Universidad Autónoma de Madrid

Escuela Politécnica Superior

Máster Universitario en en Investigación e Innovación en Inteligencia Computacional y Sistemas Interactivos (MU I2-ICSI)

TRABAJO DE FIN DE MÁSTER

REDES NEURONALES RECURRENTES PARA LA PREDICCIÓN DE ENERGÍA EÓLICA Y FOTOVOLTAICA

Rubén Martínez Sastre
Tutor: José Ramón Dorronsoro Ibero

2021-09-03

REDES NEURONALES RECURRENTES PARA LA PREDICCIÓN DE ENERGÍA EÓLICA Y FOTOVOLTAICA

Autor: Rubén Martínez Sastre
Tutor: José Ramón Dorronsoro Ibero

Departamento de Ingeniería Informática
Escuela Politécnica Superior
Universidad Autónoma de Madrid

2021-09-03

Agradecimientos

En primer lugar quisiera agradecer a mi tutor José Dorronsoro la oportunidad de haber podido aprender de él durante este proyecto.

También quisiera agradecer a Red Eléctrica de España, al Centro Europeo y a AEMET por haberme proporcionado los datos sin los cuales este trabajo no podría haberse llevado a cabo.

Al Instituto de Ingeniería del Conocimiento por haber confiado en mi en todo momento.

Y por último y no menos importante, a todos aquellos que, por suerte o por desgracia, les ha tocado compartir este momento conmigo. A mi familia por todo el apoyo constante a lo largo de estos meses. A mis amigos por su alta estima y confianza en mí. Y en especial a Alba, que como buena compañera de vida, está siempre ahí, para lo bueno y para aguantarme.

"Hazlo o no lo hagas, pero no lo intentes" Yoda

Abstract

Abstract —

Renewable energies are the order of the day, due to their competitiveness and necessity compared to fossil fuels, which cause climate change. Green energies ended 2020 with a 44 % share of total generation, and today they are around 51 %, according to REE (Red Eléctrica de España). The increase in installed capacity, mainly wind and solar energy, meant that almost half of the total energy generated in Spain in 2020 was renewable, with 110,566 GWh, 12.9 % more than the previous year. It is on these two energies that this TFM focuses, with the aim of predicting the behavior of short-term production over time. This aspect is of vital importance for electric companies, which rely on this type of estimates for the control of each of their plants or the management of the electricity market.

For the development of this project, we have chosen to study the following recurrent neural networks: *SimpleRNN* from *keras*, *Gated Recurrent Unit* and *Long Short-Term Memory*. In addition, the results will be compared with the *Ridge Regression* and *Multilayer Perceptron Regression* regression models. It is important that these models are mainly fed by the time series of production in each of the different experiments.

In this project we will first carry out an analysis of the behavior of the models in peninsular Spain for the two types of energy and then we will carry out two experiments located in more specific areas: in the wind field, the Sotavento wind farm will be studied and in the photovoltaic field, the behavior on the island of Mallorca will be analyzed. For the latter two problems, in addition to the corresponding production series, a time series of meteorological variables provided by the ECMWF (*European Centre for Medium-Range Weather Forecasts*) will be used.

Resumen

Resumen —

Las energías renovables están a la orden del día, por su competitividad y necesidad frente a los combustibles fósiles, causantes del cambio climático. Las energías verdes cerraron 2020 con una cuota del 44 % del total de generación, y a día de hoy se encuentran alrededor del 51 %, según comenta REE (Red Eléctrica de España). El aumento de la potencia instalada, principalmente energía eólica y solar, permitió que casi la mitad del total de la energía generada en España en 2020 fuera renovable, con 110.566 GWh, un 12,9 % más que en el año anterior. Son en estas dos energías en las que se centra este TFM, con el objetivo de predecir el comportamiento de la producción a corto plazo a lo largo del tiempo. Este aspecto es de vital importancia para las compañías eléctricas, las cuales se nutren de este tipo de estimaciones para el control de cada una de sus plantas o el manejo del mercado eléctrico.

Para el desarrollo de este proyecto, se ha optado por el estudio de las siguientes redes neuronales recurrentes: *SimpleRNN* de *keras*, *Gated Recurrent Unit* y *Long Short-Term Memory*. Además, se comparán los resultados con los modelos regresivos *Ridge Regression* y *Multilayer Perceptron Regression*. Es importante que estos modelos se nutran principalmente de las series temporales de producción en cada unos de los distintos experimentos.

En este proyecto se llevará a cabo en primer lugar un análisis del comportamiento de los modelos en la España peninsular para los dos tipos de energía y a continuación, se realizarán dos experimentos localizados en áreas más expecíficas: en el ámbito eólico se estudiará el parque de Sotavento y en el fotovoltaico se analizará el comportamiento en la isla de Mallorca. Para estos dos últimos estudios se contará, además de la serie de producción correspondiente, con una serie temporal de variables meteorológicas proporcionadas por el ECMWF (*European Centre for Medium-Range Weather Forecasts*).

Índice general

Índice de tablas

Índice de figuras

1 Introducción

1.1. Motivación del proyecto

Las energías renovables son fuentes de energía limpia, inagotable y cada vez más competitivas y necesarias. Se diferencian de los combustibles fósiles principalmente por su diversidad, abundancia y potencial de uso en cualquier lugar del planeta, pero sobre todo porque no producen gases de efecto invernadero, causantes del cambio climático, ni emisiones contaminantes.

Además, sus costes están disminuyendo a un ritmo creciente, mientras que la tendencia general de los combustibles fósiles va en dirección contraria a pesar de su actual volatilidad. El desarrollo de energías limpias es vital para combatir el cambio climático y limitar sus devastadores efectos.

Este proyecto se focaliza en dos de las fuentes de energía renovable más utilizadas en España: la energía eólica y fotovoltaica, con el objetivo de predecir el comportamiento de la producción a corto plazo a lo largo del tiempo. Este aspecto es de vital importancia para las compañías eléctricas, las cuales se nutren de este tipo de estimaciones para el control de cada una de sus plantas o el manejo del mercado eléctrico.

1.2. Objetivos y enfoque

Este proyecto tiene como objetivos el estudio del comportamiento a corto plazo de la energía eólica y fotovoltaica, así como los modelos que propone la literatura. Los modelos servirán de base para predecir la energía eólica y fotovoltaica de la España peninsular, así

como el parque eólico de Sotavento y la fotovoltaica de la isla de Mallorca. Partiendo de la base de que los modelos regresivos funcionan bien para series temporales, se proponen las redes recurrentes como candidatos a mejorar los resultados.

Para verificar el correcto funcionamiento de los modelos se presentan distintos escenarios. En los problemas más generales (Península), donde la predicción se realiza para un terreno mucho más amplio, se generará una predicción cada diez minutos hasta llegar a las dos horas de predicción. A este tipo de predicción se la denomina de corto plazo. El resultado de la predicción vendrá dado por el histórico de producciones de la región, en forma de serie temporal.

Aunque en el resto de experimentos (Sotavento y Mallorca) el objetivo sigue siendo el mismo, los modelos no sólo se apoyarán es estas series temporales anteriormente mencionadas, si no que también utilizarán otro tipo de variables. De este modo, se podrá observar la evolución de cada uno de los modelos en las distintas vistas: sólo con la serie temporal o añadiendo más variables.

1.3. Estructura del documento

La memoria de este trabajo de fin de máster engloba los siguientes capítulos:

2. **Introducción a la Energía Renovable**. Se efectuará una breve introducción a la energía renovable, entrando más en detalle en la energía eólica y fotovoltaica. También se expondrá el uso de modelos de predicción numérica del tiempo atmosférico conocidos como *NWP* y la selección de la meteorología para cada uno de los problemas expuestos.

3. **Modelos principales**. Se hará un desarrollo de los modelos a implementar entre los que se distinguen dos tipos: los modelos regresivos en los que se hablará de *Ridge Regression* y *Multilayer Perceptron Regression*, y los modelos recurrentes tales como la *SimpleRNN*, *Gated Recurrent Unit* y *Long Short-Term Memory*.

4. **Experimentos**. Contiene la parte experimental del proyecto donde se implementan los modelos, se explican las distintas pruebas que se han realizado y se comentan los resultados, además de una descripción de los datos usados y de la metodología experimental.

5. **Conclusiones**. Comprende todo lo mencionado anteriormente, comentando las conclusiones del proyecto y el trabajo futuro.

6. **Apéndices**. Por último, se muestran una serie de algoritmos y funciones de utilidad.

2

Introducción a la Energía Renovable

La necesidad de predecir qué cantidad de energía va a producir un parque eólico es fundamental para la empresa que lo mantiene. Su gestión y plan de acción corren de la cuenta de estas predicciones, las cuales llevan años siendo un factor indispensable para estas empresas.

Del mismo modo, la energía fotovoltaica también necesita de estos métodos para su optimización. Actualmente en España se está creando una gran inversión de cara a las energías renovables, y aprovechando el innegable buen ecosistema que tiene la zona sur de la Península, todos los ojos están sobre la energía solar, mucho más la fotovoltaica que la termosolar.

El comportamiento de energía producida por un parque fotovoltaico tiene una forma aproximada a la de una gaussiana, con un crecimiento en las primeras horas de la mañana, llegando a un clímax y un posterior decrecimiento en la hora del ocaso. Las horas en las que no hay ningún tipo de radiación la energía que se produce es igual a cero. Esto último es más que importante, pues hay que tenerlo en cuenta a la hora de generar las predicciones, que cómo si no, son generadas con diferentes técnicas de *machine learning*.

2.1. Energía Eólica

La energía eólica es un tipo de energía renovable, muy eficiente eficiente y que es clave para la transición energética y la descarbonización de la economía. Su uso está aumentando en todo el mundo, en parte porque los costes están bajando. La energía eólica se obtiene de la fuerza del viento a través de un aerogenerador que transforma la energía cinética de las corrientes de aire en energía eléctrica. y con el generador, que transforma esta energía

2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
19.561	21.018	22.609	22.853	22.871	22.801	22.847	22.874	23.126	25.442	26.811

Tabla 2.1: Potencia instalada de energía eólica en MW en España entre 2010 y 2020.

mecánica en energía eléctrica.

El proceso de generación eléctrica se puede resumir en los siguientes pasos:

1. El viento mueve las palas del aerogenerador provocando un movimiento en el rotor.
2. El rotor transforma la energía cinética en energía mecánica.
3. El generador transforma esta energía mecánica en energía eléctrica.
4. La energía eléctrica continua producida por los aerogeneradores es convertida en corriente alterna por los inversores
5. Los transformadores elevan la electricidad a media tensión.
6. La energía eléctrica alterna, ahora apta para el consumo, se distribuye por las líneas de transmisión.

De entre las tecnologías de generación de energía que existen en España, la energía eólica es la que cuenta con mayor presencia. La energía eólica en España es la tecnología de generación con más potencia instalada. En 2020 esta energía contaba con una potencia instalada de 26.811 MW, correspondiente a la cuarta parte de la energía total en España.

En la página ESIOS perteneciente a Red Eléctrica de España (REE), existe un histórico de potencias que se puede consultar en la tabla 2.2.

2.2. Energía Fotovoltaica

La energía fotovoltaica es una fuente de energía limpia y renovable que utiliza la radiación solar para producir electricidad. Se basa en el llamado efecto fotoeléctrico, por el que ciertos materiales son capaces de absorber fotones (partículas de luz) y liberar electrones, generando una corriente eléctrica.

El proceso de generación eléctrica se puede resumir en los siguientes pasos:

1. Gracias al efecto fotoeléctrico, los paneles fotovoltaicos absorben partículas de luz (fotones) y liberan electrones o corriente eléctrica directa, convirtiendo la radiación emitida por el sol en energía eléctrica.
2. La energía eléctrica continua producida por los paneles es convertida en corriente alterna por los inversores.

2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
3.645	4.032	4.294	4.397	4.403	4.432	4.436	4.438	4.464	8.665	11.277

Tabla 2.2: Potencia instalada de energía fotovoltaica en MW en España entre 2010 y 2020.

3. Los transformadores elevan la electricidad a media tensión (hasta 36 kW).

4. La energía eléctrica alterna, ahora apta para el consumo, se distribuye por las líneas de transmisión.

La Península Ibérica se ubica en una localización idónea para la generación de energía fotovoltaica. Actualmente España se sitúa entre los países con mejor reputación en este ámbito a nivel mundial, teniendo en cuenta la exportación tecnológica gracias a su continuo desarrollo e innovación. El pronóstico dice que en los próximos años se podría llegar a duplicar la potencia instalada actual.

Igual que en el ejemplo anterior, en la tabla 2.2 se observa que la potencia instalada actual en energía fotovoltaica es de 11.277 MW.

2.3. Predicciones Numéricas del Tiempo Atmosférico (NWP)

Este TFM estudia los modelos de predicción de energía a través de la serie temporal de producciones y también con históricos de meteorología. Estos últimos se basan en modelos de predicción numérica del tiempo atmosférico también conocidos como *Numerical Weather Prediction NWP*.

La meteorología ha sido extraída a través del *European Centre for Medium-Range Weather Forecasts* (*ECMWF*) [1] con la ayuda de una de sus APIs para la obtención de meteorología por todo el globo terráqueo. El Centro Europeo es una organización intergubernamental e independiente, así como un centro de investigación que opera las 24 horas del día, produciendo y difundiendo predicciones meteorológicas numéricas.

Este tipo de modelos *NWP* están cuentan con la capacidad de detectar los cambios que se producen en nuestra atmósfera, tanto a corto plazo (prácticamente a tiempo real) como a muy largo plazo (del orden de meses).

En este trabajo se ha optado por alimentar los modelos propuestos con el modelo determinista que ofrece el Centro Europeo. Este modelo se basa en las predicciones que se obtienen a través de la mejor predicción a partir de las ecuaciones pertinentes del algoritmo propio del *ECMWF*.

Estos datos se pueden recibir en una amplia gama de resoluciones. Obviamente, si esta resolución es menor la precisión también lo será. Los ejemplos que se han desarrollado en el capítulo 4 tienen un valor de resolución igual a 0,125° (resolución máxima del Centro

Europeo hoy en día), que equivale a 11,25 Km de separación entre cada uno de los puntos de coordenadas.

En cuanto a la frecuencia del dato, se ha optado por el dato horario.

2.4. Selección de Meteorología

La labor que realiza el Centro Europeo es tan grande que alberga una cantidad masiva de variables distintas para cada tipo de problema. Es por esto que se ha querido, en un primer lugar, analizar los problemas que se exponen en este proyecto y tratar de seleccionar con suficiente precisión las variables para cada tipo de problema y modelo.

El problema eólico presenta una alta dependencia sobre el viento y sus diferente tipos de medida. La literatura da un mayor peso a la velocidad y dirección de éste [2] pero también hay que tener en cuenta sobre qué altura se encuentran los aerogeneradores. Con el paso del tiempo se ha producido una notable mejora de la tecnología en este ámbito. Además de mejoras sustanciales que repercuten directamente en la potencia y generación eólica, también se ha ido modificando el tamaño de los aerogeneradores. Actualmente la gran mayoría se encuentran a una altura de entre 80 y 100 metros con respecto a la superficie. El Centro Europeo facilita parámetros a nivel superficial, a 10 y a 100 metros.

Con esta información se ha optado por construir los modelos en base a las componentes U y V del viento a 100 metros, las cuales se han combinado entre sí para obtener la velocidad del viento horizontal a la altura mencionada. Estos parámetros son las componentes este y norte del viento a 100 metros de altura. Son la velocidad horizontal del aire que se mueve hacia el este y hacia el norte, a una altura de 100 metros sobre el modelo orográfico medida en metros por segundo. Hay que tener cuidado al comparar los parámetros del modelo con las observaciones, ya que éstas suelen ser locales en un punto concreto del espacio y del tiempo, en lugar de representar las medias de una malla del modelo y de un paso temporal del mismo.

En cuanto al problema fotovoltaico es evidente que se tendrá que tener en cuenta la radiación solar [3]. Es importante definir qué tipo de radiación solar se va a utilizar, puesto que el Centro Europeo nos proporciona un amplio abanico de posibilidades, de entre las que destacan tres:

- **Radiación solar directa (DSRP)**. Este parámetro es la cantidad de radiación directa del sol que llega a la superficie en un plano perpendicular a la dirección del sol.
- **Radiación solar superficial difusa total del cielo (FDIF)**. Este parámetro define el flujo total de la radiación difusa en superficie a partir del esquema de radiación del modelo del Centro Europeo.
- **Radiación solar superficial (SSRD)**. Este parámetro es la cantidad de radiación solar que llega a un plano horizontal, en este caso la placa solar, en la superficie de

la Tierra. Comprende tanto la radiación solar directa como la difusa por lo que es el candidato perfecto de una variable más explicativa.

La radiación del Sol es reflejada en parte al espacio por las nubes y las partículas de la atmósfera (aerosoles) y una parte es absorbida. El resto incide en la superficie de la Tierra (representada por el parámetro SSRD).

Este parámetro es el equivalente en el modelo de lo que mediría un piranómetro (instrumento de medición de la radiación solar) en la superficie terrestre. Sin embargo, hay que tener cuidado al comparar los parámetros del modelo con las observaciones, ya que éstas suelen ser locales en un punto concreto del espacio y el tiempo, en lugar de representar los promedios en una malla del modelo.

Este parámetro se acumula a lo largo de un periodo de tiempo determinado que depende de los datos extraídos. Las unidades son julios por metro cuadrado (J m^{-2}). Para convertirlos en vatios por metro cuadrado (W m^{-2}), los valores acumulados deben dividirse por el periodo de acumulación expresado en segundos.

No obstante este problema presenta una dificultad añadida, donde la gran mayoría de autores asocian la inestabilidad de la energía fotovoltaica al comportamiento del cielo en cada momento, más concretamente a las nubes. Es por esto que se ha querido dar cierta importancia a este fenómeno y contar con una segunda variable para el modelo. Esta se trata de la nubosidad total (TCC). Este parámetro es la proporción de una malla cubierta por nubes. Es calculado a partir de las nubes que se producen en los diferentes niveles del modelo a través de la atmósfera. Se hacen suposiciones sobre el grado de solapamiento/aleatoriedad entre las nubes a diferentes alturas y se da una estimación entre 0 y 1, siendo 0 un cielo totalmente despejado y 1 completamente cubierto.

3

Modelos principales

En este capítulo se van a exponer los diferentes modelos que se van a usar en este proyecto. Se distinguen dos tipos de modelos: los modelos regresivos, en los que se hablará de *Ridge Regression* y *Multilayer Perceptron Regression*, y los modelos recurrentes tales como la *SimpleRNN*, *Gated Recurrent Unit* y *Long Short-Term Memory*.

3.1. Ridge Regression

La regresión lineal [4] trata de modelizar la relación entre dos o más variables ajustando una ecuación lineal a los datos observados. Existe una variable dependiente (*target*) y el resto de variables son consideras explicativas. Este modelo se expresa de la forma

$$f(x) = \beta_0 + \sum_{j=1}^{p} x_j \beta_j \tag{3.1}$$

El método de estimación más popular es el de mínimos cuadrados, en el que se eligen los coeficientes β para minimizar la suma de cuadrados residual

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij} \beta_j)^2 \tag{3.2}$$

y cuya solución en forma matricial es

$$\hat{\beta} = (X^T X)^{-1} X^T y \tag{3.3}$$

donde X corresponde a la matriz de datos e y al *target*.

Aquí se presenta el primer problema y es que si $X^T X$ puede no ser invertible y por lo tanto no existir colinealidad. Es por esto que existen técnicas en la literatura estadística que solucionan este problema. El caso particular de un regularizador cuadrático es conocido como *ridge regression* (Hoerl y Kennard, 1970 [5]). La *ridge regression* reduce los coeficientes de regresión imponiendo una penalización a su tamaño y minimizan la suma residual de cuadrados más el término de regularización:

$$\hat{\beta}^{ridge} = \underset{\beta}{argmin} \Big\{ \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \Big\}, \tag{3.4}$$

donde los valores x_{ij} e y_i representan la entradas y *targets* del modelo.

La reducción de los coeficientes se consigue penalizando el modelo de regresión a través de la suma de los coeficientes al cuadrado. La cuantía de la penalización puede ajustarse mediante una constante λ. Aquí $\lambda \geq 0$ es un parámetro de complejidad que controla la cantidad de reducción: cuanto mayor sea el valor de λ, mayor será la cantidad de reducción, mientras que los coeficientes convergen a cero. La idea de penalizar por la suma de cuadrados de los parámetros también se utiliza en las redes neuronales, donde se conoce como disminución de pesos. Una forma equivalente de escribir la fórmula (3.4) es:

$$\hat{\beta}^{ridge} = \underset{\beta}{argmin} \sum_{i=1}^{N} \Big(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \Big)^2, \tag{3.5}$$

$$\text{sujeto a} \sum_{j=1}^{p} \beta_j^2 \leq t \tag{3.6}$$

que hace explícita la restricción de tamaño de los parámetros.

Cuando hay muchas variables correlacionadas en un modelo de regresión lineal, sus coeficientes pueden quedar mal determinados y presentar una alta varianza. Al imponer una restricción de tamaño a los coeficientes, como en (3.5), se reduce este problema.

En las *ridge regression* normalmente se normalizan las entradas antes de resolver (3.5). Además, se observa que el término β_0 se ha dejado fuera del término de penalización. La penalización del *intercept* haría que el procedimiento dependiera del origen elegido para Y.

Escribiendo la ecuación (3.4) en forma matricial,

$$RSS(\lambda) = (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta, \tag{3.7}$$

se obtiene la solución del modelo *Ridge*, expresado de la forma

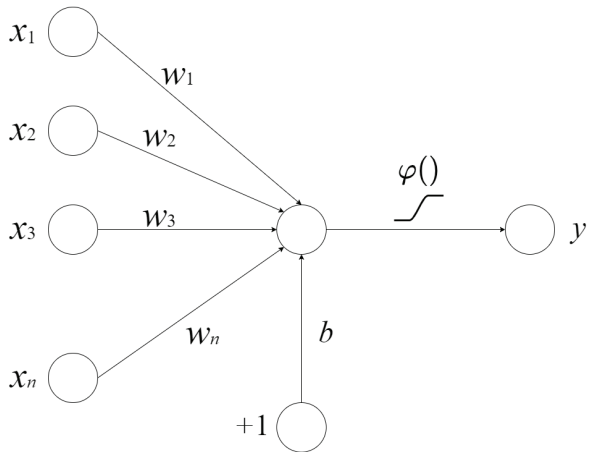

Figura 3.1: Flujo de un perceptrón.

$$\hat{\beta}^{ridge} = (X^T X + \lambda I)^{-1} X^T y, \tag{3.8}$$

3.2. Multilayer Perceptron Regression (MLP Regression)

Un *multilayer perceptron* [6] (MLP) es una red de neuronas llamadas perceptrones. El concepto básico de un perceptrón simple fue introducido por Rosenblatt en 1958 [7]. El perceptrón calcula una única salida a partir de múltiples entradas formando una combinación lineal según sus pesos de entrada y a continuación, sometiendo la salida a alguna función de activación no lineal. Matemáticamente se puede escribir como

$$y = \varphi\Big(\sum_{i=1}^{N} w_i x_i + b\Big) = \varphi(w^T x + b) \tag{3.9}$$

donde w denota el vector de pesos, x es el vector de entradas, b es el sesgo y φ es la función de activación. En la figura 3.1 se muestra un gráfico de flujo de señales de esta operación.

El perceptrón original de Rosenblatt utilizaba una función escalonada de Heaviside [8] como función de activación φ. Hoy en día, y especialmente en las redes multicapa, la función de activación se elige a menudo como la sigmoide logística $1/(1 + e^{-x})$ o la tangente hiperbólica $\tanh(x)$ [9]. Están relacionadas por $(\tanh(x) + 1)/2 = 1/(1 + e^{-2x})$. Estas funciones se utilizan porque son matemáticamente convenientes y están cerca de la linealidad cerca del origen, mientras que se saturan bastante rápidamente cuando se alejan del origen. Esto permite que las redes MLP modelen bien tanto los mapeos fuertemente lineales como los ligeramente no lineales.

Un perceptrón simple no es muy útil debido a su limitada capacidad de mapeo.

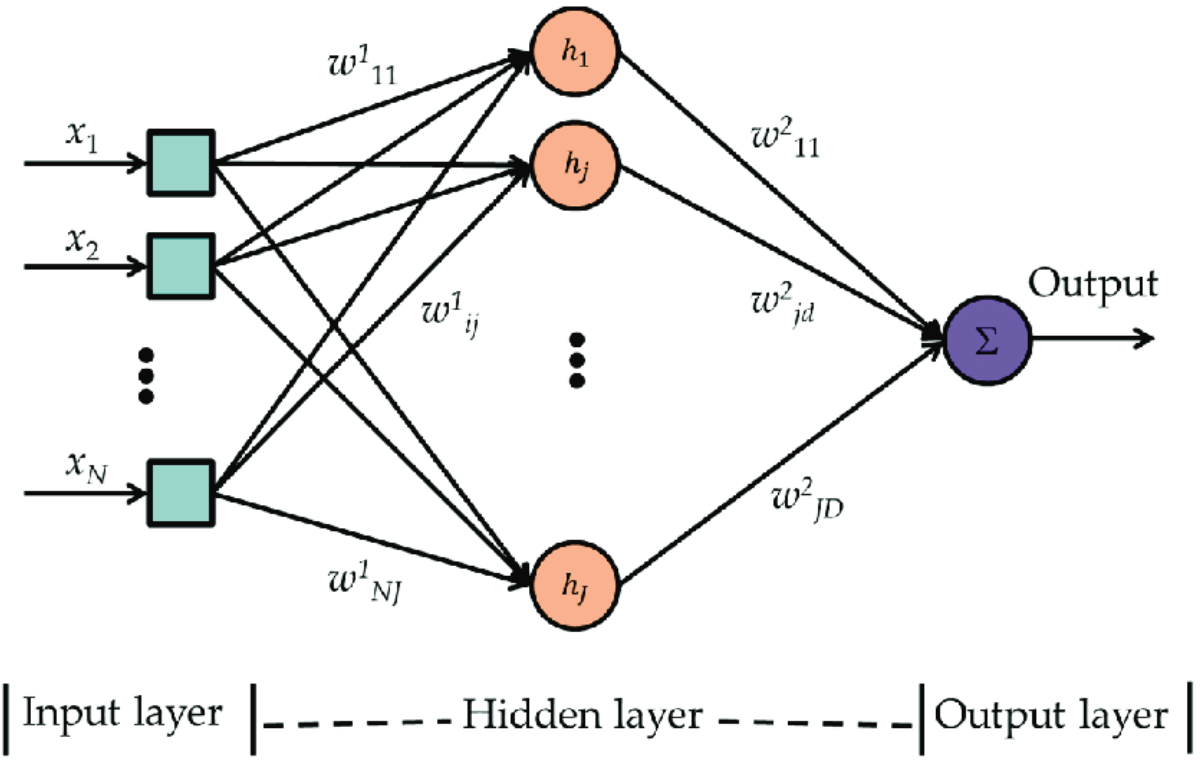

Figura 3.2: Flujo de una *MLP*.

Independientemente de la función de activación que se utilice, el perceptrón sólo es capaz de representar una función esencialmente de tipo lineal. Sin embargo, los perceptrones pueden utilizarse como bloques de construcción de una estructura más grande y mucho más práctica.

Un *MLP* típico consta de un conjunto de nodos fuente que forman la capa de entrada, una o más capas ocultas de nodos de cálculo y una capa de salida. La señal de entrada se propaga a través de la red capa por capa. El flujo de señales de una red de este tipo con una capa oculta se muestra en la figura 3.2.

Este tipo de redes se denominan *feed-forward* [10]. Una red neuronal *feed-forward* es una red neuronal en la que las conexiones entre los nodos no forman un ciclo; esto quiere decir que el dato se propaga sólo hacia delante y nunca en ambos sentidos, como se verá más adelante en las redes recurrentes.

Los cálculos realizados por este tipo de redes con una sola capa oculta con funciones de activación no lineales y una capa de salida lineal pueden escribirse matemáticamente como

$$z = f(x) = w\varphi(Wx + b) + b' \tag{3.10}$$

donde x es un vector de entradas y z un vector de salidas, W es la matriz de pesos de la primera capa, b es el vector de sesgo de la primera capa. w y b' son, respectivamente, la matriz de pesos y el vector de sesgo de la segunda capa.

Mientras que las redes sin capas ocultas están bastante limitadas en cuanto al tipo de mapeos que pueden representar, la potencia de una red MLP con una sola capa oculta es sorprendentemente grande. Tales redes, como la de la ecuación (3.10), son capaces de aproximar cualquier función continua dentro de un conjunto acotado.

Las redes MLP se utilizan normalmente en problemas de aprendizaje supervisado. Esto significa que hay un conjunto de entrenamiento de pares de entrada-salida y la red debe aprender a modelar la dependencia entre ellos. El entrenamiento aquí significa adaptar todos los pesos y sesgos (W, w, b y b' en la ecuación (3.10)) a sus valores óptimos para los pares dados (x_i, y_i). El criterio a optimizar suele ser el error cuadrático $\sum_i (f(x_i) - y_i)^2$.

El problema de aprendizaje supervisado de la MLP puede resolverse con el algoritmo de retropropagación. El algoritmo consta de dos pasos.

- En el paso $feed\text{-}forward$, las salidas predichas correspondientes a las entradas dadas se evalúan como en la ecuación (3.10).
- En el paso *back-propagation* [11], las derivadas parciales de la función de coste con respecto a los diferentes parámetros se propagan hacia atrás a través de la red.

La regla de diferenciación en cadena proporciona reglas de cálculo para el paso hacia atrás con un coste similar a las del paso hacia delante. A continuación, los pesos de la red pueden adaptarse mediante cualquier algoritmo de optimización basado en el gradiente. Todo el proceso se itera hasta que los pesos hayan convergido.

3.2.1. Descenso por gradiente

El descenso por gradiente [12] es una de las estrategias más populares utilizada en el aprendizaje automático y el aprendizaje profundo hoy en día. Es un algoritmo de optimización para encontrar un mínimo local de una función diferenciable. Se utiliza para encontrar los valores de los parámetros de una función (pesos) que minimizan una función de coste.

$$b = a - \gamma \nabla f(a) \tag{3.11}$$

La ecuación (3.11) describe lo que hace el descenso por gradiente: b es la siguiente posición, mientras que a representa su posición actual. El signo menos se refiere a la parte de minimización del descenso por gradiente. γ es un factor multiplicativo y el término de gradiente $-\nabla f(a)$ es simplemente la dirección del descenso más pronunciado.

El tamaño de los pasos que da el descenso de gradiente en la dirección del mínimo local viene determinado por la tasa de aprendizaje (3.11), que determina lo rápido o lento que nos moveremos hacia los pesos óptimos.

Para que el descenso de gradiente alcance el mínimo local se debe fijar la tasa de aprendizaje en un valor adecuado, que no sea ni demasiado bajo ni demasiado alto. Esto es importante porque si los pasos que da son demasiado grandes, es posible que no alcance el mínimo local, saltando de un lado a otro del mínimo de la función. Si se fija la tasa de aprendizaje en un valor muy pequeño, el descenso por gradiente acabará alcanzando el mínimo local, pero podría demorarse demasiado.

Hay tres tipos populares de descenso de gradiente que difieren principalmente en la cantidad de datos que utilizan:

- **Descenso de gradiente por lotes**. El descenso de gradiente por lotes calcula el gradiente para cada ejemplo dentro del conjunto de datos de entrenamiento, pero sólo después de que se hayan evaluado todos los ejemplos de entrenamiento se actualiza el modelo. Todo este proceso es como un ciclo y se llama época de entrenamiento.

  Algunas ventajas del descenso de gradiente por lotes son su eficacia computacional, que produce un gradiente de error estable y una convergencia estable. Algunas desventajas son que requiere que todo el conjunto de datos de entrenamiento esté en la memoria y disponible para el algoritmo, un posible problema dependiendo de la cantidad de datos que se tenga y la capacidad computacional de la que dispongamos.

- **Descenso de gradiente estocástico**. Por el contrario, el descenso de gradiente estocástico (SGD) hace esto para cada ejemplo de entrenamiento dentro del conjunto de datos, lo que significa que actualiza los parámetros para cada ejemplo de entrenamiento uno por uno. Dependiendo del problema, esto puede hacer que el SGD sea más rápido que el descenso de gradiente por lotes.

  Una ventaja es que las actualizaciones frecuentes nos permiten tener una tasa de mejora bastante acentuada. Sin embargo, la frecuencia de esas actualizaciones puede dar lugar a gradientes ruidosos, lo que puede hacer que el error dé saltos en lugar de disminuir lentamente.

- **Descenso de gradiente por *Mini-Batch***. El descenso de gradiente por minilotes (*Mini-Batch*) es el método más utilizado, ya que es una combinación de los conceptos de SGD y descenso de gradiente por lotes. Simplemente divide el conjunto de datos de entrenamiento en pequeños lotes y realiza una actualización para cada uno de ellos.

  Esto crea un equilibrio entre la robustez del descenso de gradiente estocástico y la eficacia del descenso de gradiente por lotes. Los tamaños habituales de los minilotes suelen ser de alrededor de 250, pero como cualquier otra técnica de aprendizaje automático, no hay una regla clara porque varía según las distintas aplicaciones.

3.2.2. Algoritmo de optimización Adam

Adam es un algoritmo de optimización [13] que puede utilizarse en lugar del procedimiento clásico de descenso de gradiente estocástico para actualizar los pesos de la

red de forma iterativa basándose en los datos de entrenamiento.

Los autores describen a Adam como una combinación de las ventajas de otras dos extensiones del descenso de gradiente estocástico. En concreto:

- Algoritmo de gradiente adaptativo (AdaGrad) que mantiene una tasa de aprendizaje por parámetro que mejora el rendimiento en problemas con gradientes dispersos (por ejemplo, problemas de lenguaje natural y *computer vision*).
- Propagación de la raíz cuadrada media (RMSProp) que también mantiene tasas de aprendizaje por parámetro que se adaptan en función de la media de las magnitudes recientes de los gradientes para el peso (por ejemplo, la rapidez con la que está cambiando).

En lugar de adaptar las tasas de aprendizaje de los parámetros basándose en el primer momento medio (la media) como en RMSProp, Adam también hace uso de la media de los segundos momentos de los gradientes (la varianza no centrada).

En concreto, el algoritmo calcula una media móvil exponencial del gradiente y del gradiente al cuadrado, y los parámetros $beta1$ y $beta2$ controlan las tasas de decaimiento de estas medias móviles.

El valor inicial de las medias móviles y los valores de $beta1$ y $beta2$ cercanos a 1 (recomendado) dan lugar a un sesgo de las estimaciones de momentos. Este sesgo se supera calculando primero las estimaciones sesgadas antes de calcular después las estimaciones corregidas por el sesgo.

3.2.3. Vanishing Gradient Problem

El *vanishing gradient problem* [14] es una dificultad que se presenta en el entrenamiento de ciertas redes neuronales artificiales con métodos basados en el gradiente. En concreto, este problema dificulta mucho el aprendizaje y el ajuste de los parámetros de las primeras capas de la red, pudiéndose agravar a medida que aumenta el número de capas en la arquitectura. No se trata de un problema fundamental de las redes neuronales, sino de un problema de los métodos de aprendizaje basados en el gradiente causado por ciertas funciones de activación.

Los métodos basados en el gradiente aprenden el valor de un peso, entendiendo que un pequeño cambio en dicho valor afectará a la salida de la red. Si un cambio en el valor del peso provoca un cambio muy pequeño en la salida de la red, la red no puede aprender el parámetro de forma efectiva, lo que puede llegar a suponer un problema.

Esto es exactamente lo que ocurre en el *vanishing gradient problem*: los gradientes de la salida de la red con respecto a los pesos en las primeras capas se vuelven extremadamente pequeños. Es una forma elegante de decir que incluso un gran cambio en el valor de los parámetros de las primeras capas, no tiene un gran efecto en la salida.

El *vanishing gradient problem* se debe en parte a la elección de la función de activación. Muchas funciones de activación comunes (por ejemplo, sigmoide o *tanh*) reducen su entrada en un rango de salida de forma muy poco lineal. Por ejemplo, la función sigmoide asigna los números reales a un rango de [0, 1], especialmente porque la función es muy plana en la mayor parte de los números reales. Como resultado, hay grandes regiones del espacio de entrada que se mapean en un rango pequeño. En estas regiones del espacio de entrada, incluso un gran cambio en la entrada producirá un pequeño cambio en la salida, por lo que el gradiente es pequeño.

Este problema empeora si se apilan múltiples capas no lineales una encima de otra. Como resultado, incluso un gran cambio en los pesos de la primera capa hará que la salida no varíe demasiado.

3.2.4. Exploding Gradient

Los *exploding gradient* son un problema en el que se acumulan grandes gradientes de error, que dan lugar a actualizaciones muy grandes de los pesos del modelo de la red neuronal durante el entrenamiento.

En las redes profundas o redes neuronales recurrentes, los gradientes de error pueden acumularse durante una actualización y dar lugar a gradientes muy grandes. Estos, a su vez, dan lugar a grandes actualizaciones de los pesos de la red y, a su vez, a una red inestable. En un extremo, los valores de los pesos pueden llegar a ser tan grandes como para desbordarse y dar lugar a valores NaN.

Este efecto se produce mediante un crecimiento exponencial al multiplicar repetidamente los gradientes a través de las capas de la red que tienen valores superiores a 1.

En las redes neuronales recurrentes, los *exploding gradients* pueden dar lugar a una red inestable incapaz de aprender de los datos de entrenamiento y, en el mejor de los casos, a una red que no puede aprender a lo largo de largas secuencias de datos de entrada.

Hay algunas señales que indican que se puede estar sufriendo un *exploding gradient* durante el entrenamiento de su red, como por ejemplo:

- El modelo es incapaz de avanzar con sus datos de entrenamiento.
- El modelo es inestable, lo que provoca grandes cambios en las pérdidas de una actualización a otra.
- La pérdida del modelo pasa a NaN durante el entrenamiento.
- Los pesos del modelo se vuelven rápidamente muy grandes durante el entrenamiento y pasan a valores NaN.

Existen varios enfoques para abordar este problema. Esta sección enumera algunos de ellos que se pueden utilizar.

1. **Rediseñar el modelo de red**. En las redes neuronales profundas, los *exploding gradients* pueden solucionarse rediseñando la red para que tenga menos capas.

   También puede ser beneficioso utilizar un tamaño de lote más pequeño mientras se entrena la red.

2. **Regularizar los pesos**. Otro enfoque, si todavía se producen *exploding gradients*, es comprobar el tamaño de los pesos de la red y aplicar una penalización a la función de pérdida de la red para valores de peso grandes.

   Esto se llama regularización de pesos y a menudo se puede utilizar una penalización L1 (pesos absolutos) o L2 (pesos al cuadrado) como en *Ridge Regression*.

3. **Usar *Gradient Clipping***. Los *exploding gradients* todavía pueden ocurrir en redes de Perceptrón Multicapa muy profundas con un tamaño de lote grande y *LSTMs* con longitudes de secuencia de entrada muy largas.

   Si todavía se producen *exploding gradients*, se puede comprobar y limitar el tamaño de los gradientes durante el entrenamiento de tu red.

4. **Utilizar *LSTM***. En las redes neuronales recurrentes, de las que se hablará más adelante, los *exploding gradients* pueden producirse dada la inestabilidad inherente al entrenamiento de este tipo de redes, por ejemplo, a través de la retropropagación en el tiempo, que esencialmente transforma la red recurrente en una red neuronal Perceptrón multicapa profunda.

   Los *exploding gradients* pueden reducirse utilizando las unidades de memoria de largo plazo.

   La adopción de unidades de memoria *LSTM* es una nueva práctica recomendada para las redes neuronales recurrentes de predicción de secuencias.

3.3. Redes Recurrentes

En esta sección se van a exponer las diferentes redes recurrentes que se van a usar en este proyecto, a saber: *SimpleRNN*, *Gated Recurrent Unit* y *Long Short-Term Memory*.

3.3.1. Introducción a las redes neuronales recurrentes (RNN)

Las redes neuronales recurrentes [15] abordan el problema de la pérdida de conceptos a lo largo de un entrenamiento prolongado, creando una cierta persistencia en el modelo. Son redes con bucles que permiten que la información persista.

En el diagrama 3.3, la unidad A observa una entrada x_t y emite un valor h_t. Un bucle permite pasar la información de un paso de la red al siguiente.

Estos bucles hacen que las redes neuronales recurrentes parezcan algo misteriosas. Sin embargo, si se piensa un poco más, resulta que no son tan diferentes de una red

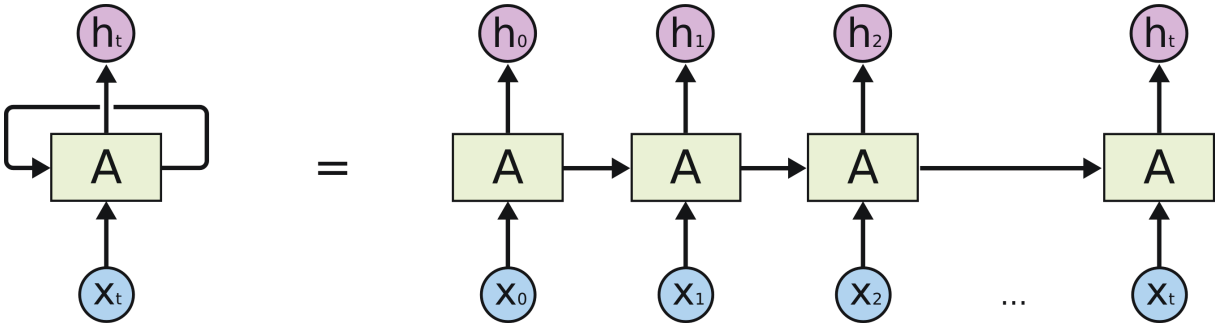

Figura 3.3: Módulo de repetición en una *RNN*.

neuronal normal. Una red neuronal recurrente puede considerarse como múltiples copias de la misma red, cada una de las cuales pasa un mensaje a su sucesora. Esta naturaleza encadenada revela que las redes neuronales recurrentes están íntimamente relacionadas con las secuencias y las listas. Son la arquitectura natural de las redes neuronales para utilizar esos datos.

En los últimos años, se han obtenido notables éxitos aplicando las RNN a una gran variedad de problemas: reconocimiento del habla, modelado del lenguaje, traducción, subtitulación de imágenes...Esencial para estos éxitos es el uso de las *LSTM*, un tipo muy especial de red neuronal recurrente que funciona, para muchas tareas, mucho mejor que la versión estándar.

Casi todos los resultados interesantes basados en redes neuronales recurrentes se consiguen con ellas. Uno de los atractivos de las RNN es la idea de que puedan conectar la información anterior con la tarea actual. A veces, sólo es necesario mirar la información reciente para realizar la tarea actual. Por ejemplo, se considera un modelo lingüístico que intenta predecir la siguiente palabra basándose en las anteriores. Si se intenta predecir la última palabra de "las nubes están en el cielo", no necesitamos más contexto: es bastante obvio que la siguiente palabra será cielo. En estos casos, en los que la distancia entre la información relevante y el lugar en el que se necesita es pequeña, las RNN pueden aprender a utilizar la información anterior.

Pero también hay casos en los que se necesita más contexto. Si se cuenta con la serie temporal de la producción fotovoltaica en un día se observa que en las primeras horas de la mañana el dato va aumentando. Si se quisiera predecir la producción de las 15h, las seis horas anteriores indican que la serie va aumentando progresivamente. Quizás si no se contara con la serie del día anterior, el valor predicho seguiría la misma tónica pero se necesita el contexto del día anterior a la misma hora para saber que el valor a predecir disminuye.

O en caso de tratar de predecir la última palabra del texto "Me crié en Portugal... Hablo con fluidez el portugués". La información reciente sugiere que la siguiente palabra es probablemente el nombre de un idioma, pero si se quiere acotar qué idioma, se necesita el contexto de Portugal, desde más atrás. Es perfectamente posible que la brecha entre la información relevante y el punto en el que se necesita sea muy grande. Por desgracia, a medida que esa brecha crece, las RNN se vuelven incapaces de aprender a conectar la

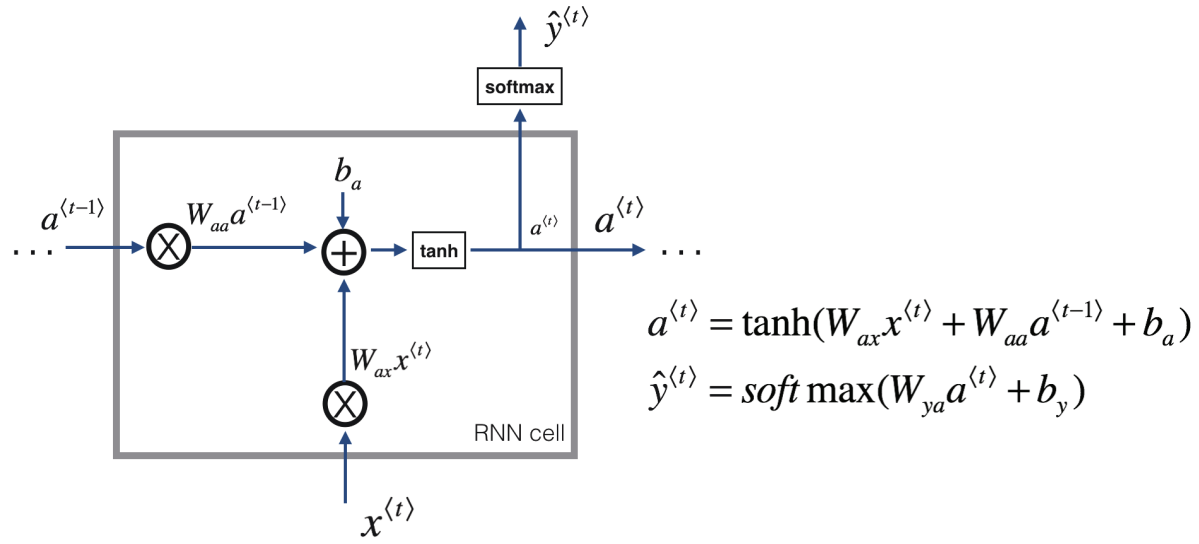

Figura 3.4: Módulo de una *RNN*.

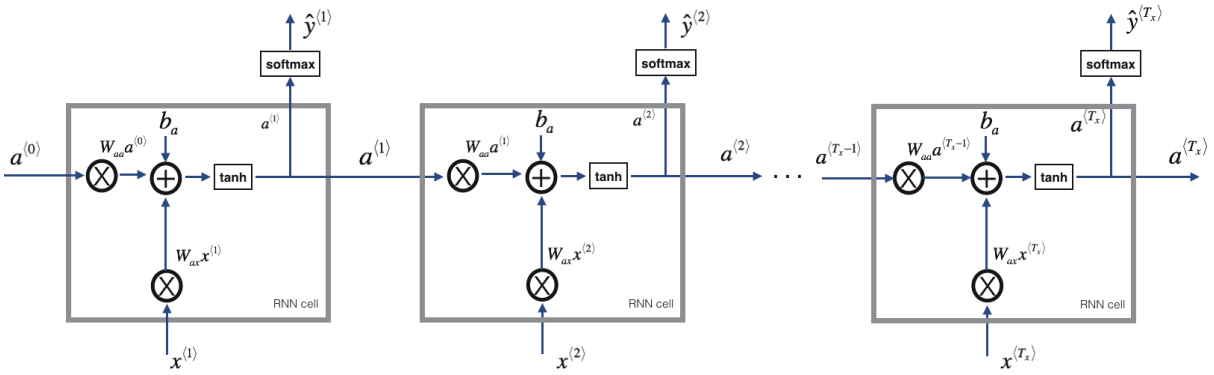

Figura 3.5: Módulo de en una *forward RNN*.

información.

En teoría, las RNN son capaces de manejar tales "dependencias a largo plazo". Un humano podría elegir cuidadosamente los parámetros para que resolvieran problemas de juguete de este tipo. En la práctica, las RNN no siempre ser capaces de aprenderlas.

3.3.2. SimpleRNN

La *SimpleRNN* ([16]) es la red neuronal que intenta mantener la información a lo largo del tiempo más sencilla. Como se puede obervar en la ecuación de la figura 3.4 la información se almacena en la capa oculta a y se actualiza en cada momento en función de las nuevas entradas. Esta red proviene de las redes de Elman [17], introducidas en 1990. El componente distribuido en el tiempo permite calcular la salida $\hat{y}^{<t>}$ a partir de la capa oculta.

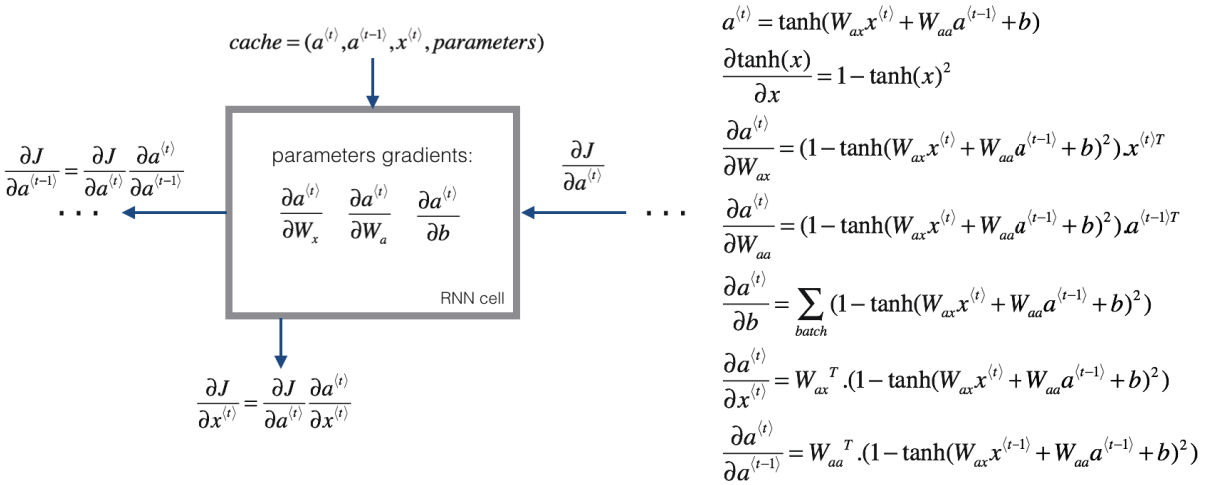

Figura 3.6: Módulo de la retropropagación de los pesos en una *RNN*.

Este método se propaga a lo largo de cada una de las neuronas de la figura 3.5, prolongando la información a modo de $feed\text{-}forward$.

La retropropagación a través del tiempo es la aplicación del algoritmo de entrenamiento de retropropagación a una red neuronal recurrente aplicada a datos secuenciales como una serie temporal, como se explica en las fórmulas de la figura 3.6. Conceptualmente funciona "desenrollando" todos los pasos del tiempo de entrada. Cada paso de tiempo tiene una entrada, una copia de la red y una salida. Los errores se calculan y se acumulan para cada paso de tiempo. Finalmente la red se vuelve a "enrollar" y se actualizan los pesos.

La retropropagación puede ser costosa desde el punto de vista computacional a medida que aumenta el número de pasos de tiempo. Si las secuencias de entrada se componen de miles de pasos de tiempo, entonces este será esencialmente el número de derivadas necesarias para una sola actualización de pesos. Esto puede hacer que los pesos se desvanezcan (*vanishing gradient problem*) o provoquen el *exploding gradient* (vayan a cero o se desborden) y que el aprendizaje sea lento.

3.3.3. Gated Recurrent Unit (GRU)

Las redes *GRU* (*Gated Recurrent Unit*), introducidas por Kyunghyun Cho (2014) [18], tienen como objetivo resolver el *vanishing gradient problem* que se produce en una red neuronal recurrente estándar. Las *GRU* también puede considerarse como una variación de la *LSTM* porque ambas están diseñadas de forma similar y, en algunos casos, producen resultados igualmente excelentes.

Para resolver el *vanishing gradient problem* de una RNN estándar, las *GRU* utilizan la denominada puerta de actualización y la puerta de reinicio. Básicamente, se trata de dos vectores que deciden qué información debe pasar a la salida. Lo especial de ellos es que pueden ser entrenados para mantener la información de hace mucho tiempo, sin perderla a través del tiempo o eliminar la información que es irrelevante para la predicción.

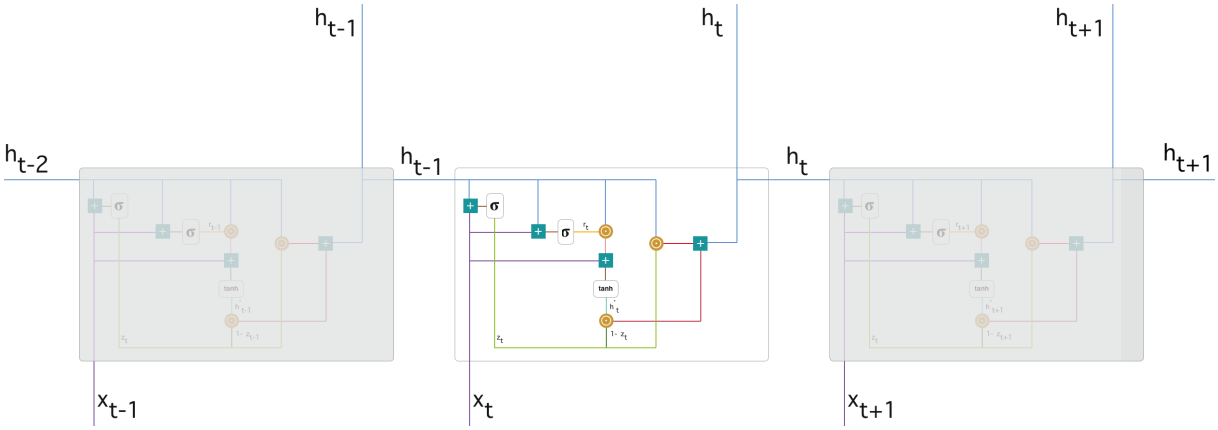

Figura 3.7: Módulo de repetición en una *GRU*.

La figura 3.7 muestra como a través de cada línea fluye el dato, mediante operaciones puntuales como sigmoides o productos. Estas redes tienen la capacidad de eliminar o agregar información al estado de la celda mediante el uso de unas estructuras denominadas puertas. Las puertas permiten que la información fluya por la red. Están compuestas por una red neuronal con una función de activación o multiplicación.

El proceso de aprendizaje comienza por cálculo de la puerta de actualización z_t para el paso de tiempo t utilizando la fórmula

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{3.12}$$

Cuando en (3.12) x_t se introduce en la unidad de red, se multiplica por su propio peso $W^{(z)}$. Lo mismo ocurre con h_{t-1}, que contiene la información de las unidades $t-1$ anteriores y se multiplica por su propio peso $U^{(z)}$. Ambos resultados se suman y se aplica una función de activación sigmoidal para acotar el resultado entre 0 y 1.

La puerta de actualización ayuda al modelo a determinar cuánta información del pasado (de los pasos temporales anteriores) debe transmitirse al siguiente paso. Esto es muy importante ya que el modelo puede decidir copiar toda la información del pasado y eliminar el riesgo de un *vanishing gradient problem*.

Esencialmente, la puerta de reinicio que actúa a continuación se utiliza para decidir qué cantidad de información pasada hay que olvidar. Para calcularlo se utiliza una fórmula similar a la de la puerta de actualización cambiando los pesos en la fórmula (3.13).

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{3.13}$$

Al igual que antes, se introduce h_{t-1} y x_t, se multiplican por sus correspondientes pesos, se suman los resultados y se aplica la función sigmoidal.

Para conocer en qué afectan exactamente las puertas a la salida final, se comienza con el uso de la puerta de reinicio. Se introducen nuevos datos a la memoria que utilizará la

puerta de reinicio para almacenar la información relevante del pasado

Antes de calcular la salida final, se introducen nuevos datos a la memoria que utilizará la puerta de reinicio para almacenar la información relevante del pasado como se muestra en (3.14).

$$h_t^{'} = \tanh(Wx_t + r_t \quad Uh_{t-1}) \tag{3.14}$$

En concreto los pasos son:

1. Se multiplica la entrada x_t con un peso W y h_{t-1} con un peso U.
2. Se calcula el producto Hadamard (elemento a elemento) entre la salida de la puerta de reinicio r_t y Uh_{t-1}.
3. Se suman los dos pasos anteriores.
4. Se aplica la función de activación no lineal *tanh*.

Como último paso, la red necesita calcular h_t, vector que contiene la información de la unidad actual y la pasa a la red. Para ello se necesita la salida de la puerta de actualización. Esta determina lo que debe permanecer del contenido de la memoria actual como muestra la fórmula (3.15).

$$h_t = z_t \quad h_{t-1} + (1 - z_t) \quad h_t^{'} \tag{3.15}$$

En concreto los pasos son:

1. Se aplica la multiplicación por elementos a la puerta de actualización z_t y h_{t-1}.
2. Se aplica la multiplicación por elementos a $(1 - z_t)$ y $h_t^{'}$.
3. Se suman los resultados de los dos pasos anteriores.

El modelo puede aprender a colocar el vector z_t cerca de 1 y mantener la mayor parte de la información anterior. Como z_t estará cerca de 1 en este paso de tiempo, $(1 - z_t)$ estará cerca de 0, lo que ignorará gran parte del contenido actual que es irrelevante para la predicción. Por último, h_t es el resultado de la suma de las salidas correspondientes a los dos últimos pasos.

3.3.4. Long Short-Term Memory (LSTM)

Las redes de memoria a largo/corto plazo, generalmente llamadas *LSTM*, son un tipo especial de RNN, capaz de aprender dependencias a largo plazo. Fueron introducidas

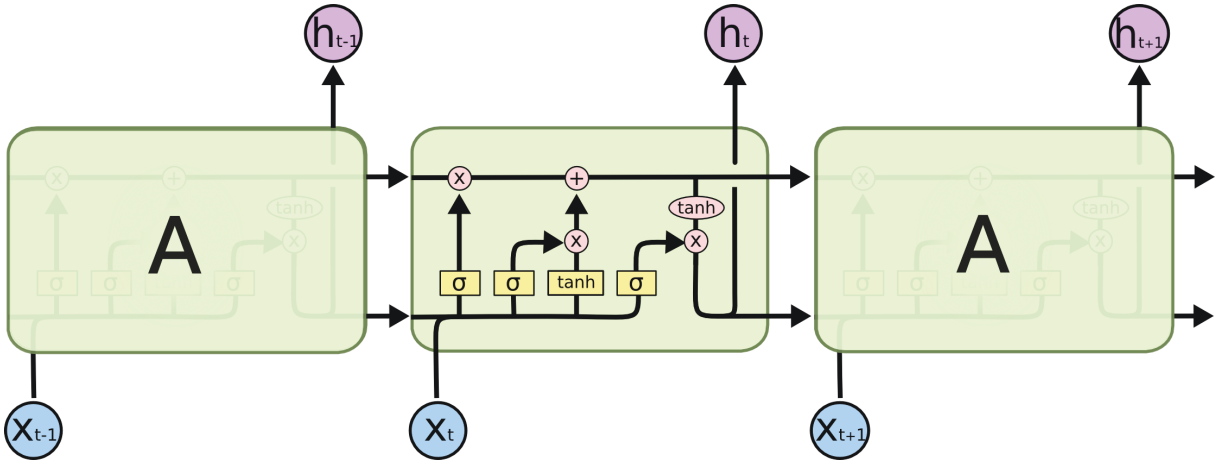

Figura 3.8: Módulo de repetición en una *LSTM*.

por Hochreiter y Schmidhuber (1997) [19]. Actualmente están en auge ya que funcionan francamente bien en una amplia variedad de problemas.

Las *LSTM* están diseñadas explícitamente para evitar el problema de dependencia a largo plazo. Recordar información durante largos períodos de tiempo es prácticamente su comportamiento predeterminado. Todas las redes neuronales recurrentes tienen la forma de una cadena de módulos repetitivos de la red neuronal. En las RNN estándar, este módulo de repetición cuenta con una estructura muy simple, con una sola función de activación *tanh*.

Las *LSTM* también tienen esta estructura de cadena, pero el módulo de repetición tiene una estructura diferente. En lugar de tener una sola celda de red neuronal, hay cuatro que interactúan de una forma muy especial.

En la figura 3.8 cada línea transporta un vector completo, desde la salida de un nodo hasta las entradas de otros. Los círculos del interior representan operaciones puntuales, como la adición de vectores, mientras que los cuadros amarillos de la figura 3.8 son las salidas de las celdas de la red neuronal. Las líneas que se fusionan denotan la concatenación, mientras que una línea de bifurcación denota el contenido que se está copiando y las copias que van a diferentes ubicaciones.

La principal clave para este tipo de redes es el estado de la unidad C_t, la línea horizontal que recorre la parte superior del diagrama. El estado de la celda tiene un funcionamiento semejante al de una cinta transportadora. Atraviesa la cadena con ligeras interacciones, aunque es muy común que no sufra cambios durante el proceso.

Como se ha mencionado en la sección anterior, estas redes tienen la capacidad de eliminar o agregar información al estado de la celda, cuidadosamente regulado por estructuras llamadas puertas. Las puertas son una forma de permitir que la información avance. Se componen de una celda con una función de activación sigmoidal y una operación de multiplicación puntual.

La celda de la sigmoide produce un resultado entre 0 y 1 que describe la cantidad

de cada componente que debe dejarse pasar. Un valor de cero significa que el valor no continuará, mientas que un valor de uno hace que se transmita la totalidad de la información. Una *LSTM* cuenta con tres de estas puertas para proteger y controlar el estado de la celda.

El primer paso de la *LSTM* es decidir qué información vamos a eliminar del estado de la celda (3.16). Esta decisión es tomada por una puerta sigmoidal llamada *forget gate*. El valor 1 representa mantener por completo el estado actual mientras que un 0 significa deshacerse de todo.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3.16}$$

El siguiente paso es decidir qué información nueva se va a almacenar en el estado de la celda. Esto tiene una serie de pasos:

1. Una puerta sigmoidal llamada *input gate* que decide qué valores actualizaremos de acuerdo a la fórmula (3.17).

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{3.17}$$

2. Una puerta *tanh* que crea un vector de nuevos valores $\tilde{C}_t$, de acuerdo a la fórmula (3.18), que podría agregarse al estado.

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3.18}$$

3. El siguiente paso consiste en actualizar el estado de la celda anterior, C_{t-1}, en el nuevo estado de la celda C_t de acuerdo a la fórmula (3.19). Se multiplica el estado antiguo por f_t olvidando las cosas que decidimos despreciar antes. Entonces se añade $i_t * \tilde{C}_t$. Estos son los nuevos valores candidatos, en función de cuánto decidimos actualizar cada valor de estado.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.19}$$

Finalmente, se genera la salida (*output gate*). La fórmula (3.20) muestra un paso intermedio basada en el estado de la celda previamente filtrada, donde una puerta sigmoidal decide qué partes del estado de la celda sirven como salida.

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{3.20}$$

Entonces, se genera la salida a través de la puerta *tanh* como se aprecia en la fórmula (3.21), para forzar a que los valores sean entre -1 y 1, y finalmente multiplicarlo por la salida de la puerta sigmoidal, de modo que sólo se emiten las partes que se decidan.

$$h_t = o_t * tanh(C_t) \tag{3.21}$$

En definitiva, las *LSTM* se resumen en tres pasos:

1. La *forget gate* decide qué es relevante mantener de los pasos anteriores.
2. La *input gate* decide qué información es relevante para añadir del paso actual.
3. Por último, se determina cuál debe ser el siguiente estado a través de la *output gate*.

4

Experimentos

4.1. Datos

El problema que se va a intentar resolver en este TFM es la predicción de producción eólica y fotovoltaica a corto plazo a través de la serie temporal de producción. Este problema parte de la premisa de obtener los datos de producción de al menos 3 años, que servirán de fuente de alimentación de los modelos de *Machine Learning*. Dichos datos han sido obtenidos a través de la plataforma ESIOS, web proporcionada por Red Eléctrica de España (REE en adelante), y que contiene, entre otros, los registros de producción eólica y fotovoltaica de los últimos años.

Para este problema se ha optado por obtener los datos de toda la Península ibérica. El rango de fechas elegido para los modelos ha sido del 1 de enero de 2016 al 31 de diciembre de 2018. Los datos extraídos tienen una frecuencia diez-minutal; esto quiere decir que cada diez minutos existe un par de datos, fecha y producción total, con un total de 157824 registros.

Para un enfoque más preciso, también se ha decidido realizar otro par de experimentos más concretos, en los que se ha tenido en cuenta a un parque en concreto y un pequeño cluster de parques. Este par de ejemplos corresponden a uno eólico y otro fotovoltaico, siendo estos el parque de Sotavento y la isla de Mallorca respectivamente.

Los datos han sido extraídos desde la página oficial del parque de Sotavento para el ejemplo eólico. La serie temporal abarca desde el 1 de enero de 2016 al 31 de diciembre de 2018 con una frecuencia horaria. Sotavento es un parque experimental creado en el año 2005, situado en Galicia. Como bien menciona su página web, su creación responde a la necesidad de distinguir las actividades comerciales de las actuaciones educativas. La

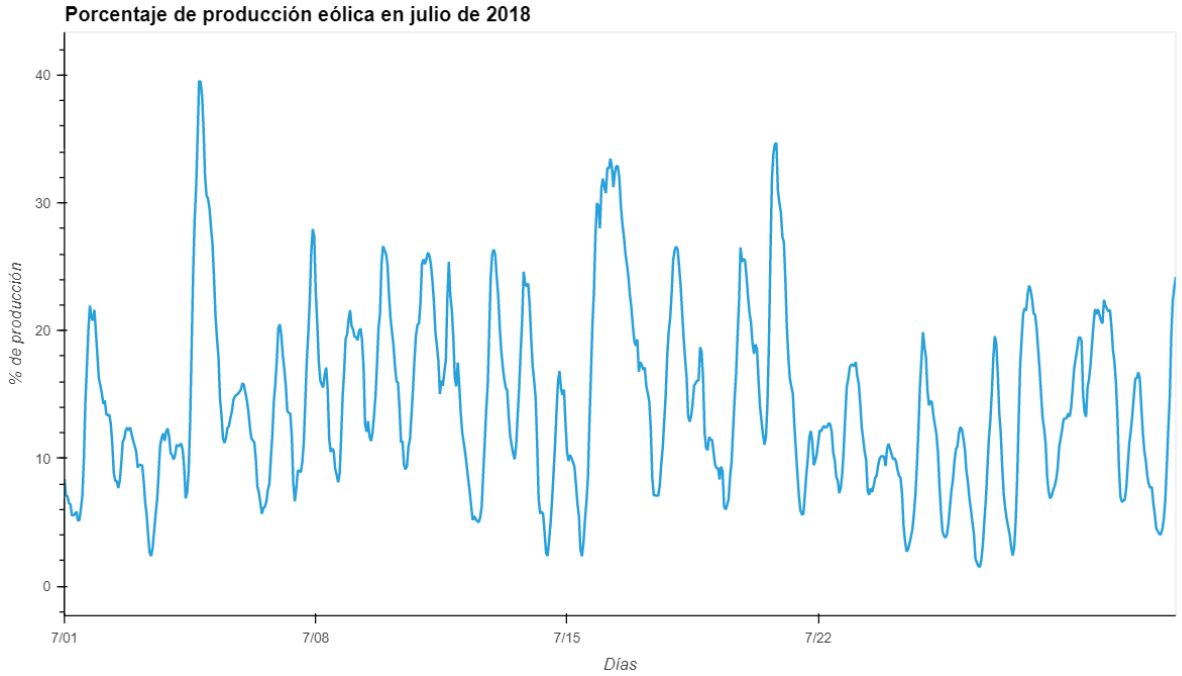

Figura 4.1: Comportamiento de la producción eólica en julio de 2018.

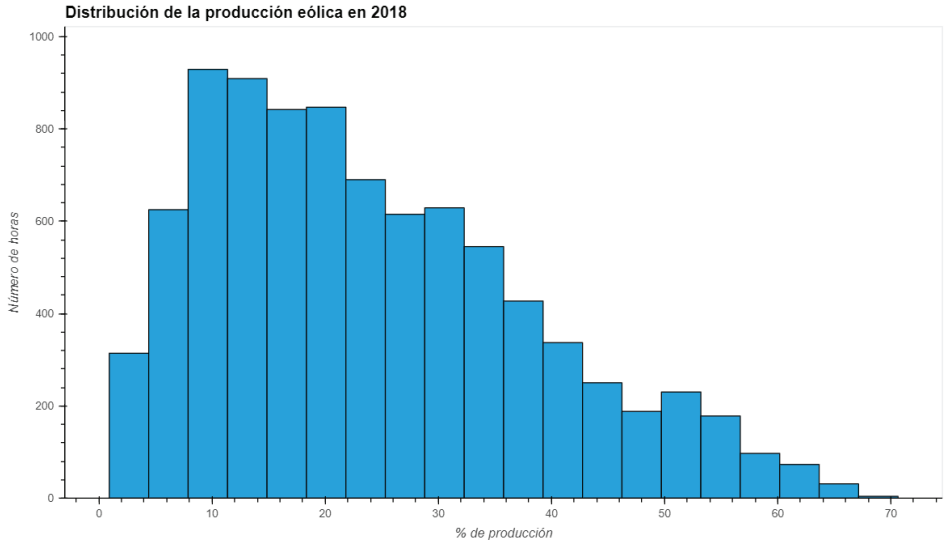

Figura 4.2: Distribución de la producción de energía eólica en Península en 2018.

potencia instalada total es de 17,56 MW.

Por otro lado, la serie temporal de producción fotovoltaica ha sido proporcionada por parte REE con la premisa de experimentar con los mismos. De igual modo que en el caso de Sotavento, la frecuencia de los datos es horaria y abarcan desde el 1 de enero de 2013 al 31 de diciembre de 2015. Cabe destacar que la fotovoltaica de Mallorca no cuenta con grandes parques, sino que alberga una serie de pequeñas instalaciones por toda la isla.

Es importante tener en cuenta que para los dos problemas que se exponen, el eólico

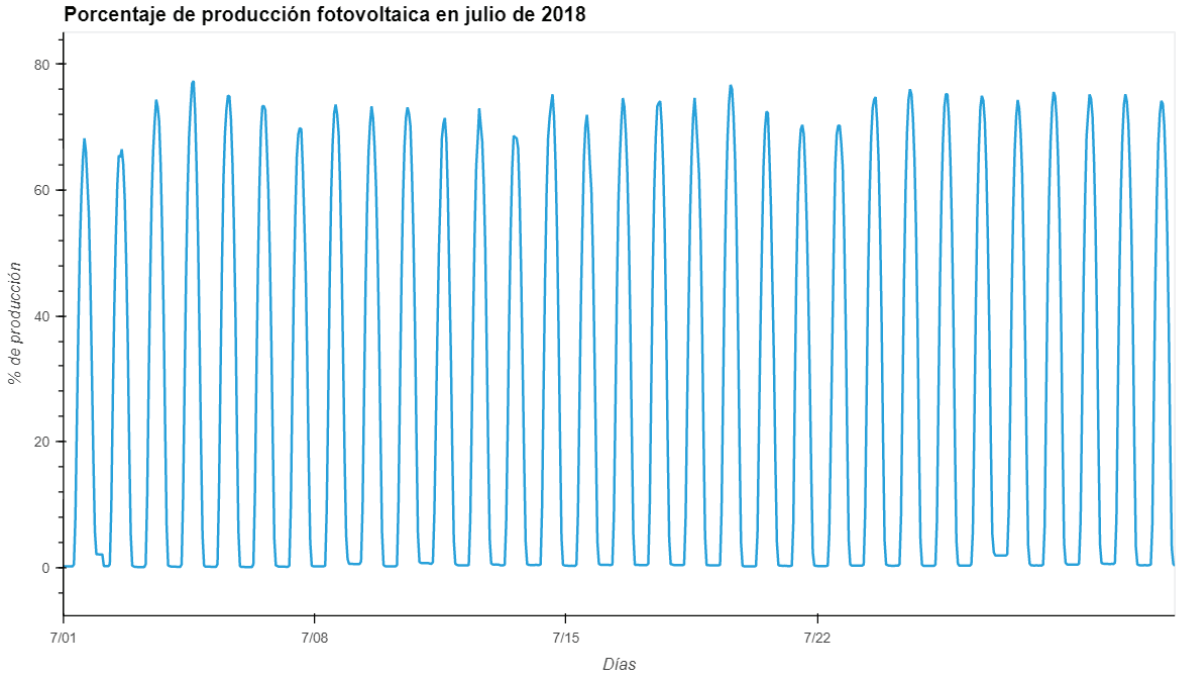

Figura 4.3: Comportamiento de la producción fotovoltaica en Península en julio de 2018.

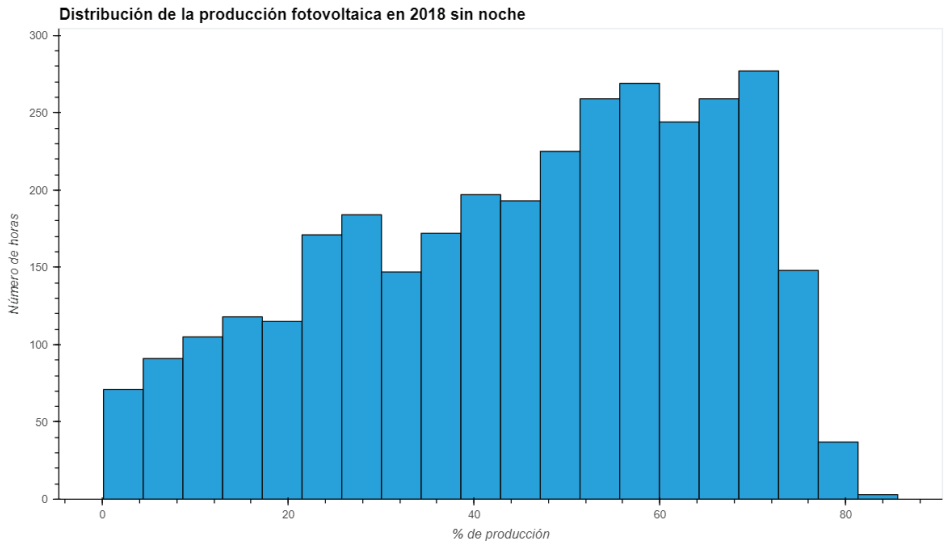

Figura 4.4: Distribución de la producción de energía fotovoltaica sin noche en 2018.

y el fotovoltaico, habrá que tratar los datos de diferente forma. Esto se debe a que su comportamiento es completamente diferente. En la figura 4.1 se puede observar el flujo de la producción eólica a lo largo del mes de julio de 2018. Al contrario que la fotovoltaica, cuyo comportamiento se aprecia en la figura 4.3 para julio de 2018.

Teniendo en cuenta que la velocidad del viento varía constantemente, si se quiere predecir la producción de una turbina eólica es necesario saber exactamente la dirección en la que sopla el viento y con qué velocidad. Normalmente, el viento se mide con un anemómetro y se registra la velocidad media del viento, en este caso, cada 10 minutos.

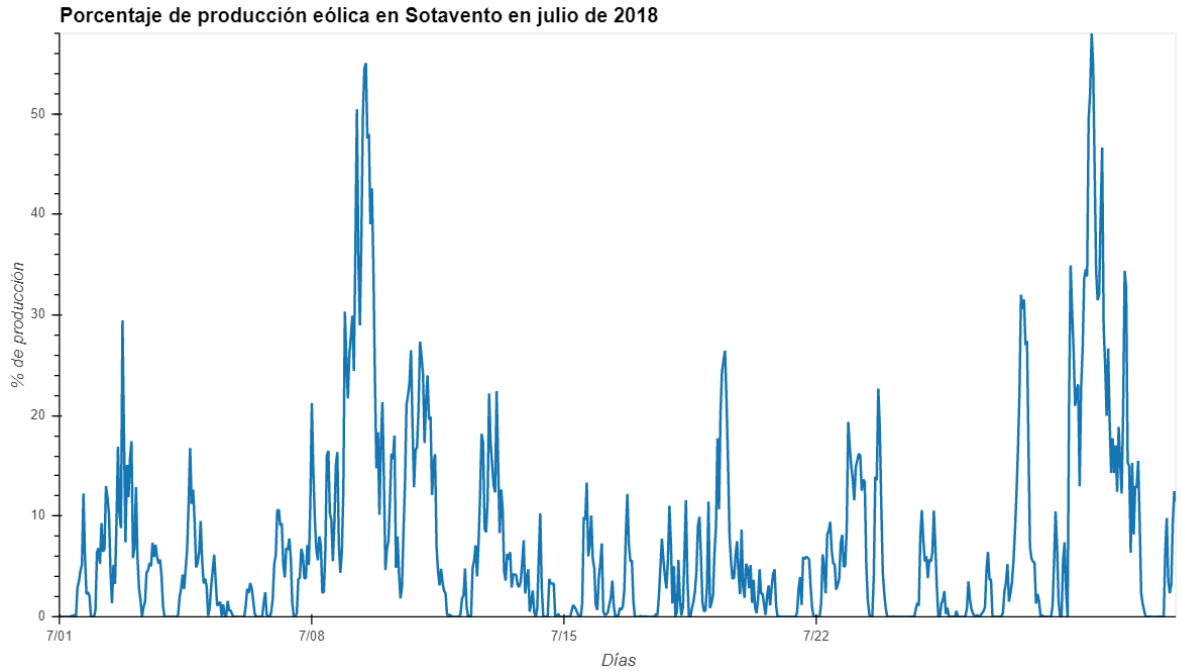

Figura 4.5: Comportamiento de la producción eólica en Sotavento en julio de 2018.

La velocidad del viento en un lugar determinado puede expresarse entonces mediante una distribución *weibull* si su función de densidad es (4.1) para valores de λ semejantes a 1 y de α a 1,5, como se observa en la figura 4.2.

$$f(x) = \lambda\alpha(\lambda x)^{\alpha-1}e^{-(\lambda x)^{\alpha}}x > 0 \tag{4.1}$$

En cambio la distribución anual fotovoltaica tiene un comportamiento distinto al de la eólica. Donde se observaba que la gran mayoría de la producción se mantenía por debajo de un 50 %, en fotovoltaica ocurre lo contrario, siempre y cuando no se tengan en cuenta las horas de noche como se aprecia en la figura 4.4.

Al igual que en los ejemplos de la España peninsular, el problema de Sotavento y el de Mallorca se deberán tratar de diferente forma.

Estos experimentos tienen una complejidad añadida, y es que no sólo se predice a través de la serie temporal de producciones, sino que también se intenta predecir dicha serie a través de predicciones meteorológicas. Para el problema eólico se ha optado por contar con la velocidad del viento a cien metros que como se ha comentado en el capítulo 2, es una de las variables meteorológicas con mayor repercusión en el comportamiento de los parques eólicos.

En el caso de experimento de Mallorca, al ser este un problema fotovoltaico no sólo se puede tener en cuenta la radiación incidente en las placas solares repartidas por toda la isla, sino que habrá que contar con otra variable que permita ajustar con mayor precisión la predicción realizada. Esta variable no es otra que el *total cloud cover* o nubosidad total.

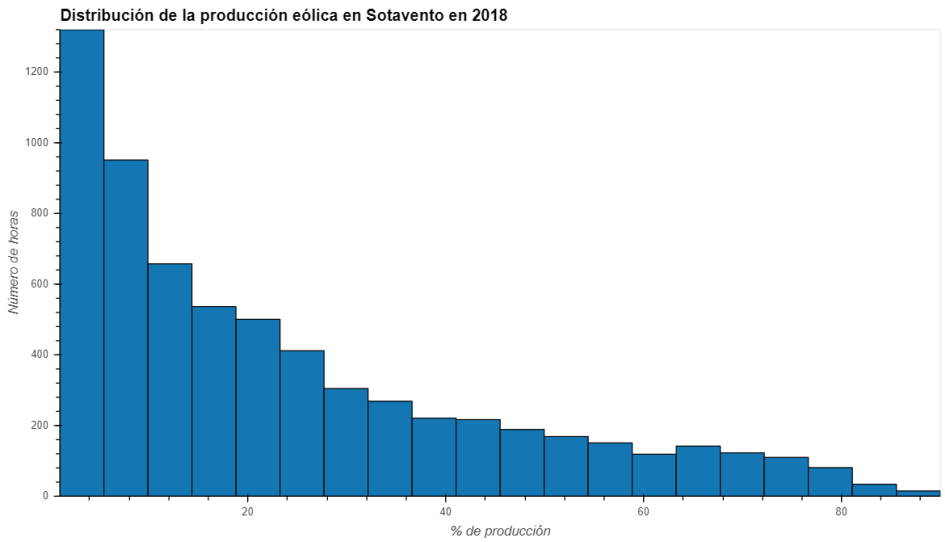

Figura 4.6: Distribución de la producción de energía eólica en Sotavento en 2018.

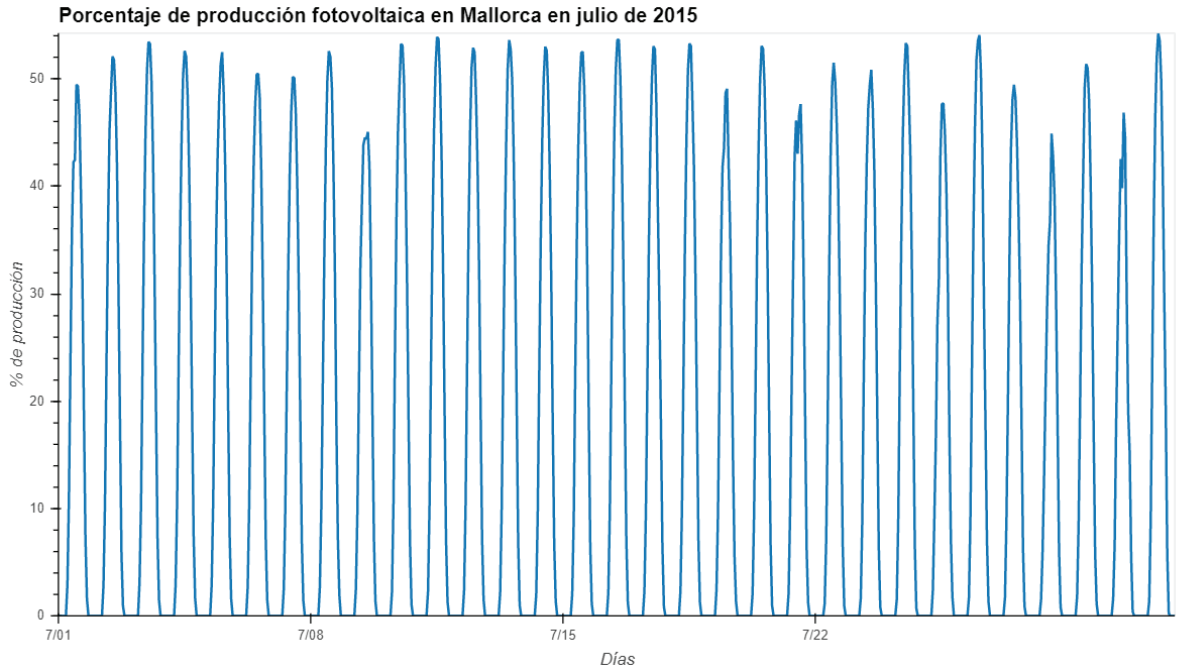

Figura 4.7: Comportamiento de la producción fotovoltaica en Mallorca en julio de 2015.

Como se ha comentado en el capítulo 2 la nubosidad es el gran enemigo de la producción fotovoltaica, causante de la gran mayoría de bajadas de producción. A continuación se exponen las particularidades de cada uno de los problemas así como sus diferencias con el caso más general de la Península ibérica.

En la figura 4.5 se puede observar la evolución de la producción eólica en Sotavento a lo largo del mes de julio de 2018. Su comportamiento es similar al de Península, salvo que por lo general no se ve esa tendencia tan suavizada de Península. En lugar de ello, se observan subidas y bajadas de producción más bruscas, además de alcanzar su mínimo en

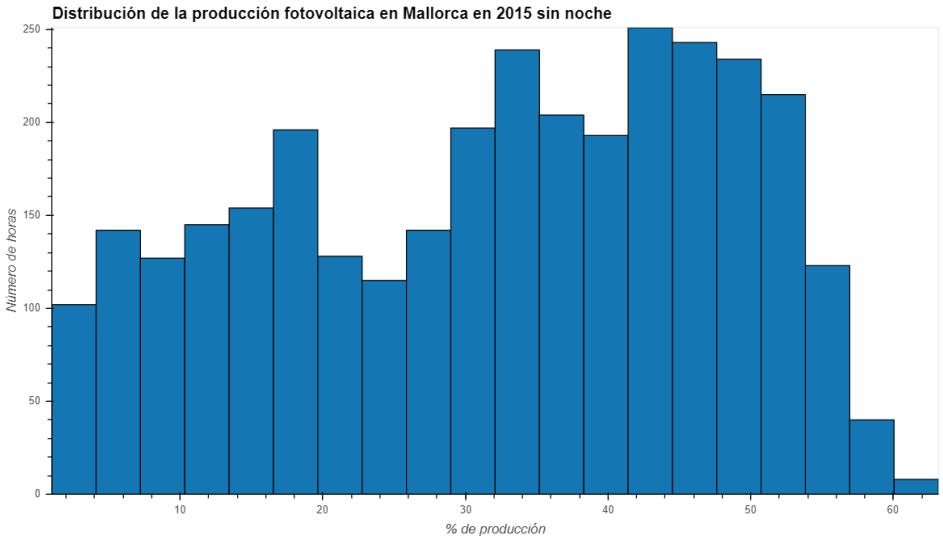

Figura 4.8: Distribución de la producción de energía fotovoltaica sin noche en Mallorca en 2015.

0. La distribución de Sotavento se puede expresar mediante una función *long tail*, donde la mayoría de la producción se encuentra por debajo del 20 %.

Por otro lado, en Mallorca se sigue apreciando el comportamiento que tanto caracteriza a la energía fotovoltaica. En la figura 4.7 se puede apreciar este fenómeno para julio de 2015. Si se analiza con más detalle, se pueden observar variaciones, como en los días 21 y 30. Esto puede deberse a una acumulación de nubes en el momento de bajada de producción. En el momento que vuelve a subir, se puede asumir que las nubes han desaparecido. Su distribución anual tiene un comportamiento particular. Se observaba que la gran mayoría de la producción se mantiene por debajo de un 55 %, siempre y cuando no se tengan en cuenta las horas de noche como se aprecia en la figura 4.8.

La predicción meteorológica ha sido extraída a través del *European Centre for Medium-Range Weather Forecasts* (*ECMWF*) con la ayuda de una de sus APIs para la obtención de meteorología por todo el globo terráqueo. La API del *ECMWF* en cuestión permite, a través de una petición en forma de fichero como el que se muestra en el apéndice A.2, configurar una serie de parámetros que facilitan la selección o filtrado de los datos que se deseen. Como resultado de la petición, se obtiene un archivo en formato *grib*, que contiene para cada fecha y coordenadas solicitadas, los valores de las variables deseadas para cada punto de la malla resultante.

En el fichero de *request* se especifican una serie de parámetros como son: el intervalo de fechas para el que se quieren los datos, el número de predicciones por día así como su resolución, la resolución espacial, las coordenadas en latitud y longitud o las variables deseadas.

4.2. Modelos de Machine Learning y metodología experimental

Son varios los modelos que se van a usar en este proyecto. Entre ellos conviven modelos de dos de las librerías de ciencia de datos más usadas en el ámbito del *machine learning*: *scikit-learn* y *keras*. Con *scikit-learn* se ha optado por desarrollar los modelos más sencillos de este proyecto, como son la *Ridge Regression* y el *Multilayer Perceptron*, mientras que con *keras* se han creado los modelos de redes neuronales recurrentes propuestos anteriormente, *SimpleRNN*, *Gated Recurrent Unit* y *Long Short-Term Memory*.

Para que los modelos de ambas librerías convivan en un mismo entorno, se ha elegido la metodología de *scikit-learn*. Esto se debe a que no sería equivalente comparar los resultados de ambos tipos de modelos si las metodologías de ambas, como pueden ser el *fit* y el *predict*, fueran independientes de cada librería. Para ello, se ha desarrollado un *wrapper* de *keras* denominado *KerasRegressor* que permite utilizar este tipo de funciones directamente sobre un modelo. De esta forma, los resultados obtenidos tendrán la mayor equivalencia posible en cuanto a implementación de los modelos.

Las funciones que se han usado en común para los experimentos son:

- ***Pipeline***. El objetivo del *Pipeline* es ensamblar varios pasos que puedan ser validados de forma cruzada mientras se establecen diferentes parámetros. Permite imponer un orden a los pasos de la implementación del modelo, creando un flujo de trabajo conveniente, que asegura la reproducibilidad del trabajo.

- ***GridSearchCV***. *GridSearchCV* ayuda a recorrer los hiperparámetros predefinidos y a ajustar el modelo en el conjunto de entrenamiento. El objetivo es poder alcanzar los mejores parámetros de la lista de hiperparámetros. Permite, entre otros aspectos, fijar el tipo de error a evaluar o el número de *cores* que ejecutarán la búsqueda de hiperparámetros, permitiendo que esta se agilice.

- ***KFold***. *KFold* permite realizar una validación cruzada [20], procedimiento de remuestreo utilizado para evaluar los modelos de *machine learning* en los datos de entrenamiento. Se puede indicar el número de particiones que se le hará a los datos. Para este proyecto se ha optado por hacer un número de *splits* igual a 2 sin *shuffle*, lo que, dada la distribución de los datos, se entrenará con un año y se validará con otro. Este proceso se repite intercambiando los años y su cometido (entrenamiento y validación).

- ***StandardScaler***. La normalización es un cambio de escala del rango original de los datos para que todos los valores estén aproximadamente dentro un mismo rango. *StandardScaler* ejecuta esta tarea sobre los datos de entrenamiento y test, haciendo que, aunque exista una diferencia notable de magnitud entre los diferentes datos, estos no se verán afectados por su sobredimensionamiento, siendo su media igual a 0 y su desviación típica 1.

- ***TransformedTargetRegressor***. Meta-estimador para hacer una regresión sobre un *target* transformado. Es útil para aplicar una transformación al *target* en problemas de regresión. Su funcionamiento es simple; transforma el *target* utilizando los mismos datos de entrenamiento utilizados para ajustar el modelo, para más tarde aplicar esa transformación inversa en cualquier dato nuevo proporcionado al llamar al método *predict*, devolviendo las predicciones en la escala correcta.

- ***mean_absolute_error***. Calcula la suma de los errores absolutos a lo largo de las observaciones. Gracias a la implementación que proporciona *scikit-learn*, se puede obtener un error de cada uno de las salidas como resultado. Si el parámetro *multioutput* es *raw_values*, entonces se devuelve el error medio absoluto para cada salida por separado, como es el caso de los modelos de este trabajo.

Además, se ha implementado una función para la creación de tensores, detallada en el apéndice A.1, que sirven como entrada a los modelos de redes recurrentes, independientemente del tamaño requerido para la matriz X e Y.

Modelo Ridge

El primer modelo, y quizás el más sencillo, es la ***Ridge Regression***, un modelo simple, rápido y efectivo en problemas de mucha complejidad en series temporales. En el capítulo 3 se ha comentado la teoría que existe detrás de este modelo, por lo que se procede a comentar su hiperparámetro a ajustar para una predicción más precisa.

- **Alpha**. Es un hiperparámetro destinado a la regularización, que combate el sobreajuste restringiendo el tamaño de los pesos. Este hiperparámetro será ajustado en cada uno de los modelos propuestos en este proyecto, aunque en los modelos de redes profundas se le denomina *kernel regularizer*.

Multilayer Perceptron

El segundo modelo con el que se realizarán las pruebas es la ***Multilayer Perceptron***, con el que se espera mejorar el resultado del modelo lineal anterior. Sus hiperparámetros a ajustar, además de alpha, son los siguientes:

- **Número de neuronas y capas ocultas**. Tanto el número de capas ocultas como el número de neuronas en cada una de estas ha de considerarse cuidadosamente, ya que tiene una enorme influencia en el resultado final. Si se utilizan muy pocas neuronas en las capas ocultas, se producirá lo que se denomina infraajuste.

  Por otro lado, el uso de demasiadas neuronas en las capas ocultas puede dar lugar a varios problemas. Puede producir un sobreajuste, además de que un número desmesurado de neuronas en las capas ocultas puede aumentar el tiempo de entrenamiento en gran medida.

Como se ha mencionado en el capítulo 3, se han usado varios optimizadores entre los que se encuentran Adam y lbfgs.

Redes neuronales recurrentes

El resto de modelos recurrentes (***SimpleRNN***, ***Gated Recurrent Unit*** y ***Long Short-Term Memory***) se hiperparametrizarán con la misma serie de hiperparámetros. Cabe destacar que la complejidad de la *SimpleRNN* es la menor, debido a que se ha elegido el modelo de *keras* para una primera toma de contacto con las redes recurrentes.

Su arquitectura se define con dos capas del modelo en cuestión, ambas de 32 o 64 unidades. La primera de estas capas recibe las entradas del modelo y a continuación distribuye las salidas a la otra capa neuronal que a su vez devolverá sus resultados a una capa *Dense*, mostrando así el resultado final del modelo.

A continuación se enumeran los hiperparámetros, además del anteriormente mencionado *kernel regularizer*:

- ***Kernel Regularizer***. Al igual que el alpha de la *Ridge*, este hiperparámetro también está destinado a la regularización, combatiendo el sobreajuste restringiendo el tamaño de los pesos. Aplica una penalización directamente en el *kernel* de la capa.

- ***Batch Size***. Define el número de muestras con las que se trabaja antes de actualizar los parámetros internos del modelo.

- **Función de activación**. La función de activación define la salida de una neurona/nodo dada una entrada o conjunto de entradas. Las funciones que se han decidido hiperparametrizar son las funciones *ReLU*, sigmoide y tangente.

- **Número de épocas**. Una época se refiere a un ciclo a través del conjunto de datos de entrenamiento. Normalmente, el entrenamiento de una red neuronal requiere más de unas pocas épocas. En otras palabras, si se alimenta una red neuronal con los datos de entrenamiento durante más de una época en diferentes patrones, se espera una mejor generalización cuando se le dé una nueva entrada "no vista"(datos de prueba).

  Una de sus características es que (especialmente para conjuntos de entrenamiento grandes) otorga a la red la oportunidad de ver los datos anteriores para reajustar los hiperparámetros del modelo, de modo que éste no esté sesgado hacia los últimos datos durante el entrenamiento.

Los hiperparámetros óptimos se obtienen mediante un *grid search*; en la tabla 4.1 se muestran los valores de los hiperparámetros utilizados para cada modelo.

Modelo	Parámetro	Rango
Ridge	Alpha	$[\ 10^{-5}, 10^{-4}, ..., 10^{4}\ ]$
MLP	Alpha	$[\ 10^{-5}, 10^{-4}, ..., 10^{4}\ ]$
	Número de capas y neuronas	{(20), (20, 20), (20, 20, 20)}
RNNs	Kernel Regularizer	$[\ 10^{-5}, 10^{-4}, ..., 10^{4}\ ]$
	Batch Size	{256, 512, 1024}
	Función de activación	{relu, sigmoid, tanh}
	Número de épocas	{100, 200, 300, 500, 750, 1000, 1500, 3000}

Tabla 4.1: Conjunto de valores de hiperparámetros para los modelos considerados.

Medida del error

Existen múltiples formas de medir el error de lo modelos que se proponen para este proyecto. De entre todos ellos, se ha elegido el Mean Absolute Error normalizado (NMAE), una variante del MAE tradicional salvo que este muestra un error porcentual basado en la potencia instalada de cada momento. A sabiendas de que el MAE es uno de los métodos más usados en predicción de series temporales, se ha optado por esta variante, cuya forma es la siguiente:

$$NMAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \cdot \frac{1}{\text{potencia}},$$

donde y muestra el valor real mientras que $\hat{y}$ el valor predicho. Por lo tanto, para calcular la precisión de los modelos se va a medir en términos del NMAE.

Este se calcula mediante el sumatorio de las diferencias de estos dos valores, normalizando el resultado a través de la potencia instalada. Finalmente se divide por el número total de muestras. Esto permite obtener un error en tanto por 100, siendo este un porcentaje del error frente a la potencia total instalada.

4.3. Predicción eólica en la España peninsular

El primer experimento de ámbito eólico tiene como objetivo la predicción de producción eólica en Península a lo largo de 2018 a partir de los datos de 2016 y 2017 con frecuencia diez-minutal. Cabe destacar, que tan sólo se realizará la predicción a partir de la serie temporal de producción eólica. De este modo, 2016 y 2017 irán al conjunto de entrenamiento (con su correspondiente permutación entre conjunto de entrenamiento y validación) y 2018 para test. El conjunto de datos de entrenamiento cuenta con 105216 registros y 24 o 48 columnas para los modelos *Ridge* y *MLP*. Estas columnas dependen del número de retrasos con los que se quiera trabajar la serie temporal, por lo que estas contienen los 24 o 48 datos anteriores al primer horizonte de predicción. Por lo tanto, 24

Modelo	1	2	3	4	5	6	7	8	9	10	11	12	Promedio
Ridge_24	0,23	0,39	0,54	0,69	0,83	0,98	1,12	1,27	1,42	1,56	1,70	1,85	1,05
Ridge_48	0,23	0,39	0,54	0,69	0,83	0,98	1,12	1,27	1,41	1,55	1,70	1,84	1,05
MLP_24	0,27	0,41	0,55	0,70	0,84	0,98	1,12	1,27	1,42	1,55	1,69	1,83	1,05
MLP_48	0,27	0,40	0,54	0,71	0,84	0,99	1,12	1,27	1,41	1,53	1,69	1,82	1,05
RNN_24	0,27	0,41	0,56	0,71	0,84	0,97	1,10	1,23	1,37	1,51	1,65	1,78	1,03
RNN_48	0,23	0,38	0,52	0,67	0,80	0,94	1,07	1,21	1,35	1,48	1,62	1,75	1,00
GRU_24	0,23	0,38	0,54	0,69	0,83	0,97	1,13	1,26	1,39	1,53	1,67	1,80	1,04
GRU_48	0,28	0,42	0,58	0,70	0,85	0,98	1,13	1,26	1,40	1,55	1,68	1,82	1,05
LSTM_24	0,25	0,40	0,55	0,69	0,83	0,97	1,11	1,24	1,38	1,52	1,65	1,79	1,03
LSTM_48	0,23	0,39	0,54	0,68	0,82	0,96	1,10	1,24	1,38	1,52	1,66	1,80	1,03

Tabla 4.2: Errores de predicción eólica en la España peninsular.

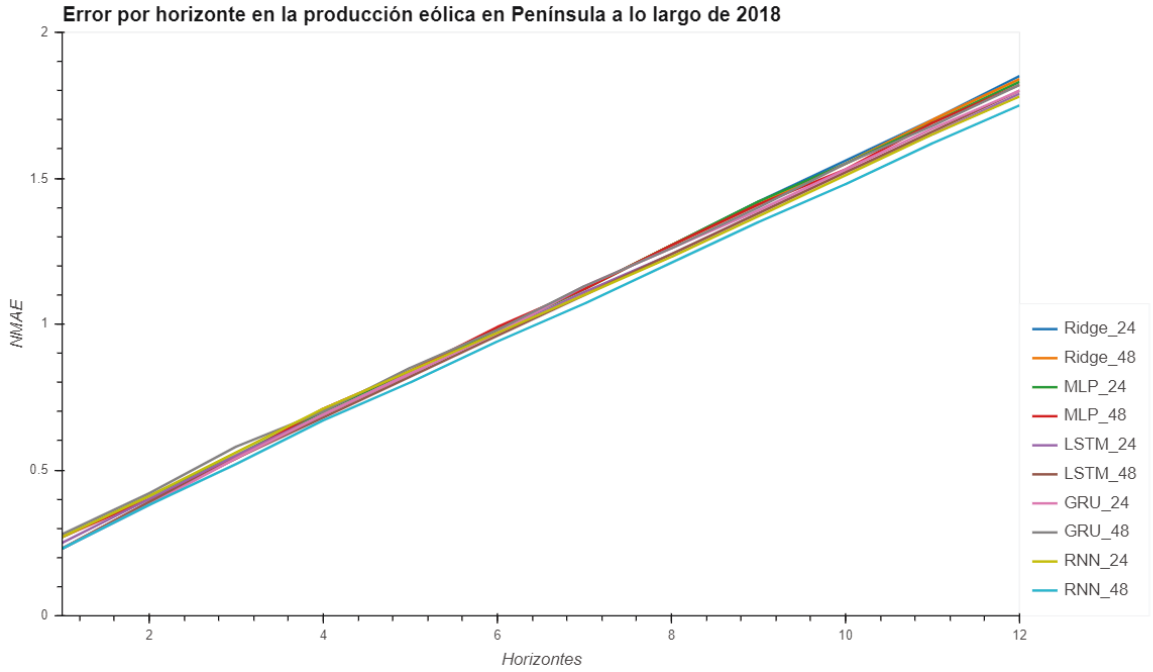

Figura 4.9: Error por horizonte en la producción eólica en Península a lo largo de 2018.

y 48 son también hiperparámetros del modelo, ya que se busca el mejor de ellos. Como se ha mencionado en el capítulo 3, los modelos de redes recurrentes son alimentados con tensores de la forma [*num_muestras, lag, num_features*] ([256/512/1024, 24/48, 1]).

El objetivo del modelo es predecir los siguientes 12 horizontes, donde horizonte equivale al tiempo que transcurre entre un dato y otro (diez minutos), a partir de los 24 o 48 horizontes anteriores. Es decir, el primer horizonte se encuentra a 10 minutos vista y el último a 2 horas vistas. Por lo tanto, existen un par de modelos para cada tipo de modelo descrito con anterioridad.

Los resultados obtenidos se pueden consultar en la tabla 4.2. La columna *Modelo* define cada tipo de modelo además del *lag* o retraso utilizado en cada uno. Cabe destacar que la *SimpleRNN* se encuentra catalogada como RNN, tanto en las tablas como en sus

respectivas gráficas. Cada una de las tablas que se exponen en este trabajo sigue el mismo patrón; modelos más sencillos como la *Ridge* o el *MLP* con sus correspondientes retrasos, y a continuación las redes recurrentes. El resto de columnas muestran el *NMAE* por horizonte, mientras que la última columna indica el promedio de los mismos.

El **mejor resultado** es el obtenido por el **modelo *SimpleRNN* con 48 horizontes** con un **NMAE promedio del 1 %**, aunque no existe una diferencia significativa ni siquiera con el peor de los modelos estudiados, con un 1,05 % de NMAE promedio. La arquitectura del modelo ganador cuenta con dos capas de 32 unidades cada una y sus hiperparámetros elegidos han sido 0,01 para el *kernel regularizer*, un *batch size* de 256, la función de activación ha sido la *relu* y el número de épocas igual a 200.

La figura 4.9 permite ver a simple vista que los errores por horizonte crecen de forma lineal y al unísono. Esto quiere decir que independientemente del modelo utilizado, el resultado no se verá afectado en gran medida ya que la frecuencia diez-minutal y el comportamiento de la eólica peninsular hacen que la elección del modelo sea poco relevante a la hora de predecir.

4.4. Predicción eólica en Sotavento

El segundo experimento de ámbito eólico tiene como objetivo la predicción de producción eólica en el parque de Sotavento a lo largo de 2018 a partir de los datos de 2016 y 2017 con frecuencia horaria. De este modo, al igual que en el experimento anterior, 2016 y 2017 irán al conjunto de entrenamiento (con su correspondiente permutación entre conjunto de entrenamiento y validación) y 2018 para test. El conjunto de datos de entrenamiento cuenta con 17520 registros.

En cuanto al número de columnas, estas podrán variar debido a las diferentes pruebas dentro del mismo experimento. Si tan sólo se usa la serie temporal de producción de Sotavento para predecir, el modelo contará con 24 o 48 columnas para los modelos *Ridge* y *MLP*. Al igual que en el problema anterior, estas columnas dependen del número de retrasos con los que se quiera trabajar la serie temporal, por lo que estas contienen los 24 o 48 datos anteriores al primer horizonte de predicción. Lo mismo ocurre con los modelos de redes recurrentes, que son alimentados con tensores de la forma [*num_muestras, lag, num_features*] ([256/512/1024, 24/48, 1]).

Como se ha comentado al comienzo de este capítulo, para este experimento también se ha contado con la velocidad del viento a cien metros. En el caso de de usar también la coordenada de meteorología más cercana al parque, el número de columnas se multiplica por dos, obteniendo 48 o 96 columnas respectivamente, correspondientes a la velocidad del viento en cada uno de esos retrasos para la coordenada más cercana al parque, más la serie temporal de producciones anteriormente mencionada. Por último, se pueden obtener 240 o 480 columnas, que corresponden la matriz tres por tres de coordenadas de meteorología calculadas a partir de la coordenada más cercana al parque, más la serie temporal de producciones anteriormente mencionada. Esta matriz tiene como punto central la coordenada más cercana al parque y alrededor de ella se encuentran los demás

Modelo	1	2	3	4	5	6	7	8	9	10	11	12	Promedio
Ridge_24	4,00	5,75	6,91	7,76	8,47	9,07	9,58	10,00	10,43	10,81	11,12	11,43	8,78
Ridge_48	3,99	5,73	6,88	7,74	8,44	9,03	9,55	9,96	10,37	10,73	11,04	11,33	8,73
MLP_24	4,12	5,47	6,33	6,98	7,47	7,82	8,21	8,44	8,70	9,06	9,37	9,81	7,65
MLP_48	4,18	5,23	5,71	5,99	6,19	6,31	6,38	6,54	6,61	6,88	7,35	7,91	6,27
RNN_24	3,79	4,86	5,34	5,51	5,60	5,62	5,70	5,89	6,18	6,50	6,89	7,41	5,77
RNN_48	4,03	5,24	5,86	6,16	6,27	6,56	6,67	6,96	7,18	7,46	7,80	8,29	6,54
GRU_24	3,92	5,45	6,48	7,20	7,86	8,38	8,81	9,29	9,59	9,91	10,04	10,36	8,11
GRU_48	4,00	5,50	6,44	7,10	7,61	8,19	8,47	8,76	9,05	9,25	9,43	9,79	7,80
LSTM_24	3,90	5,45	6,48	7,19	7,86	8,30	8,78	9,27	9,62	9,89	10,14	10,47	8,11
LSTM_48	4,44	5,73	6,84	7,39	7,85	8,41	8,75	9,05	9,35	9,54	9,73	10,06	8,10

Tabla 4.3: Errores de predicción eólica en Sotavento.

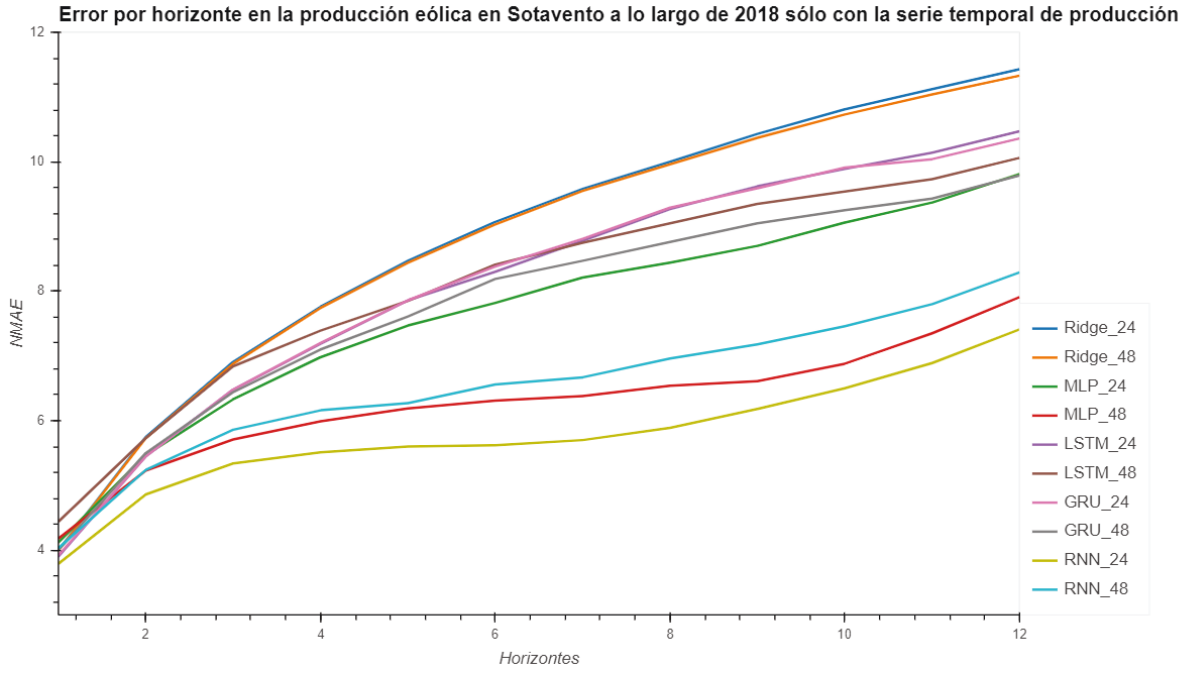

Figura 4.10: Error por horizonte en la producción eólica en Sotavento a lo largo de 2018 sólo con la serie temporal de producción.

puntos, formando nueve puntos en total.

El objetivo del modelo es predecir los siguientes 12 horizontes, donde horizonte equivale al tiempo que transcurre entre un dato y otro (una hora), a partir de los 24 o 48 horizontes anteriores. Es decir, el primer horizonte se encuentra a 1 hora vista y el último a 12 horas vistas. Por lo tanto, existen seis modelos para cada tipo de modelo descrito con anterioridad.

Los resultados obtenidos se dividen en tres tablas distintas con sus respectivas figuras, separados en los tres ejemplos mencionados en esta misma sección. En la tabla 4.3 se pueden consultar los errores de los modelos sólo con la serie temporal de producción. La figura 4.10 muestra como todos los modelos parten en torno al 4% de NMAE, y poco a poco se bifurcan hasta generar un intervalo de aproximadamente cuatro puntos de

Modelo	1	2	3	4	5	6	7	8	9	10	11	12	Promedio
Ridge_1c_24	3,96	5,58	6,64	7,48	8,15	8,75	9,28	9,72	10,17	10,54	10,87	11,21	8,53
Ridge_1c_48	3,96	5,57	6,61	7,45	8,13	8,72	9,25	9,68	10,12	10,48	10,80	11,11	8,49
MLP_1c_24	4,08	4,85	5,31	5,52	5,68	5,86	6,00	6,17	6,29	6,49	6,94	7,56	5,90
MLP_1c_48	4,00	4,58	4,72	4,73	4,76	4,82	4,79	4,83	4,80	4,86	5,13	5,78	4,82
RNN_1c_24	3,88	5,00	5,81	6,40	6,83	7,25	7,78	8,39	8,87	9,10	9,34	9,78	7,37
RNN_1c_48	3,86	5,16	5,95	6,57	7,05	7,62	7,95	8,21	8,52	8,76	8,98	9,32	7,33
GRU_1c_24	3,72	4,97	5,77	6,39	6,81	7,24	7,69	8,08	8,45	8,72	8,99	9,41	7,19
GRU_1c_48	3,90	5,03	5,66	6,12	6,41	6,83	7,00	7,23	7,44	7,65	7,88	8,25	6,62
LSTM_1c_24	3,80	4,59	5,10	5,37	5,56	5,72	5,85	6,05	6,23	6,40	6,77	7,40	5,74
LSTM_1c_48	3,93	4,80	5,38	5,67	5,85	6,09	6,18	6,42	6,59	6,78	7,01	7,42	6,01

Tabla 4.4: Errores de predicción eólica en Sotavento con una coordenada.

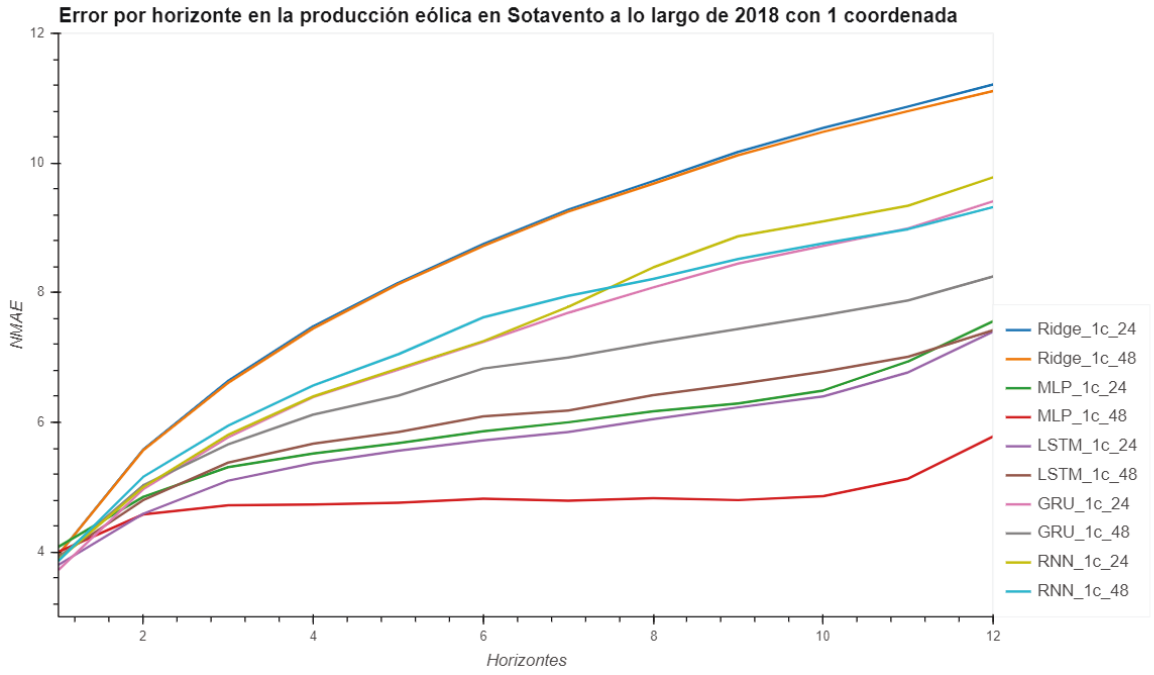

Figura 4.11: Error por horizonte en la producción eólica en Sotavento a lo largo de 2018 con 1 coordenada.

diferencia.

El **mejor resultado** es el obtenido por el **modelo *SimpleRNN* con 24 horizontes** con un **NMAE promedio del 5,77 %**. La arquitectura del modelo ganador cuenta con dos capas de 32 unidades cada una y sus hiperparámetros elegidos han sido 0,01 para el *kernel regularizer*, un *batch size* de 256, la función de activación ha sido la *relu* y el número de épocas igual a 300. Le siguen muy de cerca la *MLP* con 48 horizontes y la propia *SimpleRNN* con 48 horizontes.

En la tabla 4.4 se pueden consultar los errores de los modelos con la serie temporal de producción y la coordenada más cercana al parque. La figura 4.11 muestra como todos los modelos; al igual que en el ejemplo anterior, parten en torno al 4 % de NMAE, y poco a poco se bifurcan hasta generar un intervalo de prácticamente seis puntos de diferencia.

Modelo	1	2	3	4	5	6	7	8	9	10	11	12	Promedio
Ridge_9c_24	3,93	5,48	6,49	7,29	7,92	8,47	8,98	9,42	9,82	10,17	10,48	10,78	8,27
Ridge_9c_48	3,93	5,46	6,44	7,23	7,85	8,39	8,89	9,32	9,71	10,06	10,36	10,64	8,19
MLP_9c_24	3,90	4,34	4,50	4,55	4,50	4,52	4,57	4,57	4,58	4,68	4,95	5,64	4,61
MLP_9c_48	3,85	4,19	4,25	4,25	4,18	4,20	4,22	4,22	4,12	4,01	4,20	4,92	4,22
RNN_9c_24	3,88	4,91	5,51	5,92	6,23	6,49	6,67	6,84	7,09	7,40	7,75	8,25	6,41
RNN_9c_48	4,21	5,30	6,04	6,70	7,25	7,89	8,29	8,66	9,04	9,38	9,62	10,06	7,70
GRU_9c_24	3,78	4,43	4,71	4,79	4,76	4,78	4,74	4,74	4,72	4,80	5,13	5,73	4,76
GRU_9c_48	3,37	3,59	3,42	3,34	3,30	3,36	3,35	3,35	3,37	3,42	3,68	4,29	3,49
LSTM_9c_24	3,75	3,83	3,84	3,82	3,77	3,72	3,71	3,78	3,74	3,85	4,22	4,91	3,91
LSTM_9c_48	3,67	3,84	3,88	3,90	3,89	3,90	3,89	3,90	3,85	3,84	4,21	4,96	3,98

Tabla 4.5: Errores de predicción eólica en Sotavento con nueve coordenadas.

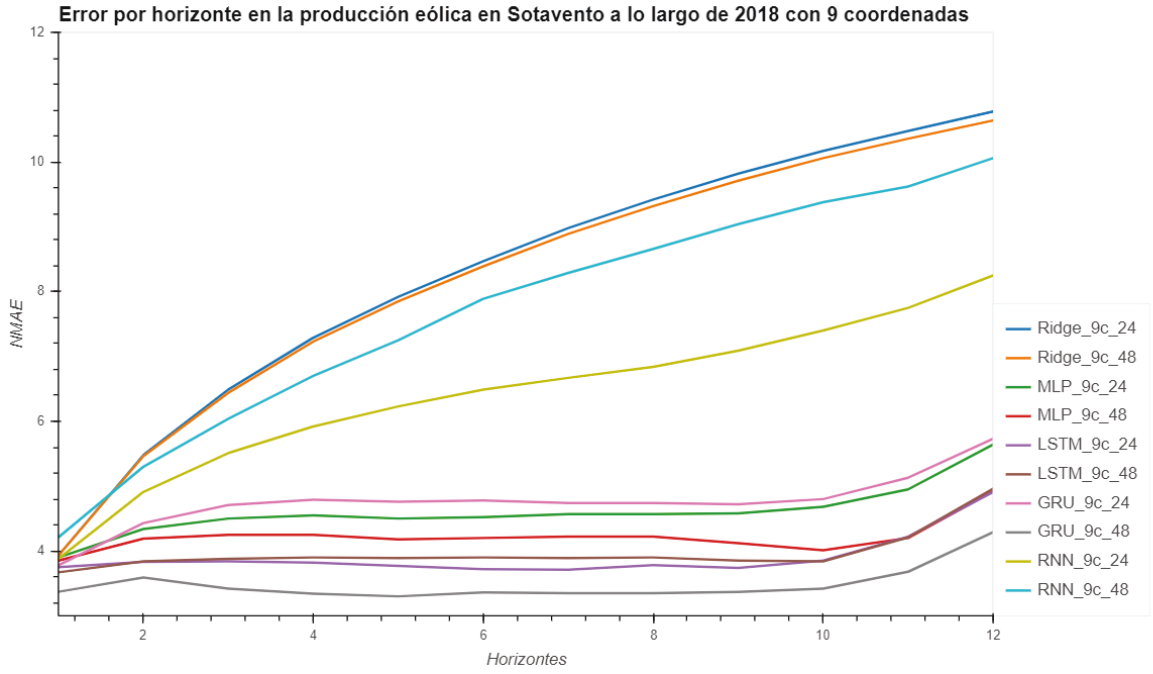

Figura 4.12: Error por horizonte en la producción eólica en Sotavento a lo largo de 2018 con 9 coordenadas.

El **mejor resultado** es el obtenido por el **modelo** *MLP* **con 48 horizontes** con un **NMAE promedio del 4,82 %**. La arquitectura del modelo ganador cuenta con dos capas de 20 unidades cada una y un alpha de 0,1.

Se observa que la mayoría de modelos mejoran su predicción salvo por el modelo *Ridge*, el cual se estanca y no varía su error. En cambio el modelo vencedor en la anterior agrupación empeora su resultado al aumentar el número de variables predictivas.

En la tabla 4.5 se pueden consultar los errores de los modelos con la serie temporal de producción y las nueve coordenadas más cercanas al parque. La figura 4.12 muestra como todos los modelos, al igual que en los ejemplos anteriores, parten en torno al 4 % de NMAE, y poco a poco se bifurcan hasta generar un intervalo de aproximadamente seis puntos de diferencia. En esta última gráfica se observa una mejora con respecto a los dos

ejemplos anteriores desde el primer horizonte, salvo por el modelo *Ridge*.

El **mejor resultado** es el obtenido por el **modelo GRU con 48 horizontes** con un **NMAE promedio del 3,49 %**, y por tanto el mejor modelo para el problema eólico en Sotavento. La arquitectura del modelo ganador cuenta con dos capas de 64 unidades cada una y sus hiperparámetros elegidos han sido 0,1 para el *kernel regularizer*, un *batch size* de 512, la función de activación ha sido la *relu* y el número de épocas igual a 500. Este modelo se muestra con una gran continuidad hasta el décimo horizonte donde despega claramente el error. Le siguen muy de cerca ambas *LSTM* y la *MLP* con 48 horizontes.

Se observa que la mayoría de modelos continúan mejorando su predicción salvo por las dos excepciones de la agrupación anterior; el modelo *Ridge* y el modelo vencedor en la primera agrupación (*SimpleRNN* con 24 horizontes), el cual claramente se ve poco favorecido al aumentar el número de variables predictivas.

Todos los modelos siguen una progresión conforme se va aumentando el número de variables predictivas. Esto tiene cierto sentido ya que como se ha hablado en el capítulo 2, se han estudiado las variables meteorológicas que mejor funcionan para cada tipo de problema. No obstante se observa que la *SimpleRNN* empeora con este fenómeno. Esto puede deberse a que el modelo no está preparado para una cantidad tan grande de variables y, de algún modo, el modelo no logra interpretar correctamente y tan sólo aporta ruido al proceso.

4.5. Predicción fotovoltaica en la España peninsular

El primer experimento de ámbito fotovoltaico tiene como objetivo la predicción de producción fotovoltaica en la España peninsular a lo largo de 2018 a partir de los datos de 2016 y 2017 con frecuencia diez-minutal. Cabe destacar, que al igual que en el ejemplo de eólica, tan sólo se realizará la predicción a partir de la serie temporal de producción fotovoltaica. De este modo, 2016 y 2017 irán al conjunto de entrenamiento (con su correspondiente permutación entre conjunto de entrenamiento y validación) y 2018 para test. El conjunto de datos de entrenamiento cuenta con 105216 registros y 24 o 48 columnas para los modelos *Ridge* y *MLP*. Al igual que en los problemas anteriores, estas columnas dependen del número de retrasos que contenga la serie temporal, por lo que estas contienen los 24 o 48 datos anteriores al primer horizonte de predicción. Lo mismo ocurre con los modelos de redes recurrentes, que son alimentados con tensores de la forma [*num_ muestras, lag, num_features*] ([256/512/1024, 24/48, 1]).

El objetivo del modelo es predecir los siguientes 12 horizontes, donde horizonte equivale al tiempo que transcurre entre un dato y otro (diez minutos), a partir de los 24 o 48 horizontes anteriores. Es decir, el primer horizonte se encuentra a 10 minutos vista y el último a 2 horas vistas. Por lo tanto, existen un par de modelos para cada tipo de modelo descrito con anterioridad. Los resultados obtenidos se pueden consultar en la tabla 4.6.

El **mejor resultado** es el obtenido por el **modelo *SimpleRNN* con 24 horizontes** con un **NMAE promedio del 1,15 %**. La arquitectura del modelo ganador cuenta con

Modelo	1	2	3	4	5	6	7	8	9	10	11	12	Promedio
Ridge_24	0,39	0,69	1,04	1,46	1,93	2,45	3,01	3,61	4,23	4,87	5,52	6,17	2,95
Ridge_48	0,39	0,68	1,03	1,44	1,90	2,41	2,95	3,53	4,12	4,73	5,34	5,96	2,87
MLP_24	0,43	0,61	0,75	0,95	1,11	1,38	1,67	1,97	2,33	2,65	3,01	3,45	1,69
MLP_48	0,48	0,58	0,74	0,91	1,05	1,31	1,54	1,83	2,09	2,41	2,72	3,07	1,56
RNN_24	0,27	0,39	0,51	0,65	0,80	0,98	1,16	1,35	1,56	1,78	2,02	2,28	1,15
RNN_48	0,30	0,42	0,56	0,71	0,88	1,08	1,27	1,49	1,72	1,96	2,22	2,50	1,26
GRU_24	0,33	0,48	0,65	0,83	1,04	1,28	1,56	1,84	2,13	2,43	2,77	3,15	1,54
GRU_48	0,49	0,69	0,93	1,22	1,56	1,87	2,26	2,70	3,12	3,56	4,07	4,59	2,26
LSTM_24	0,45	0,66	0,77	0,97	1,16	1,41	1,73	2,00	2,35	2,69	3,08	3,48	1,73
LSTM_48	0,50	0,71	0,84	1,03	1,23	1,43	1,77	2,07	2,41	2,72	3,12	3,52	1,78

Tabla 4.6: Errores de predicción fotovoltaica en la España peninsular.

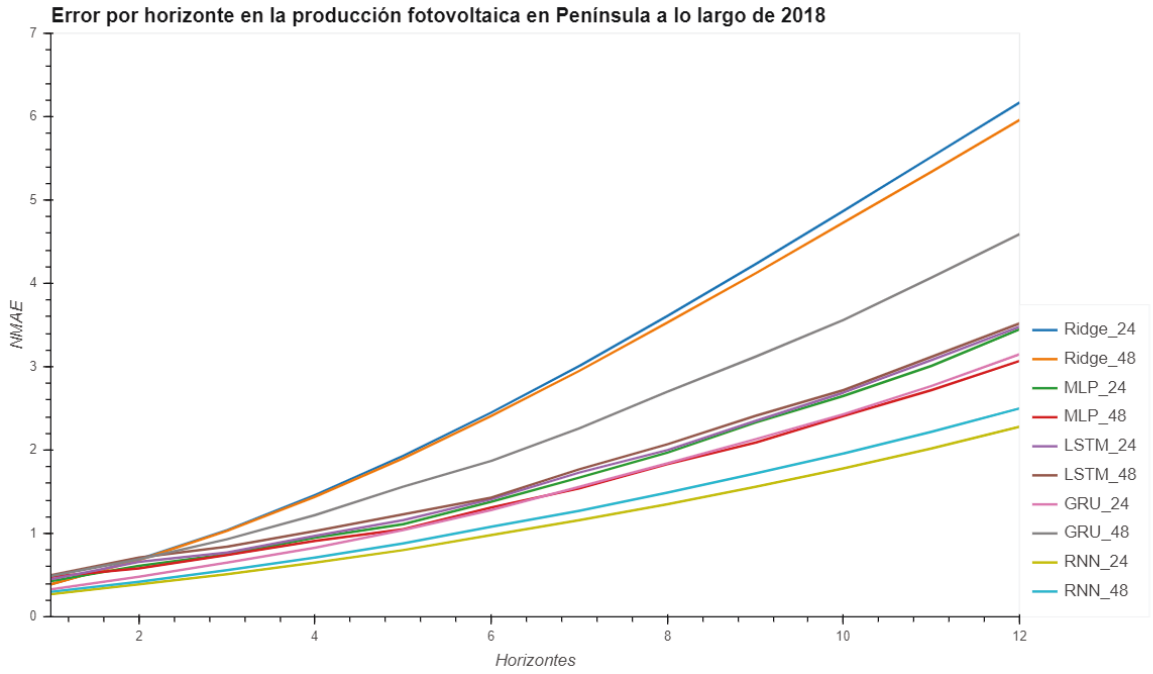

Figura 4.13: Error por horizonte en la producción fotovoltaica en Península a lo largo de 2018.

dos capas de 32 unidades cada una y sus hiperparámetros elegidos han sido 0,01 para el *kernel regularizer*, un *batch size* de 256, la función de activación ha sido la *relu* y el número de épocas igual a 300. Le siguen muy de cerca la propia *SimpleRNN* con 48 horizontes y un poco más alejados la *MLP* de 48 horizontes y la *GRU* de 24 horizontes.

La figura 4.13 muestra como todos los modelos parten de entre el 0,3 % y el 0,5 % de NMAE, y poco a poco se bifurcan hasta generar un intervalo de aproximadamente cuatro puntos de diferencia.

4.6. Predicción fotovoltaica en Mallorca

El segundo experimento de ámbito fotovoltaico tiene como objetivo la predicción de producción fotovoltaica en la isla de Mallorca a lo largo de 2015 a partir de los datos de 2013 y 2014 con frecuencia horaria. De este modo, al igual que en el experimento anterior, 2013 y 2014 irán al conjunto de entrenamiento (con su correspondiente permutación entre conjunto de entrenamiento y validación) y 2015 para test. El conjunto de datos de entrenamiento cuenta con 17520 registros.

En cuanto al número de columnas, estas podrán variar debido a las diferentes pruebas dentro del mismo experimento. Si tan sólo se usa la serie temporal de producción de Mallorca para predecir, el modelo contará con 24 o 48 columnas para los modelos *Ridge* y *MLP*. Al igual que en los problemas anteriores, estas columnas dependen del número de retrasos que contenga la serie temporal, por lo que estas contienen los 24 o 48 datos anteriores al primer horizonte de predicción. Lo mismo ocurre con los modelos de redes recurrentes, que son alimentados con tensores de la forma [*num_muestras, lag, num_features*] ([256/512/1024, 24/48, 1]).

En el caso de de usar también la coordenada de meteorología más cercana al parque, el número de columnas se multiplica por tres, obteniendo 72 o 144 columnas respectivamente, correspondientes a la radiación solar superficial y el porcentaje de nubosidad en cada uno de esos retrasos para la coordenada situada en el centro de la isla, más la serie temporal de producciones anteriormente mencionada. Por último, se pueden obtener 720 o 1440 columnas, que corresponden la matriz tres por tres de coordenadas de meteorología calculadas a partir de la coordenada situada en el centro de la isla, más la serie temporal de producciones anteriormente mencionada. Esta matriz tiene como punto central la coordenada situada en el centro de la isla y alrededor de ella se encuentran los demás puntos, formando nueve puntos en total. No obstante, estos puntos han sido elegidos con una lógica distinta al resto de problemas dada la complejidad añadida de no tener las placas localizadas en un punto concreto. Para los otros problemas existía una dilatación de 0,125 grados entre los puntos de la malla, insuficiente para englobar la isla en su totalidad, por lo que se ha decidido duplicar la dilatación y por lo tanto obtener una separación entre puntos de 0,25 grados.

El objetivo del modelo es predecir los siguientes 12 horizontes, donde horizonte equivale al tiempo que transcurre entre un dato y otro (una hora), a partir de los 24 o 48 horizontes anteriores. Es decir, el primer horizonte se encuentra a 1 hora vista y el último a 12 horas vistas. Por lo tanto, existen seis modelos para cada tipo de modelo descrito con anterioridad. A la hora de calcular el error no se tendrán en cuenta las horas de noche, ya que estas tienen una producción de 0 siempre.

Los resultados obtenidos se dividen en tres tablas distintas con sus respectivas figuras, separado en los tres ejemplos mencionados en esta misma sección. En la tabla 4.7 se pueden consultar los errores de los modelos sólo con la serie temporal de producción. La figura 4.14 muestra como todos los modelos parten en torno al 2,5 % de NMAE excepto los modelos *Ridge* que aumentan hasta el 4 %, y poco a poco se bifurcan hasta generar un intervalo de aproximadamente cinco puntos de diferencia.

Modelo	1	2	3	4	5	6	7	8	9	10	11	12	Promedio
Ridge_24	4,05	7,25	9,08	9,95	10,30	10,38	10,33	10,28	10,28	10,28	10,28	10,28	9,39
Ridge_48	3,75	6,55	8,18	9,00	9,38	9,45	9,38	9,30	9,28	9,28	9,28	9,28	8,51
MLP_24	2,51	4,21	5,19	5,56	5,69	5,73	5,72	5,70	5,76	5,77	5,79	5,80	5,29
MLP_48	2,42	3,89	4,73	5,13	5,30	5,34	5,31	5,28	5,28	5,32	5,37	5,43	4,90
RNN_24	2,61	4,42	5,50	5,99	6,22	6,28	6,27	6,21	6,20	6,20	6,19	6,20	5,69
RNN_48	2,24	3,68	4,51	4,86	5,00	5,03	5,01	4,99	5,02	5,04	5,07	5,10	4,63
GRU_24	2,48	3,70	4,43	4,50	4,50	4,53	4,55	4,55	4,70	4,73	4,78	4,80	4,35
GRU_48	2,55	3,58	4,13	4,35	4,40	4,43	4,43	4,43	4,45	4,55	4,68	4,85	4,23
LSTM_24	2,49	3,96	4,81	5,03	5,10	5,13	5,14	5,13	5,23	5,25	5,28	5,30	4,82
LSTM_48	2,49	3,73	4,43	4,74	4,85	4,88	4,87	4,85	4,86	4,93	5,02	5,14	4,57

Tabla 4.7: Errores de predicción fotovoltaica en Mallorca.

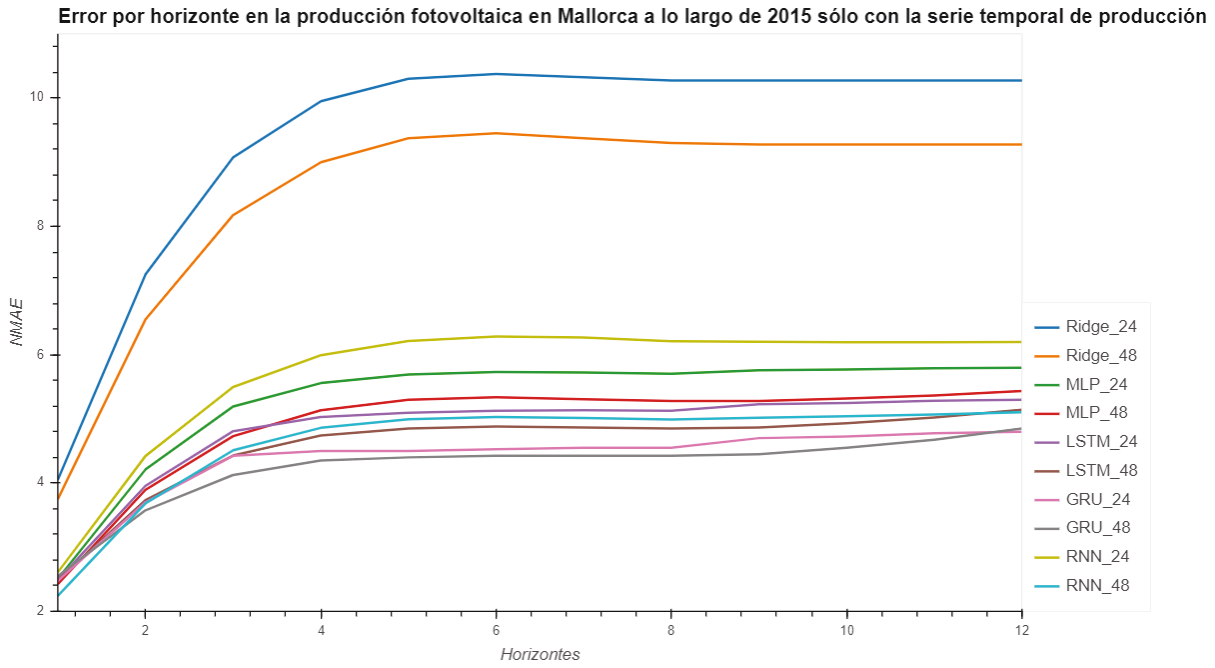

Figura 4.14: Error por horizonte en la producción fotovoltaica en Mallorca a lo largo de 2015 sólo con la serie temporal de producción.

El **mejor resultado** es el obtenido por el **modelo *GRU* con 48 horizontes** con un **NMAE promedio del 4,23 %**. La arquitectura del modelo ganador cuenta con dos capas de 32 unidades cada una y sus hiperparámetros elegidos han sido 0,1 para el *kernel regularizer*, un *batch size* de 256, la función de activación ha sido la *relu* y el número de épocas igual a 1000. Le siguen muy de cerca la propia *GRU* con 24 horizontes, la *LSTM* con 48 horizontes y la *SimpleRNN* con 48 horizontes.

En la tabla 4.8 se pueden consultar los errores de los modelos con la serie temporal de producción y la coordenada centrada en la isla. La figura 4.15 muestra como los modelos recurrentes y la *MLP* parten en torno al 2,4 % de NMAE, y poco a poco se bifurcan hasta generar un intervalo de prácticamente cinco puntos de diferencia.

El **mejor resultado** es el obtenido por el **modelo *GRU* con 48 horizontes** con

Modelo	1	2	3	4	5	6	7	8	9	10	11	12	Promedio
Ridge_1c_24	4,00	6,93	8,55	9,35	9,70	9,78	9,73	9,63	9,58	9,55	9,55	9,55	8,82
Ridge_1c_48	3,75	6,48	8,05	8,90	9,30	9,40	9,33	9,20	9,15	9,10	9,10	9,10	8,40
MLP_1c_24	2,49	3,91	4,70	5,07	5,23	5,26	5,25	5,22	5,23	5,25	5,33	5,43	4,86
MLP_1c_48	2,36	3,61	4,33	4,70	4,88	4,92	4,89	4,88	4,90	4,89	5,02	5,14	4,54
RNN_1c_24	3,25	5,57	6,87	7,45	7,70	7,75	7,72	7,66	7,67	7,66	7,67	7,67	7,05
RNN_1c_48	3,09	5,18	6,39	7,02	7,30	7,37	7,32	7,24	7,21	7,21	7,23	7,27	6,65
GRU_1c_24	2,46	3,24	3,66	3,84	3,89	3,91	3,91	3,95	4,03	4,09	4,31	4,56	3,82
GRU_1c_48	2,38	2,90	3,20	3,33	3,38	3,40	3,40	3,48	3,60	3,63	3,95	4,28	3,41
LSTM_1c_24	2,47	3,57	4,18	4,45	4,56	4,59	4,58	4,59	4,63	4,67	4,82	5,00	4,34
LSTM_1c_48	2,37	3,25	3,76	4,01	4,13	4,16	4,15	4,18	4,25	4,26	4,48	4,71	3,98

Tabla 4.8: Errores de predicción fotovoltaica en Mallorca con una coordenada.

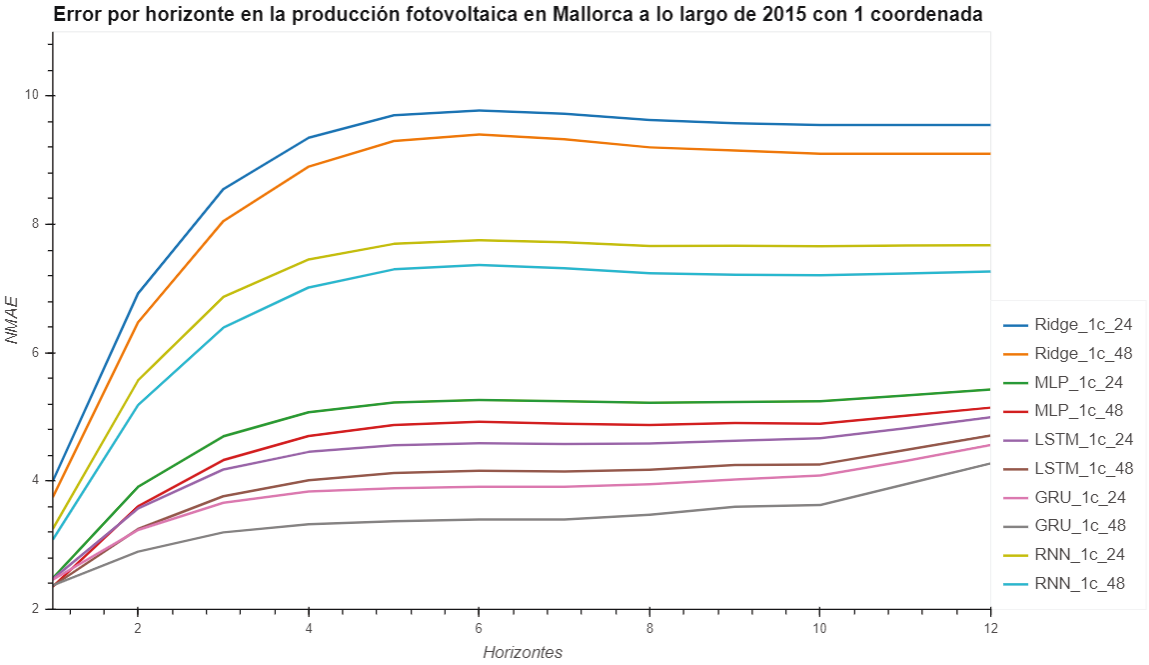

Figura 4.15: Error por horizonte en la producción fotovoltaica en Mallorca a lo largo de 2015 con 1 coordenada.

un **NMAE promedio del 3,41 %**. La arquitectura del modelo ganador cuenta con dos capas de 32 unidades cada una y sus hiperparámetros elegidos han sido 0,01 para el *kernel regularizer*, un *batch size* de 256, la función de activación ha sido la *relu* y el número de épocas igual a 1500. Le siguen muy de cerca la propia *GRU* con 24 horizontes y la *LSTM* con 48 horizontes.

Se observa que la mayoría de modelos mejoran su predicción salvo por el modelo *Ridge*, el cual se estanca y no varía su error. Al igual que sucedía en el experimento de Sotavento, la *SimpleRNN* empeora su resultado al aumentar el número de variables predictivas.

En la tabla 4.9 se pueden consultar los errores de los modelos con la serie temporal de producción y las nueve coordenadas anteriormente mencionadas. La figura 4.16 muestra como los modelos recurrentes y la *MLP* parten de un mismo punto, en torno al 2,3 % de

Modelo	1	2	3	4	5	6	7	8	9	10	11	12	Promedio
Ridge_9c_24	3,95	6,83	8,45	9,28	9,65	9,75	9,70	9,60	9,53	9,50	9,48	9,48	8,76
Ridge_9c_48	3,75	6,40	8,00	8,83	9,23	9,35	9,33	9,20	9,13	9,10	9,08	9,08	8,37
MLP_9c_24	2,42	3,68	4,39	4,75	4,91	4,95	4,94	4,93	4,96	4,95	5,09	5,23	4,60
MLP_9c_48	2,29	3,32	3,95	4,28	4,44	4,50	4,50	4,50	4,56	4,53	4,72	4,90	4,21
RNN_9c_24	3,22	5,37	6,57	7,17	7,44	7,51	7,47	7,41	7,38	7,37	7,40	7,45	6,81
RNN_9c_48	3,05	5,00	6,16	6,76	7,05	7,14	7,11	7,04	7,01	7,00	7,05	7,11	6,46
GRU_9c_24	2,33	2,73	2,97	3,07	3,11	3,13	3,14	3,23	3,38	3,38	3,76	4,12	3,19
GRU_9c_48	2,20	2,23	2,28	2,30	2,33	2,35	2,38	2,50	2,73	2,68	3,20	3,68	2,57
LSTM_9c_24	2,37	3,20	3,68	3,91	4,01	4,04	4,04	4,08	4,17	4,17	4,42	4,67	3,90
LSTM_9c_48	2,24	2,77	3,11	3,29	3,38	3,43	3,44	3,50	3,64	3,60	3,96	4,29	3,39

Tabla 4.9: Errores de predicción fotovoltaica en Mallorca con nueve coordenadas.

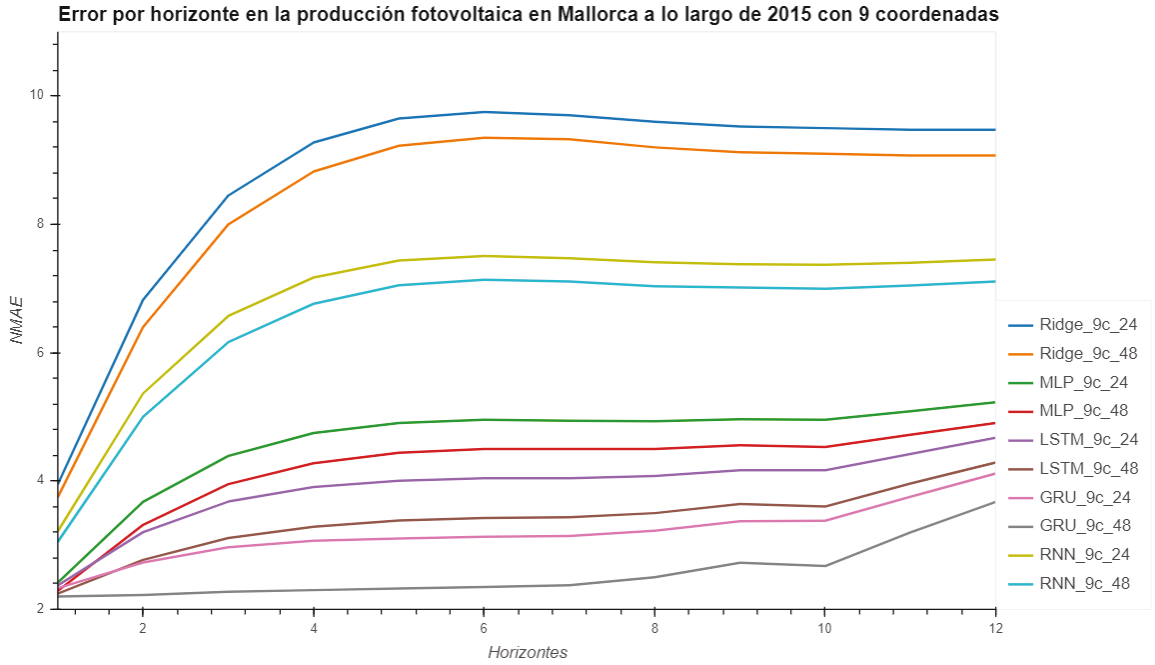

Figura 4.16: Error por horizonte en la producción fotovoltaica en Mallorca a lo largo de 2015 con 9 coordenadas.

NMAE, y poco a poco se bifurcan hasta generar un intervalo de aproximadamente seis puntos de diferencia.

El **mejor resultado** es el obtenido por el **modelo *GRU* con 48 horizontes** con un **NMAE promedio del 2,57 %**, y por tanto el mejor modelo para el problema fotovoltaico en Mallorca. La arquitectura del modelo ganador cuenta con dos capas de 64 unidades cada una y sus hiperparámetros elegidos han sido 0,01 para el *kernel regularizer*, un *batch size* de 256, la función de activación ha sido la *relu* y el número de épocas igual a 3000. Este modelo se muestra con una gran continuidad hasta el décimo horizonte donde despega claramente el error. Le siguen muy de cerca la propia *GRU* con 24 horizontes y la *LSTM* con 48 horizontes.

Se observa que la mayoría de modelos continúan mejorando su predicción salvo por

las dos excepciones de la agrupación anterior; el modelo *Ridge* y la *SimpleRNN*, el cual claramente se ve poco favorecido al aumentar el número de variables predictivas.

Al igual que en el problema de Sotavento, todos los modelos siguen una progresión conforme se va aumentando el número de variables predictivas. En cierta manera se espera que el comportamiento de la *SimpleRNN* sea semejante a este problema. Viendo los resultados de las tablas 4.7, 4.8 y 4.9 se confirma. Esto de nuevo puede deberse a que el modelo no está preparado para una cantidad tan grande de variables (mucho mayor que en el problema de Sotavento) y, de algún modo, el modelo no logra interpretar correctamente todas las variables predictivas aportando ruido a la salida.

5

Conclusiones y Trabajo futuro

5.1. Conclusiones

El objetivo principal de este TFM era comprobar el funcionamiento de las redes recurrentes en problemas de series temporales en el ámbito de las energías renovables. A continuación se enumeran los distintos experimentos desarrollados sus resultados y pertinentes conclusiones extraídas.

- **Predicción eólica en la España peninsular**. El mejor resultado ha sido proporcionado por el modelo *SimpleRNN* con 48 horizontes con un NMAE promedio del 1 %. No obstante, no existe una diferencia significativa con el resto de modelos donde el peor de ellos tan sólo difiere un 0,05 % de error con respecto al vencedor. Por lo tanto, la conclusión es que, independientemente del modelo utilizado, el resultado no se ve afectado al menos para la cantidad de horizontes expuesta; probablemente debido a la frecuencia diez-minutal y el comportamiento de la eólica peninsular.
- **Predicción eólica en Sotavento**. El mejor resultado es el obtenido por el modelo *GRU* con 48 horizontes con un NMAE promedio del 3,49 %. Se observa que la mayoría de modelos mejoran sus predicciones con el aumento de variables salvo por dos excepciones: el modelo *Ridge* que mantiene prácticamente el mismo error y la *SimpleRNN*. Este último presenta un claro problema probablemente debido a que el modelo no logra interpretar correctamente los datos provocando ruido en la salida.
- **Predicción fotovoltaica en la España peninsular**. El mejor resultado es el obtenido por el modelo *SimpleRNN* con 24 horizontes con un NMAE promedio del 1,15 %, al igual que ocurría en el experimento eólico. Mientras que para la predicción eólica la elección del modelo era prácticamente irrelevante para el resultado final,

en este problema se observa que el modelo *Ridge* se despega muy pronto del resto de modelos, provocando que este sea el peor modelo con diferencia. El *MLP* obtiene un resultado muy parejo al de la *LSTM* y al de la *GRU* con 24 horizontes, mientras que la *GRU* de 48 horizontes muestra un error inusual, probablemente debido a un error en el entrenamiento.

- **Predicción fotovoltaica en Mallorca**. El mejor resultado ha sido proporcionado por el modelo *GRU* con 48 horizontes con un NMAE promedio del 2,57 %. Viendo estos resultados y los del resto de experimentos se podría decir que en definitiva la *SimpleRNN* funciona bien para problemas con pocas variables.

Como conclusión final tras analizar los resultados obtenidos en cada uno de los experimentos, se puede afirmar que las RNN han supuesto una mejora frente a los modelos regresivos comunes. La *SimpleRNN* obtiene grandes resultados para problemas más sencillos mientras que *LSTM* y *GRU* (sobre todo esta última) presentan un magnífico comportamiento a medida que el problema va adquiriendo complejidad.

5.2. Trabajo futuro

A continuación se enumeran una serie de posibles mejoras y trabajos futuros:

- Bien es cierto que las variables seleccionada para los modelos han sido de gran ayuda, sobre todo para los modelos recurrentes, aun así, ampliar el número de variables en ambos problemas podría incluso mejorar el resultado.
- En este TFM se ha explorado relativamente poco la precisión de las predicciones a largo plazo. Una buena forma de comprobarlo sería aumentar el número de horizontes de predicción.
- Las RNN presentan una complejidad muy elevada. Es por esto que cuentan con una gran variedad de parámetros que se pueden configurar de cara a un entrenamiento mucho más preciso, por lo que se podría continuar hiperparametrizando aún más estas redes con argumentos como el *dropout*.
- En vista de que las redes recurrentes funcionan bien para este tipo de problemas, los modelos de aprendizaje profundo puede que también aporten cierta calidad al dato. Uno de los posibles modelos a probar podría ser un transformador o *Transformer*, muy usado en el mundo del procesamiento del lenguaje natural (PLN) o el *computer vision* (CV), pero que también puede funcionar muy bien en los problemas aquí planteados.

Bibliografía

[1] Ecmwf home page. [Online]. Available: https://www.ecmwf.int/

[2] P. Pinson, "Wind energy: Forecasting challenges for its operational management," *Statistical Science*, vol. 28, no. 4, 2013.

[3] J. Kleissl, *Solar energy forecasting and resource assessment*. Academic Press, 2013.

[4] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learnin," *Cited on*, p. 33, 2009.

[5] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 12, no. 1, pp. 55–67, 1970.

[6] M. W. Gardner and S. Dorling, "Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences," *Atmospheric environment*, vol. 32, no. 14-15, pp. 2627–2636, 1998.

[7] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[8] N. Van Dijk, M. Langelaar, and F. Van Keulen, "Explicit level-set-based topology optimization using an exact heaviside function and consistent sensitivity analysis," *International Journal for Numerical Methods in Engineering*, vol. 91, no. 1, pp. 67–97, 2012.

[9] S. Sharma and S. Sharma, "Activation functions in neural networks," *Towards Data Science*, vol. 6, no. 12, pp. 310–316, 2017.

[10] J. Ilonen, J.-K. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," *Neural Processing Letters*, vol. 17, no. 1, pp. 93–105, 2003.

[11] R. Caruana, S. Lawrence, and L. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," *Advances in neural information processing systems*, pp. 402–408, 2001.

[12] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *Advances in neural information processing systems*, 2016, pp. 3981–3989.

[13] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[14] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[15] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.

[16] Simplernn layer from keras. [Online]. Available: https://keras.io/api/layers/recurrent_layers/simple_rnn/

[17] Y.-c. Cheng, W.-M. Qi, and W.-y. Cai, "Dynamic properties of elman and modified elman neural network," in *Proceedings. International Conference on Machine Learning and Cybernetics*, vol. 2. IEEE, 2002, pp. 637–640.

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[19] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] C. M. Bishop, "Novelty detection and neural network validation," *IEE Proceedings-Vision, Image and Signal processing*, vol. 141, no. 4, pp. 217–222, 1994.

Apéndices

A

Algoritmos y funciones de utilidad

A.1. Algoritmos

Código A.1: Algoritmo de creación de tensores X e Y para el entrenamiento de modelos de aprendizaje profundo

```
def create_X_Y(ts: np.array, lag=1, n_ahead=1, target_index=-1)
    -> tuple:
        """
        Metodo para crear matrices X e Y a partir de una matriz
    de series temporales para el entrenamiento
    de modelos de aprendizaje profundo
        """
        # Extraer el numero de caracteristicas que se pasan de
            la matriz
        n_features = ts.shape[1]

        X, Y = [], []

        if len(ts) - lag <= 0:
                X.append(ts)
        else:
                for i in range(len(ts) - lag - n_ahead):
                        Y.append(ts[(i + lag):(i + lag + n_ahead
                            ), target_index])
                        X.append(ts[i:(i + lag)])
```

```
        X, Y = np.array(X), np.array(Y)

        # Reshaping de la matriz X a la forma de entrada de la
            ↪ RNN
        X = np.reshape(X, (X.shape[0], lag, n_features))

        return X, Y
```

A.2. Funciones de utilidad

Código A.2: Fichero *request* para la obtención de meteorología a través de la API del ECMWF

```
    retrieve,
    class=od,
    date=2013-01-01/to/2015-12-31,
    expver=1,
    levtype=sfc,
    param=169.128/164.128,
    step = 0/to/24/by/1,
    stream=oper,
    time=00,
    type=fc,
    grid = 0.125/0.125, # resolucion espacial latitudes/
       ↪ longitudes
    area = 40/2/39/4,
    target="Mallorca_Fotovoltaica_2013_2015.grib"
```