APLICACIONES DE DEEP-LEARNING A LA PREDICCIÓN DE RADIACIÓN SOLAR

AHSIN RASHID

GRADO EN INGENIERÍA DE COMPLTADORES, FACULTAD DE INFORMÁTICA,

UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Grado en Ingeniería de computadores

31 de mayo de 2019

JOSÉ IGNACIO GÓMEZ PÉREZ

CHRISTIAN TENLLADO VAN DER REIJDEN

Autorización de Difusión

AHSIN RASHID

31 mayo 2019

El abajo firmante, matriculado en el grado de Ingeniería de Computadores de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Grado: "Aplicaciones De Deep-Learning A La Predicción De Radiación Solar", realizado durante el curso académico 2018-2019 bajo la dirección de José Ignacio Gómez Pérez y Christian Tenllado van der Reijden en el Departamento de Arquitectura de Computadores y Automática, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen En Castellano

El establecimiento de un sistema energético sostenible es uno de los grandes desafíos a los que debe enfrentarse la humanidad en el siglo XXI, y la energía solar presenta la principal fuente de energía renovable en la naturaleza. Este trabajo trata de predecir, a corto plazo, la radiación solar en determinadas áreas de la superficie, lo cual cobra vital importancia en la optimización de las plantas eléctricas y favorece la integración de la energía solar en los actuales sistemas de abastecimiento eléctrico.

Este estudio emplea la tecnología de redes neuronales *fully-connected* y Convolucionales, usando lenguaje de programación *Python* con librerías de *Keras*, *Matplotlib*, *Numpy*, *Pandas* y *Pykriging*, para la creación de dos modelos de *Machine Learning* que tratan de predecir la radiación solar en una ubicación concreta o varias simultáneamente en una determinada superficie terrestre donde se hayan desplegado sensores. Por otra parte se han creado modelos de *Machine Learning* para predecir la radiación solar, en un rango que puede llegar a varios minutos, en cualquier ubicación entre los sensores desplegados usado la técnica de *kriging*.

Palabras Clave

Radiación solar, *Machine Learning*, *Deep Learning*, *kriging*, *Python*, Redes Neuronales.

Resumen En Inglés

The establishment of a sustainable energy system is one of the greatest challenges that humanity must face in the 21st century, and solar energy presents the main source of renewable energy in nature. This work tries to forcast, on the short horizons, solar radiation in certain areas of the surface, which is of vital importance in the optimization of power plants and supports the integration of solar energy in power supply systems.

This study uses the technology of Fully-Connected and Convolutional neural networks, employing *Python* programming language with libraries as *Keras*, *Matplotlib*, *Numpy*, *Pandas* and *Pykriging*, for the creation of two models of Machine Learning that try to predict irradiance in a specific location or several, simultaneously, in a certain area where sensors have been deployed. On the other hand, Machine Learning models have been created to forecast irradiance, in a range that can reach several minutes, at any location among the sensors deployed using the *Kriging* technique.

Keywords

Short-term solar radiation forecasting, Machine Learning, Deep Learning, kriging, Python, Neural Networks.

ÍNDICE

CAPÍTULO 1- INTRODUCCIÓN

1.1 Introducción

Hoy por hoy afrontamos profundas secuelas ambientales derivadas del uso de combustibles fósiles: las emisiones descontroladas de dióxido de carbono son detonantes del cambio climático que causa sequías, inundaciones y otros desastres climáticos en muchos lugares del planeta[1]. Al mismo tiempo la población mundial no para de aumentar, y alcanzará los 9.100 millones en el años 2050 (M. Pasquevich, 2016), generando sin duda nuevas demandas energéticas que los combustibles fósiles no serán capaces de abastecer porque son finitos. Parece por tanto evidente que necesitamos diversificar la matriz energética para incluir y/o aumentar el uso de fuentes renovables y limpias que combinen con los existentes modelos energéticos.

Las energías renovables son recursos limpios y casi inagotables que contribuyen a disminuir la dependencia de combustibles fósiles como el carbón, el petróleo, o el gas natural. Existen varios tipos de energías renovables, como lo son la eólica, hidráulica, geotérmica, energía marina y energía solar fotovoltaica y térmica. En este estudio nos centramos en la energía solar, que presenta la mayor cantidad absoluta de potencial energético; teniendo en cuanta que la energía solar que llega la superficie terrestre es más de tres órdenes de magnitud superior a la demanda energética mundial actual (Martín, 2012, p 27).

Sin embargo, la explotación de la energía solar entraña, por sus características, dificultades que residen principalmente en no saber cuándo se va a producir. Es una energía que depende de factores como la velocidad y dirección del viento, nubes, lluvia, humedad relativa. De aquí que predecir la radiación solar cobra vital importancia para que el uso de esta energía sea eficiente.

Estimar la radiación solar que incide en una determinada área de la superficie terrestre es de gran importancia, y la predicción de datos horarios de radiación cobra una importancia vital para el uso eficiente de la energía solar, el dimensionado de sistemas de energía solar y estimaciones meteorológica.

[1] https://www.nationalgeographic.com.es/ciencia/grandes-reportajes/energia-renovable-para-abastecer-a-todo-planeta_11706/1

También es esencial para el mantenimiento y estabilidad de las plantas eléctricas y para las distribuidoras del sistema eléctrico que integran la energía solar en sus sistemas y necesitan estimar la generación eléctrica a corto plazo para optimizar su gestión, amortiguando situaciones de reducción de carga y anticipando problemas en el abastecimiento. De este modo, saber cuándo habrá picos o bajadas en la producción solar a corto plazo permite aumentar o disminuir la producción de otro tipo de fuentes de energía para poder cubrir la demanda energética así como tomar decisiones sobre la red de distribución. En pocas palabra, la predicción es un factor crucial para integrar la energía solar en el sistema de distribución (Sayago, Bocco, Ovando, & Willington, 2011).

Partiendo de esa premisa, hay una gran cantidad de trabajo previo en el que se aplican diferentes metodologías para estimar la radiación solar diaria: modelos meteorológicos numéricos[2], imágenes satélite, imágenes de cielo para seguimiento de nubes… En este trabajo afrontas la predicción a partir de medidas radiométricas en la superficie terrestre, que son tratadas con modelos estadísticos para inferir la radiación que será recibida en el futuro próximo.

Empleamos redes neuronales artificiales (RNA), un campo muy importante dentro del ámbito Inteligencia Artificia, que mediante técnicas algorítmicas convencionales crea modelos artificiales capaces de solucionar problemas difíciles.

Ahora bien, para poder desarrollar los algoritmos y hacer efectiva la estructura neuronal se necesita de una ingente cantidad de datos que serán nuestra herramienta para entrenar los modelos neurales. Para ello usaremos datos proporcionado por el Laboratorio Nacional de Energías Renovables de Estados Unidos[3] entre el 18 de marzo de 2010 y el 31 de octubre de 2011 y con mediciones cada segundo, lo cual nos permitirá entrenar nuestras redes neuronales y comprobar la precisión de sus estimaciones.

El presente documento se estructura en cinco capítulos. El primer capítulo, el presente, es una breve introducción sobre este trabajo y los motivos del mismo. El segundo capítulo recorre brevemente las tecnologías que hemos usado y las definiciones de los conceptos básico en ámbito de *Deep Learning* relacionados con el trabajo, además de las librerías de lenguajes de programación que hemos usado. En

[2] https://instrumentosmeteorologicos.com/divulgacion/modelos-numericos-meteorologicos/

[3] http://midcdmz.nrel.gov/oahu_archive/

el tercer capítulo explicamos la elección y tratamiento de datos e entrenamiento, así como la obtención y preparación de datos y la topología e entrenamiento de la red. En el cuarto capítulo exponemos los resultados obtenidos de dicho entrenamiento así como la comparación entre los distintos modelos creados durante el entrenamiento. Finalmente, el quinto capítulo recoge las conclusiones del trabajo así como el modo de mejorarlos en el futuro.

Todos los códigos creados durante este estudio están publicados en Github. Se pueden consultar en el siguiente enlace: https://gitlabce.dacya.ucm.es/nachog/solarcasting-ml/tree/master

1.2 Introduction

Nowadays we are facing serious environmental consequences derived from fossil fuels use: the uncontrolled emissions of carbon dioxide are detonators of climate change that causes droughts, floods and other climatic disasters in many parts of the planet. At the same time, the world population continues to grow, reaching 9,100 million in the 2050s (M. Pasquevich, 2016) and will surely generates new energy demands that fossil fuels will not be able to supply because they are limited. The need for energy diversification is therefore an evidence, and so is including and increasing the use of renewable and clean energy that combines with existing energy models.

Renewable energies are clean and almost unlimited resources that help reduce dependence on fossil fuels such as coal, oil, or natural gas. There are several types of renewable energy, such as the wind energy, hydraulic energy, geothermal energy, marine energy and solar energy. In this study, we focus on solar energy.

However, due to its characteristics, the use of solar energy involves a series of difficulties that mainly reside in not knowing when it will be produced. It depends on factors such as the speed and direction of the wind, clouds, rain, and relative humidity. Hence, predicting irradiance becomes crucial for the efficient use of this energy.

Estimating the irradiance that falls upon a certain area of the earth's surface is of great relevance, and the prediction of hourly irradiance data becomes of a vital importance in many applications such as the sizing of solar energy systems and meteorological estimates (Hocaoğlu et al., 2007). It is also essential for the maintenance and stability of power plants and electric system distributors that integrate solar energy into their system and need to estimate short-term power production to optimize their management, and anticipate supply problems. In this way, knowing when there will be peaks or drops in solar production in the short term allows to increase or decrease the production of other types of energy sources to meet the energy demand as well as making decisions about distribution network. That is, prediction is a crucial factor for integrating solar energy into energy system (Sayago, Bocco, Ovando, & Willington, 2011).

Built on that premise, there is a large amount of previous work in which different methodologies are applied to estimate the daily irradiance: numerical meteorological models, satellite images, sky images for tracking clouds, *etc.* This work deals with the prediction from radiometric measurements on the earth's surface, which are treated with statistical models to infer the radiation that will be received in the near future.

We use artificial neural networks; an important field within Artificial Intelligence, which by conventional algorithmic techniques create artificial models capable of solving difficult problems.

Now, in order to develop the algorithms and make the neuronal structure effective, a large amount of data is needed, which will be our tool for training neural models. We will use data provided by the National Renewable Energy Laboratory (NREL) [4], between March 18, 2010 and October 31, 2011, with measurements every second, which will allow us to train our neural networks and verify the accuracy of their estimates.

This study is structured in five chapters. The first chapter, the present, is a brief introduction about this work and motivations. The second chapter briefly covers the technologies we have used and the definitions of basic concepts in the field of Deep Learning related to the work, in addition to the libraries of programming languages that we have used. In the third chapter, we describe the selection, treatment of data and training, as well as collection and preparation of data, the topology and training of the network. In the fourth chapter, we present the results obtained from this training as well as the comparison between the different models created during the training. Finally, the fifth chapter gathers the conclusions of the work as well as the ways to improve them in the future.

All codes created in this study are available in Github. They can be accessed at the link: https://gitlabce.dacya.ucm.es/nachog/solarcasting-ml/tree/master

[4] http://midcdmz.nrel.gov/oahu_archive/

CAPÍTULO 2 - TECNOLOGÍAS UTILIZADAS

2.1 Machine Learning

Machine Learning (ML), conocido en español como Aprendizaje Automático, es una disciplina científica en el campo de la Inteligencia Artificial (IA). Básicamente, es una rama en desarrollo de los algoritmos computacionales diseñados para simular la inteligencia humana al aprender del entorno circundante. Las técnicas basadas en *Machine Learning* se han aplicada en diferente ámbitos que van desde la ingeniería de naves espaciales, las finanzas hasta las aplicaciones médicas (El Naqa & Murphy, 2015).

El *Machine Learning*, nació como una idea de la Inteligencias Artificial en la década de los 60 y puede ser definido como:

"*Machine Learning* es la ciencia que permite que las computadoras aprendan y actúen como lo hacen los humanos, mejorando su aprendizaje a lo largo del tiempo de una forma autónoma, alimentándolas con datos e información en forma de observaciones e interacciones con el mundo real."-Dan Fagella[5].

2.1.1 ***Tipos de Aprendizaje***

Existen diferentes tipos principales de *Machine Learning*: Aprendizaje supervisado y aprendizaje no supervisado.

Aprendizaje Supervisado:

En el aprendizaje supervisado el entrenamiento utiliza una serie de patrones de entrada para los que se conoce la salida deseada. Se ajusta el modelo de estas salidas cuando se presenten como entrada estos patrones. Un ejemplo sería introducir fotos de perros y gatos con etiquetas que los definen como

[5] https://medium.com/datos-y-ciencia/introduccion-al-machine-learning-una-gu%C3%ADa-desde-cero-b696a2ead359

tal. Una vez el modelo es entrenado adecuadamente con suficiente cantidad de datos de entrada y los resultados de la batería de datos de entrenamiento, podrá clasificar correctamente como perro o gato una imagen no vista durante la fase de entrenamiento.

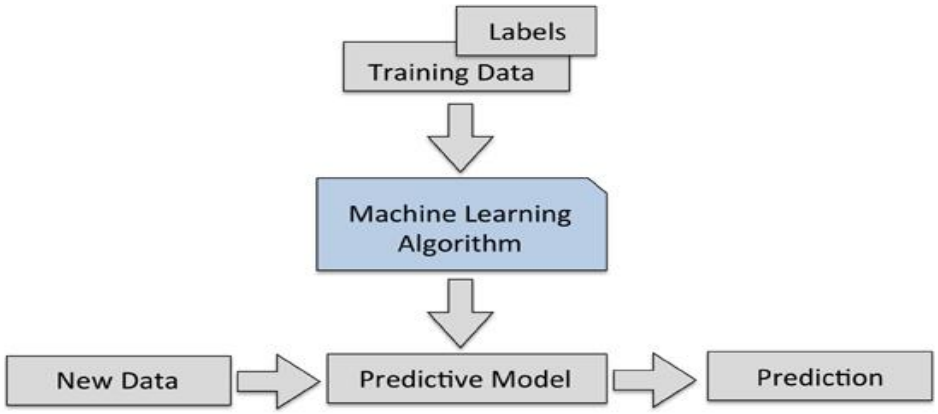


Figura 1- Proceso del *Machine Learning*[6]

Hay dos tipos principales de aprendizaje supervisado: Clasificación y regresión. El objetivo de la clasificación es predecir las clases categóricas. En otras palabras, la clasificación se usa cuando la variable de salida es una categoría, como "amarillo" o "negro" o "perros" y "caballos", es decir cuando la respuesta a una pregunta cae dentro de un conjunto finito y concreto de posibles resultados.

Existen dos tipos de clasificación: La binaria, que se da cuando la respuesta tiene dos opciones (sí o no); y la multicategoría (multi-class), que se da cuando las respuestas son más de dos categorías.

[6] https://medium.com/datos-y-ciencia/introduccion-al-machine-learning-una-gu%C3%ADa-desde-cero-b696a2ead359

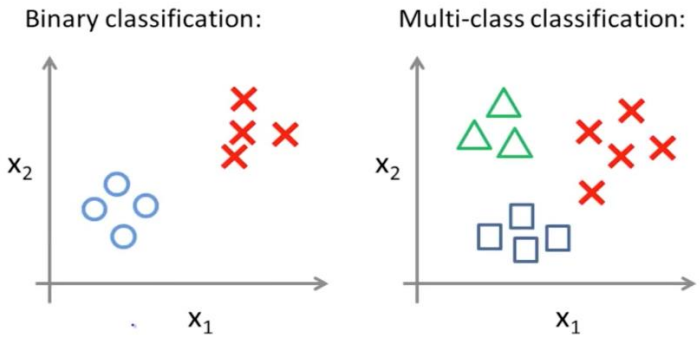


Figura 2- Ejemplo de clasificacion binaria y multigategorica[7]

Se llama en cambio regresión a un aprendizaje supervisado en el que la respuesta del sistema es un valor continuo, con infinitos posibles valores. Existen varios tipos de regresión, el más usado es la regresión lineal, en el que se intenta buscar una relación entre datos de entrada y salida.

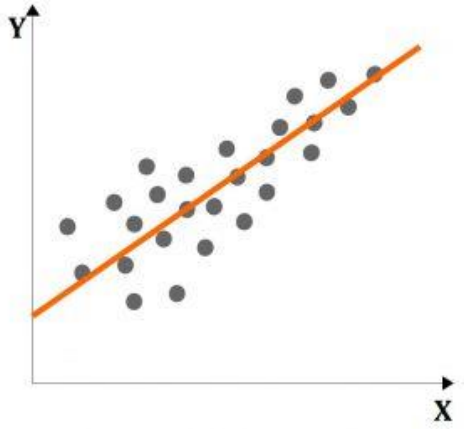


Figura 3- Ejemplo regresión[8]

La figura 3 es un ejemplo de regresión lineal, dados X e Y. El modelo trata de reducir el error medio entre el valor real que corresponde a un patrón de entrada y el valor asignado por el propio modelo.

[7] https://www.ritchieng.com/logistic-regression/

[8] https://blog.gfi.es/algoritmos-entrenamiento-machine-learning/

Los algoritmos más famosos de del aprendizaje supervisado son[9]:

1- Árboles de decisión

2- *Naïve Bayes* Clasification

3- *Ordinary Least Squares Regression*

4- *Support Vector Machines*

5- Métodos Ensemble

6- Redes Neuronales

Aprendizaje No Supervisado

Por otro lado en el aprendizaje no supervisado no se presenta al modelo la salida deseada para los patrones de entrenamiento, el modelo busca por sí sólo relaciones entre los datos de entrada sin ser forzado a dar una salida específica para cada uno de ellos. Es un método de entrenamiento más parecido al modo en que los humanos procesan la información[10], y tiene dos categorías principales: Agrupamiento (*Clustering*) y reducción dimensional.

Clustering es como una clasificación, en la que no están definidas a priori las clases, sólo su número. El sistema extrae las clases, que sería el identificador de cada *cluster*.

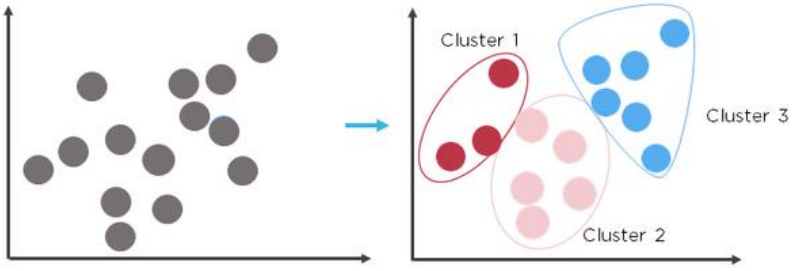


Figura 4- Ejemplo de clustring[11]

[9] https://www.raona.com/los-10-algoritmos-esenciales-machine-learning/

[10] https://www.apd.es/conoce-apd/servicios-para-asociados/

[11] https://www.quora.com/What-is-the-difference-between-k-means-and-hierarchical-clustering

Los algoritmos más famosos de Aprendizaje no supervisado son[12]:

1- Algoritmos *Clustering*

2- Análisis de Componentes Principales

3- *Singular Value Decomposition*: SVD

4- Análisis de Componentes Independientes: ICA

Por otra parte la reducción dimensional es el proceso de reducir el número de dimensiones de los variables. Se trata de eliminar variables irrelevantes o redundantes teniendo cuidado con no suprimir variable importantes.

Un ejemplo es intentar predecir el salario de una persona en una empresa dada, se necesita saber variables como el sexo, antigüedad y puesto en la impresa. En este ejemplo, variable como el color de pelo, llevar gafas, etc... Son variables irrelevantes que se deberán eliminar, para no contaminar las muestras, antes de entrenar cualquier modelo de aprendizaje, con el método de reducción dimensional.

2.2 Redes Neuronales Artificiales

Hay varios tipos de redes neuronales artificiales. En este trabajo mencionamos las redes *Fully-Connected* y redes neuronales Convolucionales. Pero primero vamos a explicar lo que sería la unidad base de las redes neuronales: las neuronas.

2.2.1 ¿Qué Es Una Neurona?

En el campo de *Machine Learning*, una neurona es la unidad básica de las redes neuronales artificiales (RNA), su nombre viene de la similitud de las estructuras de las redes neuronales artificiales con las redes neuronales de un cerebro humano.

[12] https://www.raona.com/los-10-algoritmos-esenciales-machine-learning/

La función de una neurona en una red neuronal es una función matemática que trata de proporcionar una salida. El proceso consiente de varios pasos: una neurona recibe información, la multiplica con un peso y la pasa a través de una función no lineal, llamada función de activación, para sacar una salida. En caso de varias entradas, la neurona los multiplica cada una con su peso y luego pasa la suma de todos a través de la función de activación para obtener una salida. Siendo las más habituales la sigmoidea, la tangente hiperbólica (tanh) y la función lineal rectificada (ReLU), en este trabajo hemos usado la última.

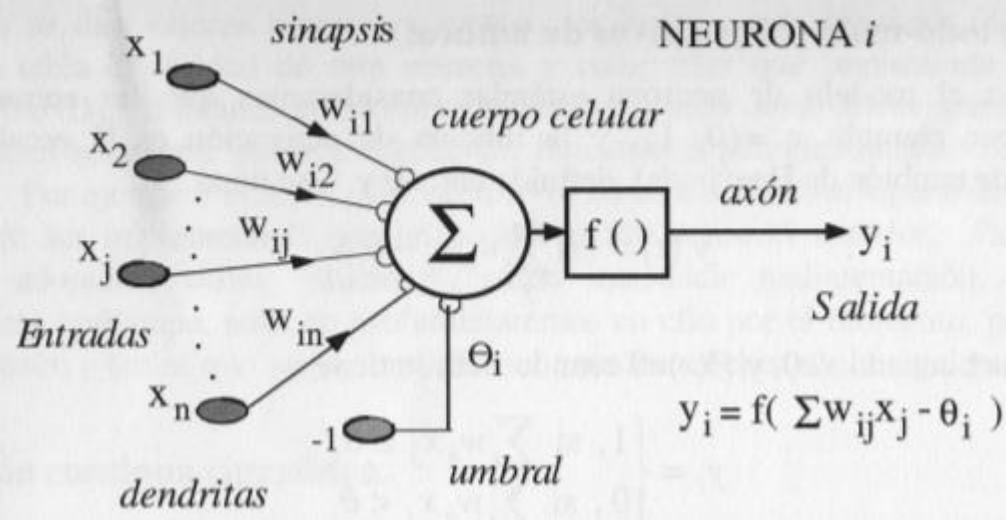


Figura 5- Una neurona artificial (Larranaga, Pedro & Inza, Iñaki & Moujahid, 2019)

2.2.2. Función ReLU

La *Rectified Linear Unit* (ReLU) es una función de activación que básicamente devuelve un valor cero si la entrada es negativa, pero cuando es entrada es positiva, devuelve ese mismo valor.

Matemáticamente se refleja así:

$$f(x) = \max(0, x)$$

Gráficamente se ve así:

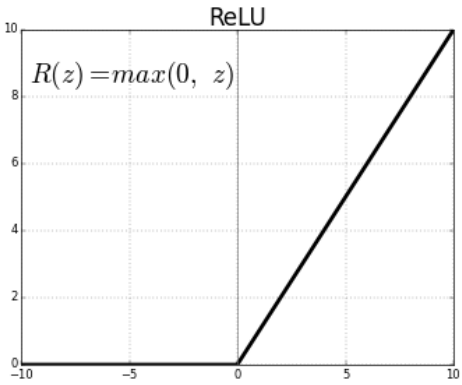


Figura 6- Función ReLU[13]

2.2.3 Redes Fully-Connected

Después de exponer qué es una neurona y su funciones de activación, profundizaremos en lo que es una red neuronal. Pero antes debemos saber lo que es una capa en una red neuronal.

Una capa es un conjunto de neuronas que no están conectadas entre sí y todas las neuronas de una capa deben tener una misma función de activación.

Una red neuronal consiste básicamente de varias capas conectadas entre sí, y tiene mínimamente una capa de entrada y una capa de salida. En una red neuronal cada capa puede tener una función de activación distinta a las demás.

Una red neuronal es *Fully-Connected*, cuando todas las neuronas de una capa están conectadas con todas las neuronas de las capas contiguas. Cuando una red neuronal tiene más de una capa oculta, es decir, situada entre dos capas interconectadas, se denomina generalmente como *Deep Neural Network*.

[13] https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

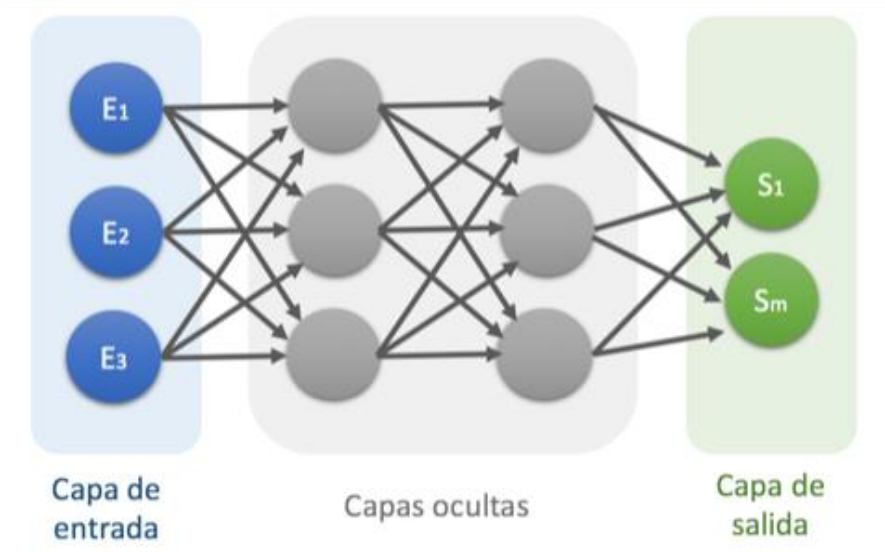


Figura 7- Ejemplo de red *Fully-Connected*[14]

En la figura 7, la capa más a la izquierda de la red es la capa de entrada y la capa más a la derecha es la capa de salida (que, en este ejemplo, tiene dos nodos). Las capas intermedias de los nodos se denominan capas ocultas porque sus valores no se observan en el conjunto de entrenamiento.

Cabe mencionar que para que una red realice su tarea con precisión, cada neurona debe tener cierto peso, que se actualiza con un algoritmo llamado *back propagation*

El ***back propagation*** es el algoritmo de entrenamiento de la red neuronal. Es la práctica de ajustar los pesos de una red neuronal en función de la tasa de error (*Loss*) obtenida en la época anterior (es decir, la iteración). El ajuste correcto de los pesos asegura tasas de error más bajas.

2.2.4 Función de Coste

La función de coste nos da una medida del error cometido por la red sobre los patrones de entrenamiento, para unos valores fijos de sus parámetros. El proceso de entrenamiento busca los parámetros que minimizan dicho error. Una de ellas es el *Mean Squared Error* (MSE), el error cuadrático

[14] http://www.diegocalvo.es/wp-content/uploads/2017/07/perceptron-multicapa.png

medio en español, que mide el promedio del error al cuadrado entre valor real y predicho. Matemáticamente se ve así:

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2}$$

Existen otras métricas para el cálculo de esta función de coste, como por ejemplo[15]:

1. Root Mean Squared Error (RMSE)
2. Mean Absolute Error (MAE)
3. R Squared (R^2)
4. Adjusted R Squared (R^2)
5. Mean Square Percentage Error (MSPE)
6. Mean Squared Error (MSE)
7. Mean Absolute Percentage Error (MAPE)
8. Root Mean Squared Logarithmic Error (RMSLE)

2.2.5 Redes Neuronales Convolucionales

Las redes neuronales Convolucionales (CNN) son una tipo de redes neuronales utilizadas en el tratamiento de datos bidimensionales y tridimensionales. Se compone de dos funciones principales: convolución y *pooling*.

[15] https://towardsdatascience.com/how-to-select-the-right-evaluation-metric-for-machine-learning-models-part-1-regrression-metrics-3606e25beae0

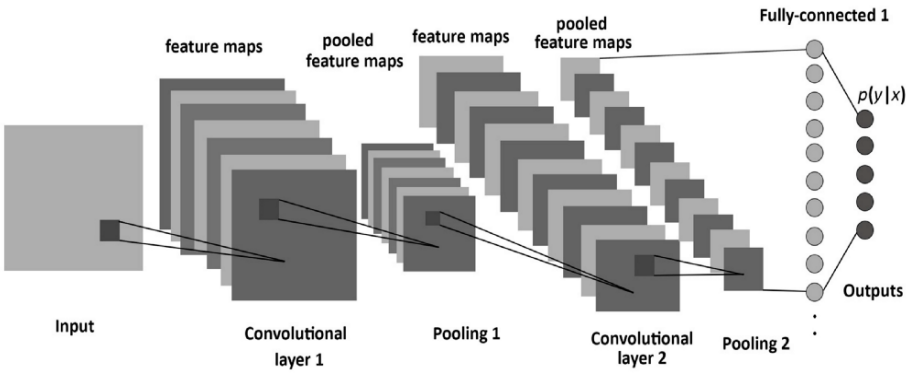


Figura 8- Ejemplo de red neuronales Convolucional[16]

Para entenderlo en detalle, hay que entender los conceptos de convolución y *pooling* que surgen en el campo de tratamiento de Imágenes y videos.

2.2.5.1 Convolución:

La convolución funciona con dos entradas: una imagen y un filtro, llamado *Kernel*. La Convolución aplica *Kernel* sobre la imagen, multiplicando sus valores, para obtener como salida otra imagen. Matemáticamente, podemos definir una convolución así:

$$(f * g)(i) = \sum_{j=1}^{m} g(j).f(i - j + m/2)$$

Es más fácil visualizar el proceso de convolución en el procesamiento de imágenes, y observar como *kernel* se desliza sobre una imagen completa cambiando los valores de cada píxel.

[16] http://ia-latam.com/2019/02/06/entendiendo-las-redes-neuronales-de-la-neurona-a-rnn-cnn-y-deep-learning/

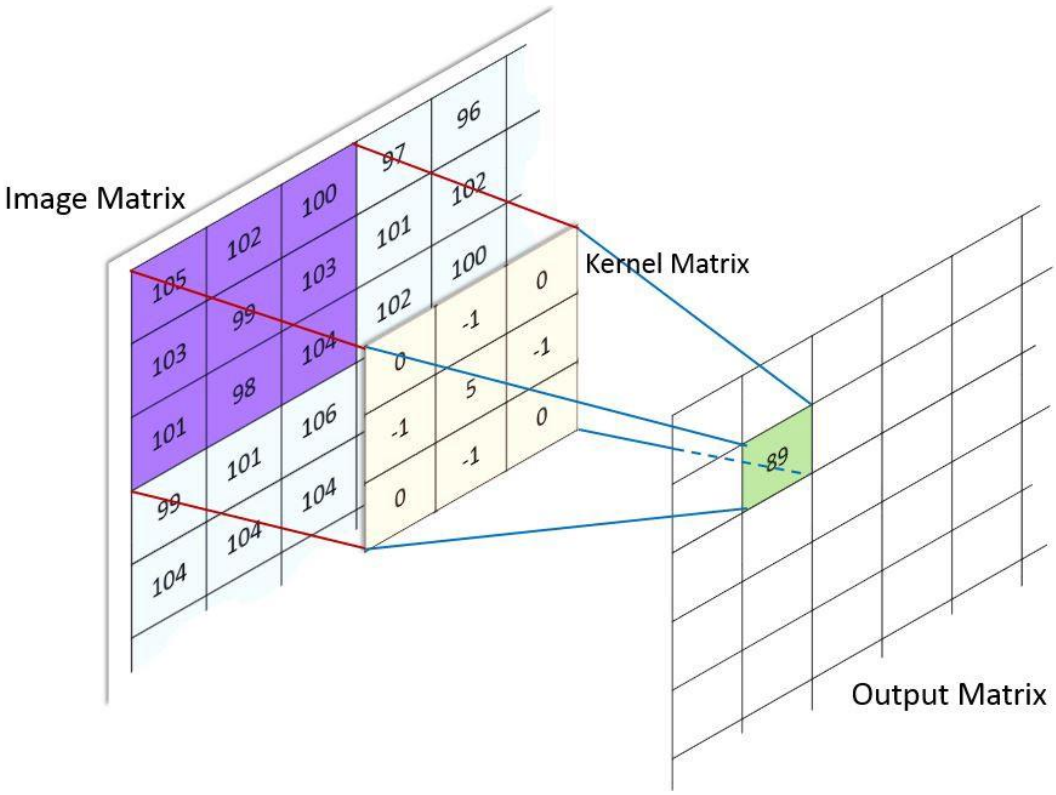


Figura 9- Ejemplo de tecnica de convolución usando *kernel* 3x3[17]

2.2.5.2 *Pooling*

El *Pooling* es un proceso de redimensionar una imagen, reduciendo su tamaño. El *pooling* recoge una imagen, la divide en regiones definidas según el *pool-size*, y luego selecciona un valor determinado de cada una de estas regiones para crear otra imagen.

Existen dos tipos de *pooling*: *MaxPooling* y *MinPooling*. El primero saca el valor más alto de cada región (dividida según el *pool-size*) para crear una imagen nueva y más pequeña. Mientras que el *MinPooling*, saca el valor más pequeño de cada región y lo convierte en una imagen nueva y más pequeña.

[17] http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

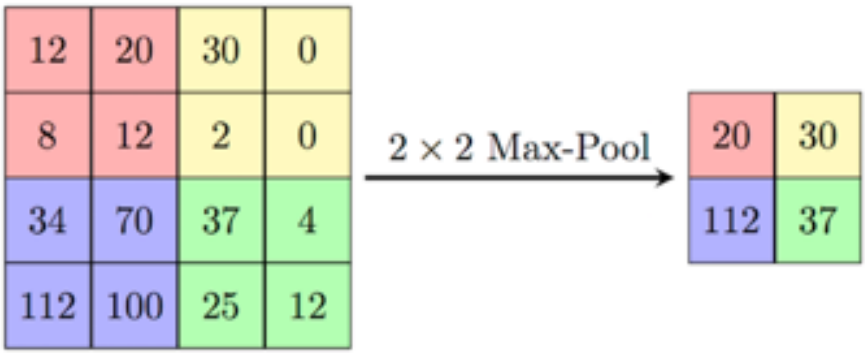


Figura 10- Ejemplo tecnica de Pooling[18]

La imagen es un ejemplo de *Maxpooling*, con *pool size* de 2X2. Podemos observar que se ha seleccionado el valor más alto de cada región para crear una nueva imagen.

2.3 Deep Learning

Se considera *Deep Learning* al uso de redes neuronales artificiales con múltiples capas ocultas, que son capaces de explotar relaciones no lineales complejas en los patrones de entrada.

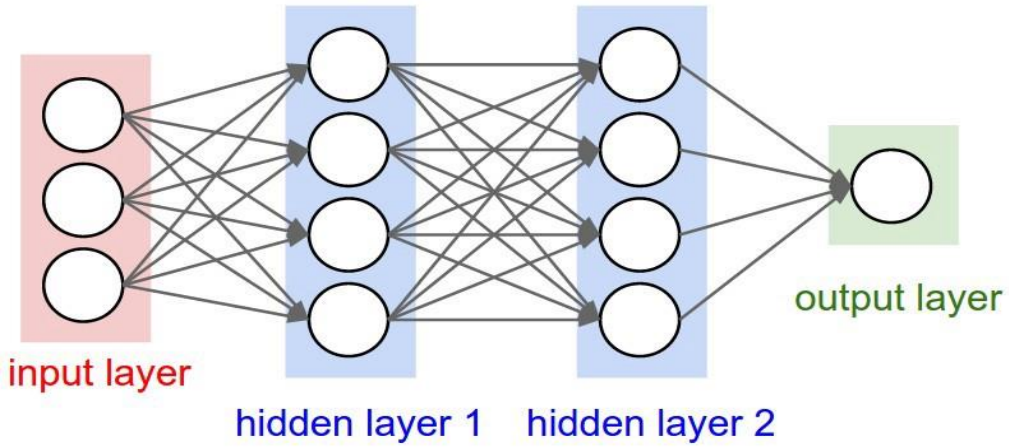


Figura 11- Arquitectura *Deep Learning*[19]

[18] http://machinelearninguru.com/computer_vision/basics/convolution/image_convolution_1.html

[19] https://hackernoon.com/challenges-in-deep-learning-57bbf6e73bb

La Figura 11 es un ejemplo de *Deep Learning*, donde la primera capa, las más a la izquierda es la capa de entrada, luego tenemos dos capas ocultas y una capa de salida.

2.4 *Kriging*

El *Kriging* es un método de interpolación que aprovecha relaciones conocidas a priori entre los datos para estimar el valor de una variable en un campo espacial, y es más efectivo que los métodos de interpolación más simples cuando hay al menos una autocorrelación espacial moderada entre los puntos de datos muestreados. A pesar de ello, la precisión de la interpolación por *Kriging* será limitada si el número de muestras es pequeño o los datos están limitados en el ámbito espacial (Liu, 2009).

Existen diferentes tipos de técnicas de *Kriging*, como el *Ordinary* Kriging (OK), el *Universal Kriging* (UK), el *Indicator* Kriging, el *Co- Kriging*, *Poisson Kriging* entre otros. La elección de qué *Kriging* usar depende de las características de los datos. El *Kriging* universal asume que hay una tendencia predominante en los datos mientras que el *Ordinary Kriging* asume que la media constante es desconocida, lo cual es un supuesto razonable, a menos que haya una razón científica para rechazarlo (Mesić, 2016). A partir de esta premisa ha sido el aplicado en este trabajo.

2.5 Herramientas para el Tratamiento De Datos

2.5.1 Python

El desarrollo del proyecto ha sido realizado enteramente con *Python*: Python es un lenguaje de programación interpretado, multiparadigma, fácil de usar que soporta programación imperativa, orientación a objetos y también, aunque en menor medida, programación funcional y está disponible en varias plataformas.

En otras palabras, *Python* es:

- **Interpretado**: capaz de ejecutarse sin necesidad de ser procesado por el compilador. Si hay errores se detectan cuando se ejecuta el código.

- **Multiparadigma**: soporta orientación a objetos, programación imperativa y programación funcional
- **Tipado dinámico**: el tipo de las variables se comprueba en tiempo de ejecución del código.
- **Multiplataforma**: disponible para *Windows*, *Linux* o *MAC*.
- **Gratuito**: Posee una licencia de código abierto[20].

Python es bastante usado en *Machine Learning* ya que está diseñado para ser legible y fácil de usar. Hay muchas librerías diseñadas para *Machine Learning* y aquí exponemos muy brevemente Las más importantes para nuestro trabajo:

Keras

Keras es una API de redes neuronales de alto nivel, que puede ejecutarse sobre *TensorFlow*, CNTK o *Theano*. Permite la creación sencilla de prototipos y admite tanto redes convolucionales como redes recurrentes. Asimismo, soporta ejecutarse sin problemas en CPU y GPU[21].

Matplotlib

Matplotlib es una librería de *Python* que permite generar, con alta calidad y con pocas líneas de código, graficos, espectoros de potencia y diagramas de dispersión etc., Está disponible para *Windows*, *Linux* o *MAC* y puede ser usada desde *Python scripts*, *Python* y *IPython shells*, *Jupyter notebook*[22] entre otros.

[20] Para más información www.python.org

[21] Para más detalle https://keras.io/

[22] Para más detalle https://matplotlib.org/

Pandas

Pandas (*Python Data Analysis Library*) es una librería de código abierto fácil de usar y una herramientas de análisis de datos para *Python*[23], que proporciona estructuras de datos de alto rendimiento. En este proyecto hemos usad *Pandas dataframe*.

NumPy

En la computación científica *NumPy* es una parte fundamental para la programación con Python. Contiene entre otras cosas[24]: Herramientas para diferentes lenguajes de programación como *C/C ++* y *Fortran*; Un poderoso objeto de array N-dimensional; y varias funciones para el álgebra lineal.

También se puede usar como un contenedor multidimensional de datos genérico y puede fácilmente integrarse en un amplio espectro de bases de datos.

PyKriging

Pykriging es una librería que se usa para implementar tecnología de *Kriging en Python*[25].

[23] Para más detalle https://pandas.pydata.org/

[24] Para más detalle https://www.numpy.org/

[25]Para más información http://pykriging.com/

CAPÍTULO 3- ELECCIÓN Y TRATAMIENTO DE DATOS E ENTRENAMIENTO

3.1 Definición del Problema:

Antes de profundizar en la parte de codificación y la parte del flujo de trabajo, necesitamos obtener la comprensión básica de nuestro trabajo, cuáles son los problemas y cuáles son las posibles soluciones.

Lo que nos interesa aquí es predecir la radiación solar a corto plazo en un área reducida y para ello utilizamos redes neuronales artificiales. Para realizar este trabajo hay que seguir los pasos generales del flujo de trabajo de *Machine learning* (ML):

1.Obtención de datos.
2.Preparación de datos.
3. Dividir datos para entrenamiento y test.
4.Seleccionar un algoritmo de ML para predecir la radiación a corto plazo para un objetivo basado en observaciones terrestres desde una red de piranómetros.
5.Entrenar los modelos.
6.Evaluar estos modelos predictivos en los datos de prueba.
7.Testear los modelos.
8.Analizar los resultados con las métricas apropiadas.

3.2 Obtención de Datos

Los datos usados para generar las predicciones se han obtenido del Laboratorio Nacional de Energías Renovables (NREL) perteneciente al Departamento de Energía de Estados Unidos[26], y corresponden a

[26] http://midcdmz.nrel.gov/oahu_archive/

la radiación global horizontal para un total de 17 estaciones ubicadas en el sur de la isla de Oahu, en Hawai, medida durante 593 días entre el 18 de marzo de 2010 y el 31 de octubre de 2011 y con mediciones cada segundo.

Estas estaciones están situadas entre sí a distancias no regulares, alrededor de 80 metros entre las estaciones en la zona con más densidad y con una distancia de 1.5 km entre las estaciones más alejadas.

Figura 12- Ubicacion de sensores en el sur de la isla de Oahu, en Hawai[27]

3.3 Preparación De Datos

El pre-procesamiento de datos es uno de los pasos más importantes que ayuda a construir modelos de *Machine Learning* con mayor precisión.

Es un proceso de limpieza de los datos en bruto. Lo datos recopilados de varias fuentes en el mundo real están en formatos distintos y probablemente con muchas variables irrelevante, según qué problema

[27] http://midcdmz.nrel.gov/oahu_archive

se busca resolver, lo cual hace que no sean factibles para el análisis. En este proceso, denominado pre-procesamiento, se limpian los datos en brutos y se convierten en un conjunto de datos limpios y listos para el uso en modelo aplicado en proyectos de *Machine Learning* y *Deep Learning*.

Para explicar la importancia de este paso, es necesario saber que los datos en brutos pueden presentar algunos problemas como[28]:

> ***1. Missing data***: La falta de datos, es cuando se interrumpe la continuidad de los mismos debido a problemas técnicos (sistema IoT) de cualquier tipo.
>
> ***2. Noisy data***: denominada también "*outliers*", ocurre debido a errores humanos, cuando los recopilan manualmente, o debido a problemas técnicos de los dispositivos encargados de la recopilación.
>
> ***3. Inconsistent data***: los errores humanos, con los nombres o los valores, o la duplicación de datos puede dar pie a datos incoherentes.

En nuestro trabajo, los datos brutos obtenidos de (NREL) son de 593 día, y después de realizar el pre-procesamiento de datos este número se reduce a 548 días, que van a ser la piedra angular de del proyecto.

Cabe mencionar que gracias a este análisis, detectamos que casi todos los datos de una estación en concreto no son válidos. Y que, para determinados días, varias estaciones presentaban datos alejados de la tendencia habitual debido, entre otros factores, a la actividad volcánica de la isla.

3.4 División de Datos para Entrenamiento y Test.

Cuando entrenamos modelos no podemos usar los mismos datos para entrenar, validar y evaluar el modelo. Por lo tanto, es preciso dividir los datos en tres conjuntos distintos que nos permitan asegurar el buen funcionamiento y por ende la fiabilidad y eficiencia de nuestro modelo en busca de los mejores resultados.

[28] https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

En este estudio, hemos destinado 466 días del total al entrenamiento y la validación. De estos 466, el 83% serán para el entrenamiento y el 17% para validación. Los otros 82 días los hemos asignado a evaluar el modelo (conjunto de test).

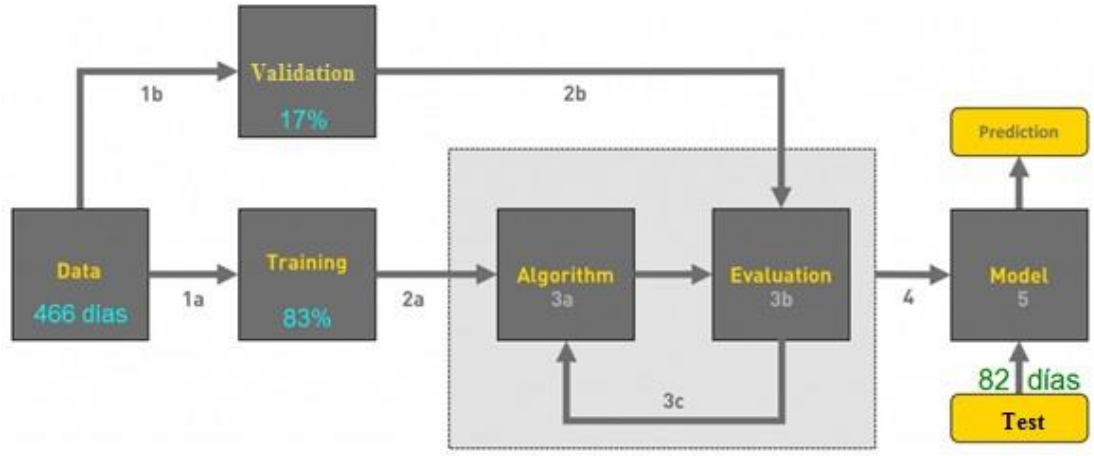


Figura 13- Proceso de divison de daots de training, validacion y test[29]

3.5 Seleccionar Modelo

Consiste en investigar y seleccionar el modelo para el tipo de datos que se manejan. Ya que partimos de un histórico de datos, podemos emplear modelos supervisados. Recordemos que, como explicamos en el capítulo anterior, hay dos tipos de aprendizaje supervisado: regresión y calificación. La regresión tiene el objetivo de predecir valores continuos (valores numéricos) mientras que la clasificación tiene la tarea de asignar una clase, es decir predecir a que clase pertenece un conjunto de datos.

En nuestro estudio aplicamos el método de regresión porque los resultados deseados son numéricos y continuos. Concretamente, vamos a usar dos tecnologías diferentes: las redes neuronales *Fully-Connected* y las redes neuronales Convolucionales.

[29] https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94

3.6 Redes Neuronales Fully-Connected

3.6.1 Creación De Entradas

Para elaborar las predicciones es necesario estructurar los datos de una manera concreta en forma de matrices para poder crear el modelo predictivo. La matriz se compone de columnas denominadas: *feature* y *label*.

Los *feature* son valores individuales que actúan con entradas, mientras que los *label* son la salida que debe obtenerse como resultados de las entradas (feature). Así pues, hay que decidir qué valores serán los *feature* y cuáles serán los *label* en función de lo que queremos predecir.

Una vez tomada la decisión, se crea un programa que toma la información de un fichero de configuración (.JSON) donde se indica qué estación es la que se quiere predecir, qué estaciones se quieren tomar en cuenta como entrada, cuántas muestras se tomarán y el horizonte de predicción.

La figura 14 muestra un ejemplo de los distintos parámetros para la selección del horizonte de predicción y la ventana temporal que se establece como entrada. En la zona sombreada, se encuentran las muestras que se usan como componentes en una entrada para el modelo para una estación de entrada. En este caso, se tomarían 4 muestras, tomando como referencia el instante actual (t-2 en la figura 14). Si la granularidad de cada muestra es de 10 segundos, la ventana temporal que consideramos es de 40s en el pasado, para predecir lo que ocurrirá dentro 20, pues el horizonte de predicción se establece a dos muestras en el futuro.

Al incluir estos parámetros en un fichero de configuración (.JSON), se pueden crear diferentes matrices de entrenamiento sin modificar el código *Python*.

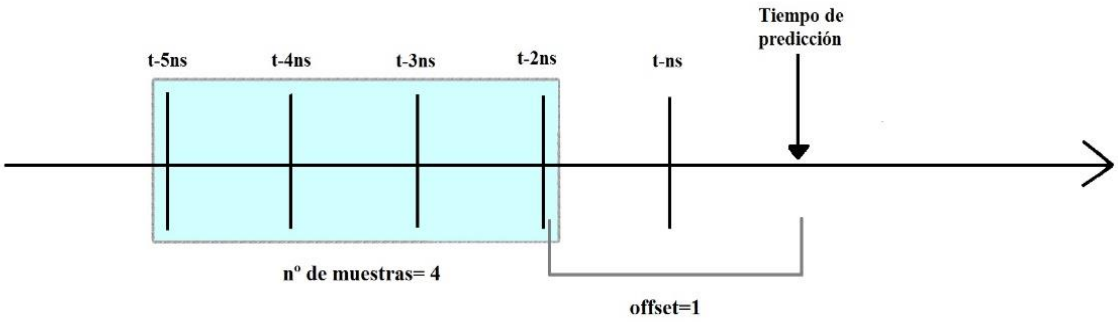


Figura 14 -Esquema de los componentes de una estación concreta en la creación de la matriz.

En nuestro estudio hemos usado cuatro muestras de cada estación. Cada muestra es de media de 10 segundos, de modo que para predecir un valor usamos los 40 segundos anteriores como *features* o entradas.

3.6.2 Tipos De Pruebas

A lo largo del estudio, hemos creado dos modelos:

Modelo 1: el entrenamiento se realiza con datos de todas las estaciones, pero la predicción se realiza para una única ubicación.

Modelo 2: el entrenamiento se realiza con datos de todas las estaciones y la predicción se realiza para cinco ubicaciones diferentes de forma simultánea (la última capa tendrá por tanto cinco neuronas).

3.6.3 Topología

La topología de las redes neuronales consiste en la organización y disposición de las neuronas en la red formando capas o agrupaciones de neuronas. Los parámetros fundamentales de la red son: número de capas, número de neuronas por capa y grado de conectividad. En nuestras pruebas hemos establecido los siguientes parámetros:

Numero de capas	4 capas en total: capas ocultas + entrada + salida
Número de neuronas por capa	300 neuronas en las capas ocultas. Una única neurona en la última capa para el modelo1 y cinco neuronas en la última capa para el modelo2.
Grado de conectividad	*Fully-Connected*
Épocas para el entrenamiento	200
Función de Activación	*ReLU*
Dimensión de Entrada	66
Dimensión de Salida	Uno (Modelo1); y cinco (modelo2)
Métricas	*Mean Absolute Error* (MAE), *Mean Squared Error* (MSE) y *Root Mean Square Error* (RMSE)
Función de coste	MSE

3.6.4 El Entrenamiento

Durante el entrenamiento se utilizan dos conjuntos de datos: entrenamiento y validación, con los porcentajes de datos anteriormente mencionados (83% y 17%, respectivamente).

El entrenamiento cosiste en proporcionar datos de entrada y nuestro algoritmo calcula una salida predicha. Esta salida predicha es comparada con la salida real que hemos proporcionado como *label*, ya que se trata de un modelo supervisado. El modelo calcula el error usando la función de coste definida y trata de minimizar dicho error modificando los pesos de las neuronas. Este proceso de optimización se denomina *backpropagation.* El proceso se repite con todos los vectores de entrada tantas veces como se indique con el número de épocas.

3.6.5 Validación

En cada época, a la vez del entrenamiento, realizamos la validación de nuestro modelo pasando los datos de la *dataset* de validación. Es un proceso similar al del entrenamiento pero ejecutado con datos diferente (los 17% restantes). Lo que se calcula aquí es el error de validación, pero sin realizar la etapa de *backpropagation*. Esto permite ganar intuición sobre los datos que tenemos y valorar si debemos modificar nuestro modelo: añadir/quitar capas, añadir/quitar neuronas en las capas ocultas, modificar el factor de regularización.

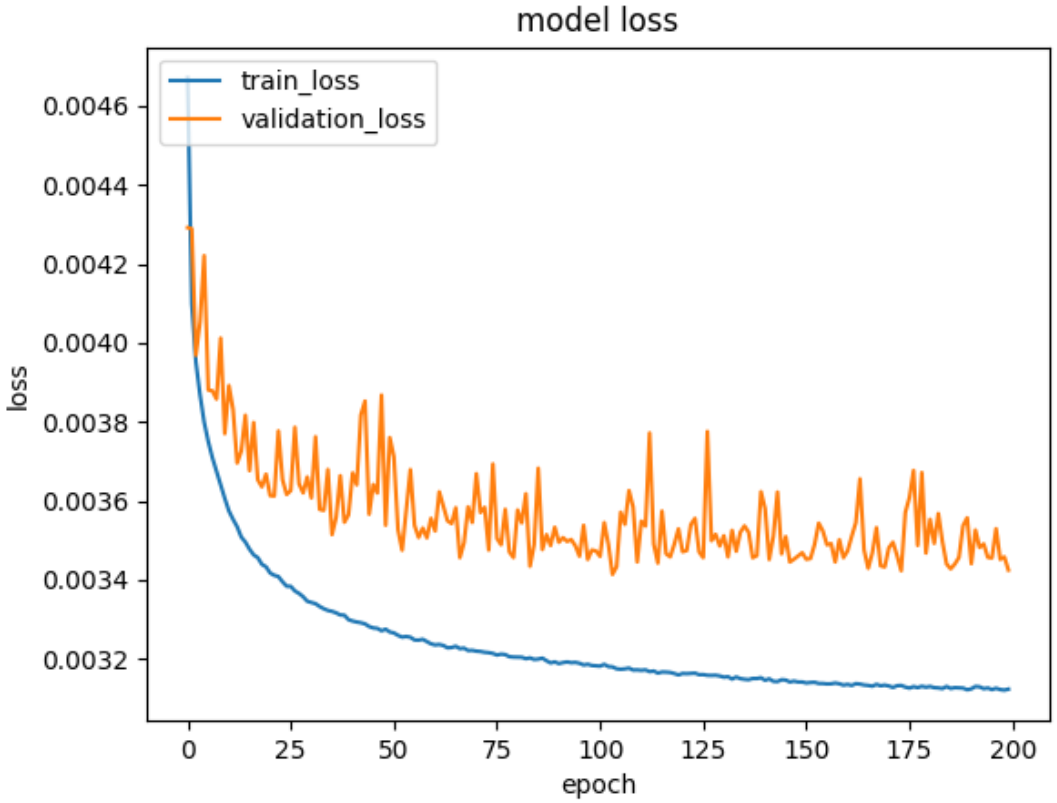


Figura 15- Muestra la evaluación de la función de coste para cada época del entrenamiento para el conjunto de entrenamiento y de validación.

La figura 15 refleja el entrenamiento de nuestro modelo. Los datos que vemos son los errores de entrenamiento y de validación. Observamos que en cada época el error de entrenamiento va bajando: empieza con 0.00525 y termina con un valor de 0.00350. También el valor de error de validación va bajando desde casi 0.00460 y termina en 0.00400.

3.6.6. Overfitting

El *overfitting* o sobreajuste ocurre cuando un modelo aprende tanto los detalles y el ruido en los datos, en su entrenamiento, lo cual repercute negativamente en el modelo cuando recibe datos nuevo. Es decir, el modelo aprende y asume el ruido como un datos valido, cuando en realidad no es más que mero ruido que afecta la eficiencia de muestro modelo. Si la precisión del modelo con los datos de entrenamiento es

muy superior a la precisión obtenida con los datos de validación, podemos estar ante un problema de sobreajuste.

3.7 Predicción De Mapas De Irradiación

En la anterior estrategia, con las redes *Fully-Connected*, solo podíamos predecir en puntos concretos y entrenando modelos diferentes para cada punto. Pero cuando hubiera un problema con algún sensor o cuando se decidiera poner más sensores teníamos que cambiar toda la topología de la red.

En búsqueda de algo más global, utilizamos el *Kriging*, que es un medio de interpolación de datos cuando existe una cierta correlación espacial en nuestro conjunto de datos. Hasta ahora, con los datos que tenemos, podíamos predecir la radiación solar solamente en aquellas ubicaciones en las que hay sensores, pero no en el resto de puntos dentro del despliegue general de los sensores. Para conseguir datos de estos puntos imaginarios, necesitamos crear datos interpolados entre dos sensores. En nuestro caso hemos elegido el *Ordinary Kriging*.

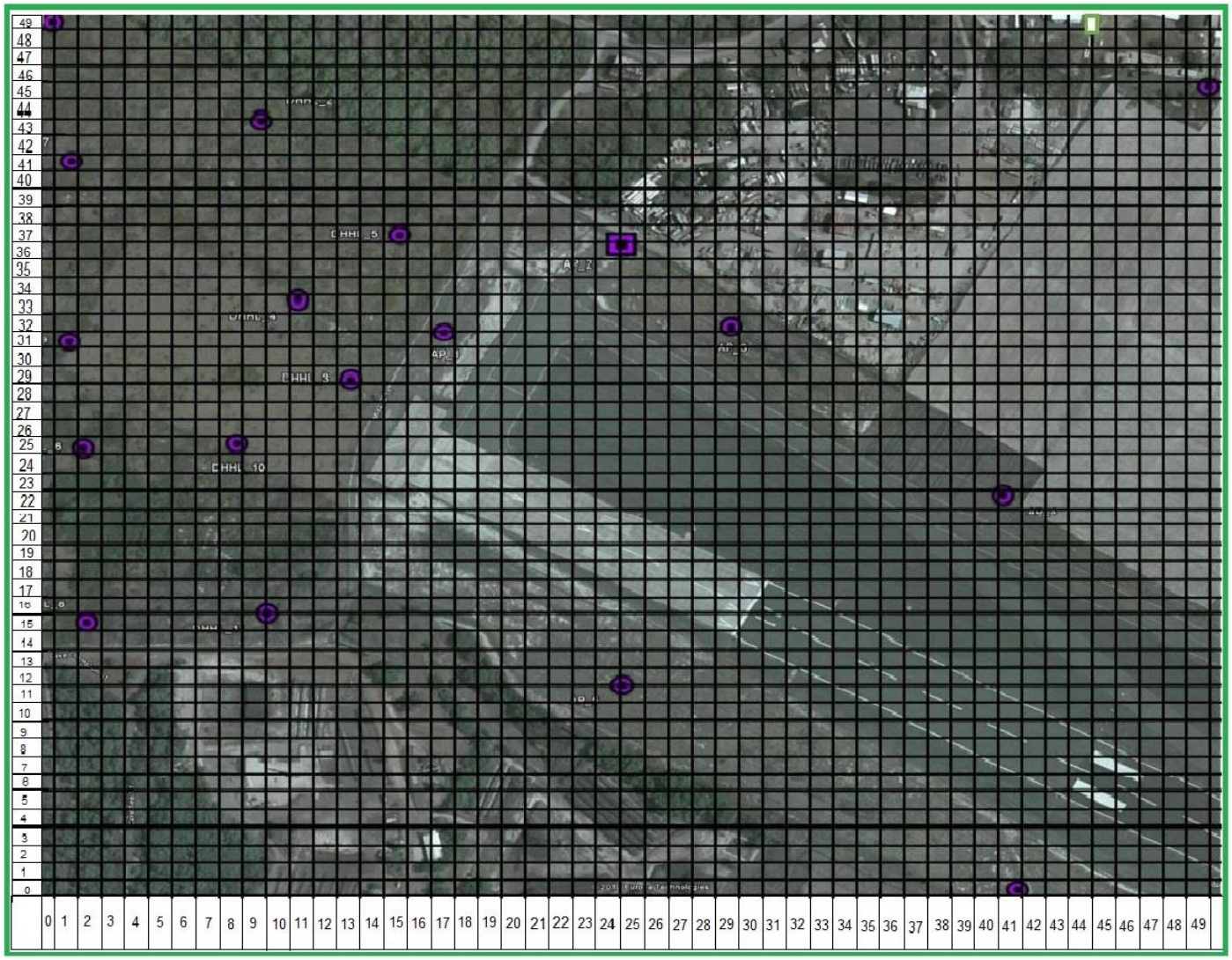


Figura 16- Mapa de *Kirging* de el area de trabajo - elaboracion propia

Hemos creado un script de *Python* usando la librería *PyKriging* proporcionando los datos de entrada de los 17 sensores y como resultados obtenemos un *grid* de 50X50.

Ahora nuestros datos dejan se ser un matriz de una dimensión y forman una matriz de dos dimensiones, donde cada entrada de matriz antigua es ahora un *grid* de 50X50.

A partir de los nuevos datos podemos predecir la radiación solar donde no hay sensores. Y las ventajas son:

1- Poder predecir la radiación solar en un punto donde haya un sensor averiado.

2- Añadir o quitar un sensor sin cambiar la topología de nuestra red y modelo.

3.8 Redes Neuronales Convolucionales

Después crear datos de *Kriging* usamos las redes neuronales Convolucionales como mecanismo de aprendizaje para solucionar este problema. Hemos creado datos con dos dimensiones usado *Kriging*, tal como hemos explicado anteriormente. En vez de tener 66 valores de una dimensión por cada entrada, ahora tenemos una matriz de 50X50 por cada entrada. Para sacar partido a esta información, vamos a usar las redes Convolucionales (CNN) ya que permiten que sus entradas sean bidimensionales y son capaces de encontrar relaciones espaciales entre los pixeles de las imágenes de entrada, que posteriormente se usarán como entrada para una red *Fully-Connected.*

3.8.1 Topología de las Redes Convolucionales

En las de redes Convolucionales (CNN), como el las *Fully-Connected*, la topologías de red consiste en la organización y disposición de las neuronas en la red formando capas o agrupaciones de neuronas. Los parámetros fundamentales son los mismos: Número de capas, número de neuronas por capa y grado de conectividad. Aquí los detallamos en cada capa:

Primera capa	Conv2D con 32 número de filtro; tamaño de *Kernel* 3X3; función de activación=RelU
Segunda capa	Conv2D con 64 número de filtro; tamaño de *Kernel* 3X3; función de activación=ReLU
Tercera capa	*Maxpooling* 2D; pool-size 2X2
Cuarta capa	*Dropout* (0.25)
Quinta capa	*Flatten*

Sexta capa	Dense; número de neuronas 128: Función de activación=Relu
Séptima capa	*Dropout* (0.5)
Octava capa	Dense; número de neuronas 1; Función de activación=Relu
Numero de épocas	200
Función de coste	*Mean Squared Error* (MSE)
Métricas	*Mean Absolute Error* (MAE), *Mean Squared Error* (MSE) y *Root Mean Square Error* (RMSE)

3.8.2 Entrenamiento

Como ya hemos visto, las redes neuronales Convolucionales (CNN) se aplican mediante convolución y *pooling*.

El proceso es el siguiente: la primera capa tiene 32 filtros con un tamaño de *kernel* de 3X3, lo que genera 32 matrices de salida para cada entrada (el resultado de aplicar cada uno de los 32 filtros de 3x3 a la imagen de entrada). Esta salida será transferida como entrada a la segunda capa que la procesa de la misma manera pero esta vez con 64 filtros dando como salida 64 imágenes para cada una de sus entradas. La tercera capa recoge esta entrada y le aplica el *MaxPooling* con tamaños 2X2, lo cual reduce el tamaño de cada imagen a una cuarta parte (eliminamos 3 de cada 4 píxeles).

La cuarta capa (*Dropout*) descarta el 0.25 de las imágenes para evitar el *Overfitting*, Finalmente, la capa (*Flatten*) convierte estos datos en una sola dimensión y los envía como entrada a la siguiente capa (Dense) de 120 neuronas. De nuevo, esta capa descarta el 0.25 de los datos y los transfiere a la última capa, que sería la salida de nuestro modelo (ver Figura 8 para una ilustración de la arquitectura típica de una CNN).

3.8.3 Validación

El proceso de validación sigue los mismos pasos que hemos explorado en Redes *Fully-Connected*, con la única diferencia de que los datos de validación representan el 20% de los datos de entrenamiento.

CAPÍTULO 4 – RESULTADOS

4.1 Los Resultados

En este capítulo vamos a exponer los resultados obtenidos del entrenamiento. Primero vamos a presentar los resultados de redes *Fully-Connected.* Recordemos que tenemos dos modelos distintos: el Modelo1 (predicción de la radiación solar en la ubicación de un solo sensor); y el Modelo2 (predicción en la ubicación de cinco sensores simultáneamente).

Para comprobar que los resultados son buenos y comparar los resultados de los dos modelos utilizamos un modelo persistente como punto de referencia. Este modelo predice que la radiación (normalizada usando un modelo de cielo claro) para el horizonte de pronóstico deseado *h* será la misma que el último valor medido en la misma ubicación:

$$Y(t+h) = X(t)$$

Como métricas para el proceso de validación hemos usado *Root Mean Square Error* (RMSE), que se muestra en la siguiente ecuación.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(Y_i - Y)^2}$$

RMSE permite comparar el rendimiento de cada modelo con diferentes valores de parámetros. También utilizamos esta métrica para comparar y analizar los resultados de nuestro experimento utilizando el conjunto de test.

Asimismo, utilizamos una métrica denominada *Skill* que representa el porcentaje relativo de mejora en RMSE con respecto al modelo persistente:

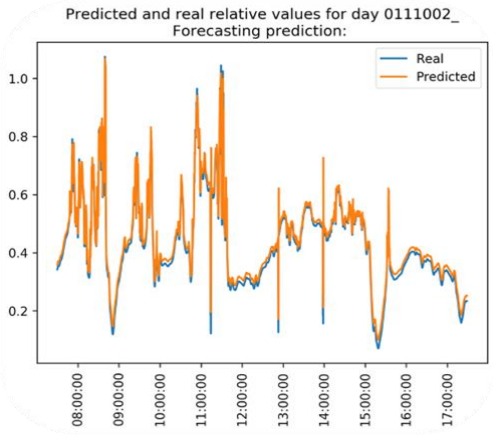
$$S = 100\left(1 - \frac{RMSE_{\mathrm{mod}\,el}}{RMSE_{persistente}}\right)$$

En el Modelo1 hemos entrenado un modelo para la predicción de la ubicación de un sensor: *ap1*, mientras que en el Modelo2, hemos entrenado un modelo para la predicción de la ubicación de cinco estaciones que son las siguientes: *ap1, dh3, dh4, dh5 y dh10.*

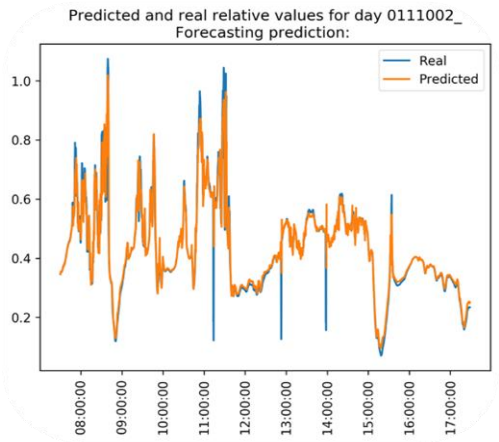
Para realizar comparaciones, en ambos modelos vamos a exponer los resultados de una única ubicación de sensor que es: *ap1*. Cabe mencionar que hemos elegido esa estación porque pertenece a la zona céntrica de nuestra área de trabajo y tenemos a su alrededor más sensores que nos proporcionan más datos. Es decir, si hubiésemos elegido una ubicación de sensor más alejada del centro y no alineada con la dirección del viento dominante como es el caso de *ap7* (figura 12) es previsible que los resultados de la predicción fueran peores.

Como ya hemos indicado anteriormente, nuestro conjunto de test es de 82 días, y los horizontes de predicción que contemplamos son los siguientes: 10 segundos, 30 segundos, 1 minutos, 2 minutos, 5 minutos y 10 minutos.

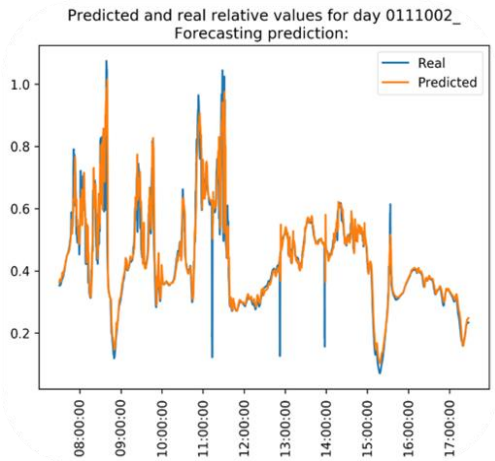
A continuación mostramos figuras (17 &18) de la radiación medida y la estimada en un día al azar que es: 2 de octubre del 2011; el horario de predicción de las imágenes a continuación va desde las 7:30 horas hasta las 17:30 horas.



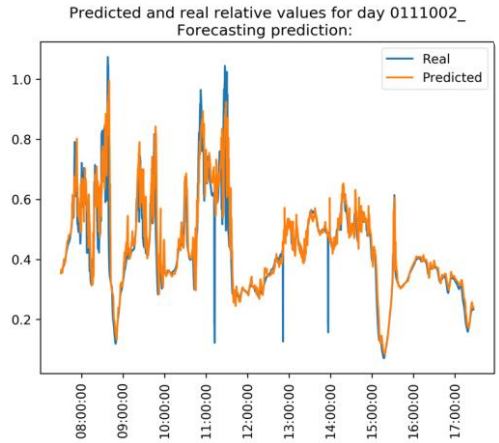
a) Horizonte de predicción 10s



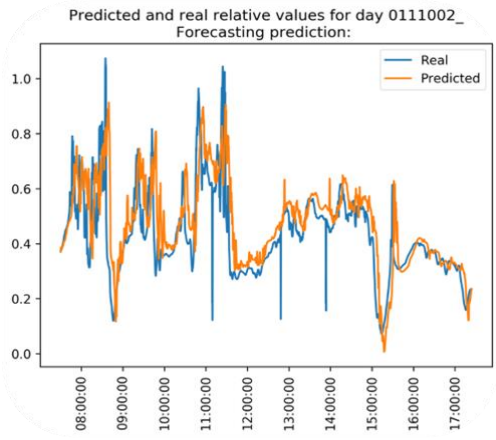
b) Horizonte de predicción 30s



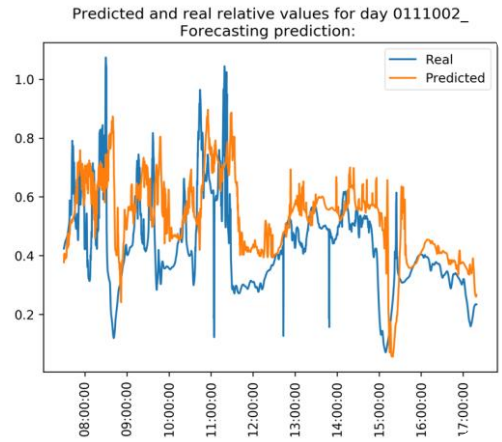
c) Horizonte de predicción 1m



d) Horizonte de predicción 2m

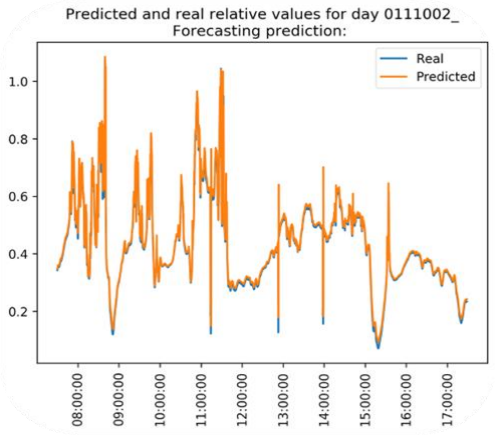


e) Horizonte de predicción 5m

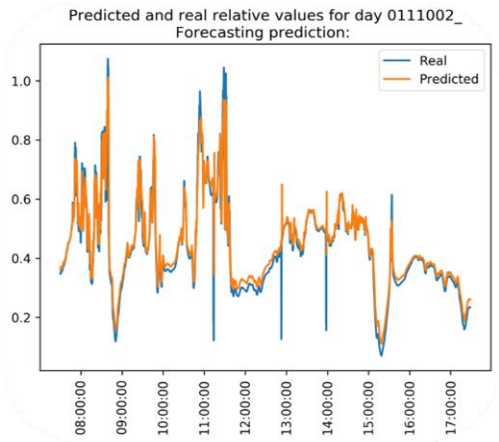


f) Horizonte de predicción 10m

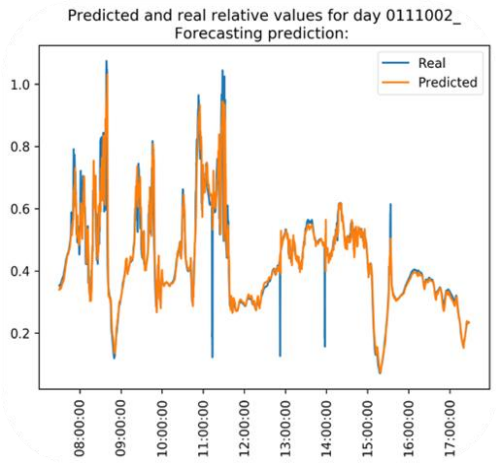
Figura 17 - Resutaldos Modelo1 para diferentes horizontes de predicción



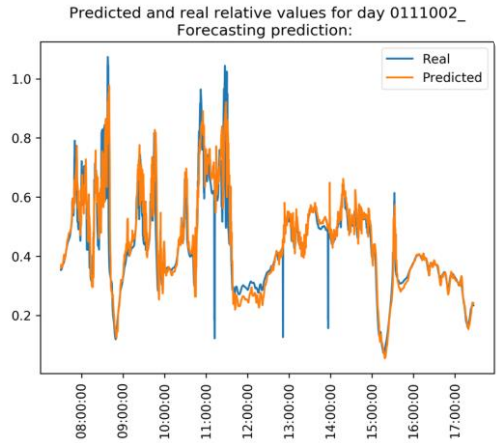
a) Horizonte de predicción 10s



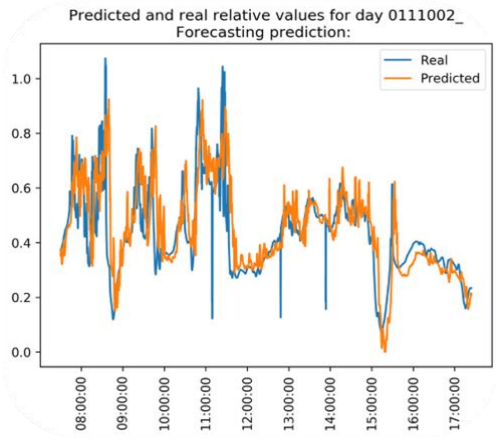
b) Horizonte de predicción 30s



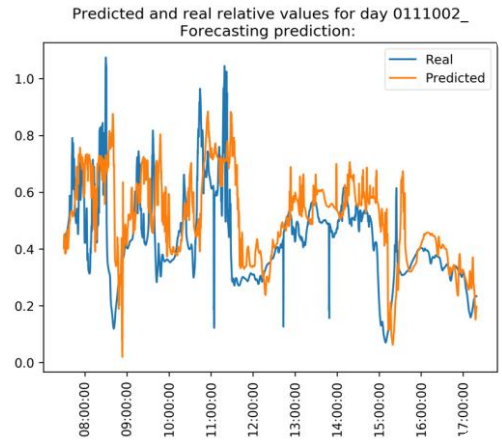
c) Horizonte de predicción 1m



d) Horizonte de predicción 2m



e) Horizonte de predicción 5m



f) Horizonte de predicción 10m

Figura 18- Resutaldos Modelo2 para diferentes horizontes de predicción

El RMSE mide las diferencias entre el valor real y el valor predicho. Es decir, el valor ideal de RMSE es (0), que se da cuando los valores real y predicho son idénticos, lo cual sería ideal pero imposible.

A partir de esta premisa, entendemos que cuando el valore de RMSE es pequeño nuestro modelo entrenado es mejor. En la siguiente figura (19) se muestran los resultados de RMSE de ambos modelos:

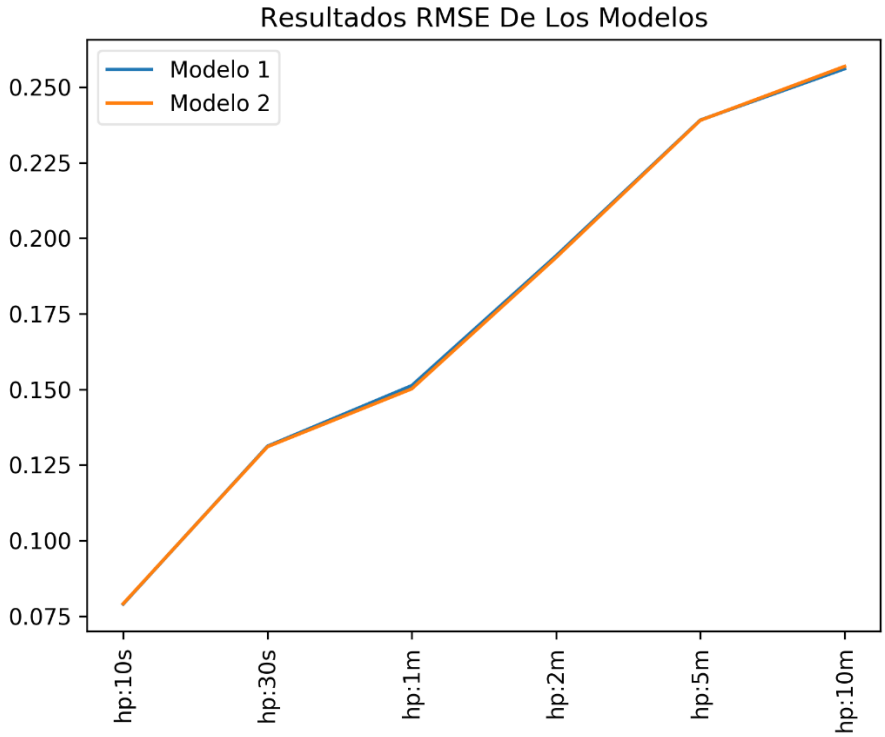


Figura 19- los Resultados de RMSE de los dos modelos para los diferentes horizontes de predicción

En la figura 19, se puede observar que cuando avanzamos en el horizonte de predicción el valor de RMSE va en aumento. Eso quiere decir que la precisión de nuestro modelo, cuando aumentamos el horizonte de predicción, empeora.

Esto se debe a que el área que estudiamos tan solo tiene 17 sensores distribuidos en menos de 1km^2 y la velocidad del viento dominante es suficiente como para que un frente de nubes atraviese toda el área en menos de 2minutos. Eso quieres decir que podemos predecir con gran éxito la radiación cuando el horizonte de predicción es menor, porque ya disponemos de información que nos proporciona sensores alrededor de nuestro ap1 y que facilitan la predicción. Cuando queremos predecir horizontes a partir de 1 minuto, la precisión cae, porque no tenemos información, a falta de sensores en zonas más alejadas. Por lo tanto los resultados de predicción para 10s son mejores en comparación con la predicción para 30s, y la de 30s es mejor que la de 1 minuto, y así sucesivamente.

Por su parte, el *Skill* mide la mejora relativa de nuestro modelo entrenado con respecto al modelo persistente. Si el valor de *Skill* es negativo eso indica que el modelo entrenado es peor que el modelo persistente. Pero ninguno de nuestras métricas ha dado este resultado negativo. Cabe reseñar que cuando el valor de *Skill* es más grande, el ratio entre ambos modelos es más grande y eso es positivo. En la siguiente figura (20) se muestran los resultados de *Skill* de ambos modelos:

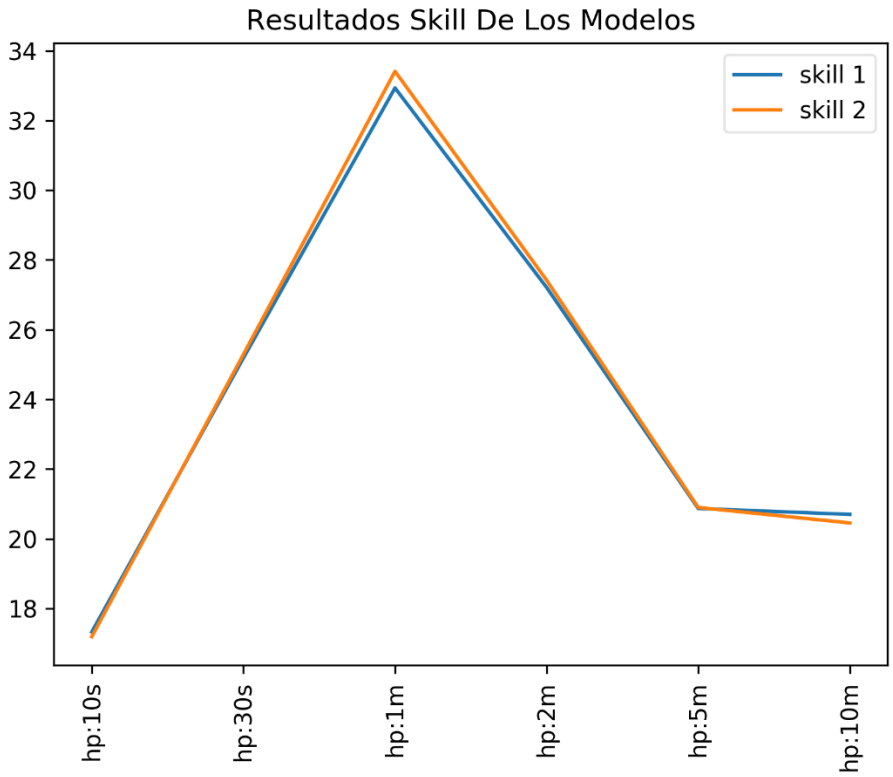


Figura 20- Resultados *Skill* de los dos Modelos

Observamos que en el intervalo de 10s hasta 1m el valor de *Skill* va aumentando, y a partir de 1m ese valor sufre una caída hasta los 10m.

El aumento de *skill* (10s - 1m) demuestra que nuestros modelos, a pesar de empeorar su precisión (medida en términos de RMSE), van mejorando en comparación con el modelos persistente. Pero a partir de 1m nuestros modelos dejan de mejorar respecto al caso base debido, como ya hemos indicado previamente, a la falta de más sensores que proporcionen información adelantada. Aun así nuestros modelos siguen siendo significativamente mejores que el modelo persistente.

Cuando observamos las figuras 17 & 18, vemos que los resultados son prácticamente iguales. Esto también queda patente en las gráficas de *RMSE* Y *Skill* (19 & 20). Por la tanto, y para distinguir las diferencias entre los dos modelos (Modelo1 y Modelo2) con más detalles, hemos creado un script de *Python* que crea otra imagen donde podemos ver las diferencias entre ambos modelos a lo largo de 82 días.

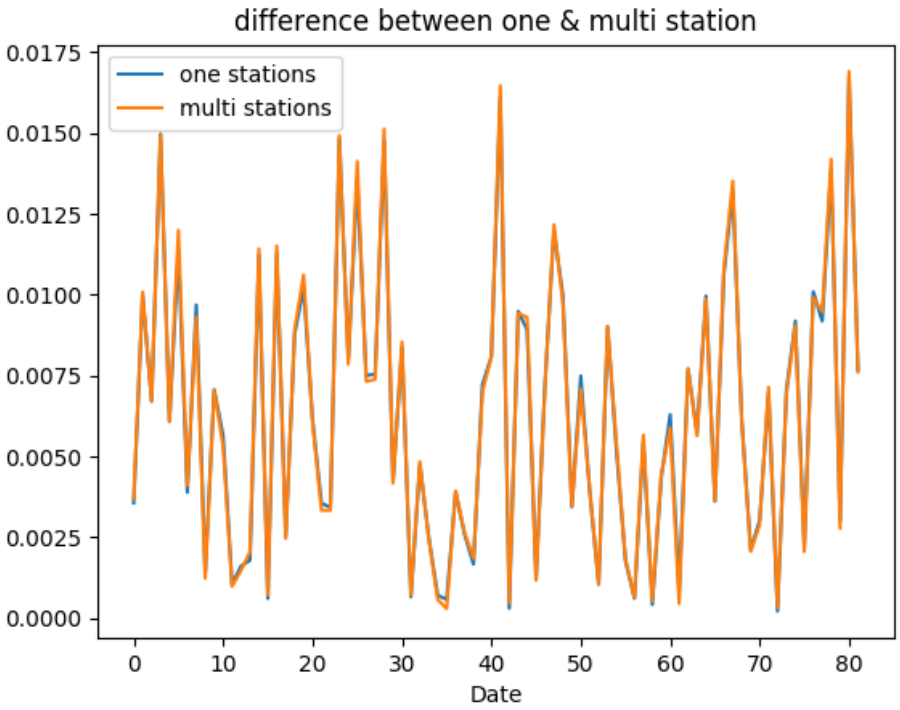


Figura 21- Diferencias entre los Modelos 1 y 2 en 82 días

La figura 21 muestra los valores *Mean Squared Error* (MSE) obtenidos de cada día y son prácticamente iguales en ambos modelos. Por lo tanto, y como los resultados de Skill y RMSE demostraron, ambos modelos son muy similares.

4.2 Resultados de las Redes Neuronales Convolucionales CNN

Ha llegado el momento de exponer los resultados de las rede neuronales Convolucionales (CNN). Pero primero hay que mencionar que los resultados de CNN no son del total de 82 días, sino sólo de 11 días de test y 60 días de entrenamiento. Esto se debe a que las Redes CNN tardan mucho en entrenarse y no teníamos el código preparado para usar máquinas con GPUs. Solo hemos hecho una prueba de 11 días de test, con un horizonte de predicción de 10 segundos.

Elegimos al azar el día es el 9 de abril de 2010 y el horario de predicción es de las 7:30 horas hasta las 17:30 horas. Los resultados fueron los siguientes:

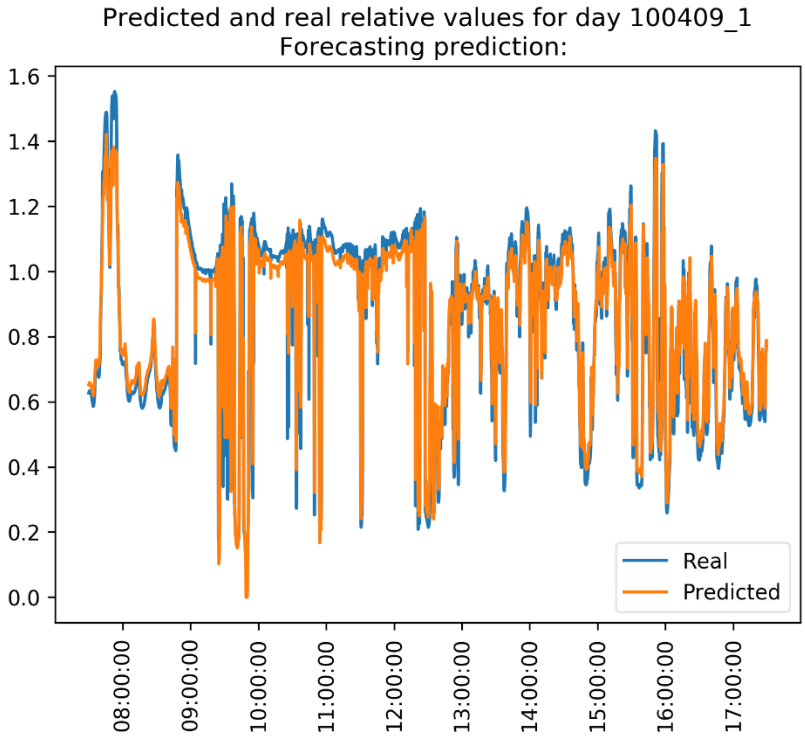


Figura 22- Modelo CNN, horizonte de predicción 10 segundos

Valores obtenidos después de aplicar métricas

RMSE modelo	0.08276310005989061
RMSE Persistente	0.12242383899302534
Skill	32.396254895579816

Vemos que a pesar de que los resultados fueron tan solo 11 días, son muy relevantes porque el valor de *RMSE* es muy similar al *RMSE* de *Fully-connected* con horizonte de predicción de 10s (0.0789535526786343) que obtuvimos del test de 82 días.

Por otra parte, cabe mencionar que el *Skill* de *Fully-connected* es (17.327273951275902) lo cual es mucho peor en comparación con el de CNN (32.396254895579816). Lo que sucede es que para esos 11 días concretos, el modelo persistente lo hace peor (en media) que para el total porque serán días con variaciones más bruscas (más nubes o cambios en el viento, etc.) lo cual explicaría que el valor de *Skill* de CNN sea más alto.

CAPÍTULO 5 - CONCLUSIONES

5.1 Las Conclusiones

El sol es la fuente de energía renovable con más potencial para hacer frente a los desafíos cada vez más importantes de la demanda energética mundial. Los estudios apuntan que la cantidad de energía solar que llega a la tierra equivale 1,7x1014 KW, lo que representa la potencia de 170 millones de reactores nucleares de 1.000 MW de potencia eléctrica unitaria, o lo que es lo mismo, 10.000 veces el consumo energético mundial[30].

A pesar de esto, todavía no hemos podido aprovechar esta fuente de energía de manera eficiente y constante. Uno de los desafíos más importantes reside en no saber con precisión cuando se va a producir en determinadas áreas geológicas, lo cual nos permitiría mejor uso de esta energía para así poder integrarla en los actuales sistemas de abastecimiento de energía en busca de modelos más sostenibles capaces de alimentar la creciente necesidad energética.

Asimismo, saber cuándo habrá picos o bajadas en la producción solar a corto plazo es un factor crucial a la hora de tomar decisiones eficaces sobre las redes de distribución. En este campo se desarrolla nuestro estudio, que intenta predecir la radiación solar en determinadas áreas con horizontes de predicción que oscilan entre los 10 segundos y los 10 minutos.

Para realizar este trabajo hemos empleado redes neuronales artificiales (RNA), usando datos proporcionados por 17 sensores en Hawai, para crear modelos que hemos usado para predecir la radiación solar.

Los resultados obtenidos demuestran que el horizonte de predicción es más preciso cuanto menor es el horizonte de predicción. Es decir, que los resultados de 10 segundos son los mejores en comparación con los de 30 segundos, y estos últimos son mejores en comparación con los de 1 minuto y así sucesivamente.

[30] Mas en http://bibing.us.es/proyectos/abreproy/5206/fichero/2.LA+RADIACI%C3%93N+SOLAR.pdf

Esto tiene sentido porque cuando predecimos un horizonte de tiempo mayor las condiciones atmosféricas pueden variar (humedad, dirección y velocidad del viento, nubes, etc.) Y esas variaciones pueden tener consecuencias en nuestros resultados, teniendo en cuenta que manejamos tan solo 17 sensores y solo medimos la radiación solar pero no las demás condiciones atmosféricas, como las antes mencionadas.

Para optimizar esta situación y mejorar los resultados es conveniente ampliar primero la zona de trabajo, es decir instalar más sensores en una superficie más amplia. También, sería favorable medir, simultáneamente, otros factores atmosféricos como la velocidad y dirección del viento o la humedad, entre otros, aunque la velocidad de viento en la superficie no es igual, ni tiene por qué estar correlacionado, a la velocidad del viento a la altura de las nubes.

La tecnología de Deep Learning aplicada a este estudio demuestra que es un método eficaz y eficiente a la hora de predecir la radiación solar a corto plazo en una determinada superficie. A pesar de sus reducidas dimensiones, este trabajo nos hace creer que no necesitamos tecnologías más caras y/o complejas para realizar esta tarea; basta con tener una amplia red de sensores y un buen modelo de Deep Learning para obtener resultados suficientemente precisos.

5.2 Conclusions

The sun is the most likely renewable energy source to meet the increasingly important challenges of global energy demand. Only a tiny fraction, 1.7x1014 kW, of the total solar radiation emitted is intercepted by the earth. However, 30 minutes of solar radiation falling on earth is equivalent to the world energy demand for one year.

Despite this, we have not yet been able to take efficient and constant advantage of this energy source. One of the most important challenges lies in not estimating precisely the amount of energy reaching a certain area. This will allow us to enhance the use of this energy in order to facilitate its integration into the global energy systems, in search of more sustainable models capable of satisfy the growing energy needs.

Likewise, knowing when there will be peaks or decreases in solar production in the short term is a crucial factor when making effective decisions about distribution networks. In the framework of this field, our work focuses on trying to forecast irradiance in certain areas with a forecast horizon ranging from 10 seconds to 10 minutes.

To carry out this work we have employed Artificial Neural Networks (ANN), using data provided by 17 sensors in Hawaii, to create models that we trained and used to forecast irradiance.

The results obtained show that the prediction is more precise with shorter forecasting horizons. That is, the estimated irradiation for the next 10 seconds is more accurate than the prediction 30 seconds ahead; and the latter is better compared to that of 1 minute and so on.

This makes sense because when we forecast a longer time horizon, atmospheric conditions may vary (humidity, wind direction and speed, clouds, etc.) and these variations can have consequences for our estimates, considering that we only manage 17 sensors and we only measure irradiance.

To optimize this situation and improve the results, it would be advantageous to first expand the work area setting up more sensors on a larger surface. In addition, it could be useful to measure, simultaneously, other atmospheric factors such as wind speed and direction or humidity, among others.

The Deep Learning technology applied to this study shows that it is an effective and efficient method for short-term solar radiation forecasting on a given surface. Despite its limited dimensions, this work makes us believe that we do not need more expensive and / or complex technologies to perform this task; it is enough to have a wide network of sensors and an optimum model of Deep Learning to obtain accurate enough results.

Lista De Términos

Best linear unbiased estimator (BLUE)

Deep Learning (DL)

Inteligencia Artificial (IA)

Internet of things (IOT)

Machine Learning (ML)

Machine Learning (ML)

Mean Squared Error (MSE)

Ordinary *Kriging* (OK)

Python Data Analysis Library (Pandas)

Rectified Linear Unit (ReLU)

Redes Neuronales artificiales (RNA)

Redes Neuronales Convolucionales (CNN)

Universal Kriging (UK)

REFERENCIAS

Convolution. (2013). https://www.cs.cornell.edu/courses/cs1114/2013sp/sections/S06_convolution.pdf

Cressie, N. (1990). The origins of kriging. *Mathematical Geology*, *22*(3), 239–252. https://doi.org/10.1007/BF00889887

El Naqa, I., & Murphy, M. J. (2015). What Is Machine Learning? In *Machine Learning in Radiation Oncology* (pp. 3–11). https://doi.org/10.1007/978-3-319-18305-3_1

I, E. D., González-Matterson, M.-L., Evans, J. M., & Zamora i Mestre, J.-L. (2017). *Simulaciones numéricas y con modelo a escala en cielo artificial. Estudio de caso con luz cenital: Instituto Nacional de Educación Física de Cataluña, Ciudad de Barcelona*. Retrieved from https://ri.conicet.gov.ar/handle/11336/30082

Javier, F., & Moreno, S. (n.d.). *Desarrollo E Implementación De Una Estrategia De Despliegue Y Mantenimiento De Red De Nodos Sensores*. From https://eprints.ucm.es/49466/1/Memoria TFM - Francisco Javier Sánchez Moreno - 48519821X.pdf

Kriging | Columbia University Mailman School of Public Health. (n.d.). From https://www.mailman.columbia.edu/research/population-health-methods/kriging

Lamigueiro, O. P. (2011). *Energía Solar Fotovoltaica*. Retrieved from https://www.researchgate.net/publication/249012821

Larranaga, Pedro & Inza, Iñaki & Moujahid, A. (2019). *Tema 8. Redes Neuronales*. Retrieved from http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf

Lichtenstern, A. (2013). *Kriging methods in spatial statistics*. Retrieved from https://mediatum.ub.tum.de/doc/1173364/1173364.pdf

Liu, H. (2009). *Taylor Kriging Metamodeling for Simulation Interpolation, Sensitivity Analysis And Optimization*. Retrieved from https://pdfs.semanticscholar.org/d3ce/affcd8a78f785bb0388138c59b2f2685defc.pdf

M. Pasquevich, D. (2016). *La Creciente Demanda Mundial de Energía Frente a Los Riesgos Ambientales*. Retrieved from http://www.cab.cnea.gov.ar/ieds/images/extras/medios/2011/aapc_la_creciente_demanda_energ_frente_riesgos_amb.pdf

Malvic, T., & Balic, D. (2009). Linearity and Lagrange Linear Multiplicator in the Equations of Ordinary Kriging. *Nafta: Exploration, Production, Processing, Petrochemistry*, *60*(1), 31–43. Retrieved from https://core.ac.uk/download/pdf/14418082.pdf

María, A., Ávila, M., Domínguez, J., Javier, B., & Puebla, G. (2014). Universidad Complutense De Madrid, Facultad de Geografía e Historia Trabajo Final. Retrieved from https://eprints.ucm.es/25543/1/TFM_ANA_M_MARTIN_AVILA.pdf

Martín, L. (2012). *Análisis y predicción de series temporales de irradiancia solar global mediante modelos estadísticos* (Universidad Complutense De Madrid). Retrieved from https://eprints.ucm.es/16446/1/T33865.pdf

Martin, L., Zarzalejo, L. F., Polo, J., Ramirez, L., & Espinar, B. (2006). Prediccion de la irradiancia global diaria a partir de imagenes de satelite. *XIII Congresso Iberico e VIII Ibero-Americano de Energia Solar*. Retrieved from https://www.researchgate.net/publication/290946420_PREDICCION_DE_LA_IRRADIANCIA_SOLAR_DIARIA_A_PARTIR_DE_IMAGENES_DE_SATELITE_MEDIANTE_TECNICAS_ESTADISTICAS

Mesić, I., & Kis. (2016). Comparison of Ordinary and Universal Kriging interpolation techniques on a depth variable (a case of linear spatial trend), case study of the Šandrovac Field. *The Mining-Geology-Petroleum Engineering Bulletin*, 41–58. https://doi.org/10.17794/rgn.2016.2.4

Ovando, G., Bocco, M., & Sayago, S. (2005). *Redes Neuronales Para Modelar Predicción De Heladas*. *Agricultura Técnica*, *65*(1), 65–73. https://doi.org/10.4067/S0365-28072005000100007

Qué es Machine Learning, cómo funciona y a qué se aplica | APD. (n.d.). Retrieved May 9, 2019, from https://www.apd.es/que-es-machine-learning/

¿Qué es Machine Learning? (n.d.). From https://cleverdata.io/que-es-machine-learning-big-data/

Sayago, S., Bocco, M., Ovando, G., & Willington, E. (2011). *Radiación Solar Horaria: Modelos De Estimación A Partir De Variables Meteorológicas Básicas*. Retrieved from https://www.researchgate.net/publication/264850041

Understanding Neural Networks. From neuron to RNN, CNN, and Deep Learning. (n.d.). From https://towardsdatascience.com/understanding-neural-networks-from-neuron-to-rnn-cnn-and-deep-learning-cd88e90e0a90

Vladimir, F., & Corea, G. (2014). *Predicción espacio-temporal de la irradiancia solar global a corto plazo en España mediante geoestadística y redes neuronales artificiales* (Tesis Doctoral), *Madrid, España, diciembre de 2014* (Los métodos de predicción se pueden clasificar en métodos numéricos de predicción). Retrieved from http://comisiones.ipgh.org/CARTOGRAFIA/Premio/Tesis_PhD_2018/IPGH_Tesis_Completa_Vladimir_Gutierrez.pdf

Wackernagel, H. (2013). *Kriging methods*. Retrieved from http://hans.wackernagel.free.fr

Anónimo. (n.d.). *La Radiación Solar*. Retrieved from http://bibing.us.es/proyectos/abreproy/5206/fichero/2.LA+RADIACIÓN+SOLAR.pdf