Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Aplicación de Reconocimiento de Imágenes para Monitorización No Intrusiva de Carga Eléctrica

Autor: Hernán Calvo Aguiar
Tutor(a): Esteban Garcia Cuesta

Madrid, Junio 2024

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática

Título: Aplicación de Reconocimiento de Imágenes para Monitorización No Intrusiva de Carga Eléctrica

Junio 2024

Autor: Hernán Calvo Aguiar
Tutor: Esteban Garcia Cuesta
Departamento de Inteligencia Artificial (DIA)
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Resumen

La Monitorización No Intrusiva de Carga, es una técnica que busca administrar los consumos de diferentes dispositivos eléctricos sin la necesidad de medir directamente cada uno de ellos.

El objetivo de este proyecto es desarrollar la pieza clave de este sistema. Una Inteligencia Artificial que permita identificar los dispositivos conectados y consumiendo a una red eléctrica, como podría ser una casa, un restaurante o una oficina.

El modelo a diseñar deberá ser capaz de identificar los dispositivos conectados a la red eléctrica, y estimar el tiempo de uso de cada uno. Para ello, se ha decidido hacer uso de CSPNet, una arquitectura potente, que, aplicado a clasificadores de imágenes mejoran su precisión reduciendo el coste computacional de entrenamiento.

El objetivo de este Trabajo de Fin de Grado es desarrollar un sistema de monitorización no intrusiva de carga eléctrica mediante el uso de reconocimiento de imágenes.

El sistema diseñado es capaz de identificar los dispositivos conectados a la red eléctrica y estimar su tiempo de uso. Se ha utilizado el dataset REFIT, el cual proporciona datos de consumo energético a lo largo del tiempo de diferentes hogares. El proceso de desarrollo incluyó la extracción, codificación y almacenamiento de datos utilizando técnicas de programación multihilo y construcción de imágenes via la librería OpenCV. Además, se implementó un modelo multitarea basado en CSPResNeXt50, adaptado para la clasificación de dispositivos eléctricos y la estimación temporal de su uso.

Los resultados obtenidos demuestran la viabilidad del enfoque propuesto, mostrando un rendimiento aceptable pero con margen de mejora en la identificación y estimación del tiempo de uso de los dispositivos.

Finalmente, se plantean futuras líneas de trabajo que incluyen la implementación de un modelo CSPResNeXt de menor tamaño, la optimización del algoritmo de codificación para su ejecución en GPU y la incorporación de mecanismos para prevenir y cuantificar el overfitting.

Abstract

Abstract

Non-Intrusive Load Monitoring (NILM) is a technique that seeks to manage the consumption of different electrical devices without the need to measure each one directly.

The goal of this project is to develop the key component of this system: an Artificial Intelligence capable of identifying devices connected to and consuming from an electrical network, such as a house, restaurant, or office.

The designed model should be able to identify the devices connected to the electrical network and estimate their usage time. To achieve this, CSPNet, a powerful architecture that improves the precision of image classifiers while reducing computational training costs, was utilized.

The aim of this Final Degree Project is to develop a non-intrusive electrical load monitoring system using image recognition.

The designed system is capable of identifying devices connected to the electrical network and estimating their usage time. The REFIT dataset, which provides energy consumption data over time from different households, was used. The development process included data extraction, encoding, and storage using multithreading programming techniques and image construction via the OpenCV library. Additionally, a multitask model based on CSPResNeXt50 was implemented, adapted for the classification of electrical devices and the temporal estimation of their usage.

The results obtained demonstrate the feasibility of the proposed approach, showing acceptable performance but with room for improvement in the identification and estimation of device usage time.

Finally, future lines of work are proposed, including the implementation of a smaller CSPResNeXt model, optimization of the encoding algorithm for execution on GPU, and the incorporation of mechanisms to prevent and quantify overfitting.

Tabla de contenidos

Capítulo 1

Introducción

1.1. Motivación y Aplicaciones

Las técnicas de Monitorización No Intrusiva de Carga (en inglés NILM o *Non Intrusive Load Monitoring*) extraen el consumo de electrodomésticos a partir de una curva de consumo agregada. [1, pág. 1]. Dicho de otra forma. El monitoreo de carga no intrusivo es un sistema que infiere los consumos de diferentes dispositivos sobre una curva de consumo en la toma de corriente principal. Por ejemplo en la toma de corriente de una casa.

En Europa, un tercio de la energía eléctrica se ha generado mediante energías no renovables (12 % o 333TWh carbón, 17 % o 569TWh gas natural) [2] [3]. Aún siendo un descenso récord, para poder continuar reduciendo el consumo de energías no renovables, deben desarrollarse herramientas que nos permitan hacer un uso más inteligente de la generación de energía. Herramientas como la monitorización de carga no intrusiva pueden ayudar a detectar patrones de uso[4, pág. 11], para entonces poder desarrollar políticas que permitan optimizar la generación de energía.

Si, hipotéticamente, se redujese un 2 % la generación de electricidad a través de la combustión de carbón. Supondría una reducción de emisiones de 96.6 Millones de Toneladadas de CO2[1].

Realizar un monitoreo de carga no intrusivo es un problema complejo. Actualmente el área de soluciones NILM se encuentra en la fase de investigación y desarrollo. Por lo tanto. Mi objetivo para este trabajo es entrenar y analizar el rendimiento de una inteligencia artificial especializada en la clasificación de imágenes, transformando los datos de entrenamiento de series temporales a imágenes. Detallado en el **Capítulo 3**. El trabajo se centrará en cómo modificar el enfoque al problema de disgregación de consumos afecta al rendimiento de la técnica en comparación con otras técnicas del estado del arte.

Este concepto fue inventado por George W. Hart, Ed Kern y Fred Schweppe en el

[1] $333 * 10^3 \text{GWh} * 290 \frac{tCO_2}{GWh}$ [5]

Instituto Tecnológico de Massachussets, durante los años ochenta [6]. Fundados por el Electric Power Research Institute, se puede encontrar el proceso básico descrito en la patente estadounidense 4,858,141.

La técnica, descrita por Hart, Kern y Schweppe en [6] modela los consumos en lo que nombran como «Modelo Total de Carga». El modelo total de carga depende de que dispositivos estén encendidos en un momento dado. Por lo que definen un 'proceso de cambio'. Esto nos lleva a un problema computacionalmente imposible de resolver salvo mediante la fuerza bruta[6, pág. 4] y poco margen de mejora en el modelado para simplificar su resolución. De manera resumida, el problema trata en encontrar el número de dispositivos y el consumo de cada uno. Este es un problema que se beneficia de hacer uso de modelos heurísticos.

Siguiendo el objetivo establecido por Hart y con el asesoramiento de Esteban García Cuesta, se decidió en hacer uso de datasets clasificados para un entrenamiento supervisado; además de una arquitectura potente, compacta, computacionalmente barata en costes de entrenamiento y con un modelo preentrenado, para aplicar transferencia de aprendizaje.

1.2. Objetivos

Previamente al comienzo del desarrollo del proyecto, se establecieron una serie de objetivos a cumplir, dividiendo el objetivo final en pasos coherentes y concretos.

Este trabajo busca investigar sobre las técnicas más actuales en NILM[2]. Dando al lector una base de las técnicas utilizadas en el estado del arte y las matemáticas detrás de estas técnicas. Los objetivos de aprendizaje son extensos, el trabajo busca además profundizar en la práctica detrás de el entrenamiento de una inteligencia artificial y de los retos que presenta el uso de la computación paralela, bases de datos, GPUs, lenguajes de bajo nivel (C++, OpenCL) y el uso de frameworks de creación, entrenamiento y análisis del rendimiento de la inteligencia artificial.

1.3. Alcance del proyecto

Al finalizar este proyecto, se busca tener un modelo capaz de realizar una monitorización de carga no intrusiva, además de una serie de progamas y herramientas útiles a la hora de realizar trabajos futuros en el área. Este modelo deberá de ser capaz, idealmente, de realizar una identificadión de dispositivos eléctricos y la regresión temporal del tiempo de operación de cada dispositivo.

1.4. Estructura de la Memoria

La memoria de esta investigación tiene los siguientes capítulos.

[2]ver Apéndice A para un glosario de términos

- **Introducción:** Se introduce el tema, sus potenciales aplicaciones, motivación, alcance.
- **Fundamentos y Panorama Actual:** Base de la teoría detrás de la tecnología, algoritmos y conceptos sobre los que se desarrolla el trabajo de fin de grado. Además de informar sobre el estado del arte en el área.
- **Desarrollo:** Los pasos tomados para el desarrollo del modelo y las herramientas que utiliza. Entrenamiento, codificación, evaluación del modelo, etc.
- **Resultados y Análisis de Impacto:** Resultados del modelo y se compara su rendimiento con otros modelos. Además recoge el análisis del impacto del modelo conforme a los Objetivos de Desarrollo Sostenible de la Agenda 2030.
- **Conclusiones y Trabajo Futuro:** Conclusiones del trabajo y el trabajo futuro que puede hacerse para mejorar el modelo y sus partes.
- **Bibliografía:** Se recogen todos los recursos bibliográficos citados durante el transcurso de la investigación.
- **Anexo:** Código e información de interés que no tiene cabida en el cuerpo del texto.

Capítulo 2

Fundamentos y Estado del Arte

2.1. Conceptos generales

El concepto de NILM desde su introducción ha sido el método preferido por ingenieros e investigadores para la disgregación de consumo eléctrico, debido a sus ventajas económicas y prácticas.[7, pág. 2, pár. 4]

En cuanto al origen de este método, George Hart buscó categorizar en [6] los dispositivos dentro de tres grandes grupos, para poder estructurar los diferentes dispositivos y generalizar el modelado de dispositivos específicos. Estos grupos son:

1. **Señales de estados fijos** Siguen la arquitecturea de una máquina de estados finita, las transiciones entre estados son consumos exactos, al igual que los estados
2. **Señales transitorias** Esto son señales que no se encuentran en un espacio discreto que permita modelar su comportamiento como una máquina de estados. Un ejemplo podría ser un calefactor automático, que regula su generación de calor en función de la diferencia de la temperatura ambiente y la temperatura objetivo.
3. **Otras** Señales que no pueden categorizarse en ninguno de los dos casos anteriores. Por ejemplo, los cambios de dirección del motor de una lavadora en el proceso de lavado.

Es importante mencionar que los propios investigadores reconocieron que en el futuro los dispositivos eléctricos que siguiesen un diseño de máquinas de estados perderían precedencia. Ya que habría más 'dispositivos inteligentes' [6]. Actualmente, una calefacción programable o un aire acondicionado programable están al alcance de cualquiera. En España, en el transcurso de 2021 el 50 % de las viviendas en alquiler constaban de aire acondicionado[8].

Y esto es sólo un dispositivo. Habrá cientos de modelos de cada dispositivo, decenas de marcas, cada dispositivo con sus características específicas. Un modelado exhaustivo sería prácticamente imposible de crear y mantener, ya que

sólo para el mercado doméstico hay una cantidad desorbitante de dispositivos. Modelar todas las posibles combinaciones no es una posibilidad real.

Un ejemplo de sistema NILM

Un sistema NILM consta de: Una fuente de datos (acometida principal). Un dispositivo de recogida de datos (puede ser de Potencia Activa Y reactiva, Voltaje e Intensidad, etc). El modelo predictivo. Y el sistema de consulta de datos, que se ha omitido en la Figura 2.1.

Debido a estos argumentos, la investigación de la monitorización no intrusiva se inclinó a los modelos heurísiticos y el aprendizaje automático. Debido a sus ventajas y prevalencia en el estado del arte (ver 2.3,2.4)

2.2. Aprendizaje Automático

En la Historia de la Inteligencia Artificial, se han definido una serie de etapas de gran crecimiento y financiación como «veranos» [9] (tres concretamente). Alternados por (dos) «inviernos»; donde el interés, la financiación y el número de publicaciones decrecían significativamente [9]. En los principios de la década de los 2010, comenzó el tercer «verano» de la Inteligencia Artificial. En 2012 ocurre un avance técnico: unos años antes, Olga Russakovsky creó el reto ImageNet [10]; un repositorio de más de un millón de imágenes de objetos a lo largo de más de mil categorías diferentes. Este reto estimuló a la comunidad científica,

Resulta lógico que el número de artículos científicos proponiendo modelos que permitiesen resolver los retos que presenta NIML aumentase, debido al comienzo de este tercer verano.

De los que se destacan los Modelos Ocultos de Markov, el Sparse Coding y finalmente las Redes Neuronales. Siendo estas últimas el foco principal del trabajo, y de esta sección.

Identificando Datos Ocultos Con Cadenas de Markov

Una cadena de Markov resulta útil cuando se tiene que modelar un comportamiento en base a una secuencia de eventos observables, siendo L.E. Baum y Ted Petrie los autores de [11], el primero de una serie de papers sobre modelos probabilísticos y estadística que introdujeron los HMM. Estos eventos observables son elementos conocidos, pero la secuencia de estados del modelo para esos eventos observables son desconocidos.

El problema, como en NILM, es que los estados no son directamente observables. Estan *ocultos* y por tanto debemos ajustar la cadena para incluir una secuencia de posibles similitudes observadas [12].

En el caso de una cadena de Markov. Dados una serie de eventos observables, podemos inferir la probabilidad de cada estado que causaron estos eventos observables: Calculando la probabilidad de una cadena de eventos observables

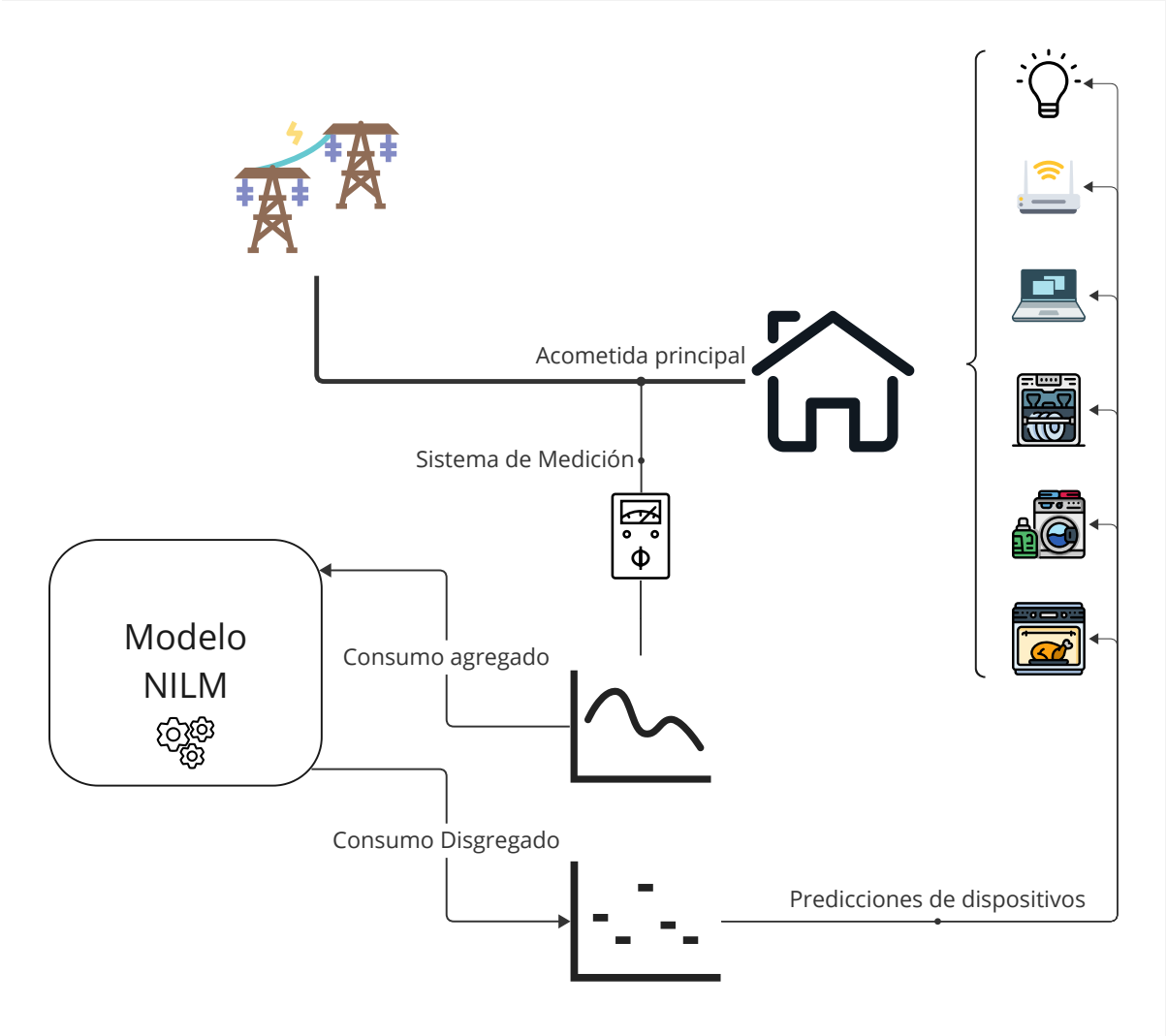

Figura 2.1: Un diagrama simple de un sistema NILM
Fuente: Elaboración propia

dada la probabilidad de una cadena de estados del conjunto, para todas las posibles combinaciones de la cadena los estados del conjunto. Este algoritmo se conoce como el algoritmo de Viterbi. Kaustav Basu propone un modelo basado en este algoritmo. Sin embargo, el modelo HMM mostró menor rendimiento que el algoritmo KNN [13].

Este es el modelado que permite inferir de manera probabilística. Se destaca sobre las cadenas de markov una serie de algoritmos para determinar la cadena de estados ocultos más probable para la cadena de eventos observables, que se omiten por brevedad, pero pueden consultarse en [12], para una detallada explicación sobre este área.

Modelos Ocultos de Markov

Explicadas brevemente las bases de los Modelos Ocultos de Markov,se detalla la aplicación de este modelo estadístico en la disgregación de consumos.

Dado que un modelo de Markov oculto infiere probabilísticamente una serie de estados, esto nos resulta útil para la disgregación de dispositivos. Para codificar diferentes estados de diferentes dispositivos se aumenta el número de cadenas del modelo. Teniendo una matriz de estados S_i^j donde S es un conjunto de estados; las observaciones dependen de todos los estados en un momento dado.

Los Modelos de Markov Ocultos se han tratado en la literatura como efectivos en comparación con otras técnicas de monitorización cuando el entrenamiento supervisado presenta una baja tasa de muestreo en el eje temporal de la obtención de datos [14]. Otras ventajas que presenta es su bajo coste computacional en el entrenamiento. Pero el uso de este tipo de elementos lo vuelven altamente susceptible a máximos locales[1].

En el caso de los Modelos Factoriales de Markov Ocultos (con siglas FHMM en inglés), presentan ciertas ventajas en comparación con los HMM [1]: La complejidad y el coste computacional de los los algoritmos de aprendizaje e inferencia son menores para los HMM. Sin embargo, tanto FHMM como HMM presentan limitaciones al volumen máximo de dispositivos simultáneos, incluso en casos como en la investigación de Kolter, J. Ziko [15] donde se propone un nuevo algoritmo no supervisado. El algoritmo desarrollado (AFMAP) no presenta los problemas de alta complejidad y máximos locales, pero requiere de etiquetación manual después de la disgregación. Y presenta bajo rendimiento para dispositivos electrónicos y de cocina [1, pág. 5].

El modelo propuesto por Parson et. al. 2014 [16] presenta un alto rendimiento para clasificar dispositivos con un número de estados finitos.

2.3. Redes Neuronales

En esta sección se tratan tres conceptos de las Redes Neuronales; se dará una introducción y se expondrán las aplicaciones pertinentes al tema del proyecto.

[1] Hidden Markov Models, o en español los Modelos Ocultos de Markov

2.3.1. Sparse Coding

Introducido por Olshausen et. al. 1997 [17]. El Sparse Coding consiste en reducir la activación de la capa oculta de las neuronas, forzando a la red neuronal a realizar un aprendizaje [18]. Su objetivo es encontrar un conjunto de vectores ϕ_i tal que podamos representar un vector x como una combinación lineal de estos vectores:

$$x = \sum_{i=1}^{k} a_i \phi_i$$

Este es un concepto que está fuertemente relacionado con los autoencoders y el aprendizaje PCA. Que tienen como objetivo es que su salida sea igual a la entrada, pero limitando la capa oculta para forzar un aprendizaje [19]. Por ejemplo, suponiendo que se buscase utilizar un autoencoder para comprimir una imagen para enviarla por internet. Si no se implementasen restricciones, dado que el objetivo de la red neuronal es reducir la pérdida de información a 0; la salida del autoencoder sería la imagen, completamente idéntica.

Para evitar esto se introduce una capa oculta de menor tamaño en comparación a la capa de entrada como restricción para forzar un aprendizaje, al tener una capa oculta, la capa de salida (de igual tamaño a la de entrada) debe «reconstruir» la entrada [19] .

Volviendo al Sparse Coding. Al tener un conjunto de vectores sobrecompleto, los coeficientes a_i ya no se determinan únicamente por el vector de entrada x. Luego se define una función de escasez[2]. Definimos esta función de escasez como: tener los menos componentes (a_i) distintos de cero o el mínimo de componentes no cerca de cero [18].

Tanto los autoencoders como el sparse coding han sido usados en la literatura NILM.

En la solución propuesta por Mengheng Xue en [20], buscan separar una señal de consumo agregada sin dar información sobre sus componentes, identificando componentes a través de una arquitectura escasa.

2.3.2. Redes Recurrentes

Introducidas por una serie de investigadores: Rumelhart et. al. 1986 [21]; introdujo el algoritmo de retropropagación, una herramienta esencial en el entrenamiento de redes neuronales. Elman(1990) avanzó sobre las propuestas de Jordan(1986)[22] creando modelos que integrasen representaciones temporales de datos [23] . Las redes recurrentes son un tipo de red neuronal profunda. Son particularmente efectivas para procesar secuencias de datos, reteniendo información sobre estados previos en su procesado de entradas. Esto es especialmente útil para tareas de clasificación de series temporales, como NILM. Un ejemplo específico de la aplicación de RNN en NILM es su uso en modelos de secuencia a secuencia (seq2seq), donde los datos de entrada son secuencias de

[2] De ahí el «Sparse»; escasez en Español

mediciones de energía y las salidas son secuencias de estados de los aparatos (encendido/apagado) o su consumo específico [24][25].

2.4. Especificación del Estado del Arte

Estado del arte más relacionado con los objetivos y el enfoque del proyecto.

2.4.1. Codificación de imágenes para NILM

En la literatura, la codificación de imágenes es un enfoque presente en varios proyectos. Los modelos propuestos por Shikha Singh et al 2017 [26] trabajan sobre imágenes codificadas con una transformada de ondícula de 2 niveles, esto permite extraer diferentes resoluciones de un rango temporal de consumo.

En cuanto al uso de codificación GAF, Ang Gao et. al. 2023 [27] es el único hasta donde llegan nuestros conocimientos que ha hecho uso de esta codificación en relación al enfoque del trabajo.

2.4.2. CSPNet

CSPNet es una arquitectura de redes neuronales introducida por Chien-Yao Wang et. al. 2020 [28]. Al aplicarse esta arquitectura a clasificadores de imágenes como ResNeXt [29] o DenseNet [30] , optimizó su rendimiento. Reduciendo costes de computación, uso de memoria y además beneficiando la velocidad de inferencia y la precisión de estos modelos [28].

Hasta donde llegan nuestros conocimientos, no se ha llegado a proponer el uso de un clasificador de imágenes con arquitectura CSPNet integrada para realizar disgregación de consumo eléctrico.

Hay ciertas similitudes en la literatura [31]. [32]

En la publicación de Michele D'Inececco [33], se explora el aprendizaje transferido con varios datasets, entre ellos REFIT e UK-DALE aplicando una red neuronal convolucional de 5 capas, el aprendizaje a transferir siendo las señales específicas de los dispositivos eléctricos via aprendizaje seq2point.

2.4.3. Validación de Modelos: NILMTK

NILMTK, publicado por Nipun Batra et. al. 2014 [34], presenta un toolkit de código abierto diseñado para facilitar la investigación y el desarrollo en el campo. Consta de interfaz de usuario, librerías, datasets estándar, algoritmos de disgregación y la flexibilidad y extensibilidad para integrar nuevos algoritmos y datasets. En 2019, publica NILMTK-contrib [35], una extensión comunitaria de NILMTK que busca acelerar la investigación en NILM mediante contribuciones comunitarias.

En lo que respecta al impacto, actualmente no ha habido grandes contribuciones al proyecto, el repositorio de nilmtk-contrib lleva tres años sin recibir ninguna actualización y sólo tres forks de 57 han tenido actividad el último año.

Capítulo 3

Desarrollo

El hardware utilizado tiene las siguientes especificaciones:

- CPU: AMD Ryzen 5 3600XT (12) @ 3.800
- GPU: AMD Radeon RX6800XT
- RAM: 16GB CL16 @ 3000
- DISCO: SSD NVME 500GB
- SSOO: Linux 5.15.0-107-generic #117 20.04.1-Ubuntu

El desarrollo fue realizado en Visual Studio Code, haciendo uso de Git y GitHub como sistema de control de versiones y copia de seguridad remota del desarrollo realizado. El repositorio consta de la siguiente estructura de carpetas:

```
datasets/
   CLEAN_REFIT_081116/
encoding
   REFIT_GAF/
      house1/
      ...
      house21/
      meta/
   thSafe/
full_model
    cspvenv/
    data/
       REFIT_GAF -> /encoding/REFIT_GAF
    first_test/
    src/
```

3.1. Planificación

Se realizó dos tandas de organización y planificación de tareas. La primera al principio del semestre y la segunda a mediados de abril. Las dificultades en la planificación fueron una combinación de motivos externos y una perspectiva algo ingenua del tiempo a invertir y el ritmo de progreso. Debido a la falta de conocimientos, se ha dedicado un tiempo considerable (unas 60-80 horas) a realizar investigación y estudio, para poder comprender las técnicas y modelos implementados el estado del arte.

En lo que respecta al desarrollo, se han dedicado unas 80-100 horas.

Los principales problemas fueron: la dificultad para programar un codificador en C++ y OpenCL, debido a la curva de dificultad que representa aprender un lenguaje para una nueva arquitectura y los retos que presenta la gran cantidad de datos usada para el entrenamiento de inteligencia artificial.

3.2. REFIT Dataset

Tras una investigación de los potenicales datasets, se decidió usar como dataset REFIT. El nombre completo del dataset es «Personalised Retrofit Decision Support Tools For UK Homes Using Smart Home Technology»[36].

3.2.1. Características principales

Puede encontrarse todo el detalle sobre el dataset en el artículo publicado por David Murray, Lina Stankovic y Vladimir Stankovic [36].

El dataset recoge datos de 20 casas durante un periodo continuo de dos años, con un periodo de medición superior a un minuto entre mediciones para todas las casas. Siendo este el único dataset de Gran Bretaña con estas características. Se miden 9 dispositivos además del consumo agregado. Todas las mediciones se realizaron en vatios.

El dataset ocupa unos 5GB de memoria, se presentan todas las mediciones de una sola casa en archivos CSV. Siguiendo la distribución:

Time, Unix, Aggregate, Appliance1, ... , Appliance9, Issues

En total, hay 20 archivos en recogen datos, y un vigésimo primer archivo conteniendo los metadatos de cada casa. Estos metadatos se limitan a especificar el tipo de dispositivo asociado al dispositivo o *Appliance*

En cuanto a la recogida de datos, Murray indica un periodo de recogida de datos de ocho segundos. Por tanto los datos se recogieron con un desplazamiento temporal. Esto quiere decir que los datos de los dispositivos y el agregado no coinciden exactamente en el tiempo, habiendo una diferencia de unos segundos entre dispositivo y dispositvo. El acumulado (o agregado) se calculaba antes de que finalizase el periodo de ocho segundos.

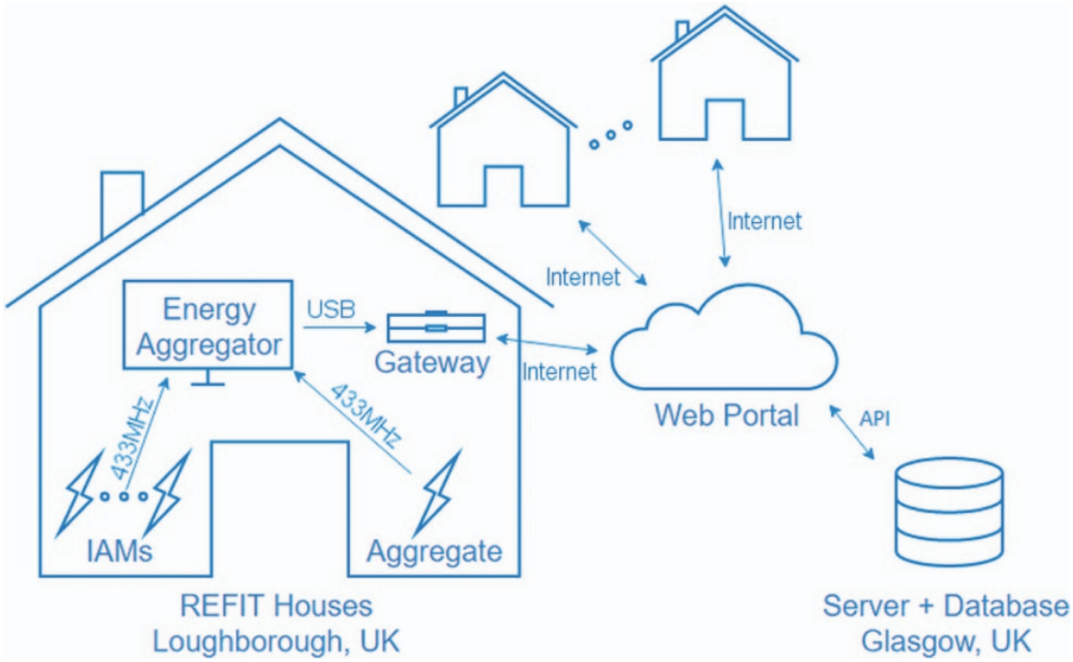

Figura 3.1: El modelo de recolección de datos (propiedad de [36])

3.2.2. Pros y contras del dataset

Las ventajas de este dataset son, su longitud en el tiempo y la facilidad de empezar a hacer un uso de los mismos. Ya que se busca investigar sobre la viabilidad de un modelo que haga uso de clasificadores de imágenes, esto habilita la posibilidad de realizar un trabajo de fin de grado con este tema.

Hay una serie de problemas conocidos sobre el dataset. En el artículo de David Murray [36] se mencionan los siguientes:

- Ocasionalmente los medidores de dispositivos reportaban consumos muy por encima de de la carga máxima para un dispositivo de uso doméstico. Estas medidas se eliminaron del dataset.
- Hubo problemas con la sincronización temporal de los medidores de dispositivos. Esto conlleva discrepancias entre el agregado y las mediciones.
- Las mediciones del acumulado en casas 3, 11 y 21 se vieron afectadas debido a que los domicilios contaban con paneles solares, y recablear para evitar afectar a las medidas del acumulado no era posible.
- En algunos casos, los valores entre acumulado y dispositivos no coinciden debido a que no se monitorizaron otras variables para ajustar estas medidas conforme el ángulo de fase o el voltaje por cargas inductivas o capacitivas.

Debido a falta de clasificación en los metadatos, las casas 12 y 13 no fueron utilizadas para el entrenamiento.

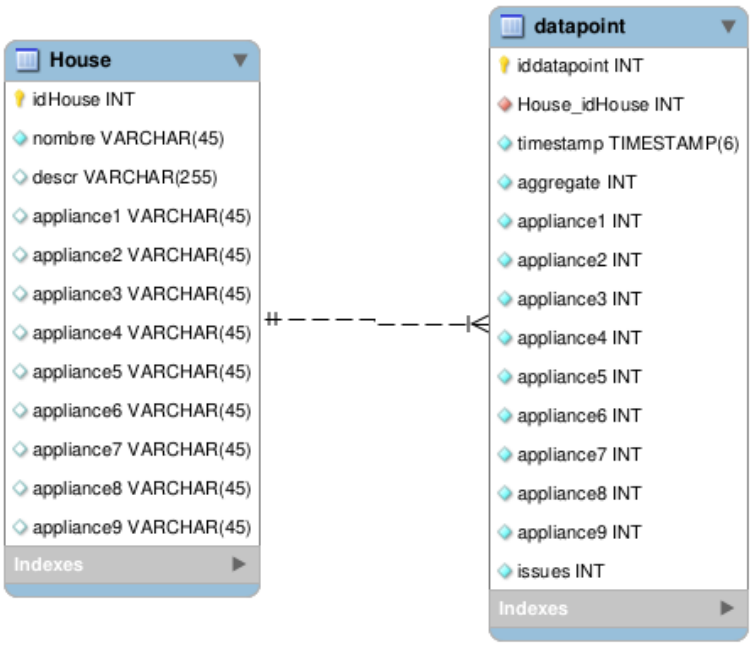

Figura 3.2: El modelo de la BBDD en el parseo. Elaboración propia via MySQL Workbench

3.3. Parseado de REFIT a MYSQL

Como primer paso, se decidió parsear todos los datos a una base de datos, escogiendo MYSQL como plataforma, para tener una lógica de acceso que permitiese ordenar los datos para su codificación conforme a diferentes algoritmos (secuencial, k-means) para poder extraer la mayor cantidad de características sobre los dispositivos en diferentes marcos temporales.

3.3.1. Inconvenientes

La inserción de datos en la base de datos fue tremendamente lenta. Para insertar un CSV entero, se tardaba en el peor de los casos cerca de las 72h de inserción continuada. En parte esto se debe a que se mantuvo durante un tiempo la configuración del servidor MYSQL *out of the box* sin realizar ningún cambio.

Más adelante se investigó cómo poder optimizar las inserciones. La solución fue reducir el número de inserciones a la base de datos y aumentar el volumen de datos por inserción. Quinientas mediciones por INSERT redujeron drásticamente el tiempo necesario, de 3 días a unos 15 minutos.

3.3.2. Modelo de la BBDD

Para este paso del desarrollo, se decidió un modelo simple, imitando la presentación de los datos en el csv.

3.3.3. Herramientas desarrolladas

Para la inserción, se diseñaron dos programas en C++. Haciendo uso de la librería de conector de C++ a mysql. Se seleccionaba el archivo manualmente y comenzaba la inserción. Una vez iniciada la inserción de datos, esta se paraba en la misma fecha, el 31 de marzo. Esto se debía a que al insertar la fecha, habiendo un cambio de hora esa madrugada, las fechas no coincidían con las esperadas.

La solución a este problema fue desarrollar un programa que permitiese saltar a la línea donde daba el fallo, además de una serie de herramientas para «debuggear» la inserción de datos, pudiendo saltar a una línea específica e insertar manualmente o continuar la inserción.

Podemos observar el comportamiento de la herramienta, llamada sin argumentos:

```
    $ ./traversecsv
    Usage: ./traversecsv <file> <house_id>
```

Y llamada con argumentos

```
$ ./traversecsv CLEAN_REFIT_081116/CLEAN_House15.csv 15

Successfully connected to the database!
    Total lines in file: 6225697
    Press 'p' to print the next line
        'i' to continue inserting
        'j' to jump to a specific line
        'r' to continue inserting until EOF
        's' to search for a line with a specific date
```

Además de desarrollar el programa, se cambió la configuración de fechas en el servidor MYSQL.

```
    SELECT @@global.time_zone, @@session.time_zone;
    SET GLOBAL time_zone = '+00:00';
    SET SESSION time_zone = '+00:00';
```

Este programa y su código fuente puede encontrarse a partir de la raíz del directorio dentro de «*datasets/*».

3.4. Codificación GAF

El método de codificación sigue una serie de pasos relativamente simples:

```
(pseudocódigo)
1. Dado un vector V de n elementos.
for (n)
    V(n)=arccos(V(n))
```

```
        if V(n) > 1
            V(n)=1
        else if V(n) < -1
            V(n)=-1
    2. construimos la matriz M
    for (i)
        for (j)
            M=cos(V(i)+V(j))
```

3.4.1. Primer Diseño

Dado el volumen de los datos, se ideó un programa multihilo, con gestión asíncrona de llamadas y computación en GPU via kernels de OpenCL.

Se decidió separar el proceso de codificación en tres partes. Extracción de datos, Codificación de datos y Almacenamiento de Metadatos.

La extracción es altamente paralelizable. Consiste en extraer de la base de datos en bloques de un cierto tamaño y preparar series de datos para guardarlos en una estructura de datos accesible para los hilos posteriores. Se decidió por un hashmap thread safe. Para notificar a los procesos de codificación las series que están listas para ser codificadas, se instanció una cola con mecanismos de sincronización para evitar posibles condiciones de carrera. Para reducir la redundancia de datos y el espacio de las estructuras de datos, sólo se introducen en la cola de series extraídas las claves resultantes de insertarlas en el hashmap. Los hilos codificadores consumen claves de la cola de series extraídas, copiando las series del hashmap a un buffer y codificando el buffer. Se hace uso de un buffer por el mismo motivo que la extracción es en bloques de datos, para reducir el overhead de hacer una llamada de sistema. Una vez codificada una serie, se introduce de vuelta en el hashmap y se introduce su clave en la cola de series codificadas. El Almacenamiento de metadatos copia el vector one-hot generado por el extractor y lo apenda a un archivo de texto. Las series codificadas siguen la misma convención para sus archivos.

La motivación de gestionar las llamadas asíncronas debido a que el conector a la BBDD MYSQL las soportaba pero no daba más alternativas que la espera activa llevó al desarrollo de la clase *EventLoop.hpp*. Se tomó prestado un concepto muy usado en servidores, en los que los sockets de un servidor (procesos individuales) traspasan sus peticiones a un proceso que se encarga de ejecutar las peticiones. Manteniendo así el coste de mantener una conexión con un servidor como un elemento separado de las operaciones del servicio. En el caso de la codificación, se implementó una solución de inserción de peticiones asíncronas, basado en promesas y futuros de funciones lambda que los procesos enviaban al EventLoop. Este se encargaba de recibirlas e instanciar un thread para ejecutar la petición, comprobando todas las peticiones de manera cíclica y devolviendo las promesas a los threads peticionantes (ver 3.3).

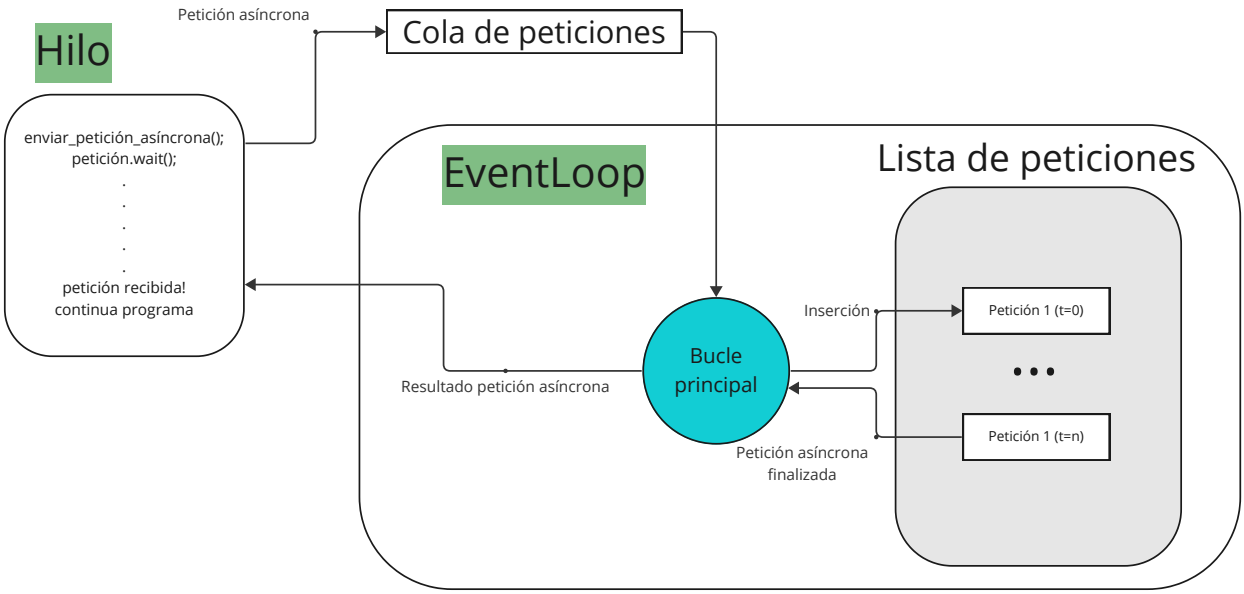

Figura 3.3: Diseño abstracto del funcionamiento del EventLoop. Elaboración propia

3.4.2. Inconvenientes

Este proceso de desarrollo se vio afectado por la optimización prematura. Sin embargo, se realizaron una serie de aprendizajes muy útiles en áreas que podrían beneficiar a futuro la escalabilidad de procesos como este en el campo de la inteligencia artificial.

EventLoop

Aún presentando una serie de ventajas a escala, la dificultad de devolver punteros sin desreferenciarlos fue el principal obstáculo de este sistema. Por lo que sólo se hizo un uso parcial de la solución desarrollada.

OpenCL

Para la codificación de datos se intentó realizar un desarrollo de kernels en OpenCL que aprovechasen las ventajas de computación paralela de la GPU. Debido a una falta de experiencia en desarrollo en OpenCL y a la naturaleza temporal del trabajo, se debió de abandonar esta rama.

3.4.3. Diseño Final

Debido a los inconvenientes expuestos, el diseño final se hizo exclusivamente en la cpu, haciendo uso de openCV para construir las imágenes.

Se decidió instanciar tríos de hilos.

Parámetros de invocación:

```
$ ./host
Usage:
./host <house number> <number of threads> <batch size> <number of batches>
SERIES SIZE IS A DEFINE
```

La última línea es un comentario relevante debido a que se consideró la posibilidad de alternar el tamaño de la serie y por ende el tamaño de las imágenes codificadas.

Tiempo transcurrido en la codificación del los datos contenidos en *datasets/CLEAN_REFIT_081116/CLEAN_House21.csv*.

```
time ./host 22 7 21031 256

real    6m28,529s
user    38m35,273s
sys     1m52,445s
```

Este programa y su código fuente puede encontrarse a partir de la raíz del directorio dentro de «*encoding/*».

3.5. Modelo y Entrenamiento

3.5.1. Modelo

Se hace uso del modelo preentrenado CSPResNeXt50 de la librería timm por HuggingFace [37]. Debido a la naturaleza multidimensional de los datos y a las predicciones deseadas del modelo, se requiere una serie de ajustes a la capa de entrada y a la de salida. Por lo tanto los pesos de esas dos capas serán perdidas.

El modelo CSPResNeXt50 es un modelo de clasificación de imágenes. El objetivo de este trabajo es aplicar transfer learning a un modelo ya preentrenado, para estudiar el rendimiento del modelo en un problema de clasificación de series temporales codificadas como imágenes.

Este modelo se basa en la arquitectura ResNeXt50, cuyas principales ventajas radican en la eficiencia de la red y el uso de la cardinalidad (dimensión extra) para aprender características más complejas, aumentando la precisión sin aumentar el ancho o profundidad de la red [29]. La arquitectura CSPNet se basa en la arquitectura ResNet, pero con la adición de un módulo CSP (Cross Stage Partial) que permite la fusión de características de diferentes etapas de la red, permitiendo una mejor generalización y aprendizaje de características más complejas [38].

Se decidió por un modelo multitarea. Ya que lo que muchos modelos no han alcanzado a realizar de forma exitosa es una clasificación de dispositivos eléctricos y una estimación del tiempo en el que se encuentran encendidos.

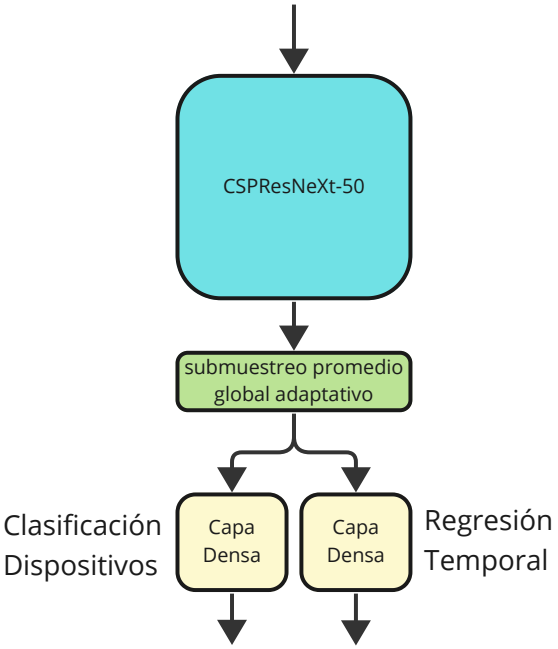

Figura 3.4: Diseño abstracto del modelo. Elaboración propia

Se extrae de la penúltima capa de CSPResNeXt50 los *features* del modelo, introduciéndose a una capa de capa de submuestreo promedio global adaptativo. Esta capa es la entrada de nuestras dos capas de salida, dos capas densas que realizan la estimación de clases y estimación temporal (ver 3.4). Estos datos son dos vectores de 23 elementos cada uno. El primero es un vector one-hot y el segundo es un vector de valores enteros que representan el número de instancias de cada clase en la imagen.

3.5.2. Pytorch

Debido a que el único modelo que se encontró que cumplía los requisitos (modelo preentrenado, clasificación de imágenes, enfoque CSPNet) fue el modelo CSPResNeXt50, se decidió hacer uso de la librería Pytorch para el entrenamiento del modelo, ya que el modelo preentrenado se encontraba en la librería timm de HuggingFace. Los modelos con enfoque CSPNet están fuertemente relacionados con la clasificación de imágenes que hacen uso de capas especializadas para la extracción de características que no resultan ventajosas para la tarea en cuestión. Por lo que la mayoría de modelos requerían una serie de modificaciones que salían del alcance del proyecto.

Para entrenar el modelo se hizo uso de la librería Pytorch, que permite el uso de la GPU para el entrenamiento de modelos de aprendizaje profundo. Para esto, se instaló el módulo ROCm de Pytorch, que permite el uso de las GPU de AMD. Y la librería de Pytorch compatible con ROCm

El proceso de entrenamiento escogido fue el de desarrollar un script que implementa el dataloader de pytorch para cargar las imágenes codificadas y sus

etiquetas. Y el desarrollo de un script que realiza las modificaciones necesarias al modelo para poder cumplir los requisitos de la tarea.

En cuanto a los datos, tanto los vectores como la predicción temporal se expandieron a 23 elementos (las 23 clases del dataset). Los vectores temporales se normalizaron, mientras que el vector de clases no se normalizó.

3.5.3. Entrenamiento

Se desarrolló un script de entrenamiento. Los parámetros escogidos fueron los siguientes:

- Número de épocas: 31
- Tamaño del lote: 20
- Learning rate: 0.001
- Optimizador: Adam
- Función de pérdida 1: CrossEntropyLoss[1]
- Función de pérdida 2: MSELoss

El entrenamiento se realizó en una GPU AMD Radeon RX6800XT. El tiempo de entrenamiento fue de en torno a las 22 horas para las 31 épocas.

Cálculo de Error

Se decidieron por dos funciones de pérdida. La primera, la función de error con Entropía cruzada. Para la clasificación de los dispositivos eléctricos. Esta función de entropía penaliza solo las predicciones correctas, calculando el error sólo para predicciones correctas.

$$CE = -\sum_{j=1}^{n} sum_{i=1}^{n} y_i \log \hat{y}_i \quad [39]$$

Para implementar una función de pérdida de entropía cruzada personalizada, se aplica la función log-softmax a las predicciones. La función log-softmax se define como:

$$\hat{y}_{\text{pred},ij} = \log\left(\frac{\exp(y_{\text{pred},ij})}{\sum_{k=1}^{C}\exp(y_{\text{pred},ik})}\right)$$

donde $y_{\text{pred},ij}$ es el logit predicho para la instancia i y la clase j, y C es el número de clases.

A continuación, se calcula el producto elemento por elemento de las etiquetas verdaderas y las predicciones log-softmax:

[1]Realmente es una función que se basa en Cross Entropy Loss

$$\mathrm{E}_i = -\sum_{j=1}^{C} y_{\text{true},ij} \cdot \hat{y}_{\text{pred},ij}$$

donde $y_{\text{true},ij}$ es la etiqueta verdadera para la instancia i y la clase j.

Luego, se suman las pérdidas a través de todas las clases para cada instancia:

$$\mathrm{E}_i = -\sum_{j=1}^{C} y_{\text{true},ij} \cdot \log\left(\frac{\exp(y_{\text{pred},ij})}{\sum_{k=1}^{C}\exp(y_{\text{pred},ik})}\right)$$

Finalmente, se calcula la media de la pérdida a través de todas las instancias en el lote:

$$\mathrm{E} = \frac{1}{N}\sum_{i=1}^{N}\mathrm{E}_i$$

donde N es el número de instancias en el lote.

Combinando estos pasos, la expresión final para la función de pérdida de entropía cruzada personalizada es:

$$\mathrm{E} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{\text{true},ij} \cdot \log\left(\frac{\exp(y_{\text{pred},ij})}{\sum_{k=1}^{C}\exp(y_{\text{pred},ik})}\right)$$

En python la implementación es la siguiente:

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class CustomCrossEntropyLoss(nn.Module):
        super(CustomCrossEntropyLoss, self).__init__()

    def forward(self, y_pred, y_true):
        y_pred = F.log_softmax(y_pred, dim=1)
        loss = -torch.sum(y_true * y_pred, dim=1)
        return loss.mean()
```

Para el segundo error, la regresión temporal de los tiempos en los que se encuentran encendidos los dispositivos eléctricos. Se escogió el error cuadrático medio.

Al hacer uso de dos funciones de error, y tener salidas diferentes para el modelo, se establece un equilibrado de pérdidas, sólo durante el periodo de entrenamiento.

Se decidió por un equilibrado de pérdidas con pesos basados en los gradientes de clases y tiempo. Se extraen los gradientes de la penúltima capa del modelo. Se normalizan y establecen los pesos como:

$$\text{peso}_k = \left(\frac{\text{grad norm}_k}{\frac{1}{N}\sum_{i=1}^{N}\text{grad norm}_i}\right)^{\alpha}$$

Donde α es un hiperparámetro que controla la sensitividad del ajuste de los pesos. Por último, se normalizan los pesos para que no se desestabilice la función total de pérdida.

Puede observarse el código en python asociado:

```
def calculate_loss(model, pred_class_count, pred_time,
        true_class_count, true_time, training, alpha=1):
# Compute the losses
class_loss = CustomCrossEntropyLoss()(pred_class_count, true_class_count)
time_loss = F.mse_loss(pred_time, true_time)

if training:
    # Zero out any existing gradients
    model.zero_grad()

    # Calculate gradients for class_loss
    class_loss.backward(retain_graph=True)
    class_grads = [p.grad.clone().detach() for p in model.parameters()
            if p.grad is not None]
    class_grad_norm = torch.sqrt(sum([g.norm() ** 2 for g in class_grads]))

    # Zero gradients before calculating for the next loss
    model.zero_grad()

    # Calculate gradients for time_loss
    time_loss.backward(retain_graph=True)
    time_grads = [p.grad.clone().detach() for p in model.parameters()
            if p.grad is not None]
    time_grad_norm = torch.sqrt(sum([g.norm() ** 2 for g in time_grads]))

    avg_grad_norm = (class_grad_norm + time_grad_norm) / 2
    class_weight = (class_grad_norm / avg_grad_norm).pow(alpha)
    time_weight = (time_grad_norm / avg_grad_norm).pow(alpha)

    sum_weights = class_weight + time_weight
    class_weight /= sum_weights
    time_weight /= sum_weights

    loss = class_loss * class_weight + time_loss * time_weight
else:
```

```
    loss = class_loss + time_loss

    return loss
```

3.5.4. Inconvenientes

El entrenamiento del modelo se vio afectado por la falta de experiencia en el uso de Pytorch y la falta de tiempo para investigar y aprender sobre la librería. Sin embargo, se logró realizar un entrenamiento exitoso del modelo.

Los principales inconvenientes que presenta el modelo se deben al tamaño de la red preentrenada. Es muy grande y podría beneficiarse de una reducción de tamaño. Además, el modelo CSPResNeXt50 no es el más ligero, por lo que el entrenamiento del modelo es lento y requiere de una GPU potente para poder realizarlo en un tiempo razonable.

Debido al hardware disponible, la GPU requirió de una serie de ajustes (undervolt, limitación de potencia y regulación de relojes) que limitaron el número de épocas y el tamaño de lote que podían realizarse en un tiempo razonable.

La validación del modelo no pudo implementarse eficientemente (alternando entre entrenamiento y validación para cada una de las épocas).

Los modelos multifuncionales son complejos de crear, entrenar y validar, este modelo no se ha mantenido exento de esos retos. Especialmente para alguien con poca o ninguna experiencia.

Capítulo 4

Resultados y Análisis de Impacto

4.1. Resultados

4.1.1. Costes computacionales

El entrenamiento del modelo ocupó en torno a los 4GB de RAM y 10GB de VRAM. El tiempo de ocupación de la gpu era en torno al 20 % del total, permitiendo el uso del ordenador para otras tareas como la redacción de esta memoria durante el transcurso del entrenamiento.

4.1.2. Evaluación del modelo

Según expuesto por Ebony Mayhorn en el reporte de 2023 del Multidisciplinary Digital Publishing Institute. Actualmente, no existe ningún estándar que permita realizar comparativas entre diferentes tecnologías NILM [40] [41]

Para evaluar el modelo, se hace uso de métricas definidas por Ang Gao en [27]: accuracy, F1 score, y weighted F1 score[1]. Por ello, y por el enfoque del proyecto, no se exponen comparaciones entre otros modelos

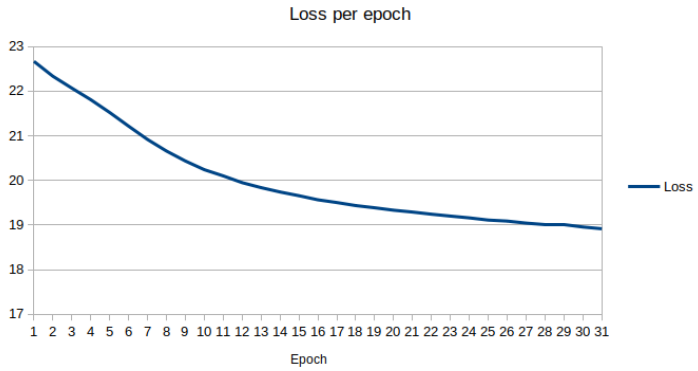

Figura 4.1: Pérdida total del modelo durante el entrenamiento

[1]modificadas para el análisis multi-etiqueta

Como podemos observar en 4.1 la pérdida disminuye consistentemente a lo largo del entrenamiento. Una pérdida decreciente típicamente significa que el modelo está aprendiendo y minimizando el error con el tiempo.

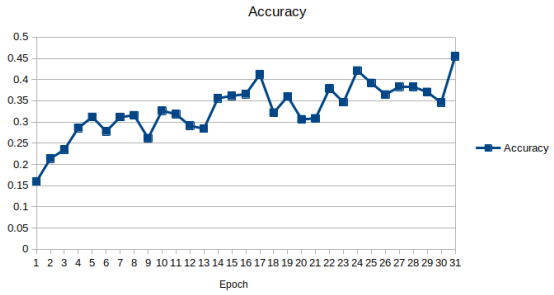

Figura 4.2: Precisión del modelo

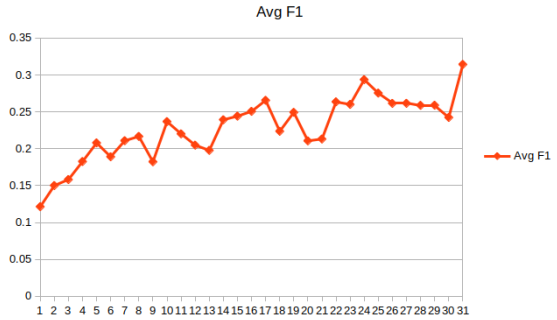

Figura 4.3: Puntuaje F1 Promedio

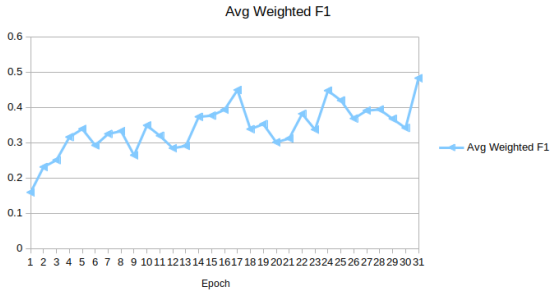

Figura 4.4: Puntaje F1 Ponderado Promedio

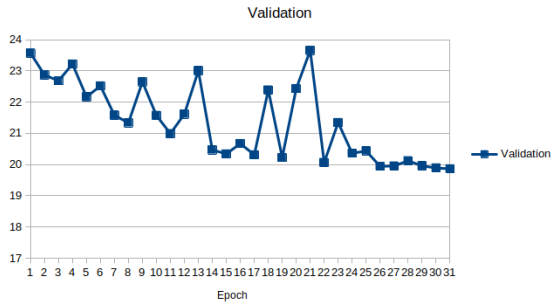

Figura 4.5: Validación

Figura 4.6: Métricas del modelo

Todas las métricas sugieren una mejora general en el rendimiento a lo largo de las épocas. Las métricas de exactitud, puntaje F1 promedio y puntaje F1 ponderado promedio presentan un incremento constante, indicando una mejor capacidad del modelo para clasificar correctamente y manejar el desequilibrio de clases. Sin embargo, se observan fluctuaciones significativas en la pérdida de validación, lo que sugiere la presencia de sobreajuste. Esto se refleja también en los picos observados en la pérdida de validación, que indican que el modelo tiene dificultades para generalizar en algunos puntos del entrenamiento.Las métricas de exactitud, puntaje F1 promedio y puntaje F1 ponderado promedio presentan un incremento constante, indicando una mejor capacidad del modelo para clasificar correctamente y manejar el desequilibrio de clases. Sin embargo, se observan fluctuaciones significativas en la pérdida de validación, lo que sugiere la presencia de sobreajuste. Esto se refleja también en los picos observados en la pérdida de validación, que indican que el modelo tiene dificultades para generalizar en algunos puntos del entrenamiento.

El sobreajuste ocurre cuando un modelo se adapta demasiado bien a los datos de entrenamiento, incluyendo el ruido y los detalles específicos de esos datos, lo que resulta en un rendimiento pobre en datos no vistos. Para prevenir el sobreajuste, se utilizan técnicas como la validación cruzada, la regularización, y la parada temprana. Técnicas que en este trabajo no dieron tiempo a implementarse.

4.2. Impacto

4.2.1. Personal

Ha sido una experiencia académicamente enriquecedora. Ya que cuando se empezó este trabajo, no se tenía prácticamente ningún conocimiento sobre la inteligencia artificial a un nivel que personalmente se considerase razonable. Se termina este trabajo habiendo disfrutado de profundizar en el área de la inteligencia artificial, más allá de lo estrictamente necesario para el desarrollo de este trabajo.

4.2.2. Social

Dados los resultados del modelo, el impacto de construir un sistema haciendo uso del modelo entrenado tendrían alguna serie de problemas con la detección de clases de dispositivos. Pero obviando esto; o clasificando manualmente los dispositivos presentes dentro de la vivienda. Podría tener un uso muy ventajoso de cara a un usuario. A cambio, podría presentar una serie de problemas de privacidad, si no se manejara de forma segura la seguridad del sistema [42].

4.2.3. Empresarial

Según Michael C. et. al. 2016 [43], la implementación de sistemas de monitorización de carga eléctrica no intrusiva en la industria podría tener un impacto positivo en el control de gasto energético, mejora de la eficiencia energética, ahorros energéticos para consumidores y una transición hacia el Internet de las Cosas. $\text{osti}_1373019$

4.2.4. Medioambiental

El potencial de un modelo como este puede beneficiar la implementación de sistemas de optimización de generación y consumo de energía, que podría potenciar el establecimiento de más comunidades solares en España. Además, establecer este tipo de sistemas da pie a la posibilidad de minar datos para poder entrenar modelos más generales que incorporen los datos de consumo de usuarios y otros, como datos metereológicos, precio de la electricidad, etc. Para poder así estimar y gestionar la oferta y demanda.

Capítulo 5

Conclusiones y trabajo futuro

5.1. Conclusiones

La monitorización de carga eléctrica no intrusiva es una tarea compleja, las ventajas que propone tienen el potencial de tener un impacto positivo en la sociedad, siempre que se tenga en consideración la seguridad del modelo y la importancia de la privaciddad de los datos. En este trabajo se ha explorado la posibilidad de aplicar transferencia de aprendizaje a un clasificador de imágenes ResNeXt optimizado por la arquitectura CSPNet. Los resultados obtenidos sugieren que el modelo CSPResNeXt es capaz de clasificar correctamente los dispositivos eléctricos presentes en una vivienda, con una precisión promedio de 0.45 y un puntaje F1 promedio de 0.48 . Sin embargo, se observan fluctuaciones significativas en la pérdida de validación, lo que sugiere la presencia de sobreajuste. Esto se refleja también en los picos observados en la pérdida de validación, que indican que el modelo tiene dificultades para generalizar en algunos puntos del entrenamiento.

5.2. Trabajo a Futuro

De cara a un trabajo de fin de máster o un doctorado, se exponen los siguientes objetivos a futuro:

- Implementar un modelo CSPResNeXt de menor tamaño.
- Obviar el uso de bases de datos, ya que solo complican el proceso.
- Implementar el algoritmo de codificación GAF para GPUs, haciendo uso de OpenCL.
- Añadir tests para prevenir el overfitting y cuantificarlo según las propuestas de James Schmidt et. al. 2023 [44].

5.3. Alegato Final

Esta tesis ha sido un reto personal, que ha requerido de un esfuerzo y dedicación considerable. A pesar de las dificultades encontradas, se ha logrado completar el trabajo con éxito. Se espera que los resultados obtenidos sean de utilidad para futuros trabajos en el área de la monitorización de carga eléctrica no intrusiva.

Bibliografía

[1] A. Faustine, N. H. Mvungi, S. F. Kaijage y M. Kisangiri, «A Survey on Non-Intrusive Load Monitoring Methodies and Techniques for Energy Disaggregation Problem», *CoRR*, vol. abs/1703.00785, 2017.

[2] *Shedding light on energy in Europe - 2024 edition - Eurostat*. DOI: 10.2785/88627. dirección: https://ec.europa.eu/eurostat/web/interactive-publications/energy-2024#about-publication.

[3] S. Brown y D. Jones. (2024). European Electricity Review 2024, Ember, dirección: https://ember-climate.org/insights/research/european-electricity-review-2024/.

[4] J. Revuelta Herrero, Á. Lozano Murciego, A. Barriuso, D. Hernández de la Iglesia, G. Villarrubia, J. Corchado Rodríguez y R. Carreira, «Non Intrusive Load Monitoring (NILM): A State of the Art», jun. de 2018, págs. 125-138, ISBN: 978-3-319-61577-6. DOI: 10.1007/978-3-319-61578-3_12.

[5] J. C. Romero Mora, *Anyone know a scientific reference about converting kwh electricity to CO2 emission?*, dic. de 2014. dirección: https://www.researchgate.net/post/Anyone-know-a-scientific-reference-about-converting-kwh-electricity-to-CO2-emission#:~:text=In%202013%2C%20it%20was%200%2C29%C2%A0Kt%20CO2/GWh.

[6] G. Hart, «Nonintrusive appliance load monitoring», *Proceedings of the IEEE*, vol. 80, n.º 12, págs. 1870-1891, 1992. DOI: 10.1109/5.192069.

[7] C. Nalmpantis y D. Vrakas, «Machine learning approaches for non-intrusive load monitoring: from qualitative to quantitative comparation», *Artificial Intelligence Review*, vol. 52, n.º 1, págs. 217-243, jun. de 2019, ISSN: 1573-7462. DOI: 10.1007/s10462-018-9613-7. dirección: https://doi.org/10.1007/s10462-018-9613-7.

[8] *El 36% de las casas en España tiene aire acondicionado*, Idealista News, Accessed: 2023-04-15, jul. de 2021. dirección: https://www.idealista.com/news/inmobiliario/vivienda/2021/07/15/791442-el-36-de-las-casas-en-espana-tiene-aire-acondicionado.

[9] A. Toosi, A. Bottino, B. Saboury, E. Siegel y A. Rahmim, «A Brief History of AI: How to Prevent Another Winter (A Critical Review)», *PET Clinics*, vol. 16, sep. de 2021. DOI: 10.1016/j.cpet.2021.07.001.

[10] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg y L. Fei-Fei, «ImageNet Large Scale Visual Recognition Challenge», *CoRR*, vol. abs/1409.0575, 2014.

[11] L. E. Baum y T. Petrie, «Statistical Inference for Probabilistic Functions of Finite State Markov Chains», *The Annals of Mathematical Statistics*, vol. 37, n.º 6, págs. 1554-1563, 1966. DOI: 10.1214/aoms/1177699147. dirección: https://doi.org/10.1214/aoms/1177699147.

[12] D. Jurafsky y J. H. Martin, *Speech and Language Processing*. 2023, cap. A. Hidden Markov Models, Copyright © 2023. All rights reserved.

[13] K. Basu, V. Debusschere, A. Douzal y B. Seddik, «Time series distance-based methods for non-intrusive load monitoring in residential buildings», *Energy and Buildings*, vol. 96, mar. de 2015. DOI: 10.1016/j.enbuild.2015.03.021.

[14] N. Miquey y E. Grover-Silva, «Non-Intrusive Load Monitoring of Single and Aggregated Profiles with a Hidden Markov Model», en *ENERGY 2021*, Valencia, Spain, mayo de 2021. dirección: https://minesparis-psl.hal.science/hal-03520223.

[15] J. Z. Kolter y T. Jaakkola, «Approximate Inference in Additive Factorial HMMs with Application to Energy Disaggregation», en *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, N. D. Lawrence y M. Girolami, eds., ép. Proceedings of Machine Learning Research, vol. 22, La Palma, Canary Islands: PMLR, abr. de 2012, págs. 1472-1482. dirección: https://proceedings.mlr.press/v22/zico12.html.

[16] O. Parson, S. Ghosh, M. Weal y A. Rogers, «An unsupervised training method for non-intrusive appliance load monitoring», *Artificial Intelligence*, vol. 217, págs. 1-19, 2014, ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2014.07.010. dirección: https://www.sciencedirect.com/science/article/pii/S0004370214001003.

[17] B. A. Olshausen y D. J. Field, «Sparse coding with an overcomplete basis set: A strategy employed by V1?», *Vision Research*, vol. 37, n.º 23, págs. 3311-3325, 1997, ISSN: 0042-6989. DOI: https://doi.org/10.1016/S0042-6989(97)00169-7. dirección: https://www.sciencedirect.com/science/article/pii/S0042698997001697.

[18] S. University. (). Sparse Coding. Accessed: 2024-04-20, dirección: http://ufldl.stanford.edu/tutorial/unsupervised/SparseCoding/.

[19] S. University. (). Autoencoding Coding. Accessed: 2024-04-20, dirección: https://ufldl.stanford.edu/tutorial/.

[20] M. Xue, S. Kappagoda y D. K. A. Mordecai, *Energy Disaggregation with Semi-supervised Sparse Coding*, 2020. arXiv: 2004.10529 [eess.SP].

[21] D. E. Rumelhart, G. E. Hinton y R. J. Williams, «Learning representations by back-propagating errors», *Nature*, vol. 323, págs. 533-536, 1986. dirección: https://api.semanticscholar.org/CorpusID:205001834.

[22] M. I. Jordan, «Chapter 25 - Serial Order: A Parallel Distributed Processing Approach», en *Neural-Network Models of Cognition*, ép. Advances in Psychology, J. W. Donahoe y V. Packard Dorsel, eds., vol. 121, North-Holland, 1986, págs. 471-495. DOI: https://doi.org/10.1016/S0166-4115(97)80111-2. dirección: https://www.sciencedirect.com/science/article/pii/S0166411597801112.

[23] J. L. Elman, «Finding structure in time», *Cognitive Science*, vol. 14, n.º 2, págs. 179-211, 1990, ISSN: 0364-0213. DOI: https://doi.org/10.1016/0364-0213(90)90002-E. dirección: https://www.sciencedirect.com/science/article/pii/036402139090002E.

[24] P. Huber, A. Calatroni, A. Rumsch y A. Paice, «Review on Deep Neural Networks Applied to Low-Frequency NILM», *Energies*, vol. 14, n.º 9, 2021, ISSN: 1996-1073. DOI: 10.3390/en14092390. dirección: https://www.mdpi.com/1996-1073/14/9/2390.

[25] K. Bao, K. Ibrahimov, M. Wagner y H. Schmeck, «Enhancing neural non-intrusive load monitoring with generative adversarial networks», *Energy Informatics*, vol. 1, oct. de 2018. DOI: 10.1186/s42162-018-0038-y.

[26] S. Singh y A. Majumdar, «Deep Sparse Coding for Non–Intrusive Load Monitoring», *IEEE Transactions on Smart Grid*, vol. 9, n.º 5, págs. 4669-4678, 2018. DOI: 10.1109/TSG.2017.2666220.

[27] A. Gao, J. Zheng, F. Mei, H. Sha, Y. Xie, K. Li e Y. Liu, «Non-intrusive multi-label load monitoring via transfer and contrastive learning architecture», *International Journal of Electrical Power Energy Systems*, vol. 154, pág. 109 443, 2023, ISSN: 0142-0615. DOI: https://doi.org/10.1016/j.ijepes.2023.109443. dirección: https://www.sciencedirect.com/science/article/pii/S0142061523005008.

[28] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen y J.-W. Hsieh, «CSPNet: A New Backbone that can Enhance Learning Capability of CNN», *CoRR*, vol. abs/1911.11929, 2019. arXiv: 1911.11929. dirección: http://arxiv.org/abs/1911.11929.

[29] S. Xie, R. Girshick, P. Dollár, Z. Tu y K. He, *Aggregated Residual Transformations for Deep Neural Networks*, 2017. arXiv: 1611.05431 [cs.CV].

[30] G. Huang, Z. Liu, L. van der Maaten y K. Q. Weinberger, *Densely Connected Convolutional Networks*, 2018. arXiv: 1608.06993 [cs.CV].

[31] R. Vargic, J. Londák y M. Medvecký, «An Approach to NILM using image-based features and transfer learning», en *2023 30th International Conference on Systems, Signals and Image Processing (IWSSIP)*, 2023, págs. 1-5. DOI: 10.1109/IWSSIP58668.2023.10180301.

[32] M. Xia, W. Liu, K. Wang, X. Zhang e Y. Xu, «Non-intrusive load disaggregation based on deep dilated residual network», *Electric Power Systems Research*, vol. 170, págs. 277-285, 2019, ISSN: 0378-7796. DOI: https://doi.org/10.1016/j.epsr.2019.01.034. dirección: https://www.sciencedirect.com/science/article/pii/S037877961930046X.

[33] M. DrIncecco, S. Squartini y M. Zhong, «Transfer Learning for Non-Intrusive Load Monitoring», *IEEE Transactions on Smart Grid*, vol. PP, págs. 1-1, ago. de 2019. DOI: 10.1109/TSG.2019.2938068.

[34] N. Batra, J. Kelly, O. Parson, H. Dutta, W. Knottenbelt, A. Rogers, A. Singh y M. Srivastava, «NILMTK: an open source toolkit for non-intrusive load monitoring», en *Proceedings of the 5th International Conference on Future Energy Systems*, ép. e-Energy '14, Cambridge, United Kingdom: Association for Computing Machinery, 2014, págs. 265-276, ISBN: 9781450328197. DOI: 10.1145/2602044.2602051. dirección: https://doi.org/10.1145/2602044.2602051.

[35] N. Batra, R. Kukunuri, A. Pandey, R. Malakar, R. Kumar, O. Krystalakos, M. Zhong, P. Meira y O. Parson, «Towards reproducible state-of-the-art energy disaggregation», en *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, ép. BuildSys '19, New York, NY, USA: Association for Computing Machinery, 2019, págs. 193-202, ISBN: 9781450370059. DOI: 10.1145/3360322.3360844. dirección: https://doi.org/10.1145/3360322.3360844.

[36] D. Murray, L. Stankovic y V. Stankovic, «An electrical load measurements dataset of United Kingdom households from a two-year longitudinal study», *Scientific Data*, vol. 4, n.º 1, pág. 160 122, 2017, ISSN: 2052-4463. DOI: 10.1038/sdata.2016.122. dirección: https://doi.org/10.1038/sdata.2016.122.

[37] C. Metinko, «Hugging Face Hits $4BValuationAfterSalesforceVentures-LedRound$», *Crunchbase News*, 2023. dirección: https://news.crunchbase.com/ai-robotics/hugging-face-unicorn-salesforce-ventures/.

[38] C.-Y. Wang, H.-Y. M. Liao, I.-H. Yeh, Y.-H. Wu, P.-Y. Chen y J.-W. Hsieh, *CSPNet: A New Backbone that can Enhance Learning Capability of CNN*, 2019. arXiv: 1911.11929 [cs.CV].

[39] A. Mao, M. Mohri e Y. Zhong, *Cross-Entropy Loss Functions: Theoretical Analysis and Applications*, 2023. arXiv: 2304.07288 [cs.LG].

[40] E. Mayhorn, J. Butzbaugh y A. Meier, «A Field Study of Nonintrusive Load Monitoring Devices and Implications for Load Disaggregation», *Sensors*, vol. 23, n.º 19, oct. de 2023, ISSN: 1424-8220. DOI: 10.3390/s23198253. dirección: https://www.osti.gov/biblio/2192971.

[41] E. Mayhorn, G. Sullivan, R. Butner, H. Hao y M. Baechler, «Characteristics and Performance of Existing Load Disaggregation Technologies», Pacific Northwest National Laboratory, Richland, WA, inf. téc., 2015.

[42] D. J. Sebastian Cardenas, M. Mukherjee y J. E. Ramirez, «A review of privacy in energy applications», oct. de 2023. DOI: 10.2172/2229911. dirección: https://www.osti.gov/biblio/2229911.

[43] M. C. Baechler y H. Hao, «Business Case for Nonintrusive Load Monitoring», mayo de 2016. DOI: 10.2172/1373019. dirección: https://www.osti.gov/biblio/1373019.

[44] J. Schmidt, *Testing for Overfitting*, 2023. arXiv: 2305.05792 [stat.ML].

Anexos

Apéndice A

Glosario

Lista de términos y abreviaciones utilizados en el trabajo. Su sentido es facilitar y familiarizar al lector la comprensión de las abreviaciones.

- NILM: Non Intrusive Load Monitoring - Monitorización de Carga No Intrusiva.
- HMM: Hidden Markov Models - Modelos Ocultos de Markov.
- Sparse Coding: Codificación de escasez.
- CSPNet: Cross Stage Partial Network
- PCA: Principal Component Analysis - Análisis de Componentes Principales

Apéndice B

Repositorios

B.1. Código Fuente

El código fuente de este trabajo se encuentra disponible en el siguiente repositorio de GitHub: `https://github.com/Hernies/tfg`

La redacción de esta memoria ha sido realizada utilizando la plantilla de LaTeX para el trabajo de fin de grado de la ETSIINF, puede encontrarse la fuente de este documento en `https://github.com/Hernies/memoria`

B.2. Repositorios externos

Para la implementación de el hashmap thread safe y el queue thread safe se han utilizado las implementaciones de:

- `https://github.com/kshk123/hashMap/tree/master`
- `https://github.com/K-Adam/SafeQueue`