UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

DISEÑO E IMPLEMENTACIÓN DE MODELOS DE TEMPERATURA MEDIANTE ALGORITMOS DE APRENDIZAJE AUTOMÁTICO PARA SISTEMAS DE COMPUTACIÓN EN INMERSIÓN EN HIDRO-FLUORO-OLEFINAS

SERGIO PÉREZ MORILLO

2018

GRADO EN INGENIERÍA DE TECNOLOGÍAS Y SERVICIOS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Título: Diseño e implementación de modelos de temperatura mediante algoritmos de aprendizaje automático para sistemas de computación en inmersión en hidro-fluoro-olefinas.

Autor: D. Sergio Pérez Morillo

Tutor: D. José Manuel Moya Fernández

Departamento: Departamento de Ingeniería Electrónica

MIEMBROS DEL TRIBUNAL

Presidente: D. ……

Vocal: D. ……..

Secretario: D. ……..

Suplente: D. ……..

Los miembros del tribunal arriba nombrados acuerdan otorgar la calificación de: ……

Madrid, a de de 2018

Resumen:

La irrupción del *big data*, la inteligencia artificial, el minado de criptomonedas o los videojuegos *online* masivos, entre otros, han elevado el consumo de datos y con ello la energía consumida globalmente. Los centros de datos, instalaciones dedicadas al manejo de datos de las aplicaciones mencionadas, cada vez ganan más importancia en la actualidad y un uso eficiente de ellos se vuelve crítico para no aumentar su consumo energético más de lo estrictamente necesario ya que a día de hoy suponen el 2% de las emisiones de CO_2 [1].

El consumo de energía en un centro de datos se puede dividir en dos partes principalmente. Una primera relacionada con el propio funcionamiento de los equipos informáticos que engloba el 60% del consumo y una segunda con los sistemas de refrigeración que equivale al 40% restante. En este proyecto se prueba un sistema de refrigeración por inmersión en líquido dieléctrico en dos fases, que potencialmente podría reducir a cero la energía consumida por sistemas de refrigeración en centros de datos. Las pocas empresas e instituciones que han usado este tipo de refrigeración lo han hecho con el líquido NOVEC 7100 de la empresa 3M [7]. Nosotros proponemos el uso del líquido dieléctrico Vertrel XF, nunca antes utilizado para este propósito. Este es un tercio más barato y potencialmente más eficiente, al tener el punto de ebullición más bajo.

La mayoría de los sistema de refrigeración en centros de datos han carecido de un método que les permita un equilibrio entre alcanzar el máximo rendimiento de los equipos y ahorrar en energía consumida por refrigeración. Han existido tanto problemas de sobreenfriamiento, consumiendo más de lo necesario en refrigeración, como de sobrecalentamiento, deteriorando los equipos por falta de ella. Para solucionarlo, proponemos el uso de metaheurísticas que nos permitirán realizar modelos predictivos de temperatura que sean capaces de predecir los cambios de temperatura en los equipos y poder actuar sobre ellos, haciendo un uso óptimo del sistema.

En este trabajo hemos realizado un primer prototipo de este sistema, que consiste en un pequeño recipiente lleno del líquido dieléctrico, en el que sumergimos un *cluster* de equipos, los cuáles están ejecutando una carga de trabajo que hemos diseñado basada en minado de criptomoneda, para poder monitorizar sus datos de temperatura, utilización, etc. Finalmente con el conjunto de datos capturados, realizamos los distintos modelos de predicción de temperatura mencionados, haciendo uso de los algoritmos metaheurísticos por excelencia en la actualidad, las redes neuronales.

Palabras clave:

Inmersión en dos fases, centros de datos, eficiencia energética, sistemas de refrigeración, Vertrel XF, redes neuronales recurrentes, criptomonedas.

Summary:

The rising of big data, artificial intelligence, cryptocurrency mining or online gaming, among others, have increased data consumption and thus the energy globally consumed. Data centers, facilities whose main goal is to handle all this data, are becoming more and more important this recent years. So, being efficient in their management have become crucial to not rise even more the energy consumed by them and their carbon emissions, which represent the 2% of the total emissions on the planet [1].

Energy consumption in data centers could be divided in two main parts. The first one is the energy consumed by their own equipment to process data, this means the 60% of all the energy consumed. The second one is the energy consumed by cooling systems, which consumed the remaining 40%. In this project we propose a two phase immersion cooling system that could potentially reduce the energy consumed by cooling systems in data centers to zero. There are few companies that have implemented a system like this, and the majority of them have used the dielectric liquid NOVEC 7100 from the company 3M [7]. We propose another dielectric liquid, Vertrel XF, which has never been used for this task before. It is a third cheaper than NOVEC and potentially more efficient because its boiling point is lower, which means it could reach more easily a two phase state.

The majority of data centers have lacked a method that allow them to maximize the performance of their equipment while saving energy consumed by their cooling systems. This generates two opposite problems, first, the data center could be overcooled, which means more energy consumed, and then, it could be overheated, leading to worst equipment performance and shortening their useful life. To solve this, we propose the use of metaheuristic algorithms to design predictive models that could predict changes in equipment's temperature, thus changing their workload to make a optimal use of the system.

In this project, we have made a first prototype of this system, which consists on a small container full of dielectric liquid where we introduce a small cluster. This cluster runs a previously designed workload (cryptocurrency mining) to monitor and gather all the data about their temperature, frequency, etc. Finally, we use this dataset to elaborate and implement predictive models based on metaheuristic algorithms, in this case, neural networks.

Keywords:

Two-Phase immersion cooling, data centers, energy efficiency, cooling systems, Vertrel XF, recurrent neural network, cryptocurrency.

Índice de contenido

Índice de figuras

Índice de tablas

1. Introducción

La cantidad de energía consumida por los centros de datos está creciendo exponencialmente. La irrupción del vídeo bajo demanda, los juegos online, la minería de datos o la inteligencia artificial, entre otros, demandan la gestión de cantidades ingentes de datos. Según un artículo en *Energies* escrito por la comisión europea [1], los centros de datos generan actualmente hasta el 2% de las emisiones de CO_2, datos similares al del sector de la aviación. En general, en el sector de las tecnologías e información (IT) hoy día consume globalmente el 7% de la electricidad y se espera que aumente hasta el 13% en 2030. En particular, los centros de datos consumen aproximadamente el 2% de la producción eléctrica mundial [2] y, con la necesidad de crecimiento, se vuelve prioritario el uso eficiente de la energía.

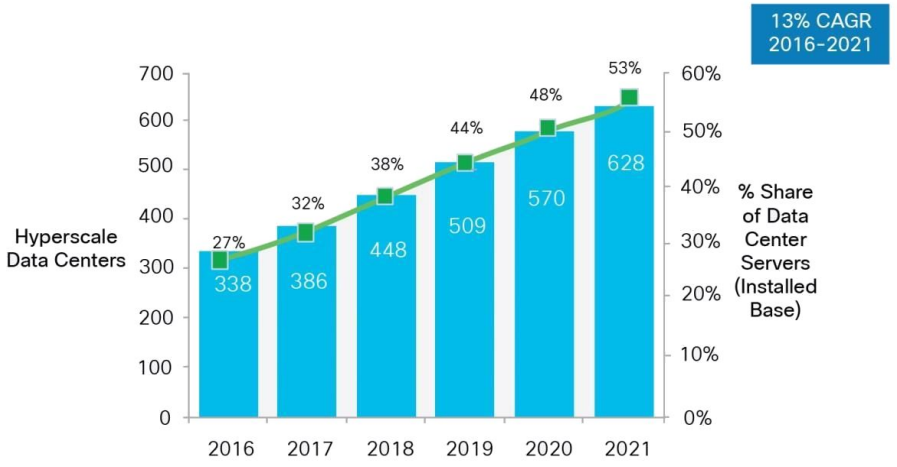


Figura 1.1: Crecimiento previsto de centros de datos de gran escala para 2021 [5].

Si analizamos el consumo de energía en centros de datos, este se distribuye en dos partes diferenciadas. El primero es el mismo procesamiento de los datos, que consume hasta el 60% energía de los centros. El segundo es el refrigeramiento de los equipos, que de media consumen un 40% pero puede llegar a ser hasta el 61% con sistemas poco eficientes [1]. La infraestructura de refrigeración es necesaria para evitar que los equipos informáticos alcancen temperaturas críticas, lo que provoca caídas en el servicio y fallos irreversibles. En este trabajo nos centraremos en un nuevo sistema de refrigeración más eficiente que permite reducir significativamente el consumo de los centros de datos.

1.1. Refrigeración en centros de datos

Tradicionalmente, los sistema de refrigeración en centros de datos han sido diseñados para enfriar las salas técnicas por medio del aire. Varias de las técnicas más frecuentes son [3]:

- *Hot Aisle* / *Cold Aisle o* Pasillo frío / Pasillo caliente: Colocación de racks en filas de forma alterna creando pasillos calientes y pasillos fríos. El aire frío circula por el

suelo hacia arriba donde se junta con el caliente y entra en equipos CRAC (*Computer Room Air Conditioning*) que lo enfrían y lo vuelven a dirigir hacia el suelo.

- *Cold Aisle Containment* o contención de pasillo frío: Evolución del anterior. Consiste en tapar huecos entre los equipos y aislar los pasillos fríos con mamparas, así se evita la fuga de aire frío o la mezcla con aire caliente que reduce la eficiencia.
- *Hot Aisle Containment* o contención de pasillo caliente: Técnica similar a la anterior pero aislando los pasillos calientes, de esta manera se simplifica los procesos de extracción de calor.
- *Free-cooling* o ventilación natural mecánica: Relativamente nuevo, consiste en el intercambio de aire del exterior (frío) por el interior de los centros de datos (caliente). Se puede usar en combinación con los demás sistema y su eficacia depende de la climatología de su ubicación.

Algunos de los sistemas más modernos utilizan la refrigeración líquida, más eficiente que el aire. Una breve resumen de estas técnicas sería [4]:

- Refrigeración con agua: Se introduce un sistema de tubos dentro de los servidores que recircula agua para realizar el intercambio de calor con los componentes que alcanzan mayor temperatura. El principal problema es que entre en contacto con los equipos, dejándolos inoperativos. Es una técnica con riesgos y se usa en casos muy específicos.
- Refrigeración por inmersión indirecta: Se usan líquidos dieléctricos que evitan el problema principal del agua ya que podrían entrar en contacto con los equipos. En esta técnica, aún así, el líquido no entraría en contacto con los equipos perdiendo eficiencia.
- Refrigeración por inmersión directa: El líquido está en contacto directo con los equipos. Existen dos sub-técnicas:
    - En una fase: El líquido nunca cambia de estado, es un baño normalmente con la tapa abierta.
    - En dos fases: El líquido cambia de estado a gas y vuelta en un bucle, es un baño con la tapa cerrada o semicerrada. En este trabajo se abordará el uso de un sistema de este tipo. Sus ventajas frente a otras técnicas se detallan en el estado del arte.

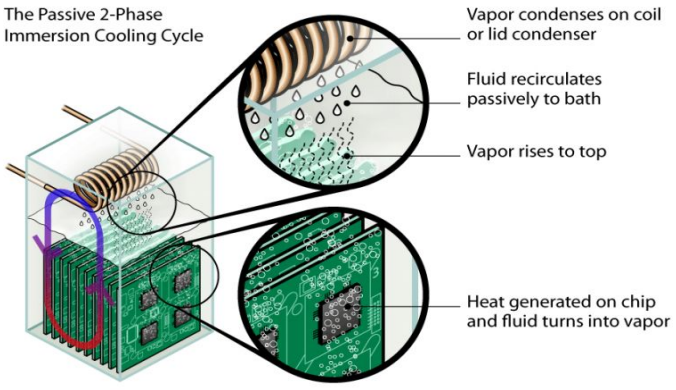


Figura 1.2: Sistemas de refrigeración moderno por inmersión directa en dos fases.

1.2. El futuro del *Cloud Computing*

Además de comentar cómo se distribuye la energía y las diferentes técnicas de refrigeración, es importante hablar sobre los diferentes tipos de centros de datos [6]. Tradicionalmente la computación en la nube (*Cloud Computing*) se ha realizado por medio de grandes centros de datos que recolectan toda la información y la procesan en el mismo centro. Hasta hace poco era una estrategia viable, pero el aumento de la demanda de las aplicaciones IoT hace casi imposible seguir usando solo este tipo de centros para abastecer la demanda. Del problema anterior surge un nuevo paradigma, la computación en el borde o *Edge Computing*. En oposición a la estructura tradicional, este busca distribuir los datos para procesarlos en los extremos de la red en vez de centralizar todo en una sola megaestructura. Se usan centros de datos de menor tamaño pero con mayor número de estos, más cercanos al origen de los datos. Tienen ventajas como menor latencia en los envíos o la mayor seguridad que ofrecen los sistemas distribuidos. Computación en la niebla o *Fog computing* se puede entender como un caso específico del *Edge Computing*, en el que la fuente de los datos tiene que estar muy cerca de un hardware con capacidad de cómputo de datos. Esto permite una latencia menor aún, pero perdemos capacidad de cómputo.

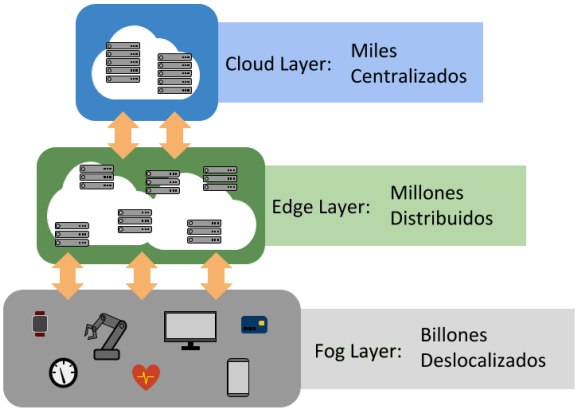


Figura 1.3: Distribución de capas por diferentes tipos de paradigma de computación.

Relacionando los tipos de centros de datos con los sistemas de refrigeración, tenemos que los sistemas tradicionales de *Cloud Computing* han usado sistemas de refrigeración por aire. Estos, debido a su gran tamaño (necesidad de unidades CRAC para refrigerar), poca eficiencia y gran coste de despliegue, son incompatibles para los nuevos centros más pequeños y de mayor número que necesita el *Edge Computing*. Además, los centros de datos tradicionales ya aprovechan localizaciones geográficas favorables, en los que el propio frío del lugar ayuda en la tarea de refrigeración. Esto es imposible en *Edge Computing*, ya que los centros de datos necesitan estar ubicados cerca de la fuente de los datos y no siempre se van a encontrar en un sitio con condiciones climatológicas favorables. El desarrollo de sistemas pequeños (al estar cerca de las fuentes de datos no pueden ser de gran tamaño), más eficientes (no puedes seguir aprovechando la climatología) y económicos (a mayor número de centros, mayor coste de despliegue), se hace necesario para un futuro en el que se pueda abastecer la necesidad de procesamiento de datos.

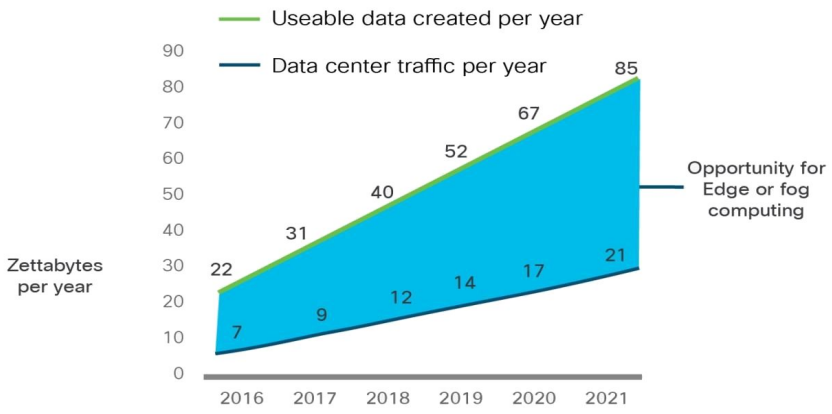


Figura 1.4: Comparativa de la previsión de datos generados y datos aprovechados [5].

1.3. Líquido dieléctrico Vertrel XF

En este trabajo se estudiará, como ya se ha mencionado, un sistema de refrigeración por inmersión directa en dos fases, usando un líquido dieléctrico que nunca antes se había utilizado con este propósito. Su nombre comercial es Vertrel XF y tiene potencial de agotamiento de ozono nulo (cantidad de destrucción de ozono) y bajo potencial de calentamiento global, lo que lo hace perfecto no solo para un sistema eficiente, sino para uno limpio [14]. Tiene propiedades físicas únicas, como: alta densidad, baja viscosidad, baja tensión superficial, no inflamable, estable química y térmicamente, baja toxicidad y facilidad de recuperación por destilación. Todo ello lo ha convertido en un líquido ideal para limpiar y aclarar equipos equipos electrónicos, secar fluido, eliminar partículas, actuar como lubricante para fluorocarbonos, medio disolvente y dispersivo, comportarse como medio para transferencia de calor y trabajar como fluido dieléctrico.

Este líquido es especialmente útil para la limpieza de partículas finas (del rango de micrómetros) de piezas metálicas o no metálicas. Pero teóricamente, mediante un estudio sobre sus propiedades previo a este trabajo, se ha evaluado que puede funcionar como refrigerante y que tendría una capacidad de arrastre de calor mucho más eficiente que los sistemas de refrigeración tradicionales. En este trabajo se probará por primera vez su uso como refrigerante. No es el primer líquido que se utiliza para estos sistemas, el más utilizado hasta la fecha ha sido el NOVEC 7100 de 3M [7]. La ventaja del Vertrel XF sobre este es su precio, un tercio más barato, y que su punto de ebullición es más bajo, permitiendo llegar a las dos fases con mayor facilidad, haciéndolo más eficiente.

1.4. Optimización en centros de datos

Después de haber explicado los diferentes sistemas de refrigeración, incluido el que vamos a estudiar, es necesario hablar sobre uno de los problemas que tienen todos ellos, el *overcooling* o sobreenfriamiento. Históricamente, los diferentes sistema de refrigeración han carecido de un método que les permita aprovechar un equilibrio entre alcanzar el máximo rendimiento de los equipos y ahorrar en energía consumida por refrigeración (fijando la temperatura de los sistemas de climatización).

Se calcula que incrementando un grado centígrado la temperatura de la sala, se puede obtener un ahorro del 7.08% en energía de refrigeración [8]. Con lo que un manejo cuidadoso de la temperatura podría resultar en un gran ahorro energético, y por tanto económico.

Además de encontrar este equilibrio, el gestor de recursos del centro de datos deberá también repartir la carga entre las diferentes máquinas para obtener el mejor rendimiento de estas en cuanto a utilización y temperatura. Para ello es necesario:

- Un sistema de monitorización que nos permita capturar datos en tiempo real de las variables deseadas, tales como: temperatura, humedad, frecuencias de los equipos, etc.
- Implementación de modelos que nos permitan predecir de forma precisa y con suficiente antelación la temperatura.
- Reparto dinámico y óptimo de la carga basado en la predicciones de los modelos.

La irrupción de la inteligencia artificial en los últimos años ha permitido facilitar el desarrollo los sistemas de predicción necesarios para esta tarea. En particular el *Machine Learning* o aprendizaje automático engloba el modelado de los sistemas de predicción mediante el uso de algoritmos metaheurísticos, en otras palabras, permite a las máquinas "aprender" de forma automática. El aprendizaje automático se divide en tres ramas: *Supervised Learning* o aprendizaje supervisado, *Unsupervised Learning* o aprendizaje no supervisado y *Reinforcement Learning* o aprendizaje por refuerzo. En este trabajo nos centraremos en el aprendizaje supervisado. La principal diferencia con las demás ramas es que para el aprendizaje se usan datos etiquetados, esto significa que para cada predicción del sistema hay un dato real etiquetado para que la máquina pueda comprobar la exactitud de la estimación y ajustar sus predicciones.

Los algoritmos metaheurísticos que utilizaremos para desarrollar los modelos predictivos en este trabajo son las redes neuronales. Éstas han alcanzado una gran popularidad en la actualidad gracias al desarrollo hardware, que ahora permite procesar grandes cantidades de datos en poco tiempo, condición necesaria que ha limitado su uso hasta ahora. Utilizando grandes volúmenes de datos, las redes neuronales son capaces de aprender por sí solas patrones, generando todo tipo de funciones no lineales. Existen en la actualidad numerosas aplicaciones que hacen uso de estas como: motores de búsqueda, análisis de mercados, robótica, reconocimiento del habla o conducción autónoma.

1.5. Carga de trabajo: minado de criptomoneda

Los centros de datos se usan para almacenar datos de una gran variedad de aplicaciones: video bajo demanda, sensores IoT, juegos online, inteligencia artificial, entre otros. En nuestro caso, la carga de trabajo que proporcionaremos a los equipos será minado de criptomonedas. Las criptomonedas son monedas virtuales que usan criptografía para garantizar el anonimato y proporcionar seguridad frente posibles falsificaciones, permitiendo a cualquier usuario realizar transacciones sin estar regulado por el gobierno o bancos privados [9]. Son tan seguras gracias a la tecnología *blockchain* en el proceso de

minado, el cual consiste en verificar transacciones de criptomonedas dentro de una cadena generando un bloque por cada operación.

Además, la verificación de transacciones por parte de usuarios independientes de estas les proporciona a estos parte del valor de la operación que realicen. Desde que el uso de criptomonedas se popularizó, se han intentado buscar nuevos métodos de rentabilización del proceso de minado, ya que este consume muchos recursos del equipo informático. Otro aspecto a destacar es que al existir más criptomonedas en circulación, se necesita cada vez mayor potencia para que el minado sea rentable.

Todo ello ha llevado a que el minado necesite de grandes estructuras, centros de datos, para que su generación sea rentable. Centros en Países Bajos [10], o en Estados Unidos [11] usan sus instalaciones para minar criptomonedas, y no solo esto, existen ya servicios en centros de datos de alquiler de servidores para este propósito, denominados como *Cloud Mining* o minado en la nube. En números, solo en minería de Bitcoin, la criptomoneda más importante y de mayor valor, se consumió el año pasado un 0.13% de la energía global (equivalente al consumo de 151 países en el mundo), [12] y se estima que para este año la cifra ascienda al 0.5%. Además, el beneficio anual total fue de 7.2 mil millones de dólares, aunque el coste fue de 1.5 mil millones de dólares.

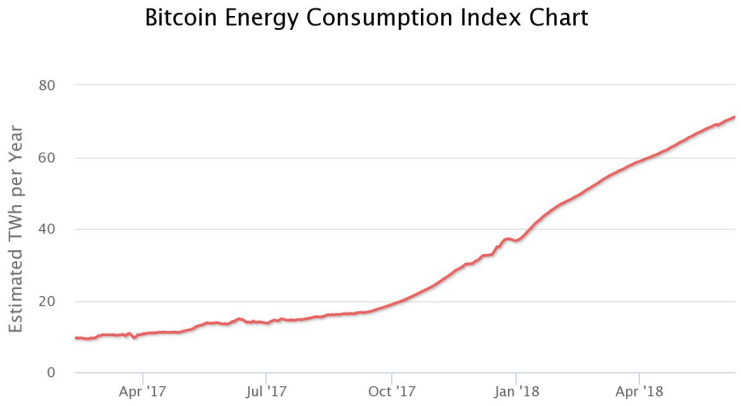


Figura 1.5: Estimación de consumo energético de Bitcoin en el último año [13].

1.6. Resumen de objetivos del proyecto

Para finalizar este apartado se resumen los objetivos de este trabajo. El presente proyecto consiste en probar la eficacia de un sistema de refrigeración, para centros de datos, por inmersión directa en dos fases que usa un líquido dieléctrico que no se había utilizado anteriormente para este propósito. Teóricamente, este sería mucho más eficiente y limpio que los sistemas convencionales. Para el desarrollo de este proyecto, se utilizarán dispositivos Raspberry Pi que estarán minado criptomoneda como carga de trabajo mientras están sumergidos en el líquido Vertrel XF. Simultáneamente, se estarán monitorizando diferentes datos como temperatura, carga o frecuencia para desarrollar modelos predictivos de temperatura que permitan hacer funcionar los equipos en el punto de trabajo óptimo.

2. Estado del arte

A continuación, se detalla el estado actual de las soluciones al problema que nos enfrentamos así como las ventajas presentadas por la solución propuesta en este trabajo. Se estudiará desde tres enfoques bien diferenciados. Primero, las técnicas de refrigeración en centros de datos. Segundo, los sistema predictivos que se han utilizado para la predicción de la temperatura. Y tercero, los algoritmos que utilizan los sistemas predictivos.

2.1. Los sistemas de refrigeración en la actualidad

En este apartado se explicaran los diferentes sistema de refrigeración en centros de datos, tanto por aire como por líquido, conceptos como el PUE para medir la eficiencia de estos y se darán ejemplos de sistemas reales (o pendientes de implementación) de diferentes empresas. En el anexo C, se resumen las ventajas e inconvenientes de todos los sistemas presentados en este apartado en una tabla.

2.1.1. Refrigeración por aire en los centros de datos

Históricamente, el problema de la refrigeración en los centros de datos se ha tratado con sistemas que hacen uso del aire para enfriar la sala. El objetivo prioritario es enfriar los equipos para que rindan de manera más eficiente. Para ello, el concepto básico en el que se basan la mayoría de los sistemas de refrigeración por aire es la colocación de *racks*, estructuras que albergan los equipos informáticos, en filas de forma alterna formando pasillos calientes (cuando se enfrentan la parte trasera de los *racks*) y pasillos fríos (cuando se enfrentan la delantera).

Los equipos suelen tomar el aire frío por la parte delantera y expulsar el aire caliente por la trasera. Para enfriarlos, y con ello mejorar su rendimiento, se colocan rejillas en el suelo de los pasillos fríos que expulsan aire frío que luego toman los equipos. El aire frío se inyecta con unidades CRAC (*Computer Room Air Conditioning*) que recogen el aire caliente en su parte superior e introducen el aire enfriado en su parte inferior. Esta técnica se denomina pasillo frío / pasillo caliente o *Hot Aisle* / *Cold Aisle*, y supone ya una primera técnica que separa las corrientes de aire frío y aire caliente para su uso más eficiente, y con ello mejorar la eficiencia energética del centro de procesamiento de datos.

Uno de los problemas que surgen con este sistema es la necesidad de un estudio minucioso de las condiciones de enfriamiento y de la distribución de aire, ya que sino se pueden dar flujos de aire no deseados. Para solucionarlo, se cierran los pasillos para contener el aire. Las técnicas que utilizan esta estrategia se denominan *Cold Aisle Containment* o contención de pasillo frío y *Hot Aisle Containment* o contención de pasillo caliente.

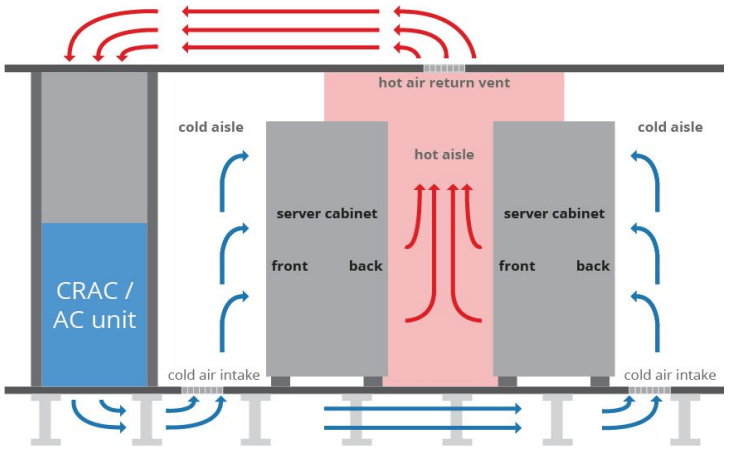


Figura 2.1: Ejemplo de sistema de refrigeración *Hot Aisle* / *Cold Aisle.*

En *Cold Aisle Containment* (CACS) se cierran los pasillos que contienen el aire frío, conservándolo para que no se mezcle con el aire caliente [15]. Además, conseguimos que el volumen de aire frío sea menor, lo que implica que las unidades CRAC consuman menos y poder aumenta la temperatura de la sala, ahorrando energía sin perder eficiencia. Las desventajas son que en caso de fallo al estar contenido y haber menos volumen, el aire frío se calentará más rápido, elevándose la temperatura de los equipos rápidamente.

En *Hot Aisle Containment* (HACS) esta vez se cierran los pasillos calientes [16]. Las ventajas principales frente a los sistemas que utilizan CACS son que al estar toda la sala fría (menos los pasillos calientes), tarda más en elevarse la temperatura de la sala en caso de error, ya que el volumen de aire frío es mayor. Además de que al tener el aire caliente contenido, se facilitan los procesos de extracción de este. Sin embargo esta técnica tiene varias desventajas, como que al haber más volumen de aire frío las unidades CRAC tienen que trabajar más o que es más caro de desplegar que un sistema CACS ya que este último solo necesita puertas y un techo mientras que en HACS necesitas un sistemas de tuberías y conectores para desplazar el aire caliente [17].

Estas dos técnicas ayudan a que no se formen corrientes indeseadas por mezclar aire caliente y frío o puntos calientes en la sala, pero no solo estos son los problema que tienen los sistemas de refrigeración. Los centros de datos están diseñados para funcionar a todas horas del día ininterrumpidamente, lo que supone un gasto de energía en las unidades CRAC para el enfriamiento del aire muy elevado. Relativamente nuevo son los sistemas de *Free Cooling*, que ayudan a mitigar este problema.

El *Free Cooling* es una técnica de refrigeración que hace uso del aire frío del exterior para climatizar la sala. No obstante, no está totalmente libre de consumo eléctrico o coste de despliegue ya que necesita de compuertas, ventiladores y filtros para controlar de forma eficiente el aire exterior, pero sí que supone un gran ahorro respecto a los otros sistemas. También mejora la calidad del aire del interior del centro de datos al estar renovándose continuamente. Por contra, el mecanismo para controlar el aire puede ser complejo y, además, depende de la climatología del lugar para poder sacar partido de esta técnica.

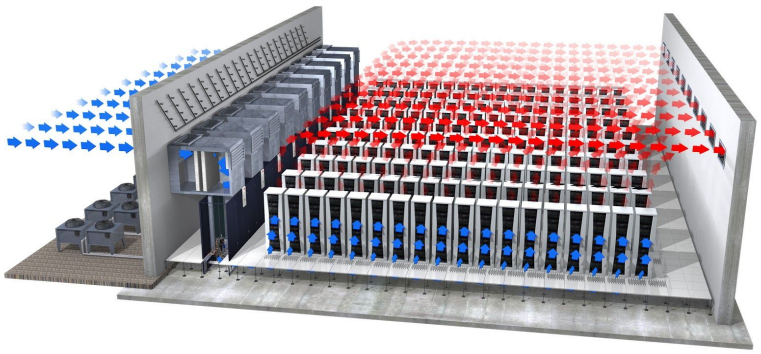


Figura 2.2: Ejemplo de sistema de refrigeración *Free Cooling*.

Muchas empresas tecnológicas como Google o Facebook aprovechan la climatología del lugar para situar sus centros de datos [18]. Países como Finlandia o Suecia son el destino principal para la construcción de nuevos edificios (sobre todo masivos, que siguen filosofías tradicionales de *Cloud Computing*). En 2017 se anunció la construcción del centro de procesamiento de datos más grande del mundo en el polo norte, en una colaboración entre Estados Unidos y Suecia, que aprovechará al máximo las ventajas climatológicas de la zona para hacer frente al elevado tráfico que se estima que tendrá [19].

2.1.2. Métrica de eficiencia energética en centros de datos

La energía utilizada en los centros de datos se usa en el alumbrado, equipos, sistemas de alimentación ininterrumpida (SAI), unidades CRAC, etc. Se puede englobar en dos partes bien diferenciadas [1]. Primera, el procesamiento de datos en los equipos de los *Racks*, y segunda, el sistema de refrigeración. Esta última puede variar entre el 40% en los sistemas más eficientes hasta el 60% en los menos eficientes.

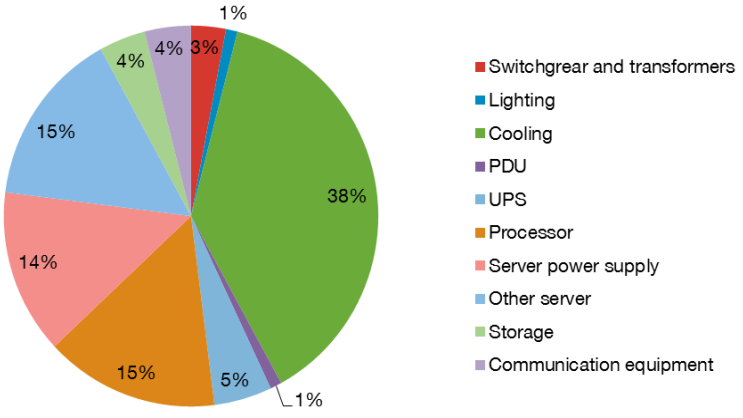


Figura 2.3: Demanda de energía en los centros de datos.

La métrica que utiliza la industria para estimar la eficiencia del sistema energético es la PUE o *Power Usage Effectiveness* , fue introducida en 2007 por *Green Grid*, es definida como el consumo de energía total de un centro de datos partido de la energía consumida por los equipos informáticos:

$$PUE = \frac{Consumo\ de\ energía\ total}{Consumo\ de\ energía\ IT}$$

Cuanto más alto es el valor del PUE, más baja es la eficiencia de la instalación ya que más energía es usada para tareas que no son la de procesamiento de datos. El valor ideal es uno, el cual indica que no hay energía desperdiciada. Este valor no es alcanzable debido a los sistemas de alimentación, transformadores, iluminación… Diferentes estudios sitúan el PUE de los centros de datos actuales entre varios rangos [1]. Koomey estima un rango entre 1.25 y 3.75 con un promedio de 1.92, basado en una encuesta a 61 centros de datos en 2010. Uptime Institute publicó una encuesta en 2013 en la que el PUE medio es de 1.65, menor al 1.89 de 2011 y 2.1 de 2007.

El enfoque que queremos dar en este trabajo es la ventaja que existe entre un sistema de refrigeración líquido (en nuestro caso por inmersión de dos fases) y los sistemas de refrigeración por aire. Para ello es de gran utilidad comparar ejemplos de valores de PUE que han dado diferentes sistema reales implementados en la actualidad:

- Un estudio de 2010 comparó los sistemas *Hot Aisle Containment* y *Cold Aisle Containment*, obteniendo que los del sistema CAC tenían una PUE de 1.87 y el sistema HAC de 1.64 [24]
- El departamento de energía de Estados Unidos comparó los dos sistemas en el centro de datos de alto rendimiento de Maui, en este caso usando un sistema de refrigeración por agua para el líquido. Los resultados dieron una media de PUE de 1.65 para el sistema por aire y de un 1.45 para el sistema por agua [20].
- Para ver cómo han evolucionado los sistemas, en octubre de 2008 los centros de datos de Google eran los líderes con un PUE de 1.21 [21], mientras que en 2015 el centro de datos de Facebook en Prineville obtuvo un PUE de 1.082 [22].
- El caso que más nos interesa es el de Allied Control, que afirmó haber conseguido un PUE de 1.2 con un líquido dieléctrico en inmersión (Novec 7100) como es nuestro caso [23].

2.1.3. Refrigeración líquida en los centros de datos

Los centros de datos refrigerados por aire tienen desventajas difíciles de tratar [25], el coste de despliegue es alto, el gasto de energía para alimentar los equipos puede llegar hasta el 15%. No solo son ineficientes, también fallan. Refrigerar con aire crea problemas más allá de energía desperdiciada o espacio. El aire, al entrar en contacto con los equipos acaba oxidándolos, y que el aire no esté totalmente limpio lo empeora. Para limpiarlo se usan filtros, pero esto hace que se caliente. Los ventiladores transmiten vibraciones que pueden aflojar soldaduras, además de que generan calor que debe ser disipado y generan ruido que puede molestar a operarios, obligándoles a llevar tapones. Elevar la temperatura en centros de datos para reducir la necesidad de refrigeración causa que los ventiladores de los diferentes equipos giren más rápido para mover más aire, concretamente, mover el doble de aire triplica la energía necesaria, se disipa calor, etc.

La mayoría de estos problemas se pueden solucionar por medio de refrigeración líquida, el agua por ejemplo, tiene una capacidad de transferencia de calor mucho mayor

que el aire. La refrigeración líquida no es nueva, ya se utilizaba antes pero los sistemas eran caros, difíciles de usar y de mantener, además una posible fuga de agua podría ser catastrófica. Era mucho más fácil para los operarios desplegar un sistema de aire acondicionado, sin entrar en profundidad de cómo resolver el problema de la refrigeración. Si los sistemas de aire han dejado de ser la opción prioritaria es porque la densidad de energía que mueven los centros de datos en la actualidad es tan grande que ya no son efectivos. Además, tecnologías de refrigeración líquida nuevas solucionan los problemas que tienen las antiguas. Ahora los sistemas son más eficientes y limpios, con fácil mantenimiento, escalables y más baratos que los antiguos.

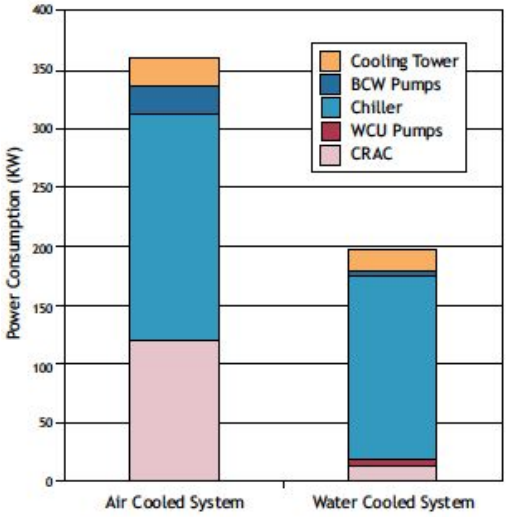


Figura 2.4: Comparativa de energía consumida en refrigeración por aire y agua [26].

Existen diferentes tipos de refrigeración líquida y se pueden agrupar de varias maneras. En este trabajo los agrupamos en dos de las que derivan las demás: refrigeración por agua y refrigeración por líquido dieléctrico.

En refrigeración por agua tenemos varias técnicas, las unidades CRAH (*Computer room air handler*) son dispositivos que a diferencia de los CRAC que utilizan refrigeración mecánica, estos utilizan ventiladores, tuberías y sistemas de enfriamiento por agua para eliminar el calor. A nivel de componentes tenemos placas frías (*Cold Plates*), que se utilizan para refrigerar partes de equipos que se calientan más como los procesadores. Estas necesitan de más sistemas como ventiladores y a veces, con lo que no son muy eficientes. Las enfriadoras por agua tipo *in-row* pueden ser usadas para grandes densidades de energía reduciendo la energía que consumiría un sistema de aire en un 60%, pero necesitan de un sistema de contención de pasillo frío o caliente con lo que se limita la eficiencia que puedan llegar a tener.

Existen refrigeradores pasivos de puertas traseras que son capaces de disipar el calor de los equipos en el punto de origen utilizando agua como refrigerante. Su contra es que necesita que los ventiladores de los equipos empujen el aire hacia el intercambiador de calor. También se ha usado enfriamiento directo en el chip con agua para *racks* de alta densidad, su ventaja es que no necesita que el agua esté tan fría como en otros sistemas y

su problema es el coste económico del sistema. Tenemos los sistema de refrigeradores activos de puertas, que incluyen un intercambiador de aire y agua con ventiladores, válvulas para controlar el agua y controladores programables. Son capaces de enfriar tanto el aire como el agua dinámicamente. Otra de sus ventajas es que ocupan aún menos a comparación por ejemplo de los sistemas *in row*.

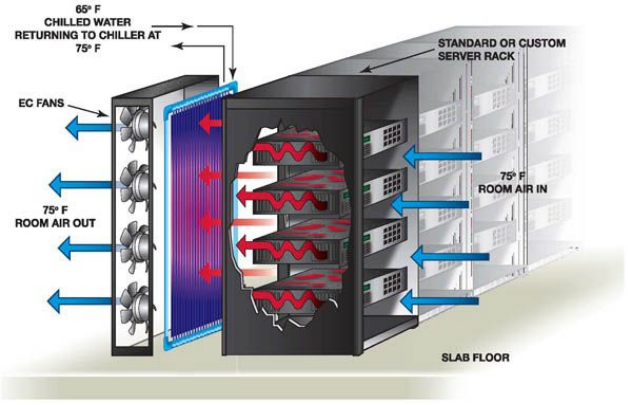


Figura 2.5: Sistema de refrigeración por agua de puerta trasera.

Los sistemas de refrigeración por agua tienen dos desventajas con respecto a los de líquido dieléctrico, primera, una fuga de agua podría ser fatal para los equipos en un centro de datos suponiendo grandes pérdidas, y segunda, aunque más eficientes que los sistemas de aire, las mejores técnicas de refrigeración por agua mueven densidades entre 30 y 40 kW por *rack*, mientras que los sistemas de refrigeración por líquido dieléctrico teóricamente pueden llegar hasta 200 kW por *rack*.

Los sistemas de refrigeración líquidos por inmersión ya existían en otros sectores desde hace años para transformadores y maquinaria industrial a principios del siglo XX [27], e incluso grandes ordenadores de la década de los ochenta llegaron a usar esta técnica. Existen dos tipos principales de dieléctricos los basados en aceite y los basados en químicos. Estos últimos suponen una ventaja, ya que a diferencia de los aceites no son inflamables ni combustibles, tienen el punto de ebullición bajo y la estabilidad térmica necesaria para realizar inmersión en dos fases y el mantenimiento de los equipos es más fácil porque al retirarlos del líquido salen completamente secos.

Podemos dividir las técnicas de refrigeración con líquido dieléctrico en cuatro tipos. Primero tenemos la refrigeración indirecta, uno de los mayores problemas que tenía el agua como hemos comentado son los riesgos de fuga cuando se aplica una refrigeración que entra en contacto directo con los equipos por medio de tubos. Este riesgo desaparece con un líquido dieléctrico gracias a sus propiedades. Por contra, esto no aprovecha todo el potencial que tienen los líquidos, perdiendo mucha eficiencia. La segunda técnica es refrigeración líquida por inmersión en equipos sellados. Los equipos del centro de datos están construidos especialmente para alojar en su interior el líquido dieléctrico, lo que los hace más manejables que si necesitaran un baño completo. Su principal desventaja es que la mayoría de los equipos no están diseñados para este propósito.

Tercero, refrigeración por inmersión en una fase. En esta técnica los equipos se sumergen en el líquido, aprovechando su potencial para arrastrar el calor. Los líquidos que se usan tienen un alto punto de ebullición que hace que no cambie de estado y permanezca líquido. Suelen ser baños con tapa abierta o semiabierta. El líquido, que suele ser un aceite, es bombeado a un intercambiador de calor donde es enfriado. También, existen métodos que usan convección natural para hacer llegar el líquido al intercambiador de calor. El problema con esta técnica es que no es completamente pasiva, perdiendo eficiencia por ello.

Y por último, refrigeración por inmersión en dos fases. Los equipos son sumergidos en el líquido en un baño cerrado. Esto genera que el líquido se evapore al calentarse (los líquidos que se utilizan con esta técnica suelen tener un punto de ebullición entorno a 50ºC en comparación al agua 100ºC) y suba al techo del cerramiento donde se condensa y se precipita otra vez hacia abajo. La eficiencia de la extracción de calor se incrementa exponencialmente ya que se trata de un método totalmente pasivo [28]. Se podría ahorrar hasta un 90% de la electricidad comparado con un sistema de refrigeración por aire, y se elimina la necesidad de operaciones costosas o operarios de más. Además, los líquidos que se usan son extremadamente limpios, no tóxicos y seguros para el medio ambiente. No tienen la necesidad de un hardware específico, se puede bañar cualquier equipo con lo que también se evita un rediseño constante. Funciona perfectamente en sitios confinados en y ambientes con situaciones adversas (calor, humedad…). Tiene alguna desventaja, como que su desarrollo todavía no está muy extendido o que se podrían generar presiones muy altas en el recinto del líquido al cambiar de fase.

2.1.4. Sistemas de refrigeración reales en funcionamiento

Hasta aquí hemos visto los diferentes tipos de refrigeración y cómo estos últimos por inmersión son una mejora considerable a cualquier otro método. En este trabajo, como ya se ha mencionado con anterioridad, se estudiará el uso de un sistema de refrigeración por inmersión en dos fases. Por lo tanto, es necesario revisar los diferentes tipos de sistemas reales que han sido ya implementados con esta técnica y en qué se pueden mejorar.

Desde 2014 el superordenador Tsubame KFC, del Tokyo Institute of Technology, usa un sistema que usa un tanque en el que sumerge servidores en un aceite mineral como si fuera un *rack* tumbado. Esto le ha valido para ser el superordenador más eficiente del mundo en su día y sigue estando entre los primeros en la actualidad (número seis en Green500 [29]). Su problema yace en usar aceites, que es una solución sucia y que solo opera en una fase, perdiendo eficiencia.

La empresa Iceotope, de origen inglés, ofrece ya soluciones de refrigeración líquida en una fase a nivel de *rack*. Su método consiste en circular líquido por carcasas que recubren los equipos. Tiene dos problemas: primero, como el anterior, solo funciona en una fase perdiendo eficiencia, y segundo, las carcasas ocupan mucho espacio, limitando el número máximo de equipos que pueda haber en el centro de datos. La empresa Allied Control, que se dedica al diseño de sistema de refrigeración, utilizó un sistema de inmersión en dos fases para el líquido dieléctrico, para un centro de datos especializado en minería de

bitcoins. BitFury compró la empresa y ahora ofrece una solución completa para minería dentro de un contenedor de barco soportable. Aunque pueda parecer que es una solución perfecta, no explota al máximo las posibilidades de la tecnología ya que no funciona en el punto de trabajo de máximo rendimiento. En esto se entrará más en profundidad en el subapartado “La necesidad de predecir la temperatura en la refrigeración (sección 2.2.1).

Figura 2.6 y 2.7: Equipo de refrigeración de la empresa Icetope y BitFury.

EXTOLL, empresa de origen alemán, ofrece un tanque de dimensiones pequeñas de muy alto rendimiento con el líquido Novec en dos fases. Su incovenientes son los mismos que para la empresa BitFury y además sus tanques están diseñados para sus placas, no siendo válidos para otro tipo de equipamiento.

Por último, comentar los sistemas de refrigeración que utilizan las empresas líderes del sector IT. En el centro de datos de Google en Hamina, Finlandia, usan agua marina para refrigerar sus equipos [30]. Bombean el agua hacia módulos de refrigeración y luego por la fuerza de la gravedad, pasan de los módulos a un edificio intermedio donde se mezcla el agua que llega con la que está pendiente de salir para que, a la hora de volver al mar, tenga una temperatura más similar a la que viene nueva.

El sistema más nuevo de Facebook, a fecha de Junio 2018 [31], es una mezcla entre un sistema de aire y uno de agua. Se denomina StatePoint y se basa en refrigeración por evaporación. Los sistemas actuales que utiliza Facebook en sus centros de datos suelen usar agua, y gracias a esta técnica podrán conseguir reducir su uso en sus sistemas en un 20% en centros con climas cálidos y hasta un 90% en centros con climas fríos.

2.2. Modelos predictivos para temperatura

En este apartado se verá por qué es necesario realizar modelos de temperatura en los sistemas de refrigeración, y en especial, en los de inmersión en líquido dieléctrico en dos fases. Además, se estudiarán diferentes algoritmos y soluciones que se hayan implementado en la actualidad.

2.2.1. La necesidad de predecir la temperatura en la refrigeración

El modelado de la temperatura en los centros de datos para un uso eficiente de la refrigeración que les permita tanto el máximo rendimiento posible de los equipos, como un

ahorro de energía consumida por refrigeración, ha sido un tema que históricamente se ha obviado y que va a ser de gran importancia en un futuro. Como comentamos en la introducción, el crecimiento del sector IT y la irrupción del *Internet of Things* (IoT) hacen necesario ser lo más eficientes posibles en los centros de datos del futuro. Esto significa que no solo es necesario encontrar nuevas técnicas para refrigerar, sino hacerlas funcionar a su rendimiento térmico óptimo, que nos permitirá aprovechar los equipos al máximo y ahorrar toda la energía que sea posible.

La empresa 3M realizó un estudio del comportamiento de la temperatura del líquido NOVEC 7100 para averiguar su rendimiento máximo posible. Aunque no existe un estudio similar para el Vertrel XF, se espera que los resultados sean similares. Por ello nos basaremos en el estudio de 3M. Si miramos la gráfica que realizaron en la siguiente figura:

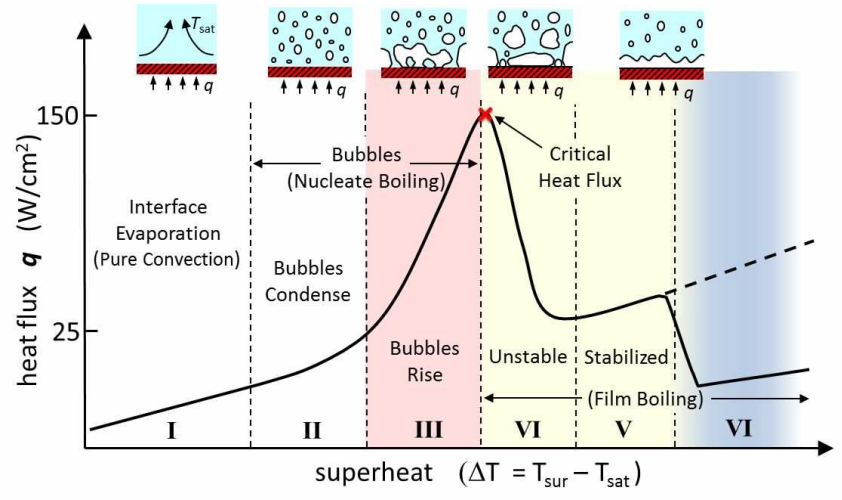


Figura 2.8: Gráfica de absorción de calor en función de la temperatura, 3M.

La figura relaciona el rendimiento máximo de refrigeración del líquido dieléctrico, expresado como cantidad de calor absorbida (W/m^2), en función de la temperatura a la que está el líquido. Existen seis franjas diferenciadas de temperatura que corresponden a los estados del líquido, en la que nos interesa trabajar es la tercera, sin llegar a pasar a la cuarta ya que el rendimiento cae en picado. Es por esto que es necesario realizar modelos predictivos de temperatura, para poder anticiparnos a las temperaturas futuras que no nos permitan trabajar en la franja de máximo rendimiento y adaptar las cargas automáticamente de los equipos en función de la temperatura para variar esta, sin necesidad de ningún tipo de sistema extra que hiciera perder la ventaja de ser un sistema de refrigeración pasivo.

Esta estrategia es totalmente nueva y será puesta a prueba en este departamento por primera vez. Hasta ahora, las soluciones que han usado inmersión en dos fases sólo han trabajado en la primera o segunda franja, obviado todo el potencial de absorción que tienen estos líquidos. Encontrar algoritmos que nos ayuden a trabajar en este rango se vuelve prioritario para hacer que nuestro sistema de refrigeración sea mejor que las demás soluciones realizadas en la actualidad.

2.2.2 Modelos de temperatura en la actualidad

Aunque como hemos mencionado, esta estrategia es la primera vez que se utiliza, no es la primera vez que se intenta mejorar la eficiencia de sistema de refrigeración mediante software creando modelos inteligentes que reparten la carga. A continuación, se mencionan diferentes trabajos destacados para la solución de este problema.

Lijun Fu, Jianxiong Wan y Ting Liu en la escuela de ingeniería de la información de Mongolia interior, China, desarrollaron un modelo de temperatura para el manejo de recursos de los equipos para una minimización holística de la energía en centros de datos [32]. Su algoritmo está basado en la optimización de Lyapunov y sirve para reducir la energía consumida de unidades CRAC mientras mantiene una temperatura estable y viable del sistema. Con este algoritmo consiguieron reducir un 6% el consumo de energía.

Justin Moore y Jeff Chase del departamento de Ciencias de la computación de la universidad de Duke, Estados Unidos, usaron heurísticas simples para modelar el comportamiento de un centro de datos [33]. Estudiaron la variación de temperatura y prueban diferentes algoritmos y experimentos para calibrar las ineficiencias del sistema. El mejor algoritmo consiguió incrementar el ahorro de energía de anteriores métodos utilizados un 165%.

Hong Xu, Chen Feng y Baochun Li del departamento de eléctrica e ingeniería de la computación de la universidad de Toronto, Canadá. Desarrollaron un sistema empírico de optimización para la refrigeración de un sistema de datos usando el algoritmo ADMM [34]. Según su estudio, modelando la temperatura y repartiendo las cargas con este algoritmo consiguieron ahorrar entre un 15% y 20% de la energía de refrigeración y entre un 5% y 20% del coste de energía general. Concluyen que el algoritmo ADMM es práctico para resolver problemas a gran escala de cargas de trabajo a pocas iteraciones que se hagan.

Anton Beloglazov, Jemal Abawajy y Rajkumar Buyya, del laboratorio de computación en la nube y sistema distribuidos de la universidad de Melbourne, Australia, desarrollaron un sistema de reparto de carga dinámico usando máquinas virtuales [35]. Sus resultados mostraron una reducción sustancial del coste de energía consumida en centros de datos destinados a *Cloud Computing*, respecto reparto de cargas estáticas. Con ello, desarrollaron una plataforma software que apoya la eficiencia energética para la colocación y manejo de recursos de centros de datos *Cloud.*

Qinghui Tang, Sandeep Kumar S. Gupta y Georgios Varsamopoulos del Instituto de Ingeniería Eléctrica y Electrónica (IEEE), desarrollaron un modelo de temperatura para minimizar los picos de temperatura llamado MPIT-TA [36]. Consiguieron ahorrar en energía un 30% de un pequeño centro de datos simulado comparado con modelos anteriores. Su acercamiento al problema se resume en: colocación de carga orientado por temperatura y un modelo de recirculación de calor preciso y de baja complejidad.

Por último Jim Gao de Google realizó quizá el estudio que más nos pueda interesa ya que los modelos que realizo utilizan las mismas metaheurísticas que los nuestros, las redes neuronales [37]. En este concluye que usando un *framework* que utilice *Machine Learning* se puede predecir la PUE de un centro de datos con un error entre 0.004 y 0.005, aproximadamente un error de un 0.4% para una PUE igual a 1.1. Afirma que el *Machine Learning* es un método efectivo para usar sensores de captura de datos y realizar modelos de eficiencia energética en centros de datos, no solo para predecir o estimar temperatura, sino también para otras tareas como colocación de equipos en una sala por ejemplo.

Podemos concluir que modelar la temperatura para variar la carga dinámica ha sido efectivo hasta la fecha para sistemas de refrigeración convencionales en centros de datos. En nuestro caso, lo pondremos a prueba por primera vez en un sistema de refrigeración por inmersión en líquido dieléctrico en dos fases.

2.3. Algoritmos metaheurísticos

El problema al que nos enfrentamos, predecir los cambios de temperatura, como muchos otros como clasificación de imágenes, aprender tareas, IA en los videojuegos, diagnósticos, etc. se enmarcan en lo que se denominan problemas de optimización. Un problema de optimización consta de dos partes: un conjunto de soluciones que forma el espacio de búsqueda y una función de evaluación que mide lo óptimas que son las soluciones [48]. La elección de la función de evaluación o lo que es lo mismo, el método de optimización, depende de la complejidad del problema. Para un problema de pequeñas dimensiones, con entornos bien definidos y de pocas soluciones, se pueden utilizar algoritmos exactos. Para problemas de gran dimensión, estos dejan de ser efectivos y se usan heurísticas, más eficientes a pesar de no garantizar una solución óptima. Dentro de las heurísticas tenemos dos ramas, las heurísticas concretas que sirven para resolver problemas concretos, y las metaheurísticas, de nuestro interés ya que son capaces de resolver problemas de forma genérica adaptándose a cualquier problema de optimización. Se basan en primero buscar los mejores subconjuntos de soluciones, para luego, intensificar la búsqueda en estos.

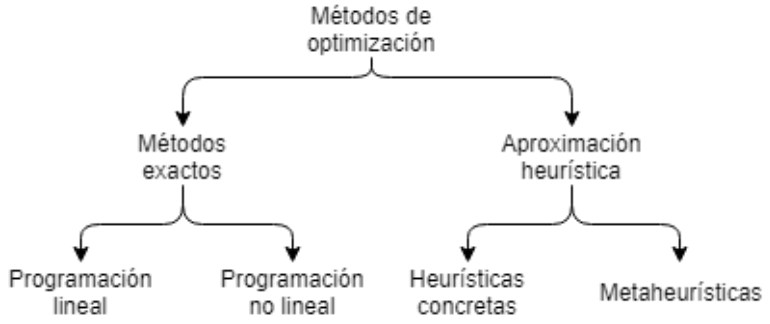


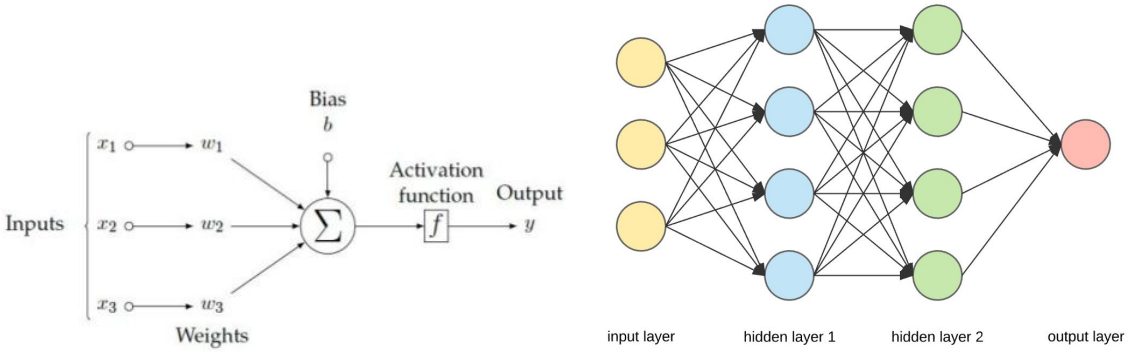
Figura 2.9: Clasificación de los métodos de optimización [48].

Para tratar con un alto número de los soluciones, las metaheurísticas siguen tres estrategias, descartan las soluciones que encuentren no factibles, penalizar las soluciones que no sean factibles (haciendo que su subconjunto sea cada vez menos adecuado) y corregir las soluciones para que sean factibles. En cualquiera de los casos, lo más

importante es definir bien la función de evaluación que nos servirá para medir la calidad de las soluciones.

Existen varias maneras de clasificar los algoritmos metaheurísticos basándonos en sus características [48]: Primero, solución única o poblaciones, la diferencias entre ellas es que la solución única se centra en intensificar la búsqueda dentro de un número de subconjuntos muy reducido, existiendo solo una solución durante el proceso, mientras que las poblaciones se enfocan en buscar nuevos subconjuntos trabajando con varios a la vez. Segundo, inspirados en la naturaleza, se basan en fenómenos como la evolución o la biología, entre otros. Tercero, iterativos o voraces, si son iterativos significa que parten de una solución o un conjunto de ellas y la van modificando durante el proceso y voraz significa que además empiezan con una solución vacía y por medio de una variable de decisión se va modificando su valor. Cuarto, con o sin memoria, si son iterativos pueden aprovechan la información que generan en cada iteración almacenando esta en diferentes estados. Quinto, estáticos o dinámicos, un algoritmo metaheurístico estático mantiene su función de evaluación durante todo el proceso mientras que uno dinámico la va adaptando.

Los algoritmos metaheurísticos que utilizaremos en este trabajo serán las redes neuronales. Estas son iterativas y voraces, ya que por medio de iteraciones y partiendo de una solución vacía, esta se va adaptando (con lo que no necesitan de un entorno específico),. También pueden tener memoria (redes neuronales recurrentes), de gran utilidad para resolver problemas temporales como el nuestro. Y por último son dinámicas, capaces de variar su función si el conjunto de temperaturas varía, pudiendo reajustarse para seguir ofreciendo la mejor solución posible. Desde hace relativamente poco, las redes neuronales se han convertido en los algoritmos por excelencia, esto es gracias al desarrollo hardware actual que ha permitido el procesado de datos masivos, factor que necesario para su utilizamiento. Su popularidad se debe a que son capaces de dar grandes resultados para todo tipo de problemas de optimización y que además son fáciles de aprender, configurar y usar.



Figuras 2.10 y 2.11: Estructura de una neurona y ejemplo de red neuronal.

La unidad básica de las redes neuronales es la neurona o perceptrón. Una neurona (Figura 2.10) no es más que una función matemática en la que cada entrada se multiplican por un peso (*weigth*) y luego se hace un sumatorio de todas junto a un *bias*, para finalmente aplicarle una función de activación. Los pesos sirven para ponderar el valor de cada entrada, y así diferenciar cual es más importante. El bias hace que la función sea no lineal,

aumentando el rango de funciones a las que la red puede aproximarse. La función de activación sirve para que los valores del sumatorio estén escalados y no tiendan a infinito, al repetirse sucesivas operaciones. Finalmente, una red neuronal (Figura 2.11) es un conjunto de neuronas organizadas en tres tipos de capas: de entrada, ocultas y de salida. Para que las redes neuronales encuentren la solución más adecuada primero hay que entrenarlas. El funcionamiento simplificado del entrenamiento en la red neuronal más simple llamada *feedforward* es el siguiente:

1. Se dividen los datos en dos subconjuntos, de entrenamiento y de test.
2. Se inician los pesos y biases de todas las neuronas (ya sea con un valor aleatorio o determinado) y se introduce una primera parte del subconjunto de datos de entrenamiento como entradas produciendo una salida, esto supone la primera iteración.
3. Como nuestro caso es *Supervised Learning* (se comentó en la introducción), los datos estarán etiquetados con una solución para que la red pueda comparar lo precisa que es su predicción. Usando una función de cálculo del error, se ajustan los pesos y biases en función del resultado de esta para la siguiente iteración.
4. Una vez se han hecho iteraciones para todo el subconjunto de datos de entrenamiento (incluso se pueden repetir las iteraciones todas las veces que se quiera, hasta que se considere que ha alcanzado la solución más adecuada), se prueba con el subconjunto de datos del test, lo precisa que es la red.

Para cada problema existe una red neuronal distinta, la forma de que la solución se aproxime a la óptima es modificando sus hiperparámetros. En la solución se explican los diferentes hiperparámetros que disponemos para crear los modelos de temperatura.

Nuestros modelos necesitan predecir en el tiempo la temperatura para poder adaptar la carga de manera óptima. Para ello, necesitaremos redes neuronales con memoria, estas se denominan redes neuronales recurrentes. Las redes neuronales recurrentes funcionan con secuencias de datos ya sean estos temporales, palabras encadenadas en frases o archivos de audio. A diferencia de las neuronas explicadas anteriormente, las neuronas de las redes recurrentes tienen un estado interno (memoria) para procesar las secuencias de entrada, estas neuronas con estados se suelen llamar nodos. Existen diferentes tipos de redes neuronales recurrentes, en este trabajo debido al software que disponemos veremos tres: completamente conectadas o recurrentes simples (*fully connected)*, GRU (*Gated Recurrent Unit*) y LSTM (*Long Short-Term Memory*). En las redes recurrentes completamente conectadas [56], los nodos son iguales que las neuronas de las redes neuronales *feedforward* pero además, las capas superiores están conectados capas inferiores. Cada nodo tiene asociado una activación con valores reales variables en el tiempo (v en la figura 2.12) y cada conexión dentro de esta activación tiene unos pesos asociados (h_t, h_{t+1}, etc. en la figura 2.12).

Este modelo de red neuronal tiene que realizar muchas más operaciones que las simples y con ello pueden surgir dos problemas. El primero es el gradiente evanescente, en este caso la red aprende muy despacio o incluso lo deja de hacer por completo, esto ocurre cuando los pesos tienen valores muy bajos. Y segundo el caso contrario, explosión de

gradientes, los pesos se vuelven tan grandes que es incapaz de converger a una solución estable porque esta cambia demasiado de una iteración a otra. Estos dos problemas surgen cuando se aumenta mucho la ventana de predicción (las operaciones necesarias crecen exponencialmente).

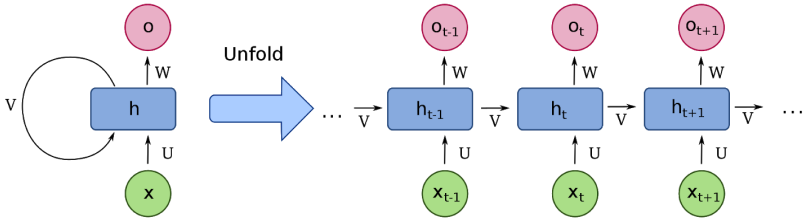


Figura 2.12: Esquema de un nodo *fully connected*.

Para hacer frente a los problemas que dan las redes completamente conectadas, se diseñaron los nodos LSTM. Los nodos LSTM utilizan un mecanismo de puertas para actualizar su estado interno u oculto [49]. Fijándonos en la figura 2.13 tenemos que i, f y o son la entrada, olvido y salida respectivamente. Estas se llaman puertas ya que son una función sigmoidal que devuelve valores entre 0 y 1. Multiplicando elementos por ellas decides cuánto de estos dejas pasar. La puerta de entrada define cuanto de la entrada actual quieres dejar pasar, la puerta de olvido cuanto del estado anterior y la puerta de salida cuanto del estado interno quieres exponer al resto de la red. Por último C es el estado interno del nodo LSTM. Los nodos LSTM permiten grandes ventanas temporales, ya que aprendiendo los parámetros de sus puertas, la red aprende como la memoria debe comportarse. Con el software que disponemos sólo es posible utilizar este modelo de LSTM pero existen varias modificaciones, se pueden leer en este artículo [50].

El comportamiento de las GRU es parecido [49], fijándonos ahora en la figura 2.14 tenemos dos puertas r (puerta de reinicio) y z (puerta de actualización). La puerta de reinicio combina la entrada nueva con la memoria previa y la puerta de actualización define cuánto de la memoria previa quieres mantener. Tanto las LSTM como las GRU pueden ser la opción óptima, depende tanto del problema como de los hiperparámetros. Como norma general, las GRU funcionan mejor con un número de datos reducido, además de ser más rápidas ya que tienen menos parámetros, mientras que las LSTM son mejores para *datasets* grandes.

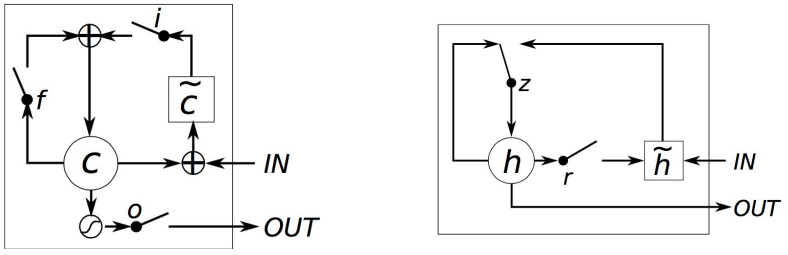


Figura 2.13 y 2.14: Esquema simplificado de un nodo LSTM y GRU.

3. Solución propuesta

En este proyecto se pondrá a prueba la eficacia de un sistema de refrigeración por inmersión en dos fases que, como hemos visto, son los sistemas más limpios y eficientes con mucho potencial ante una demanda de energía futura casi insostenible, utilizando el líquido dieléctrico Vertrel XF que nunca se ha probado para este propósito. Además, se realizarán modelos predictivos de temperatura que permitirán predecir con exactitud las temperaturas futuras en una ventana de tiempo determinada, para luego poder variar la carga de los equipos en función de esta y trabajar siempre en la franja de temperatura de máximo rendimiento. En este proyecto se obtendrá un prototipo inicial que servirá de punto de partida para la implementación de un prototipo real con equipación de mayor capacidad de cómputo.

El apartado se divide en cuatro partes. Primero, una explicación del equipo utilizado para los experimentos. Segundo, una descripción de la carga de trabajo elegida para los equipos. Tercero, una explicación del sistema de monitorización de los equipos y el tipo de datos capturados con los que vamos a realizar nuestros modelos. Y por último, una descripción de los hiperparámetros y métricas que usaremos en los modelos. En el siguiente esquema se puede ver los pasos de la solución:

Figura 3.1: Esquema de los pasos de la solución.

3.1. Diseño del equipo realizado para las pruebas

El equipo utilizado, al tratarse de una primer prueba, no es un *rack* de un centro de datos o una bañera de gran tamaño para sumergir placas de altas prestaciones ya que el coste del material sería muy elevado. En vez de eso, se utilizará el siguiente material, más acorde a una fase de prototipado inicial:

- Pecera para el hogar [38], que servirá de recipiente para sumergir los equipos informáticos en el líquido dieléctrico. Es de pequeño tamaño, para que quepan los equipos y no desperdiciar más líquido del necesario. Además viene con tapa, para no dejar que el gas producido por la evaporación del líquido escape. Como son de cristal, podemos apreciar en todo momento lo que está ocurriendo.
- Seis Raspberry Pi modelo 3B+ [39], que serán nuestro equipo informático. Nos hemos decidido por estas ya que son baratas y fáciles de usar. Dentro de los diferentes modelos de Raspberry Pi nos hemos decantado por el 3B+ ya que se probó con un *benchmark* que era el modelo que más se calentaba, pudiendo llegar hasta los 70-90ºC dependiendo si se usa la GPU o la CPU. Se han utilizado tres

para formar un pequeño *cluster* el cual monitorizaremos, para ver cómo se afectan entre sí sus temperaturas al variar la carga. Las otras tres restantes tienen el único propósito de estar funcionando a su máximo rendimiento posible sin pausa para calentar el líquido.

- Seis tarjetas microSD [40], necesarias para usar la Raspberry Pi. Se les ha añadido la distribución Raspbian Lite para programar los equipos.
- Alimentadores de tres amperios [42], para que las Raspberry Pi funcionen de manera óptima.
- *Switch* y cables *ethernet* [41], las placas están conectadas a Internet para poder monitorizar los equipos remotamente.
- Líquido dieléctrico Vertrel XF de la multinacional química Chemours [43], ya mencionado en la introducción, es un líquido química y térmicamente estable (lo que asegura que no se degrade para que pueda actuar como refrigerante por tiempo indefinido), de baja toxicidad, y no inflamable que se ha usado hasta ahora como lubricante, secado, limpiador preciso de partículas micrométricas o como dieléctrico, entre otros. Nos hemos decido por este líquido en vez de otro (principalmente al Novec 7100 [7], el más usado por la industria en la actualidad), debido a dos factores principales, su punto de ebullición es más bajo (55ºC frente a 61ºC), lo que le hace alcanzar dos fases más rápidamente, y el precio por kilo, que supone un tercio de ahorro en el coste.
- Cinta aislante para sellar la tapa de la pecera y permitir que se escape el menor volumen de gas posible, una regleta para coordinar el encendido de los equipos, y separadores de métrica 2.5mm para juntar las Raspberry Pi lo más posible al formar el *cluster*.

El montaje del equipo completo es el siguiente. Primero se llena la pecera del líquido dieléctrico, lo mínimo suficiente para cubrir los dispositivos. Por otro lado, se monta el *cluster* de Raspberry Pi (tanto el monitorizado como el que no), se introducen las tarjetas micro SD y se juntan las placas con los separadores de forma paralela con todos los procesadores en la misma dirección (ver figura 3.2). Con las cargas de trabajo preparadas en cada equipo, se conecta el *switch* a la alimentación y los alimentadores a la regleta sin encender esta. Luego se conectan los cables *ethernet* al *switch* y a las Raspberry Pi además de conectarse estas a los alimentadores.

Con todo preparado, introducimos los dos *cluster* en la pecera con cuidado de colocarlos de manera vertical (en un centro de datos con un sistema de refrigeración como este, las placas están colocadas de manera vertical para favorecer el flujo de evaporación del líquido). Por último, colocamos la tapa de la pecera, sellamos los bordes con cinta aislante y encendemos la regleta (ver figura 3.3). El líquido que empieza a temperatura ambiente, irá subiendo de temperatura según pase el tiempo debido a la carga de las Raspberry Pi, con el objetivo de que llegue al punto de ebullición, se evapore, y luego se condense en la tapa de la pecera para que finalmente se precipite en un ciclo continuo totalmente pasivo.

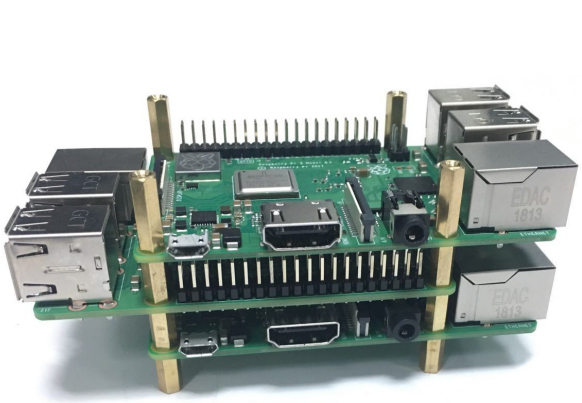


Figura 3.2 y 3.3: *Cluster* de los equipos y montaje final con la pecera medio llena.

3.2. Carga de trabajo utilizada en los equipos

Dividiremos este apartado en dos. Una primera relacionada con el tipo de carga que vamos a utilizar y qué se necesita para poder ejecutarla. Y una segunda parte para explicar el ejecutivo cíclico que ejecutan las Raspberry Pi.

3.2.1. Criptomonedas y minería de datos

Como mencionamos en la introducción, el mercado de las criptomonedas está en pleno auge y requieren en la actualidad de centros de datos para poder tener beneficios. Ejemplos como el visto de la empresa BitFury, donde dedican exclusivamente centros de datos para el minado de criptomonedas [45], no nos hace más que confirmar esta tendencia al alza en el sector. En la actualidad, las criptomonedas más importantes, como Bitcoin o Ethereum, son rentables solo usando GPUs. En nuestro caso, los equipos Raspberry Pi modelo 3B+ disponemos de una GPU incorporada, pero sus prestaciones son tan bajas, que ha dado numerosos casos de fallos al intentar minar criptomoneda en esta. Por ello, usaremos la CPU que disponen los equipos en vez de la GPU. Es importante destacar, que el objetivo de las cargas de las Raspberry Pi es generar distintos patrones de cargas reales que presenten perfiles reales de temperaturas en ejecución.

Los equipos tienen la ventaja de ser económicos, fáciles de utilizar y tener unas prestaciones aceptables. Por desgracia, el minado de criptomoneda exige una alta utilización, no siendo posible minar con todo tipo de mineros existentes que usan la CPU. Aún así, existen todavía unas pocas opciones disponibles que funcionan [46]. Nos decantamos por el minero CPUminer-multi, de Lucas Jones, ya que ofrece numerosas posibilidades como: usar varios algoritmos para minado, varios tipos de criptomonedas, variar número de cores en la ejecución…

Para usar CPUminer-multi hace falta primero, clonar e instalar en la Raspberry Pi su repositorio que se encuentra en GitHub [47] (es necesario la instalación de unas dependencias previas: *libcurl*, *jansson* y *openssl*). Una vez hecho esto, el segundo paso es crear una cuenta en MinerGate [48], una piscina de minado o *mining pool* para criptomonedas basadas en CryptoNote. Para minar es necesario usar piscinas de minado.

En la actualidad, al necesitar mayor generación de bloques para abastecer la demanda, el minado dejó de ser individual, porque los equipos con menos prestaciones ralentizaban la generación de los bloques. En las piscinas, muchos clientes diferentes contribuyen en la creación de un bloque, y cuando este es generado, se reparten los beneficios acorde a la participación en su creación.

Para participar en una piscina, es necesario crear una cuenta en un servicio como MinerGate con una cuenta de correo electrónico (al que están asociadas las ganancias del minado) y declarar en el minero CPUminer-multi la dirección tcp correspondiente a la piscina de la criptomoneda que se quiera minar. La criptomoneda que usaremos será Monero, y el algoritmo Cryptonight ya que es lo recomendado por MinerGate [57]. Se ejecutará en segundo plano para poder correr el minero dentro del *script* de Python del ejecutivo cíclico.

Lo que más variamos en la ejecución del minero son el número de cores utilizados por los equipos para minar, esto nos permite crear distintos perfiles de temperatura para las distintas utilizaciones del procesador, muy importante para luego poder variar la carga dinámicamente. Además de la utilización, otra característica que vamos a variar es la frecuencia del procesador, tiene dos disponibles: 600Mhz y 1.4Ghz. Para ello, vamos a hacer uso de la herramienta software cpufreq-set. Tanto el funcionamiento de esta como del minero se describen en el anexo D.

3.2.2. Ejecutivo cíclico para las cargas de los experimentos

Los modelos de temperatura que implementamos están diseñados para la Raspberry Pi situada en el medio del *cluster* de monitorización. Se decidió escoger la del medio ya que a esta le van afectar las temperaturas de las exteriores de forma asimétrica dado que la configuración del *cluster* es asimétrica (se hizo por aumentar la densidad lo máximo posible) y por lo tanto, la influencia de los lados no va a ser la misma. El diseño de la carga se basa en variar cuatro características: la temperatura de las Raspberry Pi situadas en los lados del *cluster*, la utilización y frecuencia. El objetivo de variar estas características es obtener más variedad en los datos capturados para que los modelos sean lo más robustos posibles.

Hasta ahora hemos explicado cómo variar la frecuencia y la utilización. Para variar la temperatura de las Raspberry Pi exteriores, ejecutaremos las cargas de cada placa en paralelo, creando cuatro instantes temporales. En ellos, cada placa puede estar haciendo dos cosas, o ejecutando el minero de una manera específica para calentarse, o bien durmiendo hasta el siguiente instante temporal. De esta manera, podemos ver cómo afecta las temperaturas de los lados a la Raspberry Pi central. En la siguiente figura se pueden observar los cuatro instantes temporales y los equipos minando en cada uno de ellos, representados con una flecha:

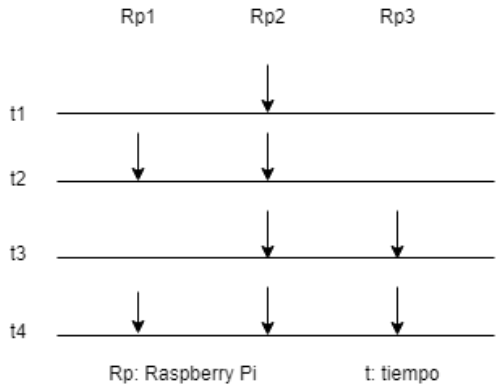


Figura 3.5: Ejecución paralela de los equipos.

Para la planificación y ejecución de la carga se ha decidido implementar con un ejecutivo cíclico mediante un script de Python. El funcionamiento de este es el siguiente:

1. Se establece la frecuencia de la CPU a uno de los tres modos posibles (anexo D).
2. Se ejecutan los cuatro instantes temporales mencionados anteriormente. Tenemos dos opciones:
    a. Que esté minando: en este caso estará minando un tiempo fijo y luego dormirá otro. Se repetirá la ejecución para cada uno de los cores en orden ascendente para un total de cuatro repeticiones.
    b. Que esté durmiendo: en este caso esperará a que las cuatro repeticiones terminen para pasar al siguiente instante temporal.
3. Una vez se han ejecutado los cuatro instantes temporales se repite para las otras dos estrategias de frecuencia restantes.

En la siguiente figura podemos observar el ejecutivo de una de las tres Raspberry Pi en forma de diagrama de flujo para facilitar su entendimiento:

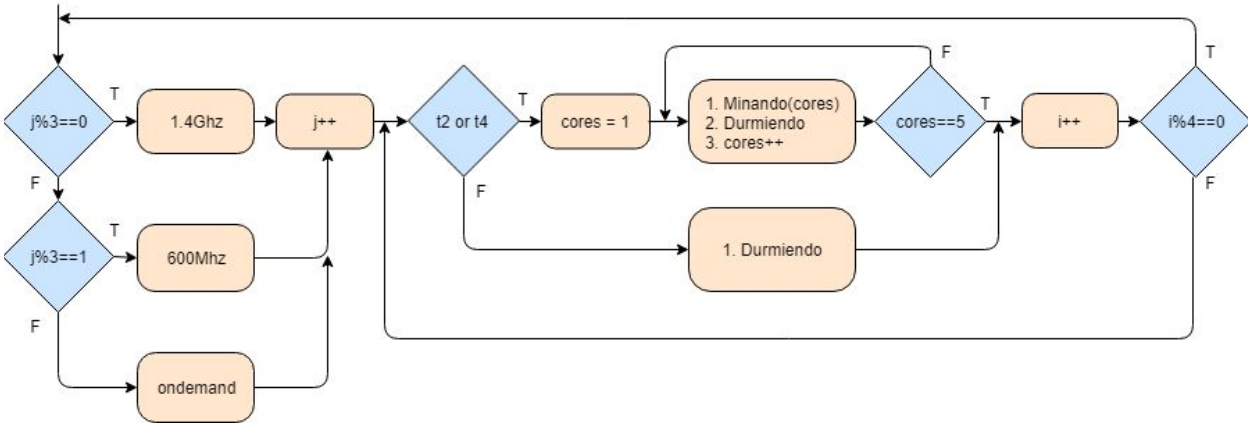


Figura 3.6: Diagrama de flujo del ejecutivo cíclico de la Raspberry Pi 1.

Finalmente, gracias a las tres Raspberry Pi extra con la única función de calentar, se consiguen un mayor número de perfiles de temperatura a la hora de ejecutar esta carga, y con ello que el conjunto de datos usado para los modelos sea más variado y las soluciones a las que converjan sean más robusta.

3.3. Monitorización y estructura de los datos capturados

Una vez tenemos el equipo montado y la carga preparada, necesitamos capturar los datos de las diferentes características para realizar nuestros modelos predictivos. Para ello, hacemos uso de una herramienta de software libre de Linux que recoge estadísticas de los equipos tales como, temperatura, frecuencia, utilización, etc. Se denominada *collectd*, es un demonio de Unix que recolecta, transfiere y almacena datos de los equipos informáticos y de red [44]. Se diseñó con el objetivo de ayudar a administradores de sistemas en el mantenimiento de los equipos.

Al instalarse solo se implementa la infraestructura para manejar datos, para recoger las diferentes características es necesario instalar *plug-ins*. Uno de los *plug-ins* que viene preinstalado es el de red, que nos permite monitorizar el equipo de manera remota, usando el protocolo UDP y soporta tanto IPv4 e IPv6. *Collectd* recolecta la información y la almacena, pero no representa los datos de una manera gráfica para facilitar su entendimiento. Para solventar esto se utiliza Graphite. Esta es una herramienta software que usa un sistema en tiempo real de gráficas. Las características que maneja *collectd* de la Raspberry Pi y nos resultan de utilidad para realizar nuestros modelos de temperatura son:

- Temperatura de la CPU: Usaremos tanto la temperatura de la Raspberry Pi que queramos modelar (es tanto una características como los datos etiquetados que necesita la red neuronal para comparar con las predicciones y poder ajustarse), como las otras dos, ya que afectan a la temperatura de la principal.
- Utilización de la CPU: Nos servirá para saber cómo varía la temperatura dependiendo si se usan uno, dos, tres o cuatro cores. Usaremos la utilización media de los cuatro cores y no la individual. En *collectd* hay muchos tipos de utilización, la que se ejecuta en primer plano se llama *user* y suele ser la que se utiliza. Pero como hemos mencionado, la carga está ejecutándose en segundo plano, con lo que se hará uso de la utilización tipo *nice* que mide la utilización de las tareas de menor prioridad (las ejecutadas en segundo plano).
- Frecuencia de la CPU: como la utilización, es otro parámetro que variamos en la carga para ver cómo afecta a la temperatura. Utilizaremos dos estrategias para variar la frecuencia (*ondemand* y *userspace*) .

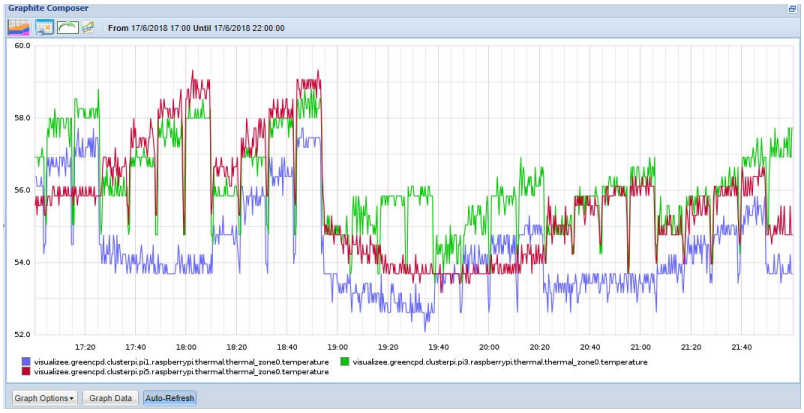


Figura 3.7: Dashboard de Graphite usando *collectd*.

Para obtener los datos de la gráfica de forma numérica y poder usarlos luego en los modelos, se eligen las características deseadas y se pintan en la pantalla. Luego se escoge la franja horaria querida, y se cambia el formato de los datos en la url de jpg a csv. Finalmente, se ejecuta un pequeño script escrito en BASH para parsear los datos y tenerlos listos para que los *scripts* de los modelos puedan interpretarlos.

La estructura final de los datos consiste en un archivo csv de cinco columnas, cada una de estas corresponde a cada una de las tres temperaturas de la CPU en cada Raspberry Pi, la media de la utilización de la CPU de todos los cores (tipo *nice*) y la frecuencia de uno de los cores ya que todos funcionan de la misma manera, estas dos últimas características pertenecen a la Raspberry Pi situada en el medio del *cluster*, como hemos comentado ya en otros apartados, los modelos de temperatura están diseñados para esta. Se ha decidido tomar los datos cada diez segundos para no sobrecargar los modelos diseñando ventanas para predecir más pequeñas en número pero de igual rango temporal, ya que la temperatura no varía tanto como para ser necesario tomarla con mayor frecuencia.

3.4. Herramientas de los modelos predictivos

Una vez explicado el funcionamiento de los equipos, pruebas realizadas y cómo se capturan los datos de estas, solo falta realizar los modelos predictivos de temperatura. Este apartado sirve para explicar cómo se trabaja con ellos. Se divide en dos partes. Primero una explicación de las herramientas software utilizadas, Jupyter Notebook, Tensorflow y Keras. Y segundo, los hiperparámetros y métricas de los modelos.

3.4.1. Software utilizado para el desarrollo de los modelos

Para desarrollar nuestro modelos utilizaremos la plataforma software Anaconda [51], que incluye diferentes herramientas software. En nuestro caso, haremos uso de la herramienta Jupyter Notebook [52] para editar, compilar y ejecutar nuestros modelos. Utilizamos Jupyter Notebook ya que es una aplicación web de código libre que permite crear documentos separados en módulos de diferentes tipos. Los dos que nos interesan son los de texto narrativo, para documentar el código, y los de código de programación, para desarrollar los modelos.

El lenguaje de programación que utilizaremos será Python debido a las librerías de *Machine Learning* que dispone, TensorFlow y Keras. TensorFlow [53] es una librería de código abierto en diferentes lenguajes (Python, C…) y válida en diferentes tipos de sistemas (Linux, MacOS, Android…) que permite la programación por grafos, herramienta muy útil para construir redes neuronales. Fue desarrollada por Google Brain (departamento de inteligencia artificial de Google) y desde su lanzamiento fue un éxito ya que facilitó en gran medida el desarrollo de modelos basados en redes neuronales.

Keras [54] es otra librería de más alto nivel capaz de correr encima de otras como Microsoft Cognitive Tooklit, Theano, MXNet o como en nuestro caso TensorFlow. La ventaja que tiene Keras es que su diseño está enfocado en conseguir una familiarización rápida con

las redes neuronales gracias a que es fácil de usar, modular y ser extensible. Por esto nos decidimos a usar Keras en el proyecto, para tardar lo menos posible en familiarizarnos con la programación de redes neuronales. En el anexo D se encuentran las librerías de Python usadas para el desarrollo de los modelos.

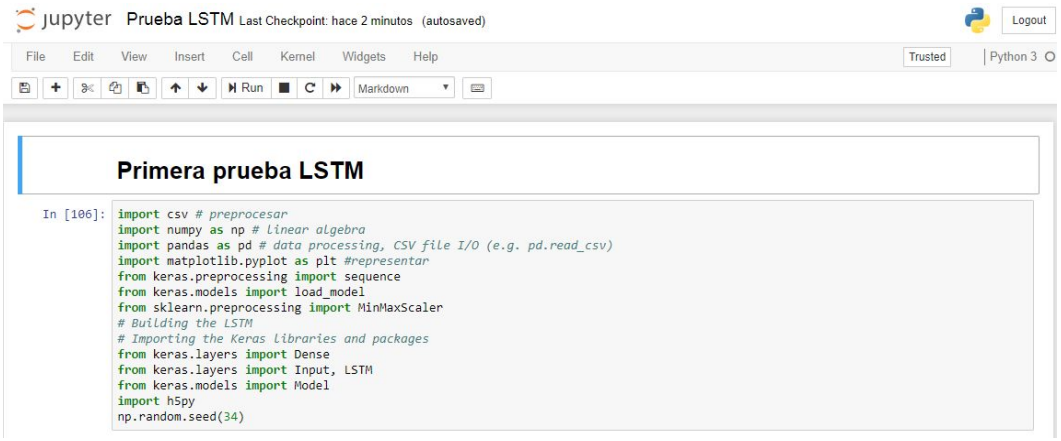


Figura 3.8: Entorno Jupyter Notebook donde se desarrollan los modelos.

3.4.2. Hiperparámetros y métricas

Como hemos mencionado anteriormente, que una red neuronal sea más o menos adecuada para resolver un determinado problema depende de como modifiquemos sus hiperparametros. A continuación se enumeran y explican los diferentes tipos de hiperparametros que disponemos para el desarrollo de nuestros modelos:

- Número de *features* o características: Las diferentes entradas a la red neuronal. Estas son los cinco características que estamos monitorizando: las tres temperaturas, la utilización y la frecuencia. No siempre utilizar todas podría ser positivo, ya que si no afectan al resultado, sobredimensionan la red, haciendo que el procesado de esta sea más lento.
- Tamaño del *batch* o lote: Al entrenar o al realizar los test, los datos tienen que ser divididos en pequeños lotes ya que los equipos informáticos no pueden procesar grandes *datasets* de una sola iteración.
- Número de *epochs* o simulaciones: Número de veces que la red neuronal procesa un mismo *dataset* durante el entrenamiento.
- Tamaño de la ventana temporal: Rango de predicción de la red neuronal. Si tiene un valor alto la red podrá predecir con más antelación. Por contra su tamaño aumentará haciendo más lento el procesado, además de que podría dejar de predecir con exactitud.
- Porcentaje del test: Porcentaje del conjunto de datos total que se utiliza para test.
- Tipo de preescalado: Forma de escalar los datos de entrada antes de introducirlos en la red. Utilizamos escalado estándar o escalado lineal [55].
- Número de capas: Total de capas que tiene la red neuronal.
- Tipo de capas, neuronas o nodos: Usaremos nodos *fully connected*, LSTM y GRU. Además, utilizaremos un tipo de capa llamada *dropout* que ayuda a solucionar problemas de *overfitting* o sobreajuste (aprender demasiado de los datos de

entrenamiento sin ser capaz de generalizar una solución que permita predecir con exactitud datos nuevos). Estas aleatoriamente reinician el valor de un porcentaje de neuronas que se indique.

- Número de neuronas o nodos por capa: Total de neuronas o nodos por capa. Como norma general, suele ser necesario mayor número contra mayor es el tamaño de los conjuntos de datos.
- Función de la pérdida o *loss*: Función que calcula el error existente entre la predicción de la red y el dato etiquetado real. Existen muchos tipos: *mean squared error*, *mean absolute percentage error*, *hinge*, etc.
- Función de optimización u *optimizer*: Función que actualiza los pesos y biases de la red dependiendo de cuánto haya sido el error. En Keras tenemos varias disponibles: SGD, *RMSprop*, *Adagrad*, *Adam*, etc.
- Grado de aprendizaje o *learning rate*: Indica cuán rápido aprende la red, un valor pequeño podría tardar demasiado en llegar a la solución óptima y un valor grande haría que hubiese saltos muy bruscos no siendo capaz de converger.

Para medir la calidad del modelo se necesitan diferentes tipos de métricas dependiendo del tipo de problema que nos enfrentemos. En nuestro caso, para redes neuronales recurrentes que predicen series temporales, se trata de un problema de regresión. Existen muchas métricas para estos problemas, nosotros utilizaremos las siguientes:

- Error medio absoluto o MAE: Mide la media del error (la diferencia entre la temperatura observada, $T_{real\ i}$, y la predicha, $T_{pred\ i}$) en valor absoluto. En esta métrica todos los errores tienen el mismo peso. Se mide grados centígrados.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} | T_{real\ i} - T_{pred\ i} |$$

- Desviación estándar de la diferencia o SD: Medida de dispersión que representa cuán dispersos están los errores (expresados como la diferencia del valor absoluto entre $T_{real\ i}$ y $T_{pred\ i}$). La presentaremos juntos al MAE. Se mide grados centígrados.

$$SD = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (Error_i - \overline{Error})^2}$$

- Error porcentual medio absoluto o MAPE: Mide el error absoluto (MAE) en términos porcentuales.

$$MAPE = \frac{100\%}{N}\sum_{i=1}^{N} | \frac{T_{real\ i} - T_{pred\ i}}{T_{real\ i}} |$$

- Error máximo (Emáx): Diferencia máxima entre todos los errores calculados.

$$Emáx = max_{i=1}^{N} | T_{real\ i} - T_{pred\ i} |$$

- Desviación de la raíz cuadrática media *o* RMSD: Es la raíz de la media de las diferencias al cuadrado. A diferencia de MAE, no trata todos los errores de igual manera, penalizando aquellos más elevados. Se mide grados centígrados.

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(T_{real\,i} - T_{pred\,i})^2}$$

- Coeficiente de determinación o R^2: Determina lo bien que puede replicar el modelo la variable que pretende predecir. Varía entre cero y uno aunque nosotros lo expresaremos porcentualmente. Cuanto más alto, más se asemeja la predicción a lo observado, llegando a ser igual que esta cuando vale uno.

$$R^2 = \frac{\sum_{i=1}^{N}\left(T_{real\,i} - T_{pred\,i} \right)^2}{\sum_{i=1}^{N}(\bar{T}_{real} - T_{real\,i})^2}$$

4. Modelos predictivos

En este apartado se detallan los experimentos realizados para modelar la temperatura. Partiremos de un modelo inicial que sirve de base en los modelos posteriores para fijar los hiperparámetros con menor sensibilidad o que funcionan independiente a los demás. Estos los dejaremos fijos durante el desarrollo de los modelos posteriores. Después, realizaremos para los tres tipos de redes recurrentes vistas (*fully connecte*d, GRU y LSTM) un proceso que consta de las siguientes cuatro partes:

1. Arquitectura: Buscamos la arquitectura más adecuada posible variando el número de neuronas y capas.
2. Entrenamiento: Entrenamos la mejor arquitectura que encontremos para diferentes tamaños del *batch* y número de *epochs*. Se usará el método de optimización de pares de hiperparámetros *grid search* [60] en el que se divide el espacio y se escogen puntos siguiendo un patrón (al contrario que random search que coge puntos aleatorios). En nuestro caso seguiremos un patrón basado en el tiempo de ejecución, contra más alto sea el número de *epochs* y menor el tamaño del *batch* más lento ejecutara pero mejores resultados teóricamente dará si se disponen de los suficientes datos.

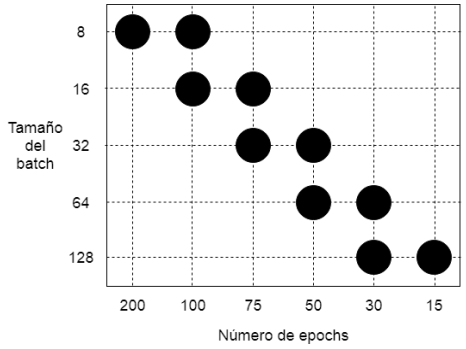


Figura 4.1: Método de optimización *grid search* con el patrón utilizado.

3. Evaluación: Probaremos cómo de grande puede ser la ventana temporal, o lo que es lo mismo, con cuánta antelación puede predecir para unos límites de precisión viables.
4. Optimización: Añadimos capas de *dropout* para comprobar que pueda mejorar el modelo y reduciremos la dimensionalidad de la red quitando características para ver comprobar si le metemos a la red información no relevante.

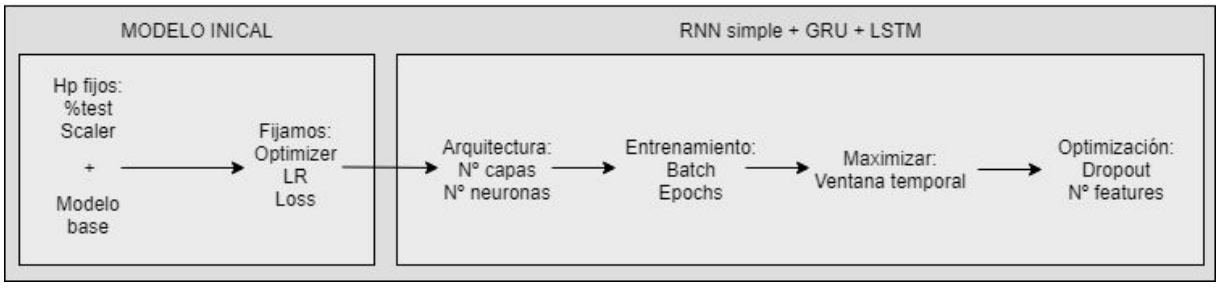


Figura 4.2: Esquema de los experimentos desarrollados.

4.1. Modelo inicial

Para el desarrollo de los modelos, partiremos de uno inicial en el que fijamos los parámetros de menor sensibilidad y buscaremos fijar los hiperparámetos más independientes de otros hiperparámetros. Un dato importante a tener en cuenta antes de empezar, es que se decidió mezclar los datos (*shuffle*), o para ser más exactos las ventanas temporales, a la hora de entrenar los modelos. Esto nos sirve para que modelos sean capaces de generalizar más y den soluciones más robustas y preparadas a los cambios.

Dicho esto empezamos con los hiperparámetros. Primero fijamos el porcentaje de test en un 10% (valor bajo respecto al de entrenamiento). Si pensamos en modelos futuros, se dispondrá un volumen enorme de datos con lo que el tamaño de los datos de entrenamiento siempre será muy superior al de test. Segundo, fijamos el preescalado, probamos varias estructuras con diferentes hiperparámetros aleatorios, y las dos técnicas de preescalado que disponíamos, distribución lineal entre cero y uno y distribución estándar, daban resultados parecidos, nos quedamos con la primera.

Para los demás hiperparámetros que variamos en los siguientes apartados pero aquí necesitamos fijarlos, pensamos en utilizar el ejemplo de redes neuronales recurrentes para series temporales del curso de inteligencia artificial de IBM [59] pero observamos que la ejecución era muy lenta y la red estaba sobredimensionada. Por ello, se ha optado por reducir los valores de varios hiperparámetros respecto al original, dado que no disponemos de un gran volumen de datos para crear grandes estructuras (sobre 4000 trazas y cinco características) y con ello ahorramos mucho tiempo de ejecución. Destacar que se ha decidido usar capas con nodos GRU ya que como vimos en el estado del arte que para *datasets* pequeños suelen funcionar mejor. El modelo inicial queda:

Modelo inicial						
Test	Preescalado	Arquitetura	Tipo	*Epochs*	*Batch*	Ventana
10%	Lineal {0,1}	1ª Capa: 40 2ª Capa: 20	GRU	30	32	60 segundos

Tabla 4.1: Configuración del modelo inicial.

Los hiperparámetros que variamos en este apartado son el optimizador (además de su *learning rate*) y la función de *loss* que suelen ser dependientes del problema pero independientes de los demás hiperparámetros. Keras ofrece distintos optimizadores y funciones de *loss*, habiendo algunos que no funcionan para problemas de regresión como el nuestro, sino que están pensados para otro tipo de problemas, por ejemplo de clasificación. En la siguiente tablas solo se representan los que han tenido resultados relevantes durante las ejecuciones. Primero probaremos las distintas funciones de *loss*, y como función de optimización fijamos *RMSProp* recomendada por Keras [58] para redes neuronales recurrentes.

	MAE±SD	MAPE	Emáx	RMSD	R^2
MSE	1.91±1.35	3.33%	6.17	2.34	17.17%
MAE	**1.11±1.11**	**1.97%**	6.87	**1.57**	**62.78%**
MAPE	1.98±1.48	3.52%	7.56	2.47	7.48%
MSLE	1.38±1.15	2.43%	5.41	1.8	51.11%
Logcosh	2.07±1.29	3.63%	5.11	2.44	9.99%
Poisson	1.52±1.15	2.69%	**4.51**	1.91	44.92%

Tabla 4.2: Modelo inicial para diferentes funciones de *loss*.

Después de realizar varias ejecuciones para cada una de las funciones de loss, vemos que la mejor es el error absoluto medio o MAE, solo siendo superado en el error máximo por poisson. Decidimos fijar la función de *loss* como MAE y ejecutar las diferentes funciones de optimización:

	MAE±SD	MAPE	Emáx	RMSD	R^2
SGD	1.1±1.18	2.0%	6.23	1.62	60.42%
RMSProp	1.5±1.16	2.63%	**5.26**	1.9	45.66%
Adagrad	**0.9±1.1**	**1.62%**	6.93	**1.42**	**69.53%**
Adadelta	1.06±0.96	1.9%	5.7	1.43	69.14%
Adam	0.99±1.08	1.78%	6.48	1.46	67.79%
Adamax	1.16±1.03	2.09%	6.49	1.55	63.61%
Nadam	1.47±1.19	2.67%	6.16	1.9	45.59%

Tabla 4.3: Modelo inicial para diferentes funciones de optimización.

Observamos que el optimizador *Adagrad* presenta los mejores resultado siendo solo superado en el error máximo por *RMSProp*. Por último, variamos el valor del *learning rate* de *Adagrad* para encontrar el óptimo:

	MAE±SD	MAPE	Emáx	RMSD	R^2
lr=0.1	1.24±1.12	2.23%	**5.67**	1.67	57.85%
lr=0.05	1.05±0.97	1.89%	6.18	**1.43**	**69.23%**
lr=0.01	**0.94±1.13**	**1.68%**	7.48	1.47	67.39%
lr=0.005	0.96±1.14	1.71%	6.9	1.49	66.51%
lr=0.001	1.66±1.38	2.95%	7.58	2.16	29.43%

Tabla 4.4: Modelo inicial para diferentes *learning rates* con *Adagrad*.

En este caso hay tres valores muy parejos. Para elegir uno de ellos nos basamos en la documentación de Keras que nos recomienda un valor de learning rate de 0.1 [58].

Finalmente, los hiperparámetros fijos para los modelos de redes recurrentes simples, GRU y LSTM son los siguientes:

Hiperparámetros fijos				
Test	Preescalado	*Loss*	Optimizador	*Learning rate*
10%	Lineal {0,1}	MAE	*Adagrad*	0.01

Tabla 4.5: Hiperparámetros para los siguientes modelos.

4.2. *Fully connected*

Como explicamos en la introducción del apartado, el desarrollo que seguiremos consta de cuatro partes: arquitectura, entrenamiento, evaluación y optimización para quedarnos con el modelo que mejor resultados nos ofrezca.

4.2.1. Arquitectura

En redes recurrentes simples tendrémos más número de neuronas que en las demás ya que sus neuronas son más simples (tienen menos parámetros). En lo referente a las capas nos centraremos en solo una y dos capas ya que en experimentos con tres los resultados fueron negativos debido al sobredimensionamiento de la red para el volumen de datos reducido que tenemos. Los hiperparámetros que variamos en los siguientes subapartados los fijamos iguales que en el modelo inicial (*batch*, *epoch* y ventana temporal). Se denota como capa uno (C1) la capa inmediata a la entrada de la red, aumentando el índice para capas superiores.

	MAE±SD	MAPE	Emáx	RMSD	R^2
C1: 64	1.01±0.87	1.78%	**6.54**	1.33	73.36%
C1: 96	0.86±0.84	1.54%	6.98	1.2	78.21%
C1: 128	**0.75±0.79**	**1.34%**	**6.54**	**1.09**	**81.89%**
C1: 192	1.09±0.9	1.95%	7.37	1.41	69.77%
C1: 256	1.29±1.2	2.31%	15.75	1.76	52.97%
C1: 16 C2: 8	1.17±1.04	2.11%	6.05	1.57	62.82%
C1: 32 C2: 16	1.54±1.19	2.77%	8.08	1.95	42.64%
C1: 64 C2: 32	1.62±1.26	2.94%	5.93	2.06	36.1%
C1: 126 C2: 64	1.52±1.14	2.74%	6.05	1.9	45.28%
C1: 256 C2: 128	2.66±1.91	4.84%	8.91	3.27	0%

Tabla 4.6: Modelos *fully connected* para diferentes arquitecturas.

Observamos que la estructura de una capa con 128 neuronas es la mejor con cierto margen a las demás. Por ello, fijamos esta arquitectura para el siguiente paso. Destacar

también el valor de R^2 en la última siendo nulo, con esto se comprueba que aumentar demasiado la dimensionalidad de la red hace que los modelos dejen de funcionar.

4.2.2. Entrenamiento

Probamos distinto número de *epochs* y tamaños de *batch* conjuntamente con el método de optimización de pareja de hiperparámetros *grid search*. Se ha añadido una columna de tiempo de ejecución en segundos T, para analizar temporalmente el entrenamiento ya que está muy ligado con estos dos hiperparámetros. Se denota *batch* como "b" y *epoch* como "e".

	T	MAE±SD	MAPE	Emáx	RMSD	R^2
b=8 e=200	243.23	0.88±1.1	1.57%	8.2	1.41	70.17%
b=8 e=100	123.59	1.29±1.06	2.3%	**5.09**	1.67	58.4%
b=16 e=100	70.31	0.74±0.88	**1.32%**	7.0	1.15	80.21%
b=16 e=75	53.3	0.83±0.87	1.48%	7.35	1.2	78.42%
b=32 e=75	29.79	**0.74±0.8**	**1.32%**	7.2	**1.09**	**82.13%**
b=32 e=50	21.71	0.79±0.81	1.4%	7.11	1.13	80.65%
b=64 e=50	14.18	1.0±0.88	1.77%	6.19	1.33	73.34%
b=64 e=30	8.32	0.81±0.8	1.45%	6.16	1.13	80.59%
b=128 e=30	6.2	0.95±0.81	1.68%	6.65	1.25	77.86%
b=128 e=15	**3.53**	1.27±0.92	2.26%	7.4	1.57	65.18%

Tabla 4.7: Modelos *fully connected* para diferentes tamaños de *batch* y número de *epochs*.

Observamos que existen muchas combinaciones con rendimientos parecidos pero destacan dos, b=16, e=100 y b=32, e=75. Nos decidimos por la segunda, ya que es ligeramente mejor y además es más del doble de rápida que la primera.

4.2.3. Evaluación

En este apartado veremos los límites que tiene el modelo para predecir con exactitud variando su ventana temporal (expresada en segundos). Los resultados son los siguientes:

	MAE±SD	MAPE	Emáx	RMSD	R^2
60	0.74±0.8	1.32%	7.2	1.09	82.13%
90	0.71±0.76	1.27%	6.76	1.04	83.54%
120	1.04±0.85	1.86%	5.47	1.34	69.98%
150	1.15±0.83	2.06%	5.58	1.42	63.2%
180	1.34±1.02	2.35%	6.21	1.69	50.13%
210	1.65±1.15	2.92%	7.55	2.01	28.05%

240	1.85±1.12	3.27%	5.24	2.17	18.59%
270	1.63±1.14	2.89%	5.56	1.99	32.71%
300	1.85±1.37	3.29%	7.44	2.3	12.91%
330	1.64±1.44	2.95%	7.37	2.18	23.73%
360	1.45±1.12	2.61%	6.38	1.83	40.23%
390	1.7±0.91	3.0%	5.06	1.93	32.69%
420	1.47±0.96	2.62%	5.61	1.76	45.01%

Tabla 4.8: Modelo *fully connected* para diferentes tamaños de ventana de predicción.

Aunque el error absoluto medio no difiere mucho, es interesante fijarse en el valor de la métrica R^2 que nos indica lo que se asemeja la predicción a la temperatura observada. Podemos diferenciar cuatro grupos. Un primero muy preciso hasta el minuto y medio, un segundo menos preciso pero aún así con predicciones válidas hasta los tres minutos. un tercero con predicciones no precisas pero que sigue aprendiendo la tendencia de la temperatura hasta cuatro minutos y medio y un cuarto que solo es capaz de seguir los cambios de temperatura vagamente.

4.2.4. Optimización

En este apartado probaremos si el uso de capas *dropout* para regularizar mejora la predicción de algunas ventanas de los grupos mencionados. Después, reduciremos las características del *dataset* para observar si reduciendo la dimensionalidad de la red los resultados se mantienen, y con ello eliminando información que no es relevante para el entrenamiento. Para reducir el tamaño de la tabla, sólo se mostrará el resultado de la ejecución con el porcentaje de *dropout* más adecuado para cada ventana. Los resultados introduciendo esta capa entre la capa de neuronas recurrentes simple y la salida son:

	Dropout	MAE±SD	MAPE	Emáx	RMSD	R^2
60	0.12	0.7±0.78	1.25%	6.78	1.05	83.33%
180	0.05	0.79±0.79	1.42%	6.43	1.12	77.92%
240	0.05	1.35±1.09	2.36%	6.16	1.73	47.9%
360	0.05	1.1±0.82	1.94%	6.38	1.37	66.56%
420	0.05	1.13±0.83	2.01%	6.11	1.4	65.05%

Tabla 4.9: Modelo *fully connected* con *dropout*.

Observamos que, menos la ventana de 60 segundos que se mantiene parecida, las demás ventanas mejoran considerablemente con el uso de *dropout*, esto podría deberse a un problema de overfitting. Ahora, reduciremos la dimensionalidad del mejor modelo conseguido (ventana temporal de 90 segundos sin *dropout*). Se eliminan por separado la utilización, la frecuencia y las temperaturas exteriores del *cluster* (conjuntamente). Se denota la temperatura de la Raspberry pi x con Tx. Los resultados son:

	MAE±SD	MAPE	Emáx	RMSD	R^2
Sin T1 y T3	0.76±0.89	1.37%	6.68	1.17	79.31%
Sin frecuencia	0.91±0.86	1.62%	6.59	1.25	76.42%
Sin utilización	0.72±0.78	1.28%	6.49	1.06	83.01%

Tabla 4.10: Modelo *fully connected* reduciendo características.

Podemos concluir que tanto la frecuencia como las temperaturas exteriores aportan al modelo información relevante aunque no hace que pierda una precisión muy elevada. Por otro lado, la utilización no aporta información prácticamente al modelo ya que los errores no varían al eliminarla del *dataset*.

4.3. GRU

De igual manera que con las redes recurrentes simples, usaremos el mismo método de optimización.

4.3.1. Arquitectura

A diferencia de las redes recurrentes simples, los nodos GRU tienen mayor número de parámetros por lo que usaremos menor número de nodos por capa. De igual manera que en el caso anterior, no se hará uso de arquitectura mayores de dos capas y se fijan los hiperparámetros del modelo inicial. Los resultados de las diferentes arquitecturas son los siguientes:

	MAE±SD	MAPE	Emáx	RMSD	R^2
C1:8	0.74±0.94	1.33%	7.04	1.19	78.52%
C1:16	0.63±0.9	1.14%	7.07	1.1	81.81%
C1:32	**0.62±0.86**	**1.11%**	7.12	**1.06**	**83.1%**
C1:64	0.65±0.91	1.17%	7.21	1.12	80.96%
C1:128	0.65±0.88	1.17%	**6.82**	1.09	81.97%
C1:8 C2 4	0.72±0.95	1.29%	6.97	1.19	78.52%
C1:16 C2 8	0.64±0.87	1.14%	7.11	1.08	82.32%
C1:32 C2:16	0.65±0.91	1.18%	6.93	1.12	80.87%
C1:64 C2 32	0.67±0.93	1.22%	6.99	1.15	79.98%
C1:128 C2:64	0.68±0.9	1.22%	7.01	1.12	80.9%

Tabla 4.11: Modelos GRU para diferentes arquitecturas.

Observamos que todos las arquitecturas presentan grandes resultados iniciales, destacando levemente la tercera estructura con una sola capa y 32 nodos. Por esto y por ser una configuración de dimensionalidad reducida es la estructura elegida.

4.3.2. Entrenamiento

Haciendo uso del método *grid search* para pares de hiperparámetros, se obtienen los siguientes resultados:

	T	MAE±SD	MAPE	Emáx	RMSD	R^2
b=16 e=100	84.22	0.7±0.96	1.26%	**6.73**	1.19	78.71%
b=16 e=75	60.44	0.71±1.04	1.29%	7.02	1.26	76.2%
b=32 e=75	31.42	0.65±0.92	1.17%	7.04	1.13	80.86%
b=32 e=50	20.62	0.64±0.91	1.16%	7.07	1.11	81.25%
b=64 e=50	12.29	0.65±0.91	1.17%	7.12	1.12	81.14%
b=64 e=30	7.35	0.65±0.88	1.17%	6.98	1.09	81.99%
b=128 e=30	4.99	0.66±0.87	1.19%	6.92	1.09	83.31%
b=128 e=15	2.78	**0.61±0.84**	**1.09%**	7.18	**1.04**	**84.73%**
b=192 e=15	2.29	0.63±0.88	1.15%	7.21	1.08	82.94%
b=192 e=10	**2.06**	0.7±0.83	1.25%	6.86	1.09	83.38%

Tabla 4.12: Modelos GRU para diferentes tamaños de *batch* y número de *epochs*.

Observamos que contra menor tamaño del *batch* y número de *epochs* peor es el resultado y más lenta es la ejecución. El mejor valor encontrado tampoco se encuentra en la combinación de mayor tamaño del *batch* y menor número se *epoch*, sino que se alcanza antes para un tamaño del batch de 128 y un número de *epochs* igual a 15.

4.3.3. Evaluación

Una vez tenemos el modelo más adecuado, probamos diferentes ventanas para ver los límites de predicción que tiene. Los resultados son los siguientes:

	MAE±SD	MAPE	Emáx	RMSD	R^2
60	0.61±0.84	1.09%	7.18	1.04	84.73%
90	0.66±0.9	1.19%	6.92	1.12	82.52%
120	0.77±0.92	1.4%	6.4	1.21	77.96%
150	0.85±0.87	1.53%	6.45	1.22	76.09%
180	0.73±0.83	1.31%	6.52	1.1	80.28%
210	1.04±0.91	1.86%	6.11	1.38	69.0%
240	1.01±0.91	1.83%	5.89	1.36	69.53%
270	1.18±0.93	2.11%	6.42	1.51	62.58%
300	1.24±1.05	2.22%	6.76	1.63	55.61%

330	1.17±1.04	2.08%	7.55	1.56	58.38%
360	1.14±0.88	2.01%	6.96	1.44	60.52%
390	1.9±1.23	3.29%	6.54	2.26	1.23%
420	1.88±0.98	3.29%	6.52	2.12	14.9%

Tabla 4.13: Modelo GRU para diferentes tamaños de ventana de predicción.

Fijándonos en la métrica R^2 los resultados podemos clasificar la precisión en tres grupos. Un primero con predicción entre un minuto y tres minutos que consigue una gran precisión en la predicción. Un segundo entre tres y seis minutos que no es tan preciso como antes pero puede seguir la tendencias de la temperatura todavía. Un tercer grupo a partir de los cinco minutos que no consigue predecir la tendencia de la temperatura.

4.3.4. Optimización

Utilizaremos varias ventanas temporales de cada grupo mencionado anteriormente para ver cómo afecta a diferentes niveles añadir capas *dropout* a la simulación. De igual manera que en redes recurrentes simples solo se indica el mejor porcentaje de *dropout* usado. Estos son los resultados:

	Dropout	MAE±SD	MAPE	Emáx	RMSD	R^2
60	0.12	0.6±0.83	1.08%	7.1	1.02	85.23%
180	0.05	0.81±0.85	1.46%	6.62	1.17	77.72%
240	0.12	0.88±0.88	1.59%	5.96	1.24	74.73%
360	0.05	0.96±0.85	1.69%	7.05	1.28	68.82%
420	0.05	1.09±0.91	1.95%	5.94	1.42	61.72%

Tabla 4.14: Modelo GRU con *dropout*.

Observamos que todos los modelos mejoran menos el segundo, siendo más llamativa está subida en los modelos de ventana temporal de mayor tamaño. Destacar que la ventana de siete minutos ha pasado de no predecir a ser una ventana con un nivel de predicción aceptable. Por último, probaremos a reducir la dimensionalidad del mejor modelo (ventana de un minuto con *dropout*) para ver si hay características que no aportan información a la red:

	MAE±SD	MAPE	Emáx	RMSD	R^2
Sin T1 y T3	0.85±0.93	1.55%	7.53	7.53	77.66%
Sin frecuencia	1.01±0.86	1.81%	6.47	1.32	75.4%
Sin utilización	0.94±0.83	1.68%	6.55	1.26	77.71%

Tabla 4.15: Modelo GRU reduciendo características.

Observamos que la ausencia de las características hace que la predicción empeoren pero aun así sigue resultando bastante precisa.

4.4. LSTM

Por último las redes LSTM, como hasta ahora, seguiremos usando el mismo método de optimización.

4.4.1. Arquitectura

Al tratarse de los nodos con más parámetros entre los tres, usaremos aún menos nodos por capa. Los resultados de las diferentes arquitecturas son los siguientes:

	MAE±SD	MAPE	Emáx	RMSD	R^2
C1:4	1.1±0.93	1.93%	7.66	1.45	68.36%
C1:8	**0.62±0.84**	**1.1%**	7.08	**1.04**	**83.55%**
C1:16	0.65±0.89	1.17%	7.43	1.1	81.61%
C1:32	0.64±0.86	1.14%	7.08	1.07	82.63%
C1:64	0.64±0.86	1.14%	7.21	1.07	82.67%
C1:8 C2 4	0.7±1.02	1.25%	7.58	1.24	76.93%
C1:16 C2 8	0.7±1.01	1.27%	6.91	1.23	77.16%
C1:32 C2:16	0.67±0.93	1.21%	7.03	1.15	80.01%
C1:48 C2 24	0.7±0.97	1.26%	**7.02**	1.19	78.6%
C1:64 C2:32	0.68±0.94	1.22%	6.96	1.16	79.76%

Tabla 4.16: Modelos LSTM para diferentes arquitecturas.

Podemos observar que sólo con dos capas ya estamos sobredimensionando la red ya que la mayoría de los resultados de una capa son mejores, entre ellos destacando la segunda arquitectura de una capa con ocho neuronas, que usaremos en los siguientes apartados.

4.4.2. Entrenamiento

Estos son los resultados del entrenamiento usando el método de optimización *grid search* para distintos pares de hiperparámetros:

	T	MAE±SD	MAPE	Emáx	RMSD	R^2
b=8 e=200	351.93	0.77±1.1	1.4%	7.45	1.34	73.07%
b=8 e=100	183.94	0.7±0.9	1.26%	6.67	1.14	80.38%
b=16 e=100	89.5	0.64±0.85	1.15%	6.74	1.06	83.07%

b=16 e=75	67.42	0.63±0.85	1.13%	**6.6**	1.06	83.13%
b=32 e=75	34.3	0.67±0.88	1.2%	6.96	1.1	81.62%
b=32 e=50	21.68	0.68±0.93	1.22%	7.2	1.15	79.98%
b=64 e=50	11.53	0.9±0.95	1.59%	7.49	1.31	74.25%
b=64 e=30	7.33	**0.6±0.86**	**1.06%**	7.2	**1.05**	**83.42%**
b=128 e=30	4.09	0.69±0.85	1.23%	6.98	1.09	83.17%
b=128 e=15	**2.54**	1.14±0.94	2.01%	7.23	1.48	69.17%

Tabla 4.17: Modelos LSTM para diferentes tamaños de *batch* y número de *epochs*.

Observamos que no tiene un comportamiento lineal, habiendo dos picos de funcionamiento. Nos decidimos por el que tiene un tiempo de ejecución menor (12.21 veces más rápido).

4.4.3. Evaluación

A continuación, se muestran los límites de predicción del modelo LSTM escogido:

	MAE±SD	MAPE	Emáx	RMSD	R^2
60	0.6±0.86	1.06%	7.2	1.05	83.42%
90	0.82±0.83	1.45%	6.9	1.17	79.4%
120	0.61±0.83	1.09%	6.69	1.03	82.3%
150	0.98±0.9	1.73%	6.58	1.33	67.29%
180	0.95±0.98	1.72%	5.84	1.37	70.68%
210	0.87±0.95	1.57%	5.96	1.28	70.68%
240	1.11±0.96	1.98%	5.93	1.47	62.47%
270	1.43±1.05	2.54%	5.29	1.77	46.69%
300	1.5±1.09	2.66%	6.41	1.86	43.0%
330	1.73±1.15	3.02%	7.02	2.08	30.21%
360	1.42±0.94	2.53%	6.46	1.7	48.4%
390	1.6±1.19	2.78%	7.84	2.0	27.8%
420	1.75±1.12	3.07%	4.88	2.08	23.33%

Tabla 4.18: Modelo LSTM para diferentes tamaños de ventana de predicción.

Diferenciamos tres grupos, un primero hasta dos minutos que predice con precisión, un segundo hasta los cuatro minutos que pierde precisión pero sigue las tendencias de los cambios de temperatura y un tercero que pierde mucha precisión respecto a los otros pero sigue todavía los cambios.

4.4.4. Optimización

A continuación se prueba para el modelo escogido diferentes ventanas de los grupos mencionados anteriormente para ver si mejora la precisión usando capas *dropout*.

	Dropout	MAE±SD	MAPE	Emáx	RMSD	R^2
60	0.12	0.78±0.8	1.38%	6.54	1.12	81.03%
180	0.05	0.86±0.95	1.54%	5.59	1.28	70.13%
240	0.12	1.01±1.0	1.81%	5.36	1.42	65.03%
360	0.12	1.04±0.91	1.83%	5.68	1.38	65.89%
420	0.12	1.27±0.97	2.26%	5.13	1.6	54.46%

Tabla 4.19: Modelo LSTM con *dropout*.

Observamos que hasta que no usamos una ventana grande (seis minutos) el uso de *dropout* empeora o no es relevante. Esto se puede deber a que la arquitectura que tenemos es muy pequeña siendo la información en cada nodos más relevante para el resultado que si utilizáramos una arquitectura más grande. Vemos si se puede reducir la dimensionalidad al mejor modelo usado:

	MAE±SD	MAPE	Emáx	RMSD	R^2
Sin T1 y T3	0.64±0.87	1.15%	6.4	1.08	82.38%
Sin frecuencia	0.77±0.88	1.35%	7.58	1.17	79.27%
Sin utilización	1.31±0.97	2.35%	5.74	5.74	59.58%

Tabla 4.20: Modelo LSTM reduciendo características.

Observamos que las dos temperaturas de los equipos exteriores no aportan información relevante a la red siendo los resultados incluso de la misma precisión, mientras que la utilización es de gran utilidad para conseguir que converja a una solución adecuada.

5. Resultados

En esta sección compararemos los resultados obtenidos en los experimentos de las tres tipos de arquitecturas utilizadas (LSTM, GRU y *fully connected*) con el objetivo de proponer el mejor modelo posible para el problema al que nos enfrentamos.

5.1. Comparativa de los modelos

5.1.1. Modelos propuestos

A continuación se muestran los hiperparámetros escogidos para cada modelo de red neuronal junto con el número de parámetros que tiene y su tiempo de ejecución:

Hiperparámetros comunes					
Test	Preescalado	Optimizador	*Learning rate*	*Loss*	*Features*
10%	Lineal {0,1}	*Adagrad*	0.01	MAE	5
Modelo *fully connected*					
Arquitectura	*Dropout*	*Batch*	*Epochs*	Parámetros	T_e (s)**
Capa 1: 128	0.05*	32	75	17281	~30
Modelo GRU					
Arquitectura	*Dropout*	*Batch*	*Epochs*	Parámetros	T_e (s)**
Capa 1: 32	0.05*	128	15	3648	~3
Modelo LSTM					
Arquitectura	*Dropout*	*Batch*	*Epochs*	Parámetros	T_e (s)**
Capa 1: 8	0.05*	64	30	457	~7

Tabla 5.1: Modelos escogidos para los tres tipos de red.

*Para ventanas de más de dos minutos.
**Tiempo de ejecución para una ventana de un minuto.

5.1.2. Análisis de los modelos

A continuación, se comparan con diferentes estrategias los mejores modelos de cada tipo de red:

1. Ventana temporal de un minuto:

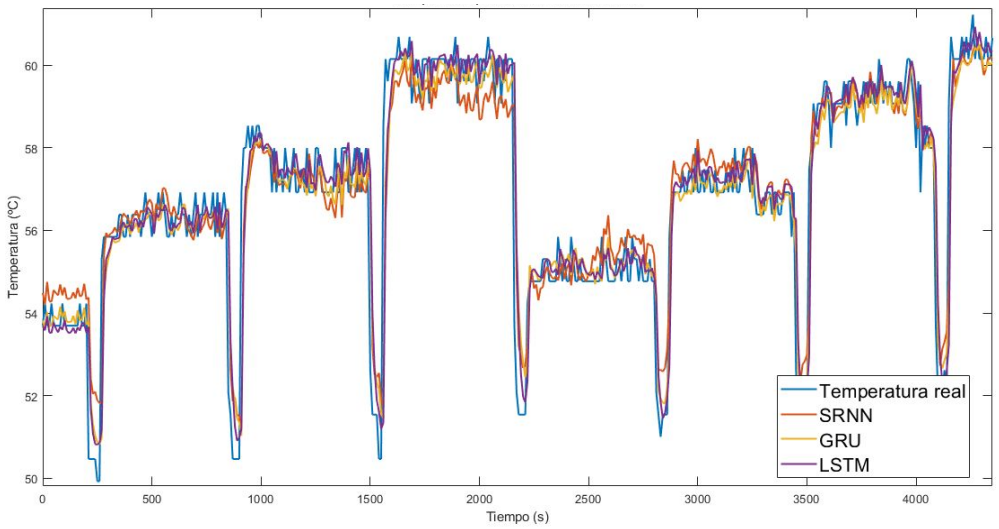


Figura 5.1: Comparativa de predicciones para ventana de un minuto.

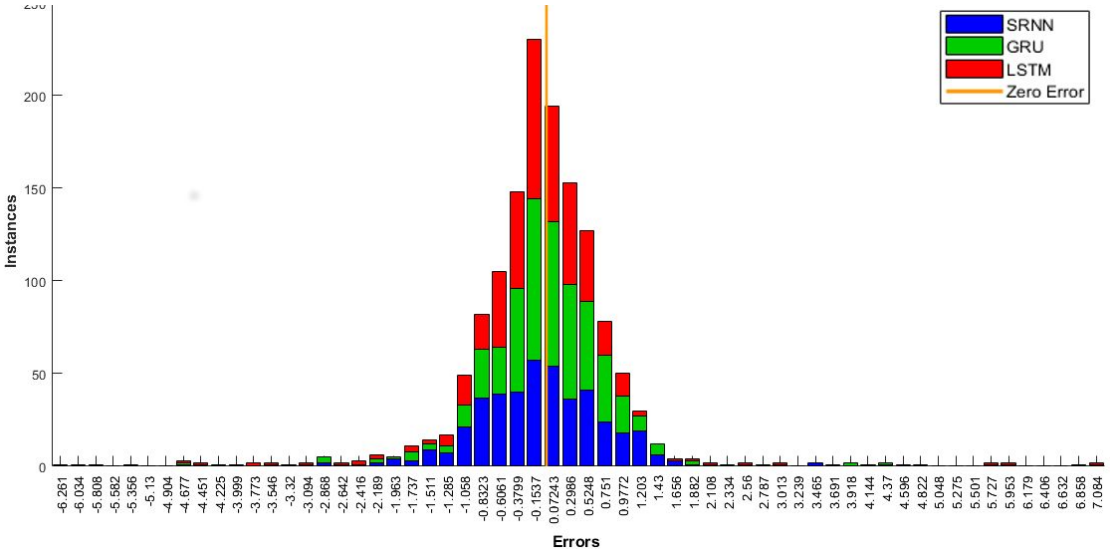


Figura 5.2: Histograma de los errores de ventana de un minuto.

	MAE±SD	MAPE	Emáx	RMSD	R^2
SRNN	0.7±0.78	1.25%	6.78	1.05	83.33%
GRU	0.6±0.83	1.08%	7.1	1.02	85.23%
LSTM	0.6±0.86	1.06%	7.2	1.05	83.42%

Tabla 5.2: Métricas de predicciones para ventana de un minuto

2. Ventana temporal de tres minutos:

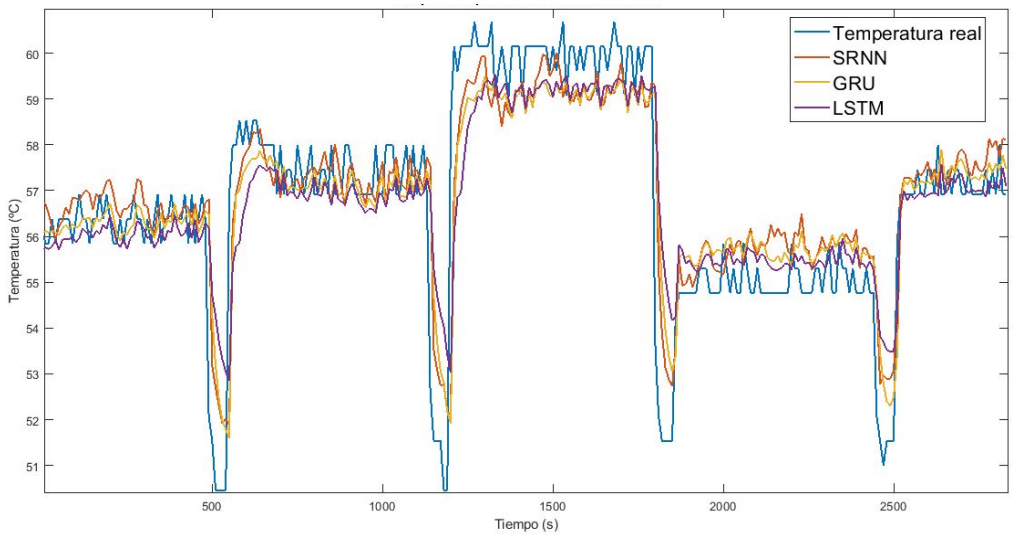


Figura 5.3: Comparativa de predicciones para ventana de tres minutos.

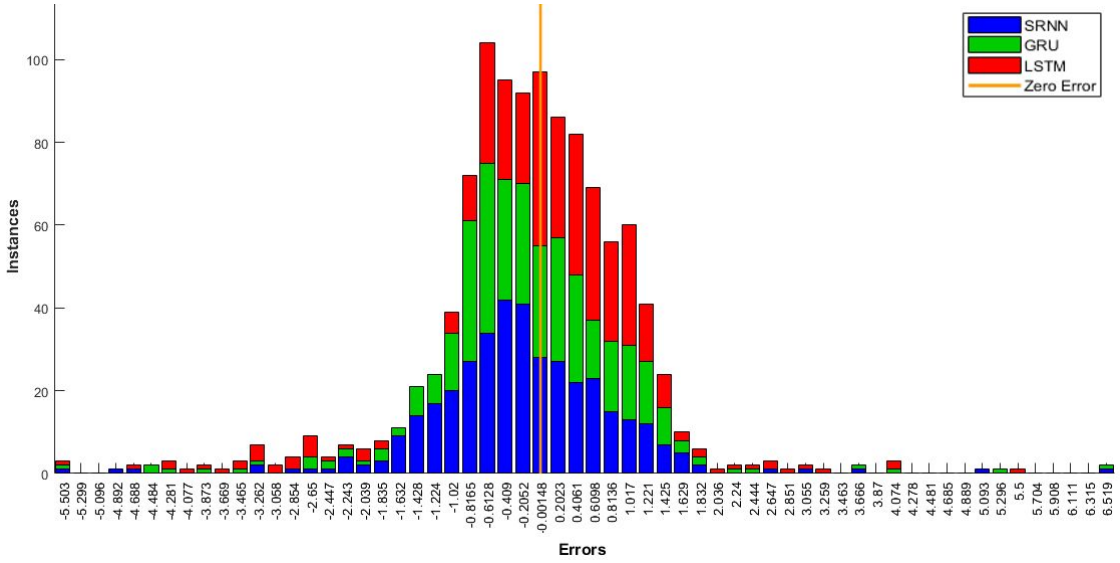


Figura 5.4: Histograma de los errores de ventana de tres minutos.

	MAE±SD	MAPE	Emáx	RMSD	R^2
SRNN	0.79±0.79	1.42%	6.43	1.12	77.92%
GRU	0.81±0.85	1.46%	6.62	1.17	77.72%
LSTM	0.86±0.95	1.54%	5.59	1.28	70.13%

Tabla 5.3: Métricas de predicciones para ventana de tres minutos.

3. Ventana temporal de seis minutos:

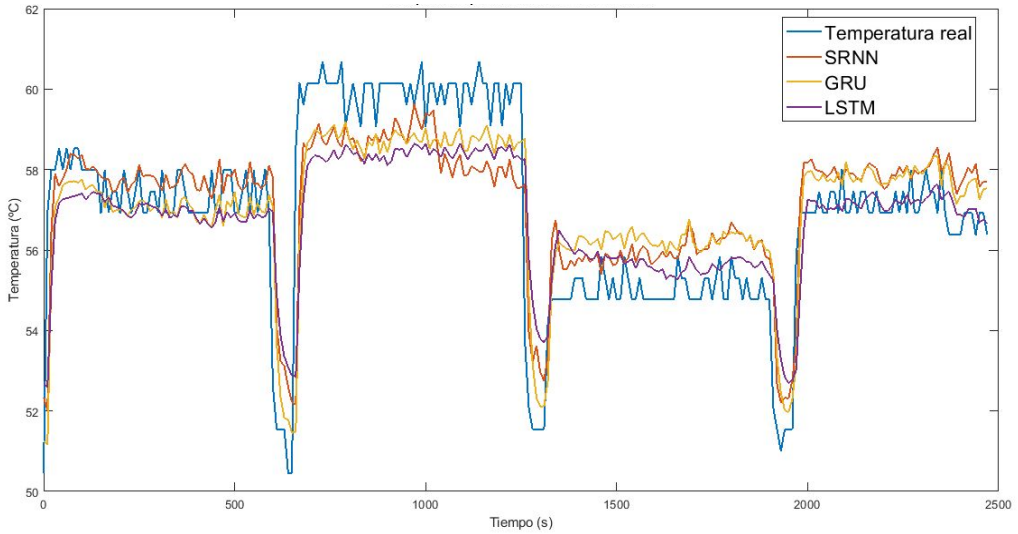


Figura 5.5: Comparativa de predicciones para ventana de seis minutos.

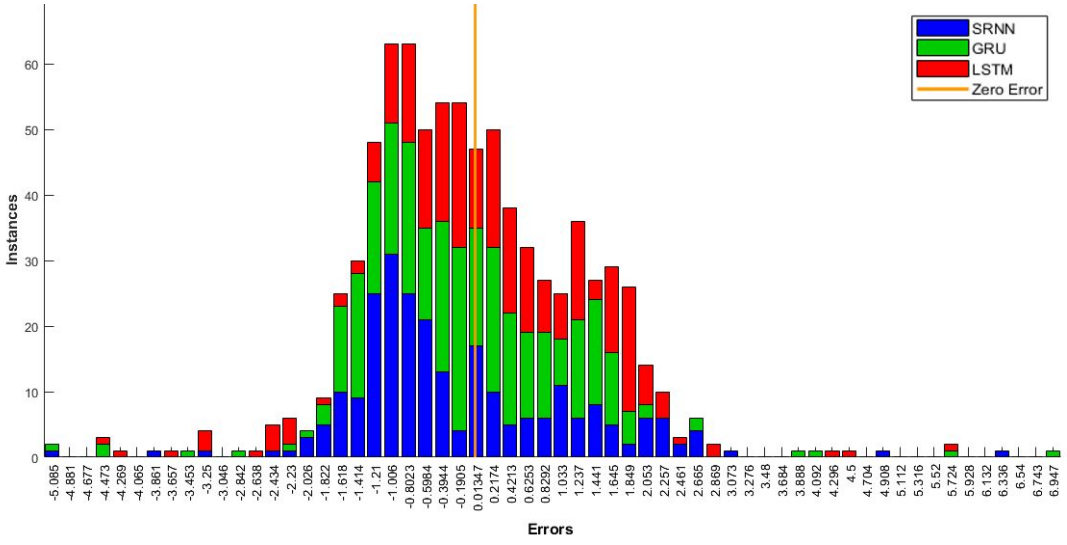


Figura 5.6: Histograma de los errores de ventana de seis minutos.

	MAE±SD	MAPE	Emáx	RMSD	R^2
SRNN	1.1±0.82	1.94%	6.38	1.37	66.56%
GRU	0.96±0.85	1.69%	7.05	1.28	68.82%
LSTM	1.04±0.91	1.83%	5.68	1.38	65.89%

Tabla 5.4: Métricas de predicciones para ventana de seis minutos.

4. Evolución del error absoluto medio y desviación estándar en función de la ventana temporal

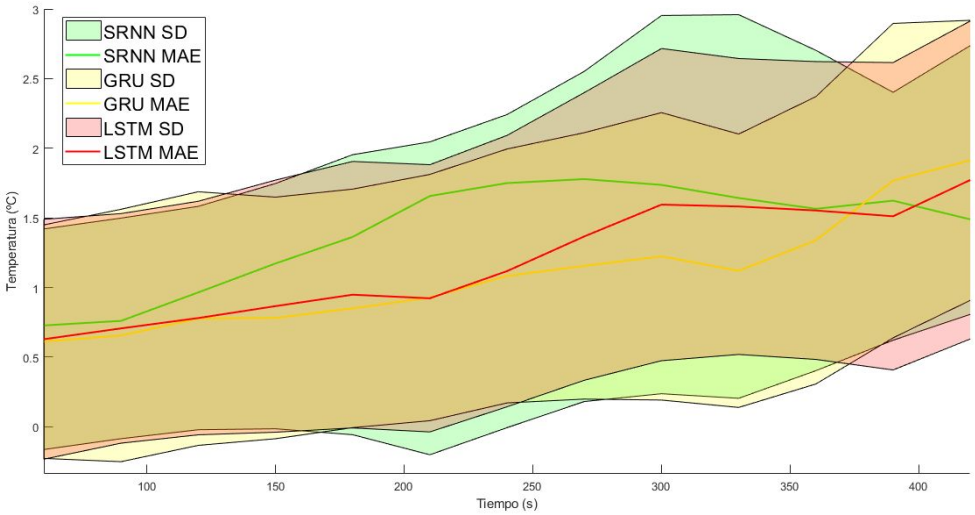


Figura 5.7: Comparativa de la métrica MAE±SD en función de la ventana temporal.

5. Evolución de la desviación de la raíz cuadrática media en función de la ventana temporal

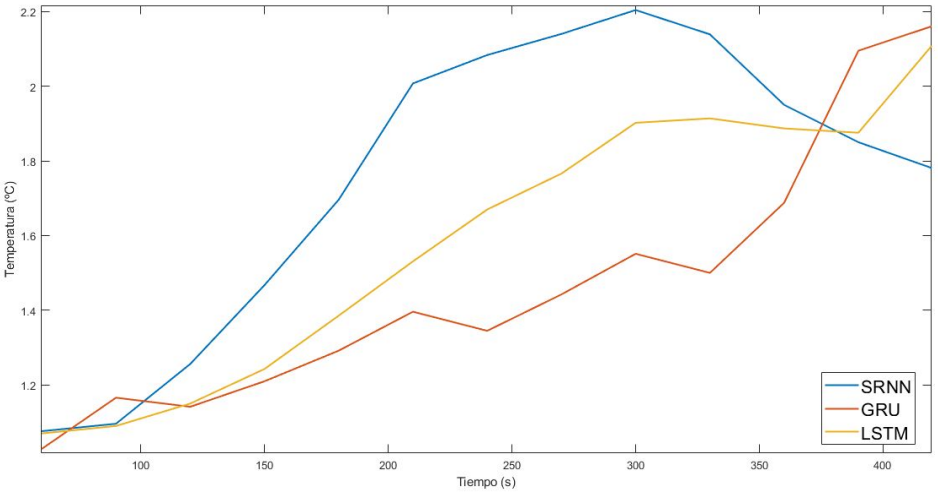


Figura 5.8: Comparativa de la métrica RMSD en función de la ventana temporal.

5.2. Modelo seleccionado

Si observamos los resultados de las métricas de las predicciones con ventana igual a un minuto, obtenemos resultados muy similares, con una ligera mejora usando redes GRU. Fijándonos en la gráfica ahora, vemos que las redes fully connected (SRNN) no llegan a predecir bien los picos mayores de temperatura que es la parte más crítica que nos conviene predecir para actuar cambiando los cargas de los equipos. Aunque, si miramos la evolución de los errores, observamos que para la ventana más grande es capaz de predecir mejor, esto se debe a que la arquitectura usada en *fully connected* es mucho más grande (17000 parámetros comparados con 3000 y 500).

Aunque sea una ventaja ya que el modelo es más robusto, no es tan relevante porque los modelos están pensados para una ventana de un minuto que es la que se ha usado durante la búsqueda de la arquitectura y el entrenamiento. Con una alta probabilidad, si optimizamos para una ventana mayor, otras arquitecturas serían más adecuadas. La desventaja de tener un modelo con más parámetros es que la ejecución es mucho más lenta. Por ello, y todo lo anteriormente comentado, descartamos la red tipo *fully connected*.

Entre la red LSTM y GRU nos decidimos por esta última, como ya hemos comentado tiene una ligera ventaja respecto a las otras en la ventana de un minuto, además si miramos la evolución de los errores también es la más precisa en las ventanas más pequeñas que interesan más. Por último, la ejecución de la red GRU es más rápida que la red LSTM con lo que si en un futuro se escalara sería un dato a tener en cuenta. Finalmente, presentamos el modelo definitivo con todos sus hiperparámetros y la simulación para la ventana de un minuto junto a un histograma de sus errores:

Modelo definitivo					
Tipo	Preescalado	Optimizador	*Learning rate*	*Loss*	*Features*
GRU	Lineal {0,1}	*Adagrad*	0.01	MAE	5
Arquitectura	*Dropout*	*Batch*	*Epochs*	Parámetros	T_e (s)
Capa 1: 32	0.05*	128	15	3648	~3

Tabla 5.5: Modelo definitivo propuesto.

*Para ventanas grandes

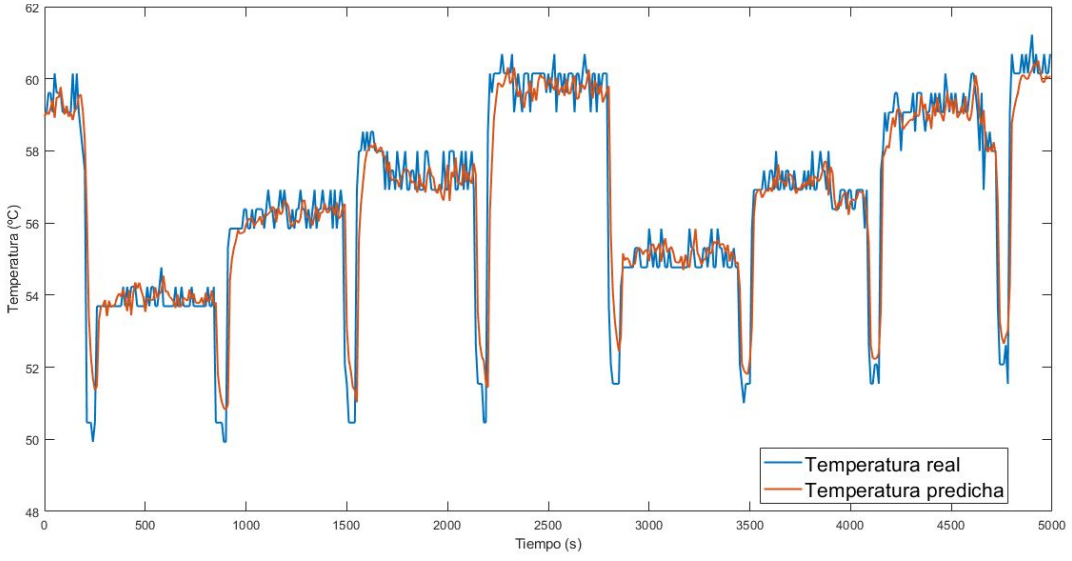


Figura 5.9: Simulación del modelo definitivo para una ventana de un minuto.

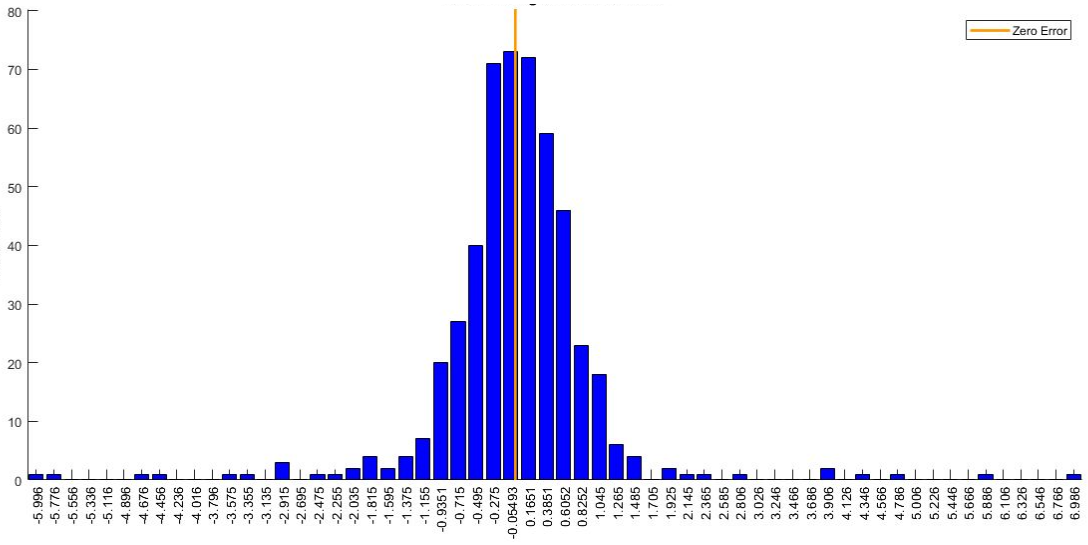


Figura 5.10: Histograma de los errores de la simulación del modelo definitivo.

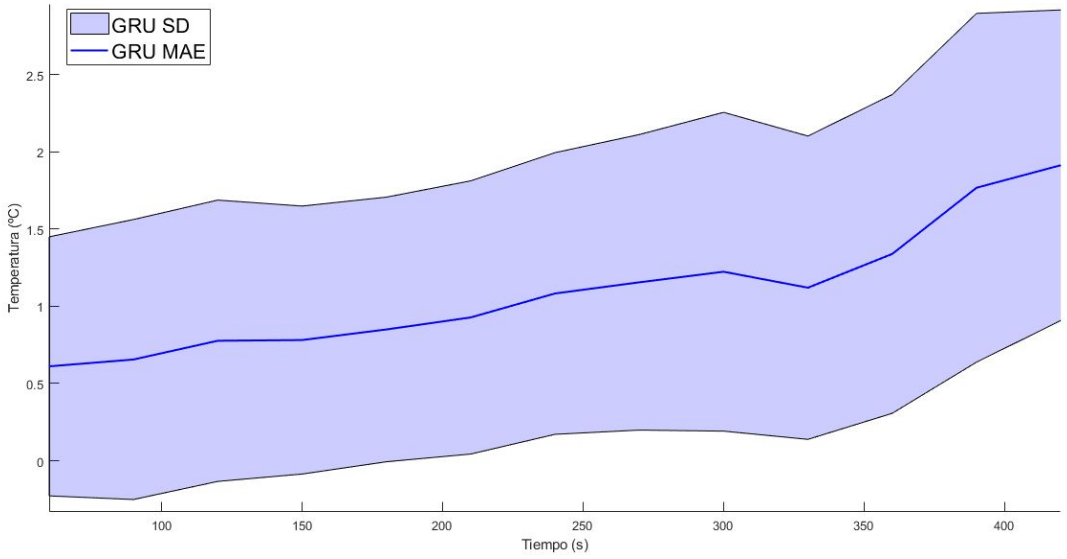


Figura 5.11: Evolución de la métrica MAE±SD en función de la ventana temporal.

6. Conclusiones y líneas futuras

Hemos conseguido probar con éxito un sistema de refrigeración por inmersión en dos fases con líquido dieléctrico. Este sistema en un centro de datos supondría una reducción potencialmente del 100% de la energía consumida por los sistemas de refrigeración o lo que es lo mismo una PUE igual a uno. Por otro lado, hemos probado que el líquido dieléctrico Vertrel XF puede ser utilizado para refrigerar en estos sistemas, lo que supone un abaratamiento de un tercio en el coste comparado con el NOVEC 7100, que ha usado la mayoría de la industria hasta ahora en sus sistemas de refrigeración por inmersión. Además, supone un aumento de la eficiencia ya que el punto de ebullición del líquido es más bajo, costando menos llegar a las dos fases.

En lo referente a los modelos de predicción de temperatura, hemos conseguido grandes resultados gracias al uso de redes neuronales recurrentes. Estas han conseguido replicar con éxito los cambios de temperatura diseñados en la carga de trabajo con una precisión muy elevada (entorno a un 0.6 ºC de error medio) para la ventana temporal que fueron diseñadas y con resultados más que aceptables en ventanas de predicción más grandes. Gracias a esta precisión en la ventana de un minuto seríamos capaces, en un entorno real, de variar la carga dinámicamente para poder hacer trabajar al líquido en su punto de máxima absorción de calor. Todos los tipos de redes neuronales recurrentes han dado grandes resultados a la hora de resolver el problema, destacando ligeramente las redes tipo GRU. Aún así, no se puede asegurar que se haya encontrado la solución óptima ya que se está trabajando con algoritmos metaheurísticos, pero los resultados obtenidos son más que suficientes para tener un uso eficiente de las cargas de trabajo de los equipos sumergidos.

Dentro de los problemas encontrados, destacar la dificultad de mantener el gas generado al evaporarse el líquido en la pecera aun sellando esta. Con ello, perdimos eficiencia a la hora de realizar los experimentos. Otro problema ha sido la limitación de los equipos Raspberry Pi calentándose, no llegando a temperaturas muy altas. Esto ha dificultado el llegar al funcionamiento en dos fases. En próximos experimentos a mayor escala, estos problemas desaparecerá al usar equipo más profesional.

Las líneas futuras las podemos dividir en dos. Primero, en lo referente al sistema de refrigeración, se debe seguir realizando experimentos cada vez más exhaustivos para conocer los límites de funcionamiento del líquido para una futura implementación eficiente en un centro de datos. El próximo paso será probarlo en el contenedor submarino facilitado por Adam para obtener resultados reales en un centro de datos de *Edge computing*. Segundo, en lo referente a los modelos de predicción, en un futuro inmediato se puede probar a optimizar para ventanas de mayor tamaño (no solo para un minuto), probar con diferentes cargas para tener modelos más robustos o introducir capas convolucionales previas a las recurrentes cuando se tengan conjuntos de datos de mayor tamaño. Cuando el sistema esté preparado para un centros de datos se podría ejecutar los modelos de forma paralela para reducir los tiempos de cómputo y en tiempo real para que los modelos se reajusten solos en función de los nuevos datos procesados.

7. Bibliografía

[1] Maria Avgerinou, Paolo Bertoldi & Luca Castellazzi, «Trends in Data Centre Energy Consumption under the European Code of Conduct for Data Centre Energy Efficiency,» European Commission, Joint Research Centre (JRC), Directorate C-Energy, Transport and Climate. Aug. 2017

[2] N. Engbers and E. Taen, Green Data Net. «Report to IT Room,» INFRA, European Commision. FP7 ICT 2013.6.2, Nov. 2014.

[3] «Técnicas de optimización de la climatización en el Datacenter,» [En línea]. Available: http://blog.aodbc.es/2012/06/04/tecnicas-de-optimizacion-de-la-climaticacion-en-el-datacenter/

[4] «Data center cooling methods,» [En línea]. Available: https://submer.com/data-center-cooling-methods/

[5] «Cisco Global Cloud Index: Forecast and Methodology, 2016–2021,» [En línea]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf

[6] «Edge Computing: The Cloud, the Fog and the Edge,» [En línea]. Available: https://www.solid-run.com/edge-computing-cloud-fog-edge/

[7] «Información del líquido NOVEC 7100 de la empresa 3M,» [En línea]. Available: https://www.3m.com/3M/en_US/company-us/all-3m-products/~/3M-Novec-7100-Engineered-Fluid/?N=5002385+3290667247&rt=rud

[8] Patricia Arroba García, «Proactive Power and Thermal Aware Optimizations for Energy-Efficient Cloud Computing,» ETSIT UPM, 2017.

[9] Bitdefender, «Cryptocurrency Mining Craze Going for Data Centers,» [En línea]. Available: https://www.bitdefender.com/files/News/CaseStudies/study/196/Bitdefender-Whitepaper-Cryptocurrency-Mining-Craze-Going-for-Data-Centers-2018.pdf

[10] «Uno de los centros de datos de mayor eficiencia energética en Europa prueba minería de BTC, ETH,» [En línea]. Available: https://es.cointelegraph.com/news/one-of-europes-most-energy-efficient-data-centers-tests-btc-eth-mining

[11] «Nuevo centro de procesamiento de datos de criptomonedas inicia operaciones en Nueva York,» [En línea]. Available: https://www.coincrispy.com/2018/06/06/centro-procesamiento-datos-criptomonedas-operaciones-nueva-york/

[12] «Bitcoin Mining Now Consuming More Electricity Than 159 Countries Including Ireland & Most Countries In Africa,» [En línea]. Available: https://powercompare.co.uk/bitcoin/

[13] «Bitcoin Energy Consumption Index,» [En línea]. Available: https://digiconomist.net/bitcoin-energy-consumption

[14] «Vertrel™ XF Specialty Fluid Technical Information,» [En línea]. Available: https://www.chemours.com/Vertrel/en_US/assets/downloads/pdf/vertrel-xf-technical-information.pdf

[15] «CACS, técnicas básicas de climatización en Datacenters (II),» [En línea]. Available:

http://blog.aodbc.es/2012/06/11/cacs-tecnicas-basicas-de-climatizacion-en-datacenters-ii/
[16] «HACS, técnicas básicas de climatización en Datacenters (III),» [En línea]. Available: http://blog.aodbc.es/2012/06/18/hacs-tecnicas-basicas-de-climatizacion-en-datacenters-iii/
[17] «The pros and cons of hot and cold aisle containment,» [En línea]. Available: https://fieldmansaccessfloorsltd.com/pros-cons-hot-cold-aisle-containment/
[18] «Super cool: Arctic data centres aren't just for Facebook,» [En línea]. Available: https://www.theregister.co.uk/2016/05/12/power_in_a_cold_climate/
[19] «World's largest data center to be built in Arctic Circle,» [En línea]. Available: https://www.cnbc.com/2017/08/15/worlds-largest-data-center-to-be-built-in-arctic-circle.html
[20] «Liquid Cooling v. Air Cooling Evaluation in the Maui High Performance Computing Center,» [En línea]. Available: https://www.energy.gov/sites/prod/files/2014/08/f18/cs_maui_high_pcc.pdf
[21] Tuf, Steve. "Power Usage Effectiveness." It.toolbox. Toolbox, n.d. Web. 17 Nov. 2014.
[22] *Open Compute Project*. Open Compute Project. 18 March 2016 "Energy Efficiency"
[23] «Two-Phase Immersion Cooling,» [En línea]. Available: http://multimedia.3m.com/mws/media/1127920O/2-phase-immersion-coolinga-revolution-in-data-center-efficiency.pdf
[24] «Hot aisle containment vs. Cold Aisle Containment for Data centers,» [En línea]. Available: https://www.missioncriticalmagazine.com/ext/resources/MC/Home/Files/PDFs/WP-APC-Hot_vs_Cold_Aisle.pdf
[25] «Is a Liquid-Cooled Data Center in Your Future?,» [En línea]. Available: http://www.liquidcoolsolutions.com/is-a-liquid-cooled-data-center-in-your-future/
[26] Ellsworth, Jr., M.J., and Iyengar, M.K., «Energy Efficiency Analyses and Comparison of Air and Water Cooled High Performance Servers,» Proceedings of the 2009 InterPACK Conference, San Francisco, CA, USA, July 19-23, 2009.
[27] «The Folsom Power Plant 1895,» [En línea]. Available: http://www.edisontechcenter.org/Folsom.htm
[28] «Allied Control: Immersion Cooling,» [En línea]. Available: http://www.allied-control.com/immersion-cooling
[29] «The Green 500: Ranking List,» [En línea]. Available: https://www.top500.org/green500/lists/2017/11/
[30] «Google's Finland data center pioneers new seawater cooling,» [En línea]. Available: http://www.datacenterdynamics.com/content-tracks/power-cooling/googles-finland-data-center-pioneers-new-seawater-cooling/32932.fullarticle
[31] «Facebook's new cooling system might make it easier to put data centers in hot places,» [En línea]. Available: https://www.geekwire.com/2018/facebooks-new-cooling-system-might-make-easier-put-data-centers-hot-places/
[32] Lijun Fu, Jianxiong Wan*, Ting Liu .«Temperature-Aware Resource Management Algorithm for Holistic Energy Minimization in Data Centers,» School of Information Engineering, Inner Mongolia University of Technology, China, 2018.
[33] Justin Moore, Jeff Chase Parthasarathy Ranganathan Ratnesh Sharma. «Making Scheduling "Cool": Temperature-Aware Workload Placement in Data Centers,» Department of Computer Science Duke University, 2005.
[34] Hong Xu, Chen Feng, Baochun Li «Temperature Aware Workload Management in Geo-distributed Datacenters,» Department of Electrical and Computer Engineering University of Toronto, 2014.

[35] Anton Beloglazov, Jemal Abawajy, Rajkumar Buyya. «Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing,» Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computer Science and Software Engineering, The University of Melbourne, Australia, 2012.
[36] Qinghui Tang, Sandeep Kumar S. Gupta, and Georgios Varsamopoulos. «Energy-Efficient Thermal-Aware Task Scheduling for Homogeneous High-Performance Computing Data Centers: A Cyber-Physical Approach,» IEEE, 2008.
[37] Jim Gao, «Machine Learning Applications for Data Center Optimization,» Google, 2016.
[38] «Umbra Pecera de cristal Fishhotel,» [En línea]. Available: https://www.amazon.es/dp/B0033FGDRS/
[39] «Raspberry Pi 3 Modelo B+,» [En línea]. Available: https://www.amazon.es/dp/B07BFH96M3/
[40] «Kingston SDC4/16GB,» [En línea]. Available: https://www.amazon.es/dp/B003WIRFD2/
[41] «D-Link DGS-108,» [En línea]. Available: https://www.amazon.es//dp/B000BCC0LO/
[42] «Cargador Raspberry,» [En línea]. Available: https://www.amazon.es/dp/B01M58O9M9/
[43] «Vertrel™ XF specialty fluid,» [En línea]. Available: https://www.chemours.com/Vertrel/en_US/products/xf.html
[44] «collectd web oficial,» [En línea]. Available: https://collectd.org
[45] «BitFury web oficial,» [En línea]. Available: https://bitfury.com/
[46] «Cryptocurrency Mining on the Raspberry Pi,» [En línea]. Available: https://www.electromaker.io/tutorial/blog/cryptocurrency-mining-on-the-raspberry-pi-60
[47] «Multi-algo CPUMiner & Reference Cryptonote Miner,» [En línea]. Available:https://github.com/lucasjones/cpuminer-multi
[48] Javier López Ruiz «Metaheurísticas para el análisis de datos en el ámbito del transporte de carretera,» Escuela técnica superior de ingenieros informáticos UPM, 2017.
[49] «Recurrent Neural Network Tutorial, Part 4 – Implementing a GRU/LSTM RNN with Python and Theano,» [En línea]. Available: http://www.wildml.com/2015/10/recurrent-neural-network-tutorial-part-4-implementing-a-grulstm-rnn-with-python-and-theano/
[50] Klaus Greff, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, Jurgen Schmidhuber «LSTM: A Search Space Odyssey,» IEEE Transactions on Neural Networks and Learning Systems (Volume: 28, Issue: 10, Oct. 2017) pp. 2222 - 2232, 2017.
[51] «Web oficial de Anaconda,» [En línea]. Available: https://anaconda.org/
[52] «Web oficial de Jupyter Notebook,» [En línea]. Available: http://jupyter.org/
[53] «Web oficial de TensorFlow,» [En línea]. Available: https://www.tensorflow.org/
[54] «Web oficial de Keras,» [En línea]. Available: https://keras.io/
[55] «Precauciones a la hora de normalizar datos en Data Science,» [En línea]. Available: http://www.synergicpartners.com/precauciones-a-la-hora-de-normalizar-datos-en-data-science/
[56] «A Beginner's Guide to Recurrent Networks and LSTMs,» [En línea]. Available: https://deeplearning4j.org/lstm.html
[57] «Mineros alternativos de MinerGate,» [En línea]. Available: https://minergate.com/altminers/cpuminer-multi-wolf
[58] «Optimizadores en Keras,» [En línea]. Available: https://keras.io/optimizers/

[59] « Ejemplo de modelo para series temporales del curso de inteligencia artificial de IBM,» [En línea]. Available: https://github.com/romeokienzler/developerWorks/blob/master/coursera/ai/week3/lstm_crude_oil_price_prediction.ipynb
[60] «Optimización de hiperparámetros ,» [En línea]. Available: https://medium.com/@elutins/grid-searching-in-machine-learning-quick-explanation-and-python-implementation-550552200596

8. Anexos

Anexo A: Aspectos éticos, económicos, sociales y ambientales

A.1 Introducción

La irrupción de las aplicaciones IoT, han elevado el consumo de datos y con ello la energía consumida globalmente. Los centros de datos, instalaciones dedicadas al manejo de estos datos, cada vez ganan más importancia en la actualidad volviéndose necesario un uso eficiente de estos para no aumentar el consumo energético y con ello las emisiones de CO_2.

El consumo de energía en un centro de datos se puede dividir en dos partes principalmente. Una primera relacionada con el funcionamiento de sus equipos que representa el 60% y una segunda con los sistemas de refrigeración que supone el 40%. En este proyecto se pondrá a prueba un sistema de inmersión en líquido dieléctrico en dos fases, que potencialmente podría reducir a cero el porcentaje de energía de refrigeración consumida en los centros de datos reduciendo así enormemente la energía consumida por estos.

A.2 Descripción de impactos relevantes relacionados con el proyecto

Podemos dividir los impactos de este proyecto en tres:

1. Impacto medioambiental: La reducción de energía consumida por el sistema gracias al sistema de refrigeración pasivo propuesto supone un ahorro inmenso a gran escala de las emisiones de CO_2 de los centros de datos que a suponen un 2% a nivel mundial actualmente.
2. Impacto económico: Además de la reducción energética, estos sistemas son mucho más pequeños y compactos que los sistemas convencionales. Esto significaría un ahorro de espacio enorme (que se traduciría en beneficios) lo que transformaría el mercado de los centros de datos, facilitando la implantación del paradigma *Fog Computing*, centros más pequeños más cerca del origen de los datos, que mitigaría el problema del aumento de la demanda de datos actual.
3. Impacto social: Al facilitar el uso del *Fog Computing*, el desarrollo de las *Smart Cities* se aceleraría y con ello se transformaría la vida de las personas de manera radical en las ciudades. Proyectos como el del coche autónomo estarían más cerca a una implementación real gracias a esto.

A.3 Análisis detallado de alguno de los principales impactos

El impacto que más hemos analizado en el presente trabajo es el medio ambiente, tanto en el inicio de la introducción (Sección 1) y su primer apartado (Sección 1.1). “Refrigeración en los centros de datos” como en el primer apartado del estado del arte

(Sección 2.1) “Los sistemas de refrigeración en la actualidad” se ha realizado un análisis detallado de todos los tipos de sistemas de refrigeración en centros de datos de carácter general con ejemplos de sistemas reales implementados con sus ventajas e inconvenientes y por qué el que proponemos teóricamente tiene un rendimiento mucho mayor facilitando la reducción de energía y con ello las emisiones de CO_2.

A.4 Conclusiones

El presente trabajo puede ayudar a plantear un futuro más sostenible en el sector IT, necesitado de ello debido a la creciente demanda de aplicaciones que necesitan grandes volúmenes de datos para ser usadas. Esto supondrá inevitablemente, que las estructuras tradicionales dejen de ser viables y haya que buscar alternativas más sostenibles, eficientes y que tengan un impacto positivo en el medio ambiente.

Anexo B: Presupuesto económico

A continuación se presenta el presupuesto del proyecto:

COSTE DE MANO DE OBRA (CD)	Horas	Precio/hora	Total
	325	15 €	4.875 €

COSTE DE RECURSOS MATERIALES (CD)	Precio de compra	Uso en meses	Amortización (en años)	Total
Ordenador personal	500,00 €	4	5	33,33 €
Líquido vertrel XF (10 Kg)	300,00 €	4	1	100,00 €
Seis Raspberry Pi	240,00 €	4	3	26,67 €
Seis alimentadores Raspberry Pi	48,00 €	4	6	2,67 €
Seis tarjetas micro SD	36,00 €	4	5	2,40 €
Switch	25,00 €	4	6	1,39 €
Pecera para hogar	30,00 €	4	8	1,25 €

COSTE TOTAL DE RECURSOS MATERIALES	167,71 €

GASTOS GENERALES (CI)	15%	sobre CD	756,41 €
BENEFICIO INDUSTRIAL	6%	sobre CD+CI	347,95 €

MATERIAL FUNGIBLE	
Cinta americana	5,00 €
Cables ethernet	15,00 €
Separadores	3,00 €

SUBTOTAL PRESUPUESTO		6.170,06 €
IVA APLICABLE	21%	1.295,71 €

TOTAL PRESUPUESTO	7.465,77 €

Anexo C: Resumen de los sistemas de refrigeración

En este anexo se resumen las ventajas e inconvenientes más importantes de todas las técnicas de refrigeración en centros de datos de carácter general vistas:

Sistemas de refrigeración	Ventajas	Inconvenientes
Refrigeración por aire	Muy extendido y fácil de usar. Puede aprovechar la climatología del lugar.	No es eficiente debido a muchos problemas, puede requerir mucha energía y espacio.
Hot Aisle / Cold Aisle	Primer acercamiento a un sistema que separa corrientes de aire frías y calientes.	Creación de puntos calientes e intercambios indeseados de corrientes.
Hot Aisle Containment	Volumen de aire frío menor,misma eficiencia de refrigeración.	SI hay fallo, hay muy poco volumen de aire frío y se calentará rápido. Consume mucho.
Cold Aisle Containment	En caso de fallo tarda más en calentarse y facilidad de extracción de aire caliente.	Coste de despliegue mayor que en HACS y como esta, consume mucho (debido a las unidades CRAC)..
Free Cooling	Usa aire exterior. Ahorras mucha energía y costes sin necesidad de unidades CRAC.	Necesita climatología favorable y sigue necesitando energía, no es completamente “gratuita”.
Refrigeración por agua	Solventa muchos problemas del aire: oxidación, vibraciones, poca capacidad de arrastre...	Capacidad de arrastre menor que un líquido dieléctrico y puede haber riesgos de fuga que son fatales.
Unidades CRAH en sustitución de unidades CRAC	Sistema de refrigeración líquido, más eficiente que sus homólogos CRAC..	Consumen mucha energía.
In-row	Reducen la energía consumida por un sistema de aire hasta un 60%.	Necesita de estructura de un centro de datos tradicional limitando eficacia
Puerta trasera pasivo	Eliminan el calor en el punto de origen utilizando agua.	Necesita ventiladores que empujen el aire hacia fuera.
Enfriamiento directo con agua	No necesita agua a una temperatura muy baja.	Coste económico.
Puerta trasera activo	Ocupan menos espacio que *in-row,* enfrían dinámicamente.	Al ser por agua, no tienen el potencial que pudieran tener los de líquido dieléctrico.
Refrigeración por líquido dieléctrico	Más seguro que el agua y mucho más eficiente al tener mayor capacidad de arrastre de calor.	Todavía por desarrollar, no está muy extendido.
Indirecta	Existe también en agua pero a diferencia de esta no existe riesgo de fuga.	Al ser indirecto pierde el potencial de los líquidos dieléctricos.
Sellado de equipos	Equipos manejables que esta vez si tienen contacto directo con el líquido.	Solución específica que necesita de un hardware determinado.
Inmersión en aceite dieléctrico	Hace uso del potencial del	No es un método completamente

en una fase*	dieléctrico al bañar el equipo entero y no genera altas presiones al ser abierto o semiabierto.	pasivo. El aceite es engorroso y poco manejable
Inmersión en químico dieléctrico en dos fases	Método pasivo que puede ahorrar hasta un 90% la energía que gasta un sistema de aire. Con el químico es más limpio y manejable.	Posibles altas presiones al cambiar de fase. No está muy extendido,por desarrollar.

Tabla 8.1: Comparativa de sistemas de refrigeración en centros de datos

*El líquido dieléctrico químico puede utilizarse en inmersión en una fase también, se ha puesto de esta manera para apreciar la diferencia entre el aceite y el químico

Anexo D: Apuntes de las herramientas software

D.1 Minero CPUMiner-multi

La ejecución del minero en la Raspberry Pi nos permite diferentes opciones de uso, las más destacadas son:

- `-a, --algo:` Especifica el algoritmo para minar. Existen 15 tipos: sha256d, quark, fresh, x15, cryptonight...
- `-o, --url:` Dirección del servidor de la pool (depende de la criptomoneda).
- `-u, --username:` Dirección de correo electrónico asociada en MinerGate.
- `-p, --pass:` Contraseña de la cuenta en MinerGate.
- `-t, --threads:` Número de cores usados para minar.
- `-B, --background:` Ejecuta el minero en segundo plano.
- `--benchmark:` Ejecuta un benchmark offline.

.

Un ejemplo del comando final para criptomoneda Monero, algoritmo CryptoNight minado en un solo core y ejecución en segundo plano, sería el siguiente:

```
./minerd -a cryptonight -o stratum+tcp://xmr.pool.minergate.com:45700 -u ejemplo@gmail.com -p X -t 1 -B
```

D.2 Herramienta cpufreq-set

La herramienta software cpufreq-set disponible en sistemas Linux nos permite configurar la frecuencia de los procesadores de múltiples maneras. Las usadas en este trabajo son las siguientes:

- `cpufreq-set -r -g userspace`: Configuración manual de la frecuencia de la CPU, la Raspberry Pi tiene dos opciones:
  - `cpufreq-set -r -f 600Mhz:` Fija la frecuencia en 600Mhz.
  - `cpufreq-set -r -f 1.4Ghz:` Fija la frecuencia en 1.4Ghz.
- `cpufreq-set -r -g ondemand:` Adapta la frecuencia de la CPU dinámicamente en función de la carga del sistema.

D.3 Librería Python útiles para el uso de redes neuronales

Además de las herramientas software para crear redes neuronales, en este trabajo se usan diferentes librerías disponibles en Python. Sus usos van desde el preprocesamiento de datos hasta el cálculo de la métricas. Son las siguientes:

- *Csv*: Transforma los datos de diferentes formatos (txt, xlsx…) a formato csv y viceversa.
- *Numpy*: Paquete de computación científica de Python, útil para hacer cálculos con arrays N-dimensionales que utilizan las redes neuronales

- *Pandas*: Paquete de herramientas para facilitar el manejo de estructuras de datos y su análisis.
- *Matplotlib*: Librería gráfica que nos sirve para representar las predicciones de los diferentes modelos entrenados.
- *Sklearn*: Paquete muy amplio para análisis y minería de datos con todo tipo de funciones desde preprocesado hasta algoritmos de clasificación. regresión, etc. En nuestro caso la utilizaremos para dos cosas, realizar las métricas para evaluar los modelos y escalar los datos antes de que entren en las redes neuronales recurrentes, ya que a diferencia de las *feedforward*, es condición necesaria que los datos de entrada sean escalados previamente.
- *H5py*: Herramienta que permite guardar los modelos en formato json para un futuro uso.