imagen: %s\n', archivos_
            imagenes(i).name);
```

```
        % Almacenar fecha, hora y minuto de la imagen dañada
        info_danada = struct('fecha', fecha_str, 'hora', [], '
            minuto', []);
        tokens = regexp(archivos_imagenes(i).name, '(\d{2})', '
            tokens');
        if numel(tokens) >= 2
            info_danada.hora = str2double(tokens{1}{1});
            info_danada.minuto = str2double(tokens{2}{1});
        end
        imagenes_danadas = [imagenes_danadas; info_danada];
        continue;
    end
    % Redimensionar y normalizar imagen
    img = imresize(img, [128 128]); % Cambia el tamaño según lo
        que necesites
    img = double(img) / 255;


    % Almacenar fecha, hora y minuto de la imagen cargada
    info_img = struct('fecha', fecha_str, 'hora', [], 'minuto',
        []);
    tokens = regexp(archivos_imagenes(i).name, '(\d{2})', 'tokens
        ');
    if numel(tokens) >= 2
        info_img.hora = str2double(tokens{5}{1});
        info_img.minuto = str2double(tokens{6}{1});
    end
    if (info_img.hora<7 || info_img.hora>22)
        continue;
    end



    % Obtener el valor de radiación correspondiente a la imagen
    % actual, dentro de 1 mnt, 2 mnt, 5mnt, 10 mnt y 20 mnt

    for l=1:6
        if(l<4) idx(l) = info_img.hora * 3600 + (info_img.minuto +
             (l-1)) * 60;
        elseif(l<6) idx(l) = info_img.hora * 3600 + (info_img.
            minuto + 5*(l-3)) * 60;
        else idx(l) = info_img.hora * 3600 + (info_img.minuto +
            20) * 60;
        end
    end

    %Vemos si no existe error en todas y cada una de las veces q
        se intenenten
    %cargar una imagen
```

```
flag1=0; %para radiacion actual
for z=1:size(datos_radiacion.Var12)
    if(datos_radiacion.Var1(z)==idx(1))
        valor_radiacion1 = datos_radiacion.Var12(z);
        flag1=1;
    end
end
flag2=0; %para radiacion dentro de 1 mnt
for z=1:size(datos_radiacion.Var12)
    if(datos_radiacion.Var1(z)==idx(2))
        valor_radiacion2 = datos_radiacion.Var12(z);
        flag2=1;
    end
end
flag3=0; %para radiacion dentro de 2 mnt
for z=1:size(datos_radiacion.Var12)
    if(datos_radiacion.Var1(z)==idx(3))
        valor_radiacion3 = datos_radiacion.Var12(z);
        flag3=1;
    end
end
flag4=0; %para radiacion dentro de 5 mnt
for z=1:size(datos_radiacion.Var12)
    if(datos_radiacion.Var1(z)==idx(4))
        valor_radiacion4 = datos_radiacion.Var12(z);
        flag4=1;
    end
end
flag5=0; %para radiacion dentro de 10 mnt
for z=1:size(datos_radiacion.Var12)
    if(datos_radiacion.Var1(z)==idx(5))
        valor_radiacion5 = datos_radiacion.Var12(z);
        flag5=1;
    end
end
flag6=0; %para radiacion dentro de 20 mnt
for z=1:size(datos_radiacion.Var12)
    if(datos_radiacion.Var1(z)==idx(6))
        valor_radiacion6 = datos_radiacion.Var12(z);
        flag6=1;
    end
end
flag7=0; %para la velocidad del viento
for z=1:size(datos_radiacion.Var14) %la columna 14 representa
     la velocidad del viento
    if(datos_radiacion.Var1(z)==idx(1)) %miro en el instante
        actual
        if datos_radiacion.Var14(z) <= 50 %miramos si la
            velocidad del viento es menor de 50 m/s, no vel.
            mayores
```

```
                valor_velocidad = datos_radiacion.Var14(z);
                flag7=1;
            else %si es menor se guarda el valor, si no lo es se
                pone el flag a 0 para que no entre en el if del
                final
                flag7=0;
            end
        end
    end
    flag8=0; %para la direccion del viento
    for z=1:size(datos_radiacion.Var15) %la columma 15 representa
         el angulo del viento
        if(datos_radiacion.Var1(z)==idx(1)) %miro el instante
            actual
            if datos_radiacion.Var15(z) >= 0 && datos_radiacion.
                Var15(z) <= 360 %miramos si el angulo se encuentra
                 entre 0 y 360º
                valor_direccion = datos_radiacion.Var15(z);
                flag8=1;
            else
                flag8=0;
            end
        end
    end

    if flag1*flag2*flag3*flag4*flag5*flag6*flag7*flag8 == 0
        fprintf('Error al cargar los datos del día %s a la hora %
            s, minuto %s y segundo %s\n', archivo_radiacion,t(4),t
            (5),t(6));
        % Almacenar fecha, hora y minuto de la rad dañada
          rad_danada = struct('fecha', fecha_str, 'hora', [], '
              minuto', [],'segundo', []);
            rad_danada.hora = info_img.hora;
            rad_danada.minuto = info_img.minuto;
            rad_danada.segundo = 0;

        rad_danadas = [rad_danadas; rad_danada];
        continue;
    end

    imagenesEst{end+1} = img;
    info_imagenes = [info_imagenes; info_img];

    radiacion1 = [radiacion1; valor_radiacion1];
    radiacion2 = [radiacion2; valor_radiacion2];
    radiacion3 = [radiacion3; valor_radiacion3];
    radiacion4 = [radiacion4; valor_radiacion4];
    radiacion5 = [radiacion5; valor_radiacion5];
    radiacion6 = [radiacion6; valor_radiacion6];
    vel_viento = [vel_viento; valor_velocidad];
```

```
            dir_viento = [dir_viento; valor_direccion];


            % Almacenar fecha, hora, minuto y segundo de los datos de
                radiación
            info_rad = struct('fecha', fecha_str, 'hora', [], 'minuto',
                [], 'segundo', []);
            info_rad.hora = info_img.hora;
            info_rad.minuto = info_img.minuto;
            info_rad.segundo = 0;
            info_radiacion = [info_radiacion; info_rad];

        end
    end

radiacion = [radiacion1, radiacion2, radiacion3, radiacion4, radiacion5,
     radiacion6, vel_viento, dir_viento];

N = length(imagenesEst); % número de imágenes
image_size = size(imagenesEst{1}); % tamaño de la imagen

% Pre-allocation
imagenes = zeros([size(imagenesEst{1}), N]);

% Llenando el array
for i = 1:N
    imagenes(:, :, :, i) = imagenesEst{i};
end
```

Código 7.6 Código empleado para generar la red neuronal que es capaz de estimar la predicción de la radiación en un horizonte temporal de 1, 2, 5, 10 y 20 minutos haciendo uso de imágenes y datos escalares.

```
%% Esta parte se lanza solo una vez
radiacion(:,1:6)=radiacion(:,1:6)/1000; %hacerlo solo una vez;
    normalizamos entre mil las radiaciones
radiacion(:,7)=radiacion(:,7)/100; %hacerlo solo una vez; normalizamos
    entre 100 ya que la velocidad del viento es un orden de magnitud
    menor que las radiaciones
radiacion(:,8)=radiacion(:,8)/10; %hacerlo solo una vez; normalizamos
    entre 10 ya que el angulo del viento es dos ordenes de magnitud
    menor que las radiaciones

% Dividir los datos en conjuntos de entrenamiento y validación
tic
numImgs=size(imagenes,4);
windInfoInputSize=size(radiacion(:,7:8),2);
idx = randperm(numImgs);
```

```
N = round(0.9 * numImgs); % Usa el 90% de los datos para el
    entrenamiento
idxTrain = idx(1:N);
idxVal = idx(N+1:end);

%ejecutar la primera vez nada mas para no cambiar los datos de
    validacion
XTrain = imagenes(:,:,:,idxTrain); %columna 1 para la radiacion actual
YTrain = radiacion(idxTrain,1); %columna 1 para la radiacion actual
YTrain_1m = radiacion(idxTrain,2); %columna 2 para la radiacion dentro
    de 1 mnt
YTrain_2m = radiacion(idxTrain,3); %columna 3 para la radiacion dentro
    de 2 mnt
YTrain_5m = radiacion(idxTrain,4); %columna 4 para la radiacion dentro
    de 5 mnt
YTrain_10m = radiacion(idxTrain,5); %columna 5 para la radiacion dentro
    de 10 mnt
YTrain_20m = radiacion(idxTrain,6); %columna 6 para la radiacion dentro
    de 20 mnt
YTrain_viento = radiacion(idxTrain,7); %columna 7 y 8 para la velocidad
    y el angulo del viento
XVal = imagenes(:,:,:,idxVal);
YVal = radiacion(idxVal,1); %columna 1 para la radiacion actual
YVal_1m = radiacion(idxVal,2); %columna 2 para la radiacion dentro de 1
    mnt
YVal_2m = radiacion(idxVal,3); %columna 3 para la radiacion dentro de 2
    mnt
YVal_5m = radiacion(idxVal,4); %columna 4 para la radiacion dentro de 5
    mnt
YVal_10m = radiacion(idxVal,5); %columna 5 para la radiacion dentro de
    10 mnt
YVal_20m = radiacion(idxVal,6); %columna 6 para la radiacion dentro de
    20 mnt
YVal_viento = radiacion(idxVal,7:8); %columnas 7 y 8 para la velocidad y
     el angulo del viento
toc

%% Hacer esta parte siempre cuando empecemos de 0
radiacion(:,1:6)=radiacion(:,1:6)/1000; %hacerlo solo una vez;
    normalizamos entre mil las radiaciones
radiacion(:,7)=radiacion(:,7)/100; %hacerlo solo una vez; normalizamos
    entre 100 ya que la velocidad del viento es un orden de magnitud
    menor que las radiaciones
radiacion(:,8)=radiacion(:,8)/10; %hacerlo solo una vez; normalizamos
    entre 10 ya que el angulo del viento es dos ordenes de magnitud
    menor que las radiaciones

%% Esta parte se realiza antes de la creación de la nueva net
% Extraemos los pesos y el sesgo de la capa fullyconnected de la net
    buena del 19 de agosto. Se hace solo en el primer experimento
```

```
%Primero vemos cuantas neuronas tiene la nueva capa fullyconnected
numFeatures = size(net.Layers(6).Weights,2);

% Copia los pesos existentes y añade los nuevos
newWeights = zeros(1, numFeatures + 2);
newWeights(:, 1:numFeatures) = net.Layers(6).Weights; %Copia los pesos
    antiguos en un vector intermedio
newWeights(:, numFeatures+1:end) = 0; % Añade los dos pesos nuevos
    correspondientes a los datos del viento que son cero al principio
newBias = net.Layers(6).Bias; % Copiamos el sesgo de la net anterior

%% Esta segunda parte se ira modificando, para dentro de 1 mnt

% Definir la arquitectura de la red
tic
%Generamos el database con el que vamos a trabajar
X1Train = XTrain; %imagenes
X2Train = radiacion(idxTrain,7); %velocidad del viento
X3Train = radiacion(idxTrain,8); %angulo del viento
TTrain = YTrain_20m; %radiacion actual, hay que cambiarla siempre

dsX1Train = arrayDatastore(X1Train,IterationDimension=4);
dsX2Train = arrayDatastore(X2Train);
dsX3Train = arrayDatastore(X3Train);
dsTTrain = arrayDatastore(TTrain);
dsTrain = combine(dsX1Train,dsX2Train,dsX3Train,dsTTrain);
toc

%Extraemos las capas que nos interesa
tic
layers = [
    net.Layers(1:end-2) %cogemos todas menos la regression y la fc
    flattenLayer
    concatenationLayer(1,3, 'Name', 'concat');
    fullyConnectedLayer(1, 'Weights', newWeights, 'Bias', newBias); %
        Generamos la capa con los nuevos pesos y el nuevo sesgo
    regressionLayer];

% Convertimos las layers en una grafica de layers
lgraph = layerGraph(layers);

%Añadimos las capas bien; en total generamos 3 ramas: 1 para la imagen y
%otras 2 para cada uno de las features nuevas
featInput = featureInputLayer(1,Name="features");
lgraph = addLayers(lgraph,featInput);
lgraph = connectLayers(lgraph,"features","concat/in2");

featInput2 = featureInputLayer(1,Name="features2");
lgraph = addLayers(lgraph,featInput2);
```

```
lgraph = connectLayers(lgraph,"features2","concat/in3");
toc

%Hay que quitar el tema de validation para que no de error
tic
options = trainingOptions('sgdm', ...
    'InitialLearnRate',0.00003, ...
    'LearnRateSchedule','piecewise', ...
    'LearnRateDropPeriod', 12, ...
    'LearnRateDropFactor', 0.75, ...
    'MaxEpochs',300, ...
    'MiniBatchSize', 1134, ... %para tener en torno a 15 iteraciones por
        epoca
    'Shuffle','every-epoch', ...
    'ValidationFrequency',30, ...
    'Verbose',false, ...
    'Plots','training-progress');
toc


% Continuar el entrenamiento anterior
% Extraer las capas de la red
% layers = net.Layers; %extrae los valores de las capas anteriores
tic
% Entrenamiento
[net,info] = trainNetwork(dsTrain, lgraph, options);
% [net,info] = trainNetwork({XTrain,YTrain_viento}, YTrain_1m, lgraph,
    options);
toc

%Generamos el database que le pasaremos como input
X1Val = XVal; %imagenes
X2Val = radiacion(idxVal,7); %velocidad del viento
X3Val = radiacion(idxVal,8); %angulo del viento

dsX1Val = arrayDatastore(X1Val,IterationDimension=4);
dsX2Val = arrayDatastore(X2Val);
dsX3Val = arrayDatastore(X3Val);
dsVal = combine(dsX1Val,dsX2Val,dsX3Val);

% Predecir los valores de radiación en los datos de validación
YPred = predict(net, dsVal); %esta línea nunca hay que cambiarla ya que
    dsXVal contiene las imágenes actuales y los datos del viento (inputs
    )
[m,~] = size(YPred);

for h=1:m
    if YPred(h) < 0
        YPred(h) = 0;
    end
```

```
end

% Calcular el RMSE
rmse = sqrt(mean((YPred - YVal_20m).^2)); %hay que cambiarla siempre
fprintf('RMSE en los datos de validación: %.4f\n', rmse);

% Graficar la pérdida de entrenamiento y validación en escala logarí
    tmica
figure
semilogy(info.TrainingLoss)
% hold on
% semilogy(info.ValidationLoss)
% hold off
% legend('Pérdida de entrenamiento', 'Pérdida de validación')
xlabel('Epoch')
ylabel('Pérdida')

% Graficar las predicciones y los valores reales
figure
plot(YPred, '.-')
hold on
plot(YVal_20m, '.-') %hay que cambiarla siempre
hold off
legend('Predicciones', 'Valores reales')
xlabel('Número de muestra')
ylabel('Radiación')
title('Predicciones de la red vs. valores reales')
```

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] José Barrientos de la Rosa, *Predicción de la radiación solar usando una cámara all-sky y redes neuronales*, (2023), 99.

[2] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger, *Understanding batch normalization*, Advances in neural information processing systems **31** (2018), 12.

[3] P. Blanc, B. Espinar, N. Geuder, C. Gueymard, R. Meyer, R. Pitz-Paal, B. Reinhardt, D. Renné, M. Sengupta, L. Wald, and S. Wilbert, *Direct normal irradiance related definitions and applications: The circumsolar issue*, Solar Energy **110** (2014), 561–577.

[4] Eduardo F Camacho and Manuel Berenguel, *Control of solar energy systems*, IFAC proceedings volumes **45** (2012), no. 15, 848–855.

[5] Eduardo F. Camacho, Manuel Berenguel, and Antonio J. Gallego, *Control of thermal solar energy plants*, Journal of Process Control **24** (2014), no. 2, 332–340.

[6] Dami Choi, Christopher J Shallue, Zachary Nado, Jaehoon Lee, Chris J Maddison, and George E Dahl, *On empirical comparisons of optimizers for deep learning*, arXiv preprint arXiv:1910.05446 (2019), 27.

[7] M. W. Gardner and S. R. Dorling, *Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences*, Atmospheric Environment **32** (1998), 2627–2636.

[8] Shab Gbémou, Julien Eynard, Stéphane Thil, Emmanuel Guillot, and Stéphane Grieu, *A comparative study of machine learning-based methods for global horizontal irradiance forecasting*, Energies **14** (2021), no. 11, 3192.

[9] Jinhwa Gene, Suntak Park, Hyung Cheol Shin, and Jong Moo Sohn, *Hybrid optical convolutional neural network with convolution kernels trained in the spatial domain*, Neurocomputing **573** (2024), 127251.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Delving deep into rectifiers: Surpassing human-level performance on imagenet classification*, Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.

[11] Rodrigo Toro Icarte, Toryn Q. Klassen, Richard Valenzano, Margarita P. Castro, Ethan Waldie, and Sheila A. McIlraith, *Learning reward machines: A study in partially observable reinforcement learning*, Artificial Intelligence **323** (2023), 103989.

[12] Najwa Syahirah Mohamed Nor Izam, Zarina Itam, Wong Leong Sing, and Agusril Syamsir, *Sustainable development perspectives of solar energy technologies with focus on solar photovoltaic—a review*, Energies **15** (2022), no. 8, 2790.

[13] Tammy Jiang, Jaimie L. Gradus, and Anthony J. Rosellini, *Supervised machine learning: A brief primer*, Behavior Therapy **51** (2020), 675–687.

[14] Raka Jovanovic, Sertac Bayhan, and Islam Safak Bayram, *A multiobjective analysis of the potential of scheduling electrical vehicle charging for flattening the duck curve*, Journal of Computational Science **48** (2021), 101262.

[15] P. Kuhn, S. Wilbert, C. Prahl, D. Schüler, T. Haase, T. Hirsch, M. Wittmann, L. Ramirez, L. Zarzalejo, A. Meyer, L. Vuilleumier, P. Blanc, and R. Pitz-Paal, *Shadow camera system for the generation of solar irradiance maps*, Solar Energy **157** (2017), 157–170.

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), no. 11, 2278–2324.

[17] Seunghye Lee, Hyunjoo Kim, Qui X. Lieu, and Jaehong Lee, *Cnn-based image recognition for topology optimization*, Knowledge-Based Systems **198** (2020), 105887.

[18] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou, *A survey of convolutional neural networks: analysis, applications, and prospects*, IEEE transactions on neural networks and learning systems **33** (2021), no. 12, 6999–7019.

[19] Benjamin Y.H. Liu and Richard C. Jordan, *The interrelationship and characteristic distribution of direct, diffuse and total solar radiation*, Solar Energy **4** (1960), no. 3, 1–19.

[20] Yu Han Liu, *Feature extraction and image recognition with convolutional neural networks*, Journal of Physics: Conference Series, vol. 1087, IOP Publishing, 2018, p. 062032.

[21] J. G. Martin, J. R.D. Frejo, R. A. García, and E. F. Camacho, *Multi-robot task allocation problem with multiple nonlinear criteria using branch and bound and genetic algorithms*, Intelligent Service Robotics **14** (2021), no. 5, 707–727.

[22] Javier G Martin, José Ramón Domínguez Frejo, Ramón A García, and Eduardo F Camacho, *Multi-robot task allocation problem with multiple nonlinear criteria using branch and bound and genetic algorithms*, Intelligent Service Robotics **14** (2021), no. 5, 707–727.

[23] Prisma Megantoro, Muhammad Akbar Syahbani, Sigit Dani Perkasa, Ahmad Rahmad Muzadi, Yusrizal Afif, Agus Mukhlisin, and Pandi Vigneshwaran, *Analysis of instrumentation system for photovoltaic pyranometer used to measure solar irradiation level*, Bulletin of Electrical Engineering and Informatics **11** (2022), no. 6, 3239–3248.

[24] Adel Mellit and Alessandro Massi Pavan, *A 24-h forecast of solar irradiance using artificial neural network: Application for performance prediction of a grid-connected pv plant at trieste, italy*, Solar Energy **84** (2010), no. 5, 807–821.

[25] JOSEPH Michalsky, Ellsworth G. Dutton, Donald Nelson, James Wendell, Stephen Wilcox, Afshin Andreas, Peter Gotseff, Daryl Myers, Ibrahim Reda, Thomas Stoffel, Klaus Behrens, Thomas Carlund, Wolfgang Finsterle, and David Halliwell, *An extensive comparison of commercial pyrheliometers under a wide range of routine observing conditions*, Journal of Atmospheric and Oceanic Technology **28** (2011), no. 6, 752–766.

[26] Sthitapragyan Mohanty, Prashanta Kumar Patra, and Sudhansu Sekhar Sahoo, *Prediction and application of solar radiation with soft computing over traditional and conventional approach - a comprehensive review*, 2016, pp. 778–796.

[27] Zhian Sun and Aixia Liu, *Fast scheme for estimation of instantaneous direct solar irradiance at the earth's surface*, Solar Energy **98** (2013), 125–137.

[28] Zhiyuan Sun, Yunhao Yuan, Xiaoxiao Dong, Zhimei Liu, Kelong Cai, Wei Cheng, Jingjing Wu, Zhiyuan Qiao, and Aiguo Chen, *Supervised machine learning: A new method to predict the outcomes following exercise intervention in children with autism spectrum disorder*, International Journal of Clinical and Health Psychology **23** (2023), 100409.