Imagen que contiene Texto

Descripción generada automáticamente

Comparativa de algoritmos para energías renovables basados en Deep Learning

Héctor A. Moreno Martín

Máster Universitario de Ingeniería Informática

Universidad Pablo de Olavide

Tutores: Francisco Martínez Álvarez y Alicia Troncoso Lora

Sevilla, 24 de febrero de 2024

# Resumen *(para publicaciones profesionales)*

Un párrafo, sin sangría. Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto Escribe aquí tu texto.

*Palabras clave:* primera línea con sangría, Escribe, aquí, tu texto

Índice

**No se encontraron entradas de tabla de contenido.**

# 

# Introducción

## 1.1 Descripción del problema

Como propuesta de los profesores Dr. Francisco Martínez Álvarez y Dra. Alicia Troncoso Lora, se llevará a cabo la investigación del funcionamiento de distintos algoritmos de aprendizaje profundo, y su posterior comparación, aplicados a datos medioambientales.

En concreto, se tratarán con datos de energía fotovoltaica, tanto su predicción de manera aislada, como con información posiblemente relevante como datos de temperatura, humedad ambiental, horas de sol o velocidad del viento.

Para realizar la comparación, se han seleccionado tres algoritmos, definidos para operar con datos temporales, como es este caso. Dos de ellos, dentro de la categoría de Redes Neuronales Recurrentes (o RNN, por su traducción al inglés, Recurrent Neural Networks): **Long-Short Term Memory (LSTM)**, y **Gated Recurrent Unit (GRU)**. Por otra parte, el último de los tres algoritmos entra en la categoría de Redes Neuronales Convolucionales (CNN): **Temporal Convolutional Network** (**TCN** en adelante).

En este documento, se hará una introducción teórica a los algoritmos anteriormente mencionados, y las variables de las redes (o hiperparámetros) que se optimizarán para buscar los mejores modelos posibles. Además, se pasarán a explicar los datos usados para la experimentación, así como las comparaciones que se podrán llevar a cabo para, posteriormente, poder sacar conclusiones claras.

## 1.2 Motivación

## 1.3 Aportaciones al problema

# Trabajos relacionados

## 2.1 Métodos de aprendizaje no profundos

Los métodos no profundos de aprendizaje para el pronóstico de series temporales se pueden dividir en dos grupos: técnicas estadísticas clásicas y técnicas de minería de datos (Martínez-Álvarez, Troncoso, Asencio-Cortés y Riquelme, 2015). En cuanto al primer grupo (métodos estadísticos), el promedio móvil integrado autoregresivo y el suavizado exponencial han sido los métodos más populares para predecir series temporales de energía solar fotovoltaica (Dong, Yang, Reindl y Walsh, 2015; Pedro y Coimbra, 2012). Con respecto al segundo grupo (métodos de minería de datos), las redes neuronales, las máquinas de vectores de soporte (SVM) y los k vecinos más cercanos se han aplicado recientemente a datos solares fotovoltaicos. Por ejemplo, Barbieri et al. (Barbieri, Rajakaruna y Ghosh, 2017) presentaron una visión general de métodos para el pronóstico muy a corto plazo de la energía solar fotovoltaica con modelado de nubes. Encontraron que pronosticar la irradiancia y la temperatura de la celda eran los mejores enfoques para predecir las fluctuaciones de potencia fotovoltaica debido a la cobertura de nubes, y que una combinación de imágenes satelitales y del cielo condujo a los mejores resultados para el pronóstico a muy corto plazo. Una red neuronal, optimizada con un algoritmo genético para pronosticar la potencia fotovoltaica intra-hora, fue propuesta en Chu et al. (2015). Se propuso un enfoque basado en agrupamientos en función de las características meteorológicas en Wang, Koprinska y Rana (2017) y Zhang et al. (2018). Un documento de revisión sobre metodologías de pronóstico para el pronóstico de energía solar se presentó en Wan et al. (2015). Se estudiaron pronósticos de intervalo utilizando SVM en Rana, Koprinska y Agelidis (2015); estos tipos de pronósticos se consideraron adecuados para la naturaleza altamente variable de los datos solares. Se propuso un método de pronóstico basado en los datos meteorológicos y de energía de los días anteriores y el pronóstico meteorológico para el día siguiente en Z. Wang y Koprinska (2017) para la predicción de la potencia fotovoltaica un día antes. Brecl y Topic (2018) propusieron un enfoque que utiliza solo pronósticos meteorológicos comunes, sin información de irradiancia solar, obteniendo resultados satisfactorios.

En los últimos años, varios estudios en el pronóstico de series temporales se han centrado en crear conjuntos de modelos de predicción. Los conjuntos combinan las predicciones de varios modelos de pronóstico y se ha demostrado que son muy competitivos y más precisos que los modelos de pronóstico individuales en Cerqueira, Torgo, Pinto y Soares (2017), Koprinska, Rana, Troncoso y Martínez-Álvarez (2013) y Oliveira y Torgo (2015), incluido el pronóstico de la potencia fotovoltaica (Z. Wang et al. (2017)). Otro método de conjunto fue propuesto por Thorey, Chaussin y Mallet (2018) - un método de aprendizaje en línea que genera una combinación ponderada de pronósticos de potencia fotovoltaica para plantas fotovoltaicas ubicadas en Francia; esta técnica se utilizó para predecir la energía solar hasta 6 días por adelantado.

## 2.2 Métodos de aprendizaje profundo

Los métodos de aprendizaje profundo han suscitado mucho interés en los últimos años debido a sus excelentes resultados, especialmente en tareas de reconocimiento de imágenes y voz (Hinton et al., 2012; Krizhevsky, Sutskever y Hinton, 2012; LeCun, Bengio y Hinton, 2015). Para encuestas sobre arquitecturas y aplicaciones de aprendizaje profundo, consulte a Kamilaris y Prenafeta-Boldú (2018), Mohammadi, Al-Fuqaha, Sorour y Guizani (2018), y Pouyanfar et al. (2018). Algunos estudios recientes han aplicado métodos de aprendizaje profundo a tareas de pronóstico, incluidas las series temporales relacionadas con la energía. Por ejemplo, Binkowski, Marti y Donnat (2017) aplicaron redes neuronales convolucionales (CNN) y redes de memoria a corto y largo plazo (LSTM) para datos de consumo financiero y eléctrico con resultados prometedores. Las redes LSTM también se aplicaron para pronósticos de calidad del aire (Zhou, Chang, Chang, Kao y Wang, 2019) y predicción de temperatura interior (Xu, Chen, Wang, Guo y Yuan, 2019), y las CNN se utilizaron para la predicción de lluvias a corto plazo (Qiu et al., 2017).

Torres, Fernández, Troncoso y Martínez-Álvarez (2017) desarrollaron una red neuronal de retroalimentación profunda para el pronóstico de la demanda eléctrica. El método se utilizó para predecir grandes series temporales de datos de consumo de electricidad española durante 10 años, con una frecuencia de muestreo de 10 minutos. En Coelho et al. (2017), se aplicó un modelo de aprendizaje profundo para el pronóstico de la demanda energética doméstica, utilizando una arquitectura paralela de GPU para un procesamiento y entrenamiento de modelos rápidos. Se presentó un modelo de pronóstico de aprendizaje profundo para una planta fotovoltaica conectada con un sistema de gestión de energía renovable en Lee, Lee y Kim (2017). Neo, Teo, Woo, Logenthiran y Sharma (2017) presentaron una aplicación de Deep Belief NN para prever la energía solar fotovoltaica.

En Koprinska, Wu y Wang (2018), se utilizaron CNN para el pronóstico de la demanda eléctrica y la energía solar y se demostró que funcionan de manera similar a las redes neuronales de retroalimentación con una capa oculta y superan a las redes LSTM. En Wang et al. (2017), se aplicó un método híbrido basado en transformadas de onda y CNN para el pronóstico de la energía fotovoltaica. La transformada de onda se utilizó para descomponer los datos originales de series temporales en varias series temporales con diferentes frecuencias; luego se utilizaron las CNN para extraer características de cada serie temporal y, finalmente, se aplicó un modelo probabilístico para predecir cada serie por separado. En Yuchi, Gergely y Brandt (2018), se utilizaron CNN para correlacionar la salida fotovoltaica con imágenes contemporáneas del cielo y pronosticar la energía fotovoltaica. También se evaluó el efecto de las diferentes CNN y parámetros de imagen en la precisión. Además, se ha demostrado que las redes neuronales recurrentes profundas (RNN) ofrecen resultados prometedores para predecir la energía fotovoltaica en Abdel-Nasser y Mahmoud (2017). Alzahrani, Shamsi, Dagli y Ferdowsi (2017) utilizaron una RNN para pronosticar la irradiancia solar y compararon su rendimiento con varios métodos comúnmente utilizados como SVR y redes neuronales de retroalimentación. Después de una amplia revisión de la literatura, hasta donde llega nuestro conocimiento, concluimos que, aunque ha habido estudios previos sobre el pronóstico de la energía solar utilizando diferentes tipos de técnicas de aprendizaje profundo, ninguno de ellos trata con series temporales de grandes datos. En este artículo, abordamos esta brecha proponiendo un algoritmo para prever grandes datos solares utilizando aprendizaje profundo y evaluando su rendimiento en múltiples fuentes de datos.

# Metodología

## 3.1 Propuesta de trabajo

Para llevar a cabo el estudio comparativo de los algoritmos de aprendizaje profundo ya mencionados (LSTM, GRU y TCN), aplicados a datos de energías renovables, tanto la predicción de datos de energía fotovoltaica de forma aislada, como junto a datos de clima, se van a definir, a continuación, ciertos pasos, explicando cada uno de ellos de forma detallada y extensa. Dichos pasos son principalmente los realizados típicamente en un estudio como el que nos ocupa. Como ayuda visual, se muestra un diagrama de flujo de estos pasos, previo a su explicación detallada.

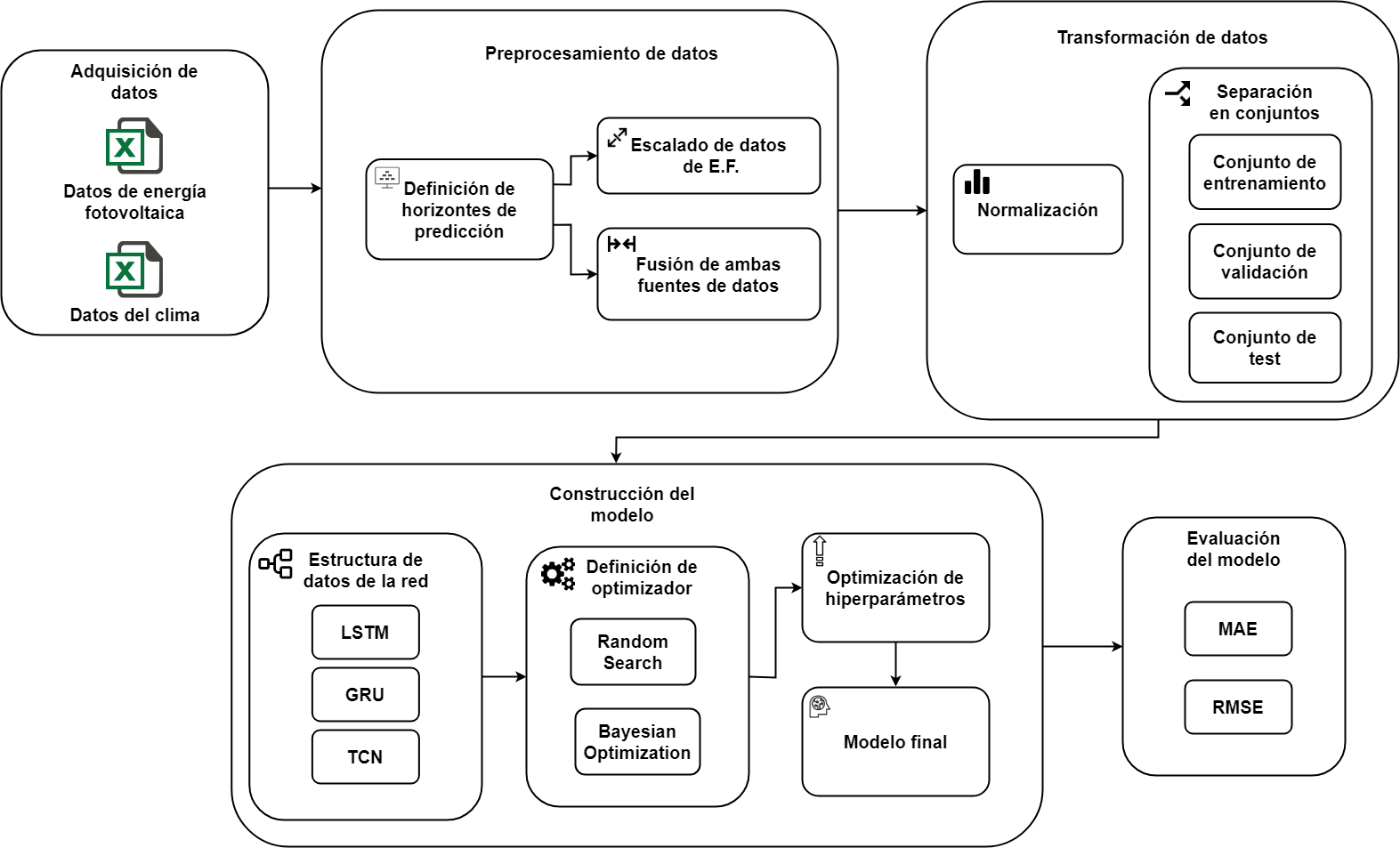


Figura 1: Modelo de flujo

### Adquisición de los datos:

### Preprocesamiento de los datos:

### Transformación de los datos:

Una vez ya con los datos preprocesados, se llevará a cabo una normalización de estos, con el objetivo de utilizar una misma escala para todas las variables que se usan, lo cual, aparte de reducir el riesgo de overfitting, ayuda a que todas tengan un peso similar de partida a la hora de construir los modelos predictivos.

Tras esto, se tendrá que definir un modus operandi para dividir los datos en tres conjuntos: entrenamiento, validación, y prueba o evaluación.

* Conjunto de entrenamiento (Training set): grupo de datos usado para entrenar el modelo predictivo. En nuestro caso, será un entrenamiento supervisado, ya que tendremos los datos de los días siguientes como información útil para ajustar mejor las predicciones.
* Conjunto de validación (Validation set): grupo de datos de soporte al modelo. Este subconjunto es usado para el ajuste de los hiperparámetros en cada iteración del entrenamiento, y también para evaluar el desempeño del modelo según se va entrenando, ayudando también a evitar el sobreajuste al conjunto de entrenamiento.
* Conjunto de prueba (Test set): subconjunto de datos que se utiliza para evaluar el rendimiento final del modelo después de que se haya entrenado y validado completamente. Es importante que el conjunto de prueba sea independiente del conjunto de entrenamiento y del conjunto de validación para obtener una evaluación imparcial del rendimiento del modelo. Normalmente, este conjunto se mantiene oculto al modelo durante todo el proceso de desarrollo hasta que se complete el entrenamiento y la validación.

### Construcción del modelo:

### Evaluación del modelo:

Una vez ya se ha construido el modelo predictivo, se lleva a cabo la evaluación del mismo, haciendo uso del subconjunto de test, que ya hemos explicado en puntos anteriores. Con esto, se sacan métricas de error interesantes, como son el Error Absoluto Medio (MAE en adelante, por su definición en inglés, Mean Absolute Error), y la Raíz del Error Cuadrático Medio (RMSE, por su definición en inglés, Root Mean Squared Error). Ambas métricas, quedarán definidas y con su uso justificado más en profundidad en su propio apartado (Ver 4.2 Métricas de error).

Aparte de las métricas, con las que podremos evaluar la eficacia de los modelos construidos, también podremos ver la eficiencia computacional a la hora del entrenamiento, a través de la obtención del tiempo de ejecución que se ha necesitado para la consecución de dichos modelos. Para ello, se deberá partir de tener parámetros lo más parecido posibles para cada una de las técnicas usadas.

Es necesario destacar, que toda la experimentación se llevará a cabo usando potencia computacional gráfica, o en otras palabras, no se ejecutará haciendo uso de memoria RAM, sino haciendo uso de GPU. Sin embargo, también llevará a cabo, a modo de recogida de información, algunas ejecuciones haciendo uso de memoria RAM, con el objetivo de poder comparar la potencia de ejecución de ambas opciones.

## 3.2 Fundamentos teóricos

### Tipos de algoritmos usados:

#### LSTM: Long-Short Term Memory.

El Long Short-Term Memory (LSTM) es un tipo de red neuronal recurrente (RNN) diseñado específicamente para modelar datos secuenciales con dependencias de largo alcance. A diferencia de las RNN tradicionales, las redes LSTM están equipadas con células de memoria especializadas que les permiten retener información sobre secuencias extensas, lo que las hace muy eficaces para tareas como la previsión de series temporales (precisamente nuestro caso) o el procesamiento de voz.

Cada una de estas células contiene tres “puertas”: la puerta de entrada, la puerta de olvido y la puerta de salida. Estas puertas regulan el flujo de información que entra y sale de la célula, lo que permite a la red recordar u olvidar información de forma selectiva en función de su relevancia en cada contexto. Además, las redes LSTM incorporan una célula de estado (cell state) que sirve, digamos, de “cinta transportadora”, llevando la información a través de los distintos timesteps y facilitando el aprendizaje de las dependencias a largo plazo.

Al igual que las RNN convencionales, las redes LSTM aprenden a actualizar sus parámetros mediante retropropagación y descenso gradiente, ajustando los pesos de las conexiones entre neuronas para minimizar la diferencia entre las salidas previstas y las reales, es decir, para realizar mejores predicciones.

A continuación, se muestra una figura representativa de la arquitectura de una neurona LSTM. No entraremos mucho más en detalle, sirva esto para esta técnica, como para las dos restantes, ya que no es el objetivo de este proyecto explicar su funcionamiento. Si se desea profundizar más en los conceptos teóricos de estas técnicas, puede consultar 6.1 Enlaces de interés

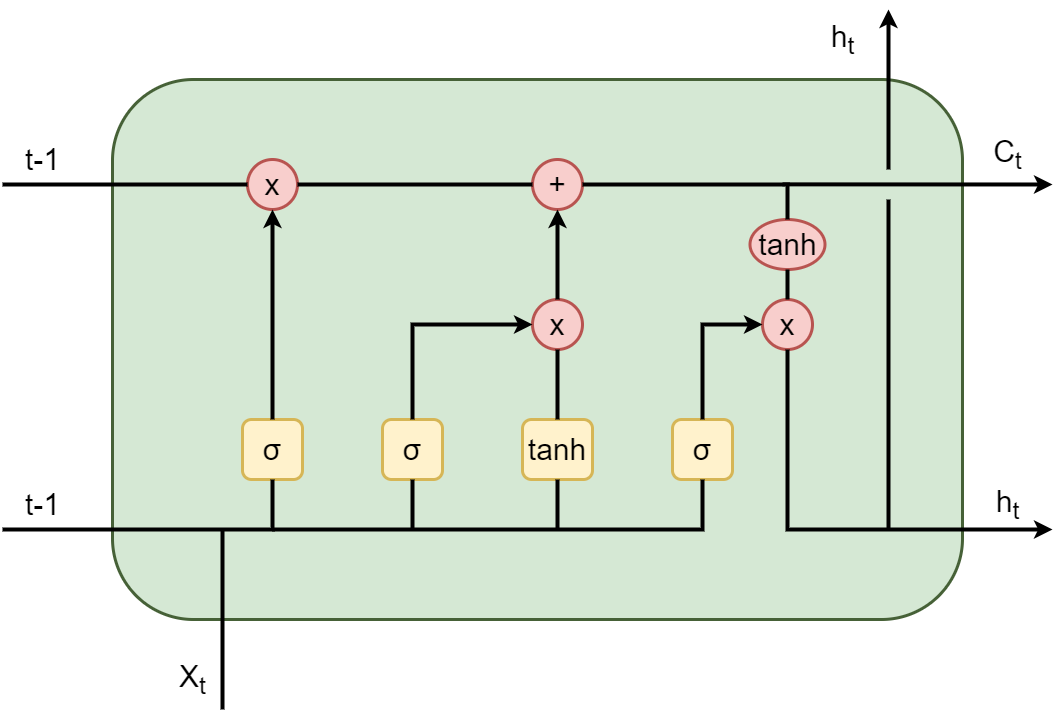


Figura 2: Neurona de una red LSTM

#### GRU: Gated Recurrent Unit.

Las redes neuronales GRU (Gated Recurrent Unit) representan una variante de las arquitecturas de redes neuronales recurrentes (RNN) diseñadas para modelar datos secuenciales al tiempo que mitigan algunos de los problemas que presentan las RNN tradicionales, como el desvanecimiento de gradientes durante el entrenamiento. Al igual que las redes LSTM, las GRU son especialmente eficaces para tareas que implican datos secuenciales, como el procesamiento del lenguaje natural, el análisis de series temporales y el reconocimiento del habla.

La estructura fundamental de una red GRU consiste en unidades recurrentes o células que mantienen estados ocultos que representan la memoria de la red. Las GRU simplifican la arquitectura en comparación con las LSTM combinando las puertas de entrada y olvido en una única puerta de actualización, y fusionando el estado de la célula y el estado oculto, lo que se traduce en un menor número de parámetros y una menor complejidad computacional.

Dentro de cada célula GRU, la puerta de actualización controla el flujo de información, determinando cuánta información pasada debe retenerse y cuánta información nueva debe incorporarse. Además, la puerta de reinicio modula la influencia de la información pasada sobre el estado actual, lo que permite a la red actualizar adaptativamente su memoria en función de la secuencia de entrada.

Al igual que se ha explicado con las redes LSTM, las redes GRU aprenden a actualizar sus parámetros mediante retropropagación y descenso del gradiente, entrenando así los modelos para realizar mejores predicciones.

A continuación, al igual que se hizo con las redes LSTM, se presenta la arquitectura de una neurona de una red GRU:

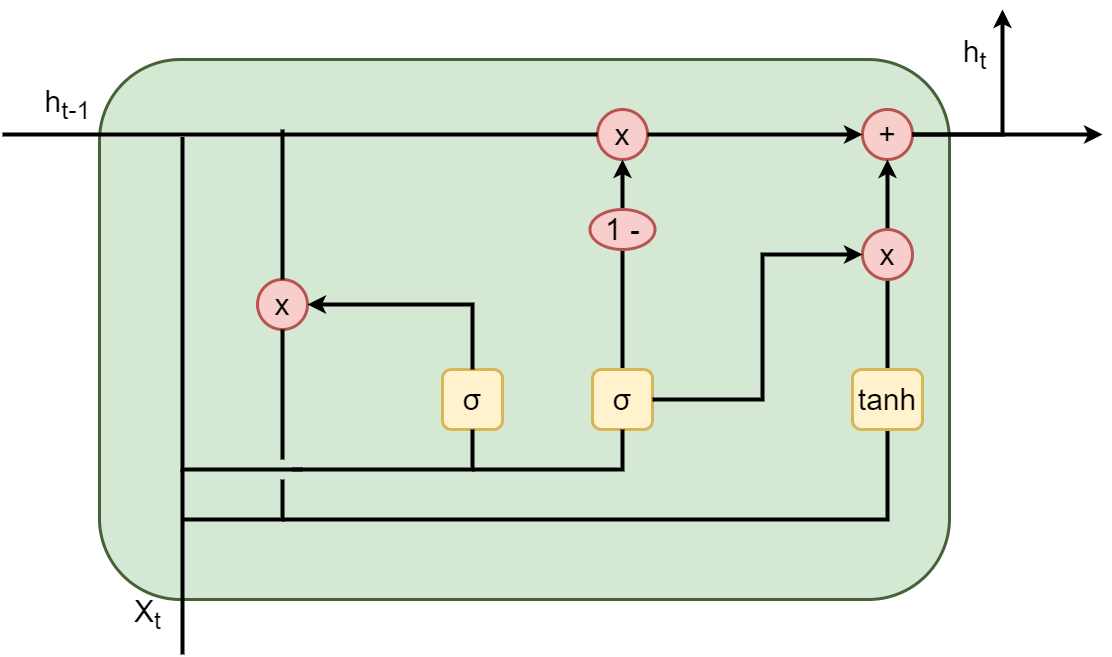


Figura 3: Neurona de una red GRU

#### TCN: Temporal Convolutional Network.

Las redes TCN representan una clase de arquitecturas de redes neuronales diseñadas específicamente para modelar datos secuenciales con dependencias de largo alcance, al igual que las LSTM y GRU, pero estando fuera de la categoría de las RNN, ya que, al fin y al cabo, es un tipo de red convolucional, o CNN. Son una alternativa a tener en cuenta frente a las mencionadas al aprovechar las operaciones convolucionales para capturar patrones temporales en datos secuenciales de forma más eficiente.

Lo que caracteriza a las redes TCN es el uso de convoluciones dilatadas unidimensionales, que permiten a la red ampliar su campo receptivo sin aumentar significativamente el número de parámetros. Esto permite captar dependencias de largo alcance en datos secuenciales manteniendo la eficiencia computacional, lo que las hace idóneas para tareas como la previsión de series temporales, como es nuestro caso.

En una arquitectura TCN, múltiples capas de estas convoluciones dilatadas que se han nombrado se apilan secuencialmente, y cada capa aumenta el campo receptivo y captura características cada vez más abstractas de la secuencia de entrada. Además, las redes TCN pueden incorporar técnicas como conexiones residuales y conexiones de salto para facilitar el flujo de gradiente y aliviar el problema del desvanecimiento de este, que suele darse en las redes profundas, y que ya ha sido nombrado anteriormente. Tal y como se ha especificado en los otros tipos de redes neuronales, las TCN también actualizan sus parámetros mediante retropropagación y descenso de gradiente, ajustando los pesos correspondientes en cada iteración del entrenamiento del modelo, intentando así reducir el error y, por tanto, lograr un mejor modelo predictivo.

A continuación, se muestra un ejemplo de arquitectura de una red TCN. Se introducen nombres como “kernel\_size” o “dilation”, que se explicarán específicamente en 4.3 Experimentación de los datos.

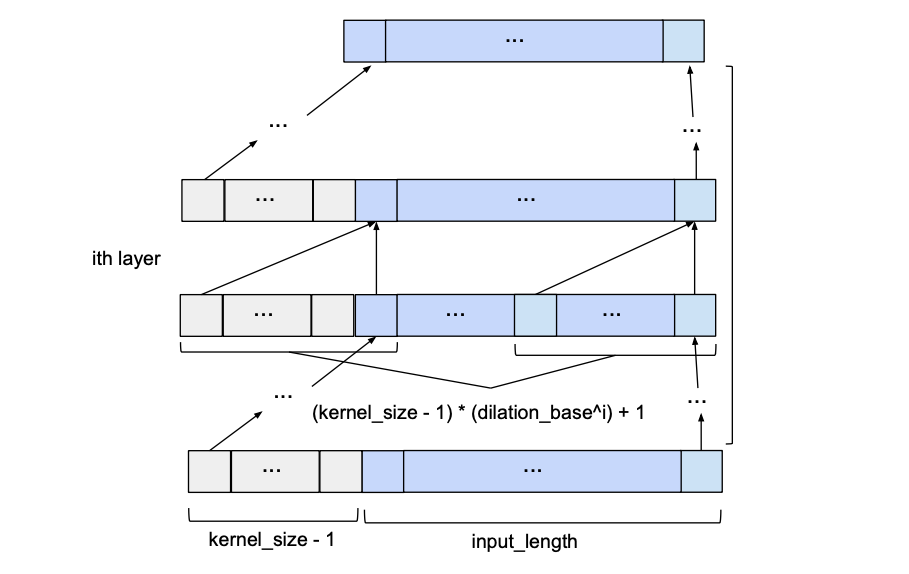


Figura 4: Arquitectura de una red TCN

### Tipos de optimizadores usados:

#### Random Search:

La optimización por búsqueda aleatoria es una técnica utilizada para buscar el mejor conjunto de

hiperparámetros para un modelo de aprendizaje automático mediante la selección aleatoria de

combinaciones a partir de un espacio de búsqueda predefinido. En el contexto del aprendizaje

automático, los hiperparámetros son parámetros que se establecen antes del proceso de

entrenamiento y no pueden aprenderse directamente de los datos.

A continuación, se explica cómo funciona normalmente la optimización por búsqueda aleatoria:

Definir el espacio de búsqueda: El espacio de búsqueda incluye todas las combinaciones posibles de hiperparámetros que desea explorar. Por ejemplo, en una red neuronal, los hiperparámetros podrían incluir la tasa de aprendizaje, el tamaño del lote, el número de capas, etc.

Muestreo aleatorio de hiperparámetros: En lugar de probar exhaustivamente todas las combinaciones posibles, la búsqueda aleatoria selecciona al azar una combinación de hiperparámetros del espacio de búsqueda. La muestra puede ser uniforme o utilizar otra distribución.

Entrenar y evaluar el modelo: Una vez elegido un conjunto de hiperparámetros, el modelo se entrena con los datos de entrenamiento y se evalúa en un conjunto de validación utilizando esos hiperparámetros.

Repetir: Este proceso se repite durante un número fijo de iteraciones o hasta que se cumple un determinado criterio de parada.

Seleccionar los mejores hiperparámetros: Una vez completadas todas las iteraciones, se seleccionan los hiperparámetros que dan como resultado el mejor rendimiento en el conjunto de validación.

La optimización por búsqueda aleatoria es beneficiosa porque no requiere tantos recursos computacionales como otros métodos, como la búsqueda en cuadrícula, que evalúa exhaustivamente todas las combinaciones posibles de hiperparámetros. Además, la búsqueda aleatoria a menudo puede encontrar buenas configuraciones de hiperparámetros de manera más eficiente, especialmente en espacios de alta dimensión donde la búsqueda exhaustiva se vuelve poco práctica.

#### Bayesian Optimization:

La optimización bayesiana es una técnica avanzada utilizada para buscar los mejores conjuntos de hiperparámetros para modelos de aprendizaje automático mediante la construcción de un modelo probabilístico de la función objetivo y su posterior optimización. A continuación, se explica cómo funciona típicamente la optimización bayesiana:

Definir el Espacio de Búsqueda: Al igual que con la búsqueda aleatoria, se define un espacio de búsqueda que incluye todas las combinaciones posibles de hiperparámetros que se desean explorar.

Construir un Modelo Probabilístico: La optimización bayesiana construye un modelo probabilístico (generalmente una distribución gaussiana) que estima cómo se relacionan los hiperparámetros con el rendimiento del modelo. Inicialmente, este modelo es incierto y se basa en pocas observaciones.

Seleccionar Hiperparámetros para Evaluar: Utilizando el modelo probabilístico, se seleccionan los siguientes hiperparámetros para evaluar en función de una estrategia de selección (por ejemplo, elegir los hiperparámetros con la mayor incertidumbre).

Evaluar el Modelo: Se entrena y evalúa el modelo utilizando los hiperparámetros seleccionados. Se recopila el rendimiento del modelo en función de estos hiperparámetros.

Actualizar el Modelo Probabilístico: Después de evaluar el modelo, el modelo probabilístico se actualiza con la nueva información. Esto implica ajustar la distribución para reflejar el rendimiento observado del modelo y reducir la incertidumbre sobre cómo se relacionan los hiperparámetros con el rendimiento.

Repetir: Se repiten los pasos 3 a 5 hasta que se alcance un cierto criterio de parada, como un número máximo de iteraciones o una mejora mínima en el rendimiento.

Seleccionar los Mejores Hiperparámetros: Al final del proceso, se seleccionan los hiperparámetros que maximizan la función objetivo estimada por el modelo probabilístico.

La optimización bayesiana es útil porque aprovecha la información recopilada en las evaluaciones anteriores para dirigir la búsqueda hacia regiones prometedoras del espacio de hiperparámetros. Esto la hace más eficiente que la búsqueda aleatoria, especialmente en espacios de búsqueda de alta dimensionalidad o cuando el costo de evaluar cada conjunto de hiperparámetros es alto.

# Resultados

## 4.1 Descripción de los datos

Se utilizan datos de dos fuentes: energía fotovoltaica, por un lado, y datos de clima por otro. Los datasets utilizados muestran 2 años de datos: desde el 1 de enero de 2015 al 31 de diciembre de 2016. Estos datos son los usados también por Wang et al. (2017) y Torres Maldonado et al. (2019).

El conjunto de datos de energía fotovoltaica es generado a través de la recopilación de datos de una planta fotovoltaica en una azotea ubicada en la Universidad de Queensland en Brisbane, Australia, y los datos están disponibles públicamente (<http://www.uq.edu.au/solarenergy/>).

Para cada día, solo seleccionamos los datos durante un período de 10 horas de luz solar, desde las 7:00 hasta las 17:00. Los datos originales de energía fotovoltaica se midieron en intervalos de 1 minuto y se agregaron a intervalos de 30 minutos tomando el valor promedio de dicho intervalo de tiempo. Como resultado, este conjunto de datos contiene 14,620 puntos de datos: (365 + 366) días × 20 mediciones por día.

En cuanto a los conjuntos de datos de información climática, se obtuvieron del Australian Bureau of Meteorology (<http://www.bom.gov.au/>). Para cada día, recopilamos 14 variables meteorológicas, descritas en la siguiente tabla (Tabla 1). En total, este conjunto de datos contiene 731 días y 14 mediciones por día, lo que resulta en 10,234 puntos de datos.

|  |  |  |
| --- | --- | --- |
| ID | Abreviación | Descripción |
| 1 | DMIN | Temperatura mínima diaria |
| 2 | DMAX | Temperatura máxima diaria |
| 3 | DRAIN | Lluvia diaria |
| 4 | DSUN | Horas de sol diaria |
| 5 | DMAXWIND | Máximo diario de velocidad del viento |
| 6 | TEMP9 | Temperatura a las 9:00 |
| 7 | HUM9 | Humedad relativa a las 9:00 |
| 8 | CLOUD9 | Nivel de nubes a las 9:00 |
| 9 | WIND9 | Velocidad del viento a las 9:00 |
| 10 | TEMP3 | Temperatura a las 15:00 |
| 11 | HUM3 | Humedad relativa a las 15:00 |
| 12 | CLOUD3 | Nivel de nubes a las 15:00 |
| 13 | WIND3 | Velocidad del viento a las 15:00 |
| 14 | DSOLARIRR | Radiación solar diaria |

Tabla 1: Variables de los datos climáticos

## 4.2 Métricas de error

Tal y como se han introducido en 3.1 Propuesta de trabajo, las métricas elegidas para la evaluación de los modelos construidos son dos: Mean Absolute Error (MAE) y Root Mean Squared Error (RMSE). A continuación, se define y explica la finalidad de cada uno, mostrando su fórmula de cálculo.

* **Mean Absolute Error (MAE):** es la media de las diferencias absolutas entre las predicciones del modelo y los valores reales. Se calcula como la suma de las diferencias absolutas dividida por el número total de muestras.

La fórmula para calcular el MAE es:

Donde:

* es el número de muestras en el conjunto de datos.
* es el valor real.
* es el valor predicho por el modelo para la muestra .
* **Root Mean Squared Error (RMSE):** es la raíz cuadrada de la media de las diferencias cuadráticas entre las predicciones del modelo y los valores reales. Se calcula como la raíz cuadrada de la media de los errores cuadráticos.

La fórmula para calcular el RMSE es:

Donde:

* es el número de muestras en el conjunto de datos.
* es el valor real.
* es el valor predicho por el modelo para la muestra.

En resumen, el MAE proporciona una medida de la magnitud promedio de los errores en las predicciones del modelo, mientras que el RMSE también penaliza más los errores grandes debido a su naturaleza cuadrática, lo que lo hace más sensible a los valores atípicos.

## 4.3 Experimentación de los datos

Partiendo de las dos fuentes de datos que se han explicado en los apartados anteriores, se establecen dos criterios de agrupación para obtener información y sacar conclusiones a partir de ella: la bondad del modelo, o cómo de eficaz es cada modelo realizando las predicciones, la cual se mide y compara a través de las métricas definidas en el apartado anterior, 4.2 Métricas de error; y los tiempos de ejecución. Estos últimos, nos dicen cómo de rápido trabajan las diferentes técnicas para construir los modelos. A su vez, este apartado se divide en dos subapartados: escalabilidad de los datos, y RAM vs GPU.

Para lograr la escalabilidad de los datos, se construyen 7 ficheros de datos solo con datos de energía fotovoltaica, partiendo de los datos originales, y en los demás se repiten los datos N veces hasta 7. En cada fila, tendremos los 20 datos de cada día repetidos N veces, más el horizonte de predicción (H), es decir, las 20 mediciones del día siguiente.

Por otra parte, para llevar a cabo la construcción de los modelos usando los datos climáticos, también tenemos como resultados 7 ficheros de datos. Tal y como se explica en 4.1 Descripción de los datos, tenemos 14 variables, o columnas, representativas de la medición de cada día. En el primer fichero de datos tenemos solo un día de medición por fila como datos para el entrenamiento, en el segundo fichero, dos días por fila; y así sucesivamente, hasta el séptimo fichero de datos, donde por fila tenemos una semana de datos para cada predicción, con lo cual en este último tenemos 14 x 7 = 98 columnas útiles para los modelos.

Es necesario definir también, cómo se dividen los datos, ya que, para construir el modelo, necesitamos tres conjuntos de datos: conjunto de entrenamiento, conjunto de validación, y conjunto de test. Tal y como se define en apartados previos, tenemos datos de dos años: 2015, y 2016. La partición se hace de la siguiente manera: para los conjuntos de entrenamiento y validación, se usan los datos de 2015. Un 70% de estos datos están destinados al conjunto de entrenamiento, y el 30% restante, al conjunto de validación del modelo. Por otro lado, para el conjunto de test, es decir, el conjunto con el que se evalúa cada modelo final generado se hace con los datos al completo de 2016. A la hora de números concretos, los ficheros de 2015 con 365 registros, uno por día, generan un conjunto de entrenamiento de 255 filas, y un conjunto de validación de 110 filas; y por otro lado, su par correspondiente a 2016, generará un conjunto de test con 366 filas.

Se ha optado también por una normalización de los datos. Esta es una práctica muy común en el ámbito del aprendizaje profundo, y que consiste en reconvertir los datos a un rango más pequeño y por ello más sencillo de interpretar, en valores entre 0 y 1. Entre los beneficios que puede aportar la normalización están:

* Una mejora de la convergencia del modelo, ya que al escalar los datos a un rango más pequeño y centrarlos alrededor de cero, se evitan problemas como el desvanecimiento de gradiente, lo que permite una optimización más estable.
* Equilibra las características: en muchos conjuntos de datos, las características pueden tener diferentes escalas y distribuciones. La normalización permite que todas las características contribuyan de manera más equitativa al modelo, evitando que las características con escalas más grandes tengan más peso que otras con escalas más pequeñas.
* Aumenta la generalización: la normalización puede ayudar a mejorar la capacidad de generalización del modelo, y, por tanto, a reducir el riesgo de overfitting. Al estandarizar los datos, se reducen las posibilidades de que el modelo se sobreajuste a características específicas del conjunto de entrenamiento que pueden no ser relevantes para otros conjuntos de datos.
* Facilita la interpretación: los modelos normalizados son más fáciles de interpretar, ya que las características están en la misma escala y es más sencillo comparar sus efectos relativos en la predicción.

A continuación, se presenta la fórmula matemática que se aplica para llevar a cabo la normalización de los datos:

Para llevar a cabo la experimentación, y poder comparar unos resultados con otros, a parte de la ventaja que nos supone la normalización de los datos, también es necesario que tanto el entorno de ejecución como los parámetros que se usen sean lo más parecidos posibles, solo difiriendo en aquellos parámetros (o hiperparámetros) propios de cada tipo de red.

El equipo elegido para llevar a cabo la ejecución de la experimentación tiene las siguientes características:

* Procesador: Intel(R) Core (TM) i7-10750H CPU @ 2.60GHz
* Memoria RAM: 32,0 GB
* Memoria GPU: NVIDIA GeForce RTX 2070 - 8GB

Tal y como se introduce en 3.1 Propuesta de trabajo, toda la experimentación se llevará a cabo haciendo uso de la potencia computacional gráfica. Para ello, es necesario configurar el equipo con las herramientas necesarias para poder ejecutar el proyecto haciendo uso de la GPU. Las herramientas y paquetes que son estrictamente necesarios son los siguientes:

|  |  |  |
| --- | --- | --- |
| Herramienta | Versión | Descripción |
| CUDA | 10.1 | O Compute Unified Device Architecture, plataforma de computación paralela de NVIDIA que permite la ejecución de software haciendo uso de GPU. |
| cuDNN | 6.4.7 | O CUDA Deep Neural Network, biblioteca de primitivas con aceleración de GPU para usar con redes neuronales profundas |
| Python | 3.6.13 | Lenguaje de programación usado para la creación del código necesario para llevar a cabo la experimentación. |
| TensorFlow | 2.1.0 | Biblioteca de código abierto para aprendizaje automático |
| KerasTuner | 1.0.1 | Biblioteca de Python que permite realizar la optimización de hiperparámetros de modelos de aprendizaje automático creados con TensorFlow y Keras. |

Tabla 2: Herramientas necesarias para llevar a cabo la experimentación

Aparte, se usan otros paquetes de Python para la ejecución del código, pero el objetivo de la tabla es que queden recogidos las versiones concretas que se usan ya que, de no ser así, podría haber errores de compatibilidad entre dichas herramientas.

Por otra parte, se deben definir cuáles son los hiperparámetros que vamos a optimizar en cada red, así como los valores que pueden asignarse a dichos hiperparámetros, los cuales pueden ayudar a construir mejores modelos predictivos. Para ello, se procede a definir una lista con dichos hiperparámetros definidos para cada técnica, pero antes, es necesario explicar brevemente una serie de conceptos, usados dentro de este apartado:

* Valor del paso: de cuánto será el salto del valor del hiperparámetro. Por ejemplo, si el valor es 10, el optimizador irá comprobando valores de 10 en 10 para ese hiperparámetro.
* Nodo: equivalente a neurona dentro de la red.
* Dropout: técnica de regularización utilizada para prevenir el overfitting. Durante el entrenamiento, se apaga aleatoriamente, en cada iteración, un porcentaje de los nodos de la capa de la red neuronal donde se defina, lo cual ayuda a la generalización del modelo.
* LSTM:
  + Número de nodos: número de neuronas de la red.
    - Valor mínimo: 10
    - Valor máximo: 50
    - Valor del paso: 10
    - Valor por defecto: 30
  + Valor del Dropout:
    - Valor mínimo: 0
    - Valor máximo: 0.33
    - Valor por defecto: 0.25
  + Ratio de aprendizaje:
    - Valores válidos: 0, 0.01, 0.001, 0.0001
* GRU:
  + Número de nodos: número de neuronas de la red.
    - Valor mínimo: 10
    - Valor máximo: 50
    - Valor del paso: 10
    - Valor por defecto: 30
  + Valor del Dropout:
    - Valor mínimo: 0
    - Valor máximo: 0.33
    - Valor por defecto: 0.25
  + Ratio de aprendizaje:
    - Valores válidos: 0, 0.01, 0.001, 0.0001
* TCN:
  + Número de filtros:
    - Valor mínimo: 32
    - Valor máximo: 128
    - Valor del paso: 32
    - Valor por defecto: 32
  + Número de kernels:
    - Valor mínimo: 1
    - Valor máximo: 5
    - Valor del paso: 1
    - Valor por defecto: 3
  + Dilataciones:
    - Valores válidos: 1, 2, 4, 8, 16
  + Valor del Dropout:
    - Valor mínimo: 0
    - Valor máximo: 0.33
    - Valor por defecto: 0.25
  + Ratio de aprendizaje:
    - Valores válidos: 0, 0.01, 0.001, 0.0001

Aparte de los hiperparámetros definidos, también hay otros parámetros a la hora de ejecutar el entrenamiento que han sido necesarios definir, para que sean iguales para todas las ejecuciones de las 3 técnicas. Estos son:

* Épocas (epochs): una época es una iteración completa a través de todo el conjunto de datos de entrenamiento. Durante una época, el modelo utiliza todos los datos de entrenamiento una vez para actualizar sus pesos y ajustarse mejor a los datos. Después de cada época, el modelo se evalúa utilizando el conjunto de datos de validación, para evaluar su desempeño.
  + Valor elegido: 100.
* Ejecuciones (executions): una ejecución es la acción de entrenamiento del modelo. En otras palabras, este parámetro se poder hacer múltiples ejecuciones sin necesidad de hacerlas manualmente. La librería Keras-Tuner proporciona esta opción, guardando los X mejores modelos de cada ejecución para su posible posterior análisis.
  + Valor elegido: 2
* Intentos (trials): cada trial representa la evaluación de una tupla de hiperparámetros en un modelo. Es otras palabras, con X trials, se probarán X combinaciones de los hiperparámetros definidos.
  + Valor elegido: 40

A la hora de mostrar los resultados, se mostrarán los valores de los mejores modelos conseguidos.

Tal y como se ha introducido en este mismo apartado, como contrapunto, se han ejecutado también de forma “tradicional”, es decir, haciendo uso de memoria RAM, algunos modelos, con el objetivo de comparar tiempos de ejecución entre unos y otros.

## 4.4 Discusión y análisis de los resultados

# Conclusiones

# Referencias

## 6.1 Enlaces de interés

[1] Explicación de las redes LSTM: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

[2] Explicación de las redes GRU: <https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be>

[3] Explicación de las redes TCN: <https://unit8.com/resources/temporal-convolutional-networks-and-forecasting/>

## 6.2 Citas bibliográficas

Abdel-Nasser,M., & Mahmoud, K. (2017). Accurate photovoltaic power forecasting models using deep LSTM-RNN. NeuralComputing and Applications, 1–14.

Alzahrani, A., Shamsi, P., Dagli, C., & Ferdowsi,M. (2017). Solar irradiance forecasting using deep neural networks. Procedia Computer Science, 114, 304–313.

Barbieri, F., Rajakaruna, S., & Ghosh, A. (2017). Very short-term photovoltaic power forecasting with cloud modeling: A review. Renewable and Sustainable Energy Reviews, 75, 242–263.

Binkowski, M., Marti, G., & Donnat, P. (2017). Autoregressive convolutional neural networks for asynchronous time series. In Time Series Workshop at International Conference on Machine Learning (ICML), Stockholm, Sweden.

Brecl, K., & Topic, M. (2018). Photovoltaics (PV) system energy forecast on the basis of the local weather forecast: Problems, uncertainties and solutions. Energies, 11(5), 1143.

Cerqueira, V., Torgo, L., Pinto, F., & Soares, C. (2017). Arbitrated ensemble for time series forecasting. In Proceedings of the European Conference on Machine Learning and Principles of Knowledge Discovery in Databases, Cham, pp. 478–494.

Chu, Y., Urquhart, B., Gohari, S. M. I., Pedro, H. T. C., Kleissl, J., & Coimbra, C. F. M. (2015). Short-term reforecasting of power output from a 48 mwe solar pv plant. Solar Energy, 112, 68–77.

Coelho, I. M., Coelho, V. N., da Luz, E. J. S., Ochi, L. S., Guimarães, F. G., & Rios, E. (2017). A GPU deep learning metaheuristic based model for time series forecasting. Applied Energy, 201, 412–418.

Dong, Z., Yang, D., Reindl, T., & Walsh,W. M. (2015). A novel hybrid approach based on self-organizing maps, support vector regression and particle swarm optimization to forecast solar irradiance. Energy, 82, 570–577.

Ervural, B. C., & Ervural, B. (2018). Improvement of grey prediction models and their usage for energy demand forecasting. Journal of Intelligent & Fuzzy Systems, 24, 2679–2688.

Flannery, T. F., & Sahajwalla, V. (2013). The critical decade: Australia's future: Solar energy: Climate Commission Secretariat, Department of Industry, Innovation, Climate Change, Science, Research and Tertiary Education. http://apo.org.au/sites/default/files/docs/ClimateCommission\_Australias-

Future-Solar-Energy\_2013.pdf

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A., Jaitly, N., ... Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine, 29(6), 82–97.

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. Computers and Electronics in Agriculture, 147, 70–90.

Koprinska, I., Rana, M., & Agelidis, V. G. (2011). Yearly and seasonal models for electricity load forecasting. In International Joint Conference on Neural Networks (IJCNN), San Jose, CA, USA, pp. 1474–1481.

Koprinska, I., Rana, M., Troncoso, A., & Martínez-Álvarez, F. (2013). Combining pattern sequence similarity with neural networks for forecasting electricity demand time series. In Proceedings of the International Joint Conference on Neural Networks, Dallas, TX, USA, pp. 1–8.

Koprinska, I., Wu, D., & Wang, Z. (2018). Convolutional neural networks for energy time series forecasting. In International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, pp. 1–8.

Kostylev, V., & Pavlovski, A. (2011). Solar power forecasting performance—Towards industry standards. In First International Workshop on Integration of Solar Power Into Power Systems, Aarhus, Denmark, pp. 1–11.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, Lake Tahoe, Nevada, pp. 1097–1105.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539

Lee, J., Lee, I., & Kim, S. (2017). Multi-site photovoltaic power generation forecasts based on deep-learning algorithm. In 2017 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, South Korea, pp. 1118–1120.

Livingstone, D. J., Manallack, D. T., & Tetko, I. V. (1997). Data modelling with neural networks: Advantages and limitations. Journal of Computer-Aided Molecular Design, 11, 135–142.

Martínez-Álvarez, F., Troncoso, A., Asencio-Cortés, G., & Riquelme, J. C. (2015). A survey on data mining techniques applied to energy time series forecasting. Energies, 8, 1–32.

Martínez-Álvarez, F., Troncoso, A., Riquelme, J. C., & Aguilar, J. S. (2011). Energy time series forecasting based on pattern sequence similarity. IEEE Transactions on Knowledge and Data Engineering, 23, 1230–1243.

Mohammadi, M., Al-Fuqaha, A., Sorour, S., & Guizani, M. (2018). Deep learning for IoT big data and streaming analytics: A survey. IEEE Communications Surveys Tutorials, 20(4), 2923–2960.

Neo, Y. Q., Teo, T. T., Woo, W. L., Logenthiran, T., & Sharma, A. (2017). Forecasting of photovoltaic power using deep belief network. In Tencon 2017 - 2017 IEEE Region 10 Conference, Penang, Malaysia, pp. 1189–1194.

Oliveira, M., & Torgo, L. (2015). Ensembles for time series forecasting. In Proceedings of the Sixth Asian Conference on Machine Learning, Nha Trang City, Vietnam, pp. 360–370.

Pedro, H. T. C., & Coimbra, C. F. M. (2012). Assessment of forecasting techniques for solar power production with no exogenous inputs. Solar Energy, 86, 2017–2028.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., ... Iyengar, S. S. (2018). A survey on deep learning: Algorithms, techniques, and applications. ACM Computing Surveys, 51(5), 92:1–92:36. https://doi.org/10.1145/3234150

Qiu, M., Zhao, P., Zhang, K., Huang, J., Shi, X., Wang, X., & Chu, W. (2017). A short-term rainfall prediction model using multi-task convolutional neural networks. In 2017 IEEE International Conference on Data Mining (ICDM), New Orleans, LA, USA, pp. 395–404.

Rana, M., Koprinska, I., & Agelidis, V. G. (2015). 2d-interval forecasts for solar power production. Solar Energy, 122, 191–203.

Reikard, G. (2009). Predicting solar radiation at high resolutions: A comparison of time series forecasts. Solar Energy, 83, 342–349.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85–117.

SolarPowerEurope (2016). Global market outlook for solar power / 2016 - 2020.

Thorey, J., Chaussin, C., & Mallet, V. (2018). Ensemble forecast of photovoltaic power with online crps learning. International Journal of Forecasting, 34(4), 762–773.

Torres, J. F., Fernández, A. M., Troncoso, A., & Martínez-Álvarez, F. (2017). Deep learning-based approach for time series forecasting with application to electricity load. In Biomedical Applications Based on Natural and Artificial Computing, Cham, pp. 203–212.

Wan, C., Zhao, J., Song, Y., Xu, Z., Lin, J., & Hu, Z. (2015). Photovoltaic and solar power forecasting for smart grid energy management. CSEE Journal of Power and Energy Systems, 1(1), 38–46.

Wang, Z., & Koprinska, I. (2017). Solar power prediction with data source weighted nearest neighbors. In Proceedings of the International Joint Conference on Neural Networks, Anchorage, AK, USA, pp. 1411–1418.

Wang, Z., Koprinska, I., & Rana, M. (2017). Solar power forecasting using pattern sequences. In Artificial Neural Networks and Machine Learning (ICANN), Cham, pp. 486–494.

Wang, Z., Koprinska, I., & Rana, M. (2017). Solar power prediction using weather type pair patterns. In Proceedings of the International Joint Conference on Neural Networks, Anchorage, AK, USA, pp. 4259–4266.

Wang, H., Yi, H., Peng, J., Wang, G., Liu, Y., Jiang, H., & Liu, W. (2017). Deterministic and probabilistic forecasting of photovoltaic power based on deep convolutional neural network. Energy Conversion and Management, 153, 409–422.

Xu, C., Chen, H., Wang, J., Guo, Y., & Yuan, Y. (2019). Improving prediction performance for indoor temperature in public buildings based on a novel deep learning method. Building and Environment, 148, 128–135.

Yuchi, S., Gergely, S., & Brandt, B. A. R. (2018). Solar pv output prediction from video streams using convolutional neural networks. Energy and Environmental Science, 11, 1811–1818.

Zhang, X., Li, Y., Lu, S., Hamann, H., Hodge, B. S., & Lehman, B. (2018). A solar time-based analog ensemble method for regional solar power forecasting. IEEE Transactions on Sustainable Energy, 10, 268–279.

Zhou, Y., Chang, F., Chang, L., Kao, I., & Wang, Y. (2019). Explore a deep learning multi-output neural network for regional multi-step ahead air quality forecasts. Journal of Cleaner Production, 209, 134–145.