UNIVERSIDAD POLITÉCNICA SALESIANA ECUADOR

POSGRADOS

MAESTRÍA EN

ELECTRÓNICA Y AUTOMATIZACIÓN

RPC-SO-30-No.507-2019

OPCIÓN DE TITULACIÓN:

PROYECTOS DE DESARROLLO

TEMA:

MODELO DE INTERACCIÓN DE REDES GENERATIVAS PARA LA DESAGREGACIÓN ENERGÉTICA EN EL HOGAR

AUTOR:

KEVIN SANTIAGO CHACA BENAVIDES

DIRECTOR:

DIEGO ROMÁN CABRERA MENDIETA

CUENCA - ECUADOR

2022

COHORTE 2019-2020

Autor:

Kevin Santiago Chaca Benavides
Ingeniero en Electrónica y Telecomunicaciones.
Candidato a Magíster en Electrónica y Automatización, Mención en Informática Industrial por la Universidad Politécnica Salesiana - Sede Cuenca.
kchacab@est.ups.edu.ec

Dirigido por:

Diego Román Cabrera Mendieta
Ingeniero Electrónico.
Master Universitario en Lógica, Computación e Inteligencia Artificial.
Doctor dentro del programa Ingeniería Informática.
dcabrera@ups.edu.ec

MODELO DE INTERACCIÓN DE REDES GENERATIVAS PARA LA DESAGREGACIÓN ENERGÉTICA EN EL HOGAR

Índice general

Índice de figuras

Índice de tablas

Resumen

La tarea de desagregación energética no intrusiva se ha convertido en un objetivo investigado ampliamente en los últimos años. La aplicación de diferentes metodologías y procedimientos innovadores han logrado buenos resultados al momento de clasificar los dispositivos conectados en la red y distinguir los diferentes estados en los que están trabajando. Sin embargo, el verdadero objetivo de esta tarea es obtener la señal de potencia consumida por los electrodomésticos conectados en la red del hogar. El presente trabajo propone una nueva metodología para la resolución de la tarea NILM a partir de la señal de potencia medida en la acometida principal de un hogar.

Dicha metodología se basa en la descomposición de la señal en un árbol de componentes con ayuda de la transformada de wavelet. En cada uno de los nodos de este árbol se entrena una WGAN-GP. Luego, con las predicciones de los modelos entrenados se procede a reconstruir la señal aplicando la transformada de wavelet inversa en cada nivel hasta llegar al nodo principal, en donde se representa la potencia consumida por el electrodoméstico. Además, se aplica un algoritmo de optimización para ponderar cada uno de los nodos en el proceso de reconstrucción de la señal.

Esta metodología se aplicó a la desagregación energética de cuatro dispositivos: refrigeradora, lavadora, secadora y microondas. Para el entrenamiento de los modelos se utilizó la base de datos REDD. Los resultados permiten obtener la señal de la potencia consumida por cada dispositivo. Finalmente, se implementó un sistema de desagregación energética en la nube utilizando los modelos resultantes de la metodología. Esta aplicación web esta operativa en la plataforma Google Cloud y se ha puesto a prueba su funcionamiento.

Abstract

In last years, Non-intrusive energy disaggregation has become widely researched. The application of different methodologies and novel architectures have achieved good results classifying the devices connected to the network and identifying their different working states. However, the main aim of NILM is to obtain the power signal consumed by the appliances connected in the home. The present work proposes a new methodology to achieve this objective, using only the power signal measured in the main supply of a home.

This methodology is based on the signal decomposition, by creating a tree of components using the wavelet transform. In each node of this tree a WGAN-GP is trained. The signal that represents the power consumed by the appliance is reconstructed by the inverse wavelet transform in the predictions of the trained models. This process is repeated in each level until reaching the main node. In addition, an optimization algorithm based on PSO is applied for searching the best weights for each node in the signal reconstruction process. This methodology was applied to four devices: refrigerator, washing machine, dryer and microwave. The REDD database was used to train the models. As result, the method obtains the power signal consumed by each device. Finally, an energy disaggregation system was implemented in the cloud using the models resulting from the methodology. This web application was implemented on the Google Cloud platform and its operation has been tested.

Capítulo 1

Introducción

La desagregación energética no instrusiva ha sido un reto que atrae la atencion de muchos investigadores en los últimos años. Sin embargo, la mayoría se han enfocado en métodos para la identificación de estado prendido/apagado y la clasificación de aparatos, dejando de lado el objetivo de obtener información confiable del consumo de energía de los aparatos de un hogar en tiempo real, información relevante para la administración o inyección de energía a la red eléctrica [Cominola et al., 2017].

El avance tecnológico en los últimos años ha permitido el desarrollo de herramientas cada vez más potentes y robustas en distintos campos de las maquinas de aprendizaje. Aprovechando está herramienta se propone utilizar un modelo novedoso para la resolución de la tarea NILM.

1.1 Descripción general del problema

Hasta el momento, los métodos utilizados para la extracción de esta información son computacionalmente costosos, por lo que su implementación no puede ser ejecutada en tiempo real, y la cantidad de información que se debe recopilar para el proceso es muy extensa [Ji et al., 2019]. Concretamente, los métodos de aprendizaje profundo con mejores resultados no son capaces de retornar una secuencia de la estimación de la energía desagregada de igual longitud que la serie temporal de la señal agregada; por ejemplo, el método de secuencia a subsecuencia ofrece como salida máximo la mitad de la longitud de la señal que requiere como entrada. De la misma manera trabaja el método secuencia a punto en donde como resultado se obtiene un solo dato de toda la amplia información que se requiere como entrada al método.

En [Liang et al., 2019] se demuestra que para obtener información de la desagregación de los aparatos en la red eléctrica se debe adquirir datos durante 30 minutos de la acometida principal del hogar.

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar un modelo de interacción de redes generativas para la desagregación energética de distintos tipos de dispositivos en el hogar a partir de una medición energética agregada en la acometida principal de la red.

1.2.2 Objetivos específicos

- Obtener una base de datos de señales agregadas y desagregadas con al menos cuatro dispositivos distintos dentro del hogar.
- Crear el modelo de interacción de redes generativas entrenadas a partir de versiones descompuestas y simplificadas de las señales agregadas y desagregadas.
- Optimizar el modelo a través de técnicas basadas en enjambre mediante la ponderación de los nodos del árbol de recomposición de la señal desagregada.
- Desplegar el modelo en un sistema en línea con conexión a la nube.

1.3 Contribuciones

Las contribuciones del presente trabajo son las siguientes:

1. El desarrollo de una nueva metodología a partir del entrenamiento de modelos generativos (GAN).

2. El desarrollo de un sistema online de procesamiento en la nube basado en la nueva metodología creada.

1.4 Organización del manuscrito

El presente documento se encuentra organizado de la siguiente manera:

El capítulo 2 se encuentra constituido por los antecedentes de la temática tratada en este proyecto como es la desagregación de energía usando métodos no intrusivos. Además, contiene la condimentación teórica necesaria para el desarrollo del modelo abordando los conceptos sobre maquinas de aprendizaje, redes generativas adversarias (GAN), transformada de wavelet y las redes generativas adversarias con penalidad por gradiente (WGAN-GP).

En el capítulo 3 se describe el flujo de los datos y los procedimientos seguidos para diseñar la nueva metodología. Se describe los componentes fundamentales del sistema y se detalla cada uno de los procedimientos desarrollados para la generación de la metodología.

En el capítulo 4 se describe el proceso del diseño e implementación del sistema en la nube, sus requerimientos y el funcionamiento.

El capítulo 5 contiene el análisis de los resultados obtenidos en cada una de las etapas de desarrollo.

Por último en el capítulo 6 se exponen las conclusiones y recomendaciones obtenidas una vez finalizado el proyecto, además de los trabajos futuros que se podrían desarrollar para mejorar los resultados.

Capítulo 2

Antecedentes y fundamentos teóricos

El capítulo se encuentra conformado por los antecedentes sobre los métodos de desagregación energética no intrusiva en hogares. Además, contiene los fundamentos teóricos necesarios para el manejo de la temática abordando los conceptos matemáticos sobre modelos de aprendizaje, GAN y WGAN-GP.

2.1 Antecedentes

La desagregación energética es el proceso de extraer el consumo de energía de diferentes dispositivos conectados a la misma red eléctrica a partir de la medición agregada, que puede ser adquirida en la acometida de la red [Hosseini et al., 2017]. Además, la desagregación de la energía permite al usuario extraer eventos de encendido y apagado así como patrones de comportamiento de la casa, utilizando un único punto de medición [Welikala et al., 2019].

La desagregación de energía fue propuesta en [Krystalakos et al., 2018] por primera vez como Monitoreo No Intrusivo del Consumo Eléctrico de Aparatos (NALM o NILM por sus siglas en inglés) por George Hart en la década de 1980. Su algoritmo original para NILM se basó en técnicas de optimización combinatoria. La idea principal era encontrar los estados óptimos de los aparatos monitoreados para que la suma del consumo de energía fuera la misma que la lectura del medidor.

Existen distintos tipos de cargas según los comportamientos de los electrodomésticos. Esto ha impulsado el desarrollo de nuevos algoritmos para el NILM con el objetivo de disminuir el error en la clasificación de electrodomésticos, como también disminuir la carga computacional y tiempo de procesamiento de los datos [Sun et al., 2019]. Le et al. [2016], por ejemplo, utiliza una red neuronal recurrente tipo Gated Recurrent Unit (GRU) consiguiendo una exactitud de 89-98 % en la clasificación de diferentes electrodomésticos. En [Kumar and Bhattacharjee, 2018] se demuestra que una red neuronal convolucional (CNN) con selección de hyperparámetros tiene una alta eficiencia en la clasificación de aparatos que tienen múltiples estados, tales como el aire acondicionado, la lavadora, la refrigeradora, el microondas, entre otros. [Gillis and Morsi, 2017], por otro lado, utiliza un algoritmo de árboles de decisión y de K-vecinos cercanos; demostrando que al aplicar filtros de Wavelet a las señales el rendimiento de la clasificación de aparatos mejora, así como también su robustez ante el ruido.

Algunos han optado por métodos de aprendizaje profundo. En [Singh and Majumdar, 2018] se compara el aprendizaje profundo con los métodos de aprendizaje tradicional para la tarea NILM, obteniendo mejores resultados con técnicas de aprendizaje profundo en la clasificación de aparatos como lavadora, secadora, microondas y refrigeradora. Así también [Çavdar and Faryad, 2019] propone un modelo que combina una red convolucional de una dimensión con una red recurrente (1D CNN-RNN); de esta manera logran mejorar en notablemente los resultados de clasificación de aparatos entre microondas, lavadora de platos y refrigerador. Otro algoritmo de

aprendizaje profundo utilizado para la tarea NILM es el Long short-term memory (LSTM); en [Tongta and Chooruang, 2020] se demuestra que con este modelo disminuye en 86 % el error cuadrático medio en comparación con el modelo de árboles de decisión al identificar la señal del refrigerador en la casa. Este mismo modelo es estudiado en [Nambiar et al., 2019]; sin embargo, concluyen que es un método lento, pues mientras más capas intermedias se agreguen tiene una mejor respuesta disminuyendo el error, pero aumenta el tiempo de respuesta. En [Osathanunkul and Osathanunkul, 2019] mejoran la respuesta de LSTM aplicando una red neuronal recurrente sobre un conjunto de datos con una baja tasa de muestreo de datos. Incluso lo proponen como opción para implementarlo en tiempo real. Otra mejora del algoritmo LSTM se detalla en [Kim et al., 2017], aquí se agrega una señal dedicada a distinguir el comportamiento de aparatos con múltiples estados como la lavadora obteniendo un sistema robusto.

Como se aprecia en los trabajos anteriores, una preocupación importante en este tema es el tiempo de respuesta prolongado de los modelos, mismo que se puede mejorar cuando se enfocan específicamente a un hogar y pocos electrodomésticos. En [Miyasawa et al., 2017] se utiliza el método de factorización de matriz no negativa ponderada, separando el procesamiento por turnos semi-supervisado con retroalimentación; el resultado de la tarea NILM en el calefactor, ventilador y aire acondicionado es computacionalmente eficiente, simple de implementar y robusto. Otra idea se muestra en [Meziane et al., 2017], donde se usa un algoritmo que detecta variaciones de la envoltura de la señal. Como resultados la simulación muestra un 100 % de precisión en clasificación y relación señal-ruido de 50dB. Con el mismo objetivo de reducir tiempos de procesamiento, [Sirojan et al., 2018] propone un Autocodificador Convolucional Variacional para identificar el estado de prendido/apagado de aparatos como la lavadora, el microondas, el lavaplatos y la refrigeradora, consiguiendo buenos resultados al aplicar a datos de baja frecuencia lo que le hace más económico y rápido. Con una tasa de muestreo de 1/60 Hz también funciona el método BPNN explicado en [Tian et al., 2017], donde obtienen alta precisión en clasificación de aparatos en comparación con métodos tradicionales. A pesar que los trabajos anteriores presentan resultados con poco error en la tarea NILM, se enfocan solamente en la clasificación e identificación del aparato conectado y no en su consumo energético.

Por otro lado, en [Pan et al., 2020] proponen una red neuronal generativa para la estimación de la energía consumida en el refrigerador, el microondas y lavaplatos. Para ello mapean una secuencia de la medición de la acometida a una sub-secuencia (generalmente la mitad de la secuencia) del aparato en

cuestión, de esta manera consiguen una buena exactitud y bajo error absoluto medio en la identificación del consumo energético de los aparatos. En [Zhang et al., 2016] mejoran este concepto aplicando el algoritmo sequence-to-point donde la entrada es una ventana de la señal y la salida es la información de una única medición de un electrodoméstico, con esto se reduce el error en 83 %. Los dos métodos anteriores estiman el consumo energético del aparato conectado; sin embargo, no son computacionalmente eficientes debido a que requieren procesar grandes cantidades de información para devolver una porción del tamaño de la secuencia en el caso de [Pan et al., 2020] y un solo punto en [Zhang et al., 2016].

2.2 Fundamento teórico

2.2.1 Red Generativa Adversaria

La Red Generativa Adversaria GAN es un tipo de red compuesta por dos partes; por un lado, tiene un modelo generativo y por otro lado, un modelo discriminador. Ambos modelos son antagónicos, por lo que están enfrentándose entre sí. En algunos documentos explicativos ([Goodfellow et al., 2014], [Goodfellow, 2017], [Hitawala, 2018], [You et al., 2022]) se describe el funcionamiento representando al modelo generativo como un falsificador de billetes y al modelo discriminador como el policía. El objetivo en esta representación es que el falsificador intenta engañar al policía mejorando su técnica para crear billetes más parecidos a los verdaderos, al mismo tiempo que el policía se entrena para distinguir con mayor precisión los billetes falsos.

El funcionamiento empieza en el modelo generativo, el cual a partir de una señal aleatoria o ruido blanco crea una muestra que es enviada al modelo discriminador junto con muestras reales. En este paso, el discriminador intenta reconocer qué muestra ha sido generada y qué muestra es real y, en base a su predicción, se entrena el discriminador para mejorar su capacidad de distinguir las muestras. Además, se entrena el generador para mejorar su capacidad de engañar al discriminador usando como información los aciertos que tubo el discriminador. Este proceso de competencia entre los dos modelos se repite varias veces hasta que el generador sea capaz de crear muestras lo suficiente buenas o parecidas a las reales.

En [Goodfellow et al., 2014] se propone la GAN original, aquí estipula que su aplicación es mas simple cuando ambos modelos (generador y discriminador) son perceptrones multicapa. La entrada del modelo generador G es z, del cual se conoce que tiene una distribución P_z, obteniendo como

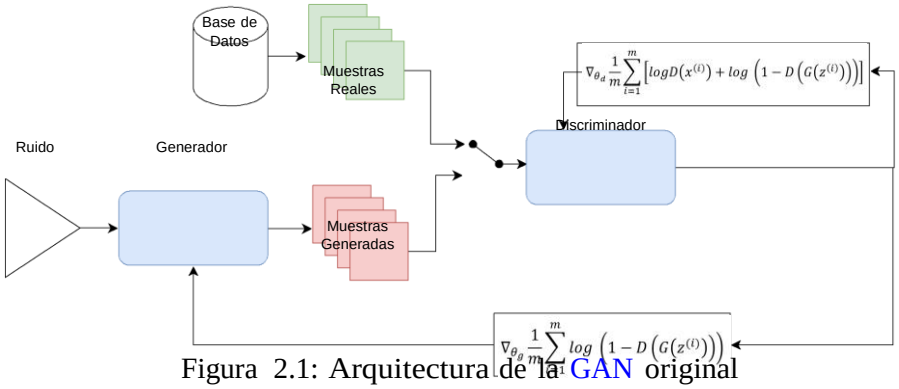

Figura 2.1: Arquitectura de la GAN original

resultado $x = G(z)$ con la distribución P_g. Por otro lado, la respuesta del modelo discriminador D es un valor escalar $D(x)$ que representa la probabilidad de que x provenga de la distribución de datos reales P_{data} en lugar de P_g. El proceso de entrenamiento se describe como el juego mínimo-máximo ([Zhang and Zhang, 2003]) descrito con la siguiente formula:

$$\min_G \max_D V(D, G) = E_{x \sim p_{data}(x)}[\log D(x)] + E_{z \sim p_z(z)}[\log D(1 - D(G(z)))] \tag{2.1}$$

Este juego de máximo-mínimo continua desarrollándose en el algoritmo hasta que se consiga el equilibrio de Nash ([Holt and Roth, 2004]). En la Figura 2.1 se muestra la arquitectura de la GAN. El generador y el discriminador son entrenados en función a los gradientes estocásticos de retroalimentación en busca de los modelos óptimos. La propuesta de esta arquitectura es innovadora; sin embargo, tiene algunos problemas en el entrenamiento.

El principal problema que se puede mencionar es que un modelo muy bueno de discriminador va a producir un error en salida muy bajo, lo que significara que el gradiente de retroalimentación para el entrenamiento del modelo generador es muy pequeño y por lo tanto no va a converger el entrenamiento.

Además, el entrenamiento del modelo GAN puede caer en el problema de colapso de modos. Esto se refiere a cuando el generador ha encontrado especializarse en un conjunto limitado de muestras o modos; al validar estas

muestras generadas con el discriminador se obtienen buenos resultados y su entrenamiento parece ser exitoso; sin embargo, no es capaz de generar toda la distribución de datos objetivo del modelo ([Bau et al., 2019]).

2.2.2 Wasserstein GAN

Una mejora propuesta para resolver los problemas en el entrenamiento de la GAN es el uso de la distancia de Wasserstein detallada en [Arjovsky et al., 2017]. El problema se genera por la métrica que se utiliza en la GAN original. Ésta es la combinación de dos métricas de distancia entre la distribución de datos generados P_g y la distribución de datos reales P_r. Estas métricas son la divergencia Kullback-Leibler (KL) y la divergencia Jensen-Shannon (JS) que se muestran a continuación:

$$KL\ (P_r\ ||\ P_g) = \int \log \frac{P_r(x)}{P_g(x)}\ P_r(x)du(x) \tag{2.2}$$

$$JS\ (P_r, P_g) = KL\ (P_r\ ||\ P_m)\ + KL\ (P_g\ ||\ P_m) \tag{2.3}$$

$$P_m\ = \frac{P_r + P_g}{2} \tag{2.4}$$

Esta métrica no cumple con la propiedad de simetría que establece que la distancia desde A hacia B es la misma que desde B hacia A ([Demey et al., 2011]). Como solución a este problema, se propone el uso de la distancia Wassertstein:

$$W\ (P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} E_{(x,y)\sim\gamma}\ [\|x - y\|] \tag{2.5}$$

Donde $\Pi\,(P_r, P_g)$ denota el grupo de todas las distribuciones conjuntas $\gamma(x, y)$ con sus marginales respectivas de P_r y P_g. Al aplicar la dualidad de Kantorovich-Rubinstein resuelta en [Edwards, 2011] para la distancia W se obtiene una expresión dependiente de la función 1-Lipschitz f:

$$W\ (P_r, P_\theta) = \sup_{\|f\|_L \leq 1} E_{(x,y)\sim P_r}\ [f(x)] - E_{(x,y)\sim P_\theta}[f(x)] \tag{2.6}$$

Encontrar una función que cumpla la propiedad de que su derivada sea menor a 1 en todos sus puntos (función 1-Lipschitz [Cobzaş et al., 2019])

es un ejercicio difícil de solucionar, así que se propone cambiar la función 1-Lipschitz por una función K-Lipschitz donde K es una constante. De esta manera, el problema de optimización del modelo generador se convierte en solucionar el problema de maximización:

$$\max_{w \in W} E_{x \sim P_r}[f_w(x)] - E_{z \sim p(z)}[f_w(g\theta(x))] \tag{2.7}$$

Donde f_w es la familia de funciones K-Lipschitz y se le nombra como el optimizador crítico ya que deja de cumplir la función de clasificar; en cambio, ahora produce un porcentual escalar y no una probabilidad. La ventaja en comparación a la GAN es que en la WGAN una buena optimización del optimizador crítico ayuda a que el generador encuentre su modelo óptimo; es decir, se logra una mayor estabilidad en el proceso de entrenamiento.

2.2.3 WGAN Gradient Penalty

El algoritmo de entrenamiento de la WGAN propone un recorte de pesos con un parámetro c al momento de optimizar el discriminador crítico; esto lo hacen para compensar la distancia KW generada al utilizar funciones K-Lipschitz en lugar de la 1-Lipschitz establecida como solución matemática a la distancia de Wasserstein, métrica de distancia entre las distribuciones generadas $q(x')$ y realesp(x).

$$KW(P_r, P_r) = KW(p(x), q(x')) \tag{2.8}$$

$$KW = \max_{w \in W} E_{x \sim P_r}[f_w(x)] - E_{z \sim p(z)}[f_w(g\theta(x))] \tag{2.9}$$

A pesar de que funciona, el acotamiento para en ésta metodología no es la forma óptima para compensar la distancia KW . Tomando en cuenta que una función diferenciable f es 1-Lipschitz si y sólo si tiene gradientes con norma 1 en todas partes como maximo superior; entonces Los puntos interpolados entre los datos reales y los generados deben tener una norma de gradiente de 1 para f. [Gulrajani et al., 2017] plantea el uso de la penalidad de gradiente en vez de este recorte:

$$L = E[D(e \quad x)] - E[D(x))] + \lambda E\left[\left(\|\nabla_b D(b \quad x)\|_2 - 1\right)^2\right] \tag{2.10}$$

e ____ $x \sim P$ ____ $e \sim P_x$ ____ ____

$x \sim P_g$ $\{z \quad r\}$ $x \quad b$ $\{z \quad\}$

Función de perdida original | Penalidad de gradiente

Donde b x y x con t uniformemente muestreado entre 0 y 1.

$$\hat{x} = t\tilde{x} + (1 - t)x \text{ con } 0 \leq t \leq 1 \tag{2.11}$$

Así, en lugar de aplicar el recorte, WGAN-GP penaliza el modelo si la norma de gradiente se aleja de su valor objetivo de norma 1. Los resultados muestran que la arquitectura WGAN-GP tiene mejoras en la estabilidad durante el entrenamiento en comparación con la GAN, por lo tanto, también es mejor que la GAN original.

2.2.4 Transformadas Wavelet

La transformada de wavelet descompone una señal en un conjunto de componentes, separando las altas frecuencias de las bajas frecuencias ([Mallat, 1989]). A diferencia de la transformada de Fourier, la transformada de wavelet utiliza una serie de funciones de duración finita, conocidas como wavelets madres, que permiten extraer información tanto en el tiempo como en la frecuencia ajustando su resolución, [Ngui et al., 2013] recomienda varios criterios al momento de seleccionar la wavelet madre. Se debe tener en cuenta que para eventos de alta frecuencia se tiene una mejor resolución en tiempo y para componentes en baja frecuencia una mejor resolución en frecuencia.

Debido al principio de incertidumbre de Heisenberg([Busch et al., 2007]), al cruzar cierto umbral no se puede obtener una mejora en la resolución del tiempo sin empeorar la resolución de la frecuencia y viceversa.

[Burrus et al., 1998] describe a la wavelet como una "pequeña onda", que tiene su energía concentrada en tiempo para ofrecer una herramienta de análisis de fenómenos transitorios, no estacionarios o variables en el tiempo. Esto es justamente el fenómeno que se analiza en la problemática NILM, es por esta razón que se utiliza esta transformada en la metodología propuesta en la siguiente sección.

Capítulo 3

Metodología

En este capítulo se describe con detalle los procedimientos desarrollados en cada etapa para resolver la tarea NILM propuesta. En primer lugar, se describen los componentes fundamentales necesarios para la construcción de la arquitectura del modelo trabajado. En segundo lugar, se especifica las etapas que componen la arquitectura del modelo así como el efecto que dicha etapa tiene sobre los datos procesados. Finalmente, muestra la implementación de la arquitectura para obtener los resultados propuestos en la desagregación de la señal de potencia de cada aparato.

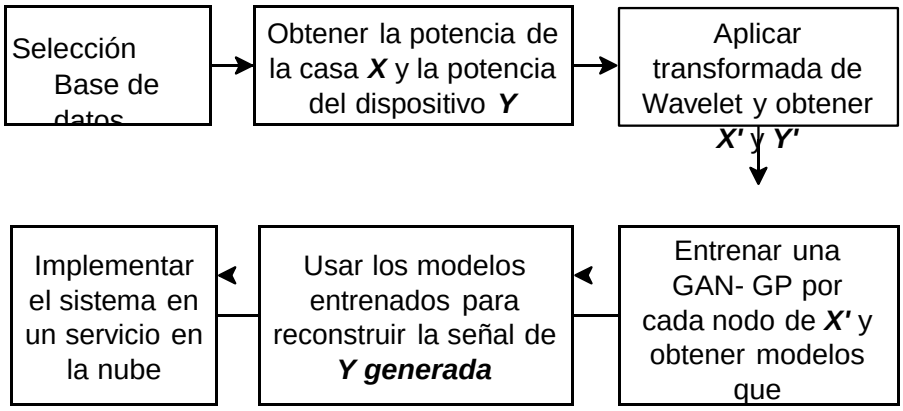

Figura 3.1: Esquema de la metodología propuesta.

3.1 Diseño de la metodología

En la literatura se muestra que al momento los métodos más utilizados para la problemática NILM se basan en algoritmos de modelos ocultos de Markov, procesamiento de señales basado en grafos y modelos de aprendizaje profundo. El último método es el que se utiliza con la propuesta de una arquitectura que combina las características en tiempo y frecuencia que nos brinda la transformada de wavelet.

La metodología propuesta se muestra en la Figura 3.1. En síntesis se compone por las siguientes etapas:

1. Selección de datos: De entre las múltiples bases de datos disponibles para resolver la tarea NILM en hogares, se escoge una que cumpla con los requisitos necesarios para le arquitectura propuesta.

2. Preparar la base de datos: A través de un preprocesamiento de la base de datos, se filtra y selecciona las muestras válidas de la señal de potencia agregada consumida en la casa (X) y la señal de potencia del dispositivo (Y), estos datos serán utilizados en las siguientes etapas.

3. Descomposición de la señal: se aplica la transformada wavelet packet en ambas señales (X y Y) N veces, obteniendo nodos X' y Y' en cada nivel.

4. Modelamiento de cada red: Cada nodo X' creado en la etapa anterior tiene una GAN, en esta etapa se modela todas las GAN con el fin de

obtener el valor de Y' estimada $\hat{Y}'$ lo más aproximada posible al valor Y'.

5. Reconstrucción de la señal desagregada: en esta etapa se usa la predicción de los modelos WGAN-GP entrenados en conjunto con la transformada inversa de wavelet para reconstruir la señal de potencia del dispositivo Y desde el nivel inferior.

6. Optimizar los resultados: con el objetivo de minimizar el error en la reconstrucción de las señales, se aplica un algoritmo de optimización por enjambres encargado de escoger valores ponderados para cada resultado de los modelos entrenados.

7. Implementación del servicio web: cargar el algoritmo de la solución como un servicio en la nube disponible en un servidor publico.

A continuación, se explica cada uno de los pasos a seguir en la metodología:

3.2 Selección de la base de datos

En la actualidad existen varias opciones de bases de datos recolectadas con el objetivo de implementar soluciones a la problemática NILM en los dispositivos de los hogares. Algunas de las mas conocidas y utilizadas en la investigación durante varios años atrás. Entre estas bases de datos se pueden citar REDD, BLUED, PLAID, UK-DALE, COOLL, BLOND, LIT y muchas otras más. A continuación se detalla breves características de algunas de estas bases de datos.

3.2.1 LIT-Dataset

Una base de datos propuesta en el año 2020, el Laboratorio de Innovación y Tecnología en Sistemas Embebidos (LIT) trabaja en su propio sistema NILM. El conjunto de datos ha tenido un preprocesamiento por parte de la empresa basados en las características y requisitos de las formas de ondas de tensión y corriente, Además reúne información de cargas individuales y múltiples en diversas condiciones de funcionamiento ([Renaux et al., 2018]).

Una característica distintiva del conjunto de datos LIT es que se compone de tres clases distintas de monitorización de cargas:

- Sintética: las formas de onda han sido muestreadas con un dispositivo que controla con precisión los tiempos de encendido y apagado en

un escenario de conformación controlada de la carga. El dispositivo controla cada carga individual, hasta ocho, mediante TRIACs y relés, lo que permite tiempos de encendido en ángulos específicos de la onda sinusoidal de la red. Los tiempos de encendido y apagado se registran en la forma de onda con una resolución mejor que 5 ms, lo que permite identificar el semiciclo de la red en el que se produjo el evento de carga "encendido" o "apagado". Las formas de onda suelen tener una duración de 30 segundos a velocidades de muestreo superiores a 15 KHz, alcanzando 256 muestras por ciclo.

- Simulación: el laboratorio ha simulado varias clases de cargas en un entorno MATLAB/SIMULINK y se han validado con cargas reales. El uso de cargas simuladas permite modificar las características de la carga para crear escenarios que serían muy difíciles/complejos de crear en el mundo real.

- Natural: las formas de onda se toman en un entorno real (residencial, de investigación, comercial, industrial) durante períodos de tiempo más largos. Los sensores están conectados a cada carga para detectar y registrar con precisión los tiempos de encendido y apagado de cada carga, mientras que las formas de onda registran la corriente y el voltaje agregados de todas las cargas monitorizadas.

3.2.2 REDD-Dataset

Una base de datos propuesta en el año 2011, el objetivo principal de esta base de datos denominado Conjunto de datos de desagregación de energía de referencia (REDD) fue que sea publico, pues en ese entonces muy pocas bases de datos habían disponibles para la investigación. Los datos están específicamente orientados a la tarea de desagregación de energía: determinar los dispositivos componentes a partir de una señal de eléctrica principal.

REDD consiste en el consumo de electricidad de toda la casa y de un circuito/dispositivo específico para varias casas reales durante varios meses. Para cada casa monitorizada, se ha registrado toda la señal de electricidad de la casa (monitores de corriente en ambas fases de potencia y un monitor de voltaje en una fase) muestreada en alta frecuencia (15kHz); y también se registra hasta 24 circuitos individuales en el hogar, cada uno etiquetado con su categoría de aparato o aparatos, muestreados a 0,5 Hz; finalmente almacena hasta 20 monitores de nivel de enchufe en el hogar, registrados a 1 Hz ([Kolter and Johnson, 2011]),

La base de datos se compone de 10 viviendas monitoreadas, con un total de 119 días de datos (combinados en todos los hogares), 268 monitores únicos y más de 1 terabyte de datos sin procesar. Considerada como el mayor conjunto de datos disponible públicamente para su desagregación con las cargas reales de cada dispositivo en el momento que fue creado.

La organización de la información dentro de la base de datos es simple, existen dos carpetas principales que tienen la información, la primera es "low_freq" y la segunda es "high_freq". Dentro de cada una de las carpetas existen carpetas que distinguen cada casa, por ejemplo la carpeta de baja frecuencia tiene dentro 10 carpetas cada una con la información de cada vivienda. Dentro de la carpeta de cada una de las casas se un archivo "labels.dat", en este archivo se detalla un directorio de cada canal de información, por ejemplo detalla que el primer canal representa la fase 1 de la potencia principal, el canal 2 la segunda fase de la potencia principal, el canal 3 la potencia del refrigerador, etc; además de este archivo, dentro de la carpeta de cada casas esta las muestras tomadas de cada canal en archivos con extinción ".dat", cada canal almacena la potencia muestreada con el tiempo en el que fue tomada la muestra en formato UTC timestamps.

3.2.3 COOLL

COOLL (Controlled On/Off Loads Library por sus siglas en inglés) es un conjunto de datos de mediciones eléctricas a nivel de enchufe. Contiene mediciones de corriente y tensión de encendido de 42 aparatos muestreados a una frecuencia de 100 kHz (840 mediciones de corriente y 840 mediciones de tensión).

Los aparatos del conjunto de datos del COOLL se miden individualmente, de uno en uno. El conjunto de datos no contiene un escenario en el que se midan varios aparatos simultáneamente. Además, los aparatos seleccionados se eligen de forma que sea posible controlar el instante de encendido, es decir pulsando previamente el botón de encendido, el aparato puede funcionar electrónicamente.

Cada medición dura 6 segundos con una duración de pre-disparo de 0,5 segundos (el pre-disparo de las primeras mediciones es diferente e igual a 1 segundo) y una duración post-disparo de 1 segundo. Estas duraciones corresponden, respectivamente, al tiempo en que el aparato está apagado antes el encendido y después del apagado.

Para cada aparato, se realizan 20 mediciones controladas. Cada medición corresponde a un retardo de acción específico que va de 0 a 19 ms con un paso de 1 ms (de este modo, se cubre toda la duración del ciclo de tiempo de la

tensión de red de 50 Hz, es decir, 20 ms). Este retardo de acción corresponde al tiempo con el que se retrasa la acción de encendido con respecto al inicio de un ciclo de tiempo específico de la tensión de red 1 . El retardo de la acción de apagado se fija en 0 ms para todas las mediciones ([Picon et al., 2016]).

3.2.4 UK-DALE

Creada en el año 2015 con el objetivo de investigar los algoritmos de desagregación, UK-DALE es un conjunto de datos de libre acceso del Reino Unido que registra la electricidad a nivel de aparatos domésticos a una frecuencia de muestreo de 16 kHz para toda la casa y a 1/6 Hz para los aparatos individuales.

Se trata del primer conjunto de datos del Reino Unido de libre acceso con esta resolución temporal. Registra cinco casas, una de las cuales fue registrada durante 655 días, la mayor duración que se conocía al momento para cualquier conjunto de datos de energía con esta tasa de muestreo ([Kelly and Knottenbelt, 2015]).

Cada seis segundos se muestrea la potencia activa consumida por los aparatos individuales y la demanda de potencia aparente de toda la casa. Además, en tres casas, muestrea el voltaje y la corriente de toda la casa a 44,1 kHz (muestreados a 16 kHz para su almacenamiento) y también calcula la potencia activa, la potencia aparente y el voltaje RMS a 1 Hz. En la casa 1, almacena durante 655 días y registra individualmente casi todos los electrodomésticos de la casa, lo que dio lugar a una grabación de 54 canales distintos (aunque se registraron menos canales al principio del conjunto de datos).

En las otras cuatro casas se almaceno durante varios meses; en cada una de ellas se registraron entre 5 y 26 canales de datos de aparatos individuales. esta información esta destinado a los investigadores que trabajan en:

- Modelación de la red eléctrica
- Exploración del potencial de la respuesta automática a la demanda
- Comportamiento del uso de los aparatos

3.2.5 WHITED

Es un conjunto de datos de mediciones de de aparatos en varios lugares. Los aparatos se registraron con un medidor de tarjeta de sonido de bajo coste.

Tabla 3.1: Comparación de bases de datos.

Dataset	REDD	COOLL	UK-DALE	WHITED	LIT
Año	2011	2016	2017	2016	2020
Residencial	Si	Si	Si	Si	Si
Comercial	No	No	No	Si	Si
Industrial	No	No	No	Si	Si
Muestreo Alta Frecuencia	15kHz	100kHz	16Khz	44.1kHz	100kHz
Muestreo Baja Frecuencia	0.5Hz	N.A.	1-6Hz	N.A.	1Hz
Duración	Meses(6 casas)	840 señales de 6s	Mas de un año(5 casas)	5123 señales de 5s	1664 señales de 40s
Dispositivos	N.A.	42	N.A.	109	54

El registro de La grabación se realizó principalmente en hogares y pequeñas industrias en diferentes regiones del mundo. Por lo tanto, puede puede ser posible extraer las características de la red específicas de cada región a partir de las formas de onda de la tensión en los datos. Para cubrir todos los transitorios correspondientes, registra los primeros 5 segundos de los 110 aparatos diferentes. El objetivo de este conjunto de datos es proporcionar un amplio espectro de diferentes tipos de aparatos en regiones de todo el mundo.

Las señales se registraron con una resolución temporal de 44,1 kHz y 16 bits de resolución. Para poder realizar múltiples mediciones en diferentes lugares, se usó 3 kits de medición idénticos con sus respectivos factores de calibración ligeramente diferentes.

La base de datos comprende 1.100 registros diferentes para 110 aparatos diferentes que pueden agruparse en 47 tipos diferentes (clases) en 6 regiones distintas. Para la mayoría de los de los electrodomésticos, tiene una foto de su etiqueta de especificaciones eléctricas. Estas imágenes se encuentran en la subcarpeta de imágenes y etiquetas de tipo([Kahl et al., 2016])

La Tabla 3.1 resume las características mas importantes de cada base de datos analizadas; para éste trabajo se ha seleccionado la base de datos REDD, entre las ventajas que presenta en comparación con las demás bases de datos se destaca la diversidad de información bien detallada y su buena organización en sus carpetas. El hecho de que la información no sea preprocesada permite trabajar en un sistema que puede ser usado en una aplicación real; además, la base de datos recopila sus datos con diferentes frecuencias de muestreo que se adaptan a las necesidades del desarrollo de la metodología propuesta.

3.3 Preparación de los datos

La base de datos seleccionada dispone de datos tomados a alta frecuencia, a media frecuencia y a baja frecuencia. La frecuencia alta digitaliza señales de voltaje y corriente a tiempo real, la frecuencia media también son señales de corriente y voltaje agrupadas mientras que las señales de baja frecuencia son señales de potencias, potencias medias de las dos líneas de alimentación principal de la casa y la potencia consumida por cada dispositivo electrónico etiquetado en dicha casa. La base de datos completa almacena los datos tomados a cada frecuencia en carpetas diferentes, para la aplicación del presente trabajo se utiliza únicamente la carpeta con la información de datos tomados a baja frecuencia.

La Figura 3.2 representa la organización de los datos dentro de esta carpeta; los datos son recolectados de 6 casas diferentes, la información de cada casa esta almacenada en una carpeta que contiene el numero de la casa como por ejemplo “house_1”. Dentro de la carpeta de cada una de la casa se tiene almacenado la información recolectada en cada canal almacenado en un archivo con el nombre del canal, por ejemplo “channel_4.dat”; además en esta misma carpeta existe un archivo llamado “labels.dat” que almacena las etiquetas de cada canal.

El archivo “labels” identifica cada uno de los canales con sus etiquetas de potencia medida, típicamente el canal 1 almacena la potencia principal media de la linea 1 y el canal de la linea 2, los demás canales tienen las muestras de potencia de un electrodoméstico como la refrigeradora, la secadora, el microondas, entre otros. El archivo de cada canal “channel_n” tiene el valor real de la potencia muestreada asociado al momento en el que fue tomada la muestra en formato tiempo UTC.

3.3.1 Recolección de datos válidos

El objetivo de este proceso es realizar un preprocesamiento de las muestras para crear los conjuntos de datos que se utilizarán en los siguientes pasos. Los datos son muestreados a baja frecuencia, sin embargo, un previo análisis demuestra que en algunas muestras la frecuencia de 1Hz no se mantiene, esta varianza puede causar incertidumbres en los resultados. Como solución a esta problemática se plantea un algoritmo para filtrar los datos y seleccionar conjuntos de datos validados.

En la Figura 3.3 se ilustra el diagrama de flujo del algoritmo utilizado para dicha validación. A continuación se detalla cada paso de este algoritmo:

- Primero se cargan los datos de un canal(por ejemplo, la potencia del

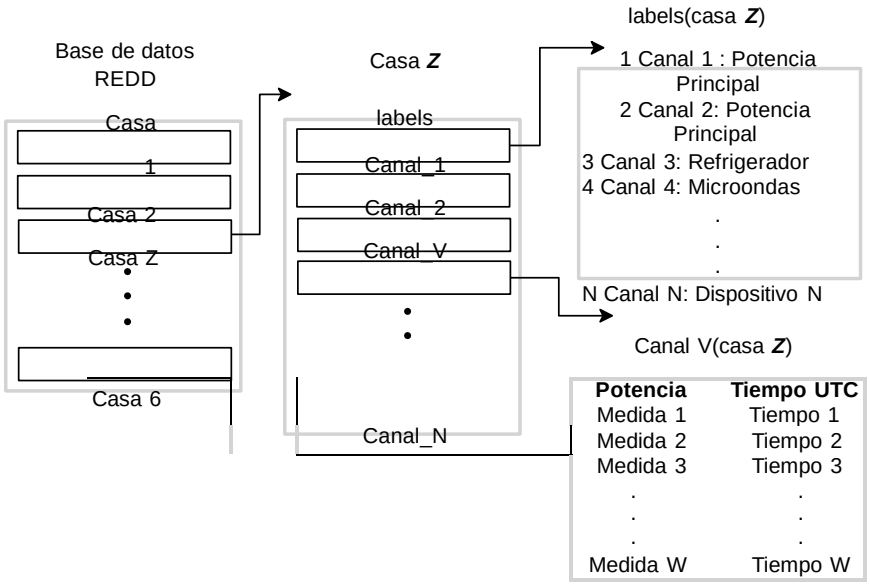

Figura 3.2: Arquitectura de la base de datos REDD.

refrigerador); para esto se debe navegar dentro de la carpeta de baja frecuencia, dentro de la carpeta de una de las casas.

- Segundo se inicia un bucle que se repetirá la cantidad de conjuntos de datos que se necesitan para el procesamiento de la arquitectura, a cada conjunto de datos se le conoce como batch. En este trabajo se utilizaron 128 batchs.
- Tercero se inicia un bucle que se encarga de completar cada batch, en este trabajo se almacenaron 1024 muestras en cada batch.
- Cuarto se hace una comparación de los datos muestreados para asegurarse que han sido tomados exactamente a la frecuencia de 1Hz; en el caso de que no cumpla con este requisito, el dato es despreciado y no se incluye en el batch. Este proceso se repite hasta que se complete de datos el batch y continua con el siguiente batch.

Finalmente se tiene la cantidad de batchs deseados con datos validados. Este mismo algoritmo se repite para cada uno de los canales que se necesitan; es decir, para el canal de la potencia principal media de la linea 1, el canal de la potencia principal media de la linea 2, el canal del refrigerador, el canal del microondas, el canal de la lavadora y el canal de la secadora. Estos conjuntos de datos se almacenan en archivos de datos para su futuro uso.

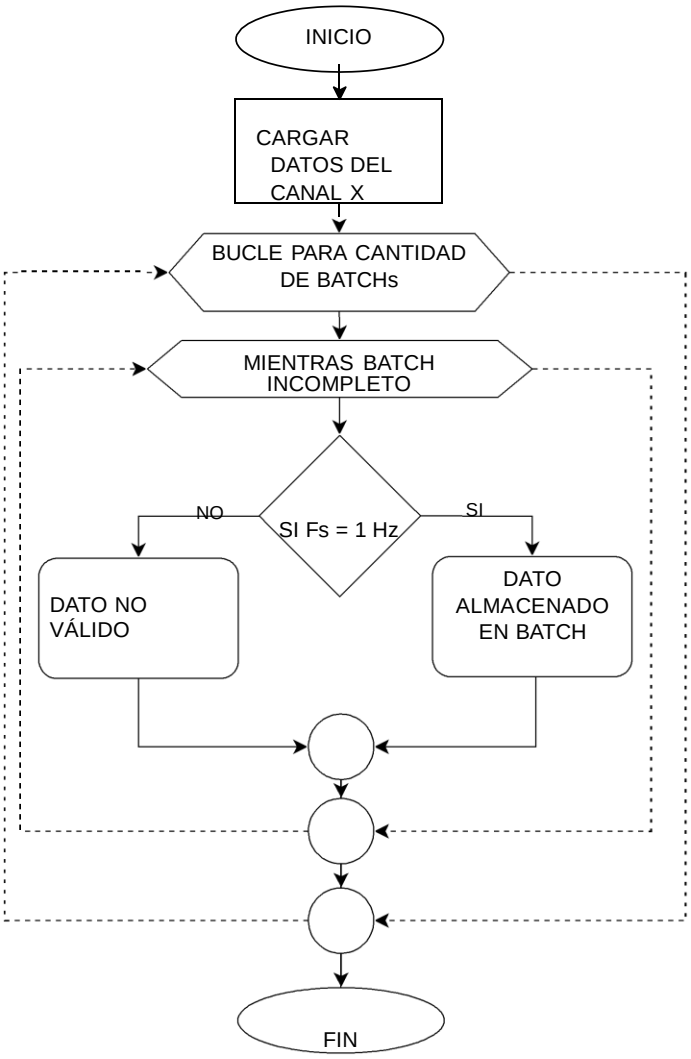

Figura 3.3: Validación de datos.

La tabla 3.2 resumen las variables que describen el uso de la base de datos. La cantidad de Batch empleados para el proceso de entrenamiento y pruebas del modelo, el tamaño de cada Batch, entre otros.

Tabla 3.2: Descripción del uso de la base de datos.

Variable	Base de Datos	Utilizados	Unidad
Cantidad de Casas	6	1	N.A.
Dispositivos	18	4	N.A.
Potencias de Entrada	2	2	[Watts]
Duración de muestreo	180	180	[días]
Frecuencia de muestreo	0.5-15000	1	[Hz]
Tamaño de Batch	N.A.	1024	N.A.
Cantidad de Batch	N.A.	1000	N.A.

3.3.2 Separación de datos de entrenamiento, validación y prueba

Los batch validados anteriormente se separan y almacenan en dos archivos diferentes, el primer archivo contiene los batchs que se utilizarán para el proceso de entrenamiento del modelo mientras que el segundo archivo contiene los batchs que se utilizaran para probar el modelo. No puede probarse el funcionamiento del sistema con los mismos datos con los que se entrenó pues no se estimaría un resultado real del funcionamiento. El archivo de datos destinado a probar el modelo se subdivide en dos partes, un 66 % para la creación y retroalimentación de los modelos en cada nodo y un 33 % para la validación de cada uno de esos modelos una vez que están entrenados; además estos datos de validación se utilizan en el proceso de optimización de reconstrucción de la señal como se muestra en la Figura 3.4.

3.4 Arquitectura del modelo

Con los datos listos, se empieza por definir la arquitectura que se utilizará para resolver la problemática NILM. En síntesis, el objetivo es crear un modelo que en el futuro sea capaz de diferir por sí solo la potencia del dispositivo conociendo únicamente la potencia principal consumida en el hogar. Para lograr este objetivo se utilizará como datos de entrada de la arquitectura la información de las potencias principales y la potencia de cada dispositivo, la salida será un sistema preparado para desagregar la señal de potencia del dispositivo.

Se representa a toda la arquitectura como un modelo F, capaz de desagregar la señal de un dispositivo Y a partir de la señal agregada de potencia X. Como muestra la Figura 3.5, los datos validados se utilizan como datos de entrada para entrenar el modelo F. Para el proceso de entrenamiento

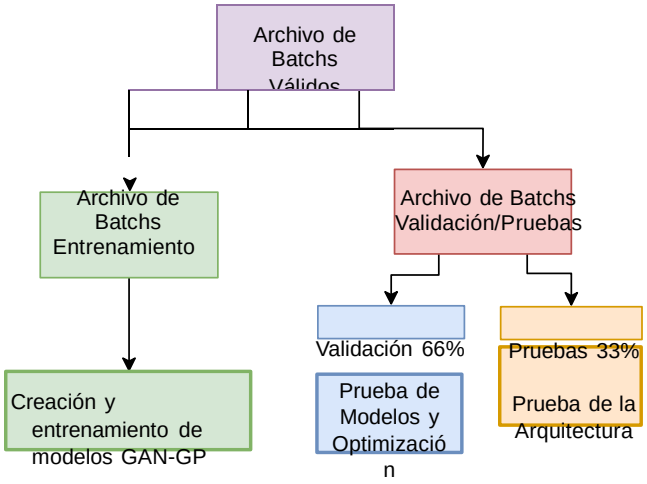

Figura 3.4: Separación de datos.

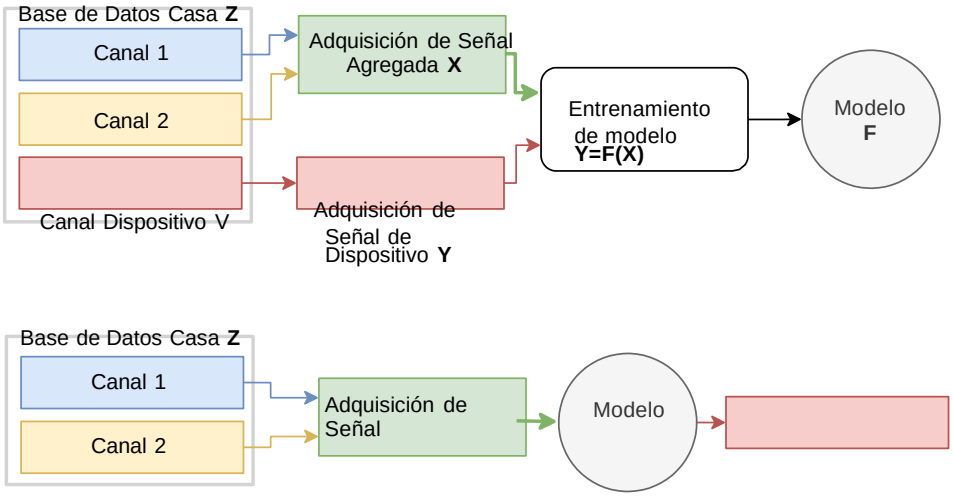

Figura 3.5: Esquema general de la problemática NILM.

del modelo es necesario tener la señal de potencia del dispositivo (Y). Sin embargo, una vez que el proceso de entrenamiento esté finalizado, se podrá obtener la señal de potencia Y del dispositivo a partir de la señal agregada X utilizando el modelo F, esto quiere decir que ya no es necesario censar la señal de potencia de un dispositivo sino solamente obtenerla desde el modelo

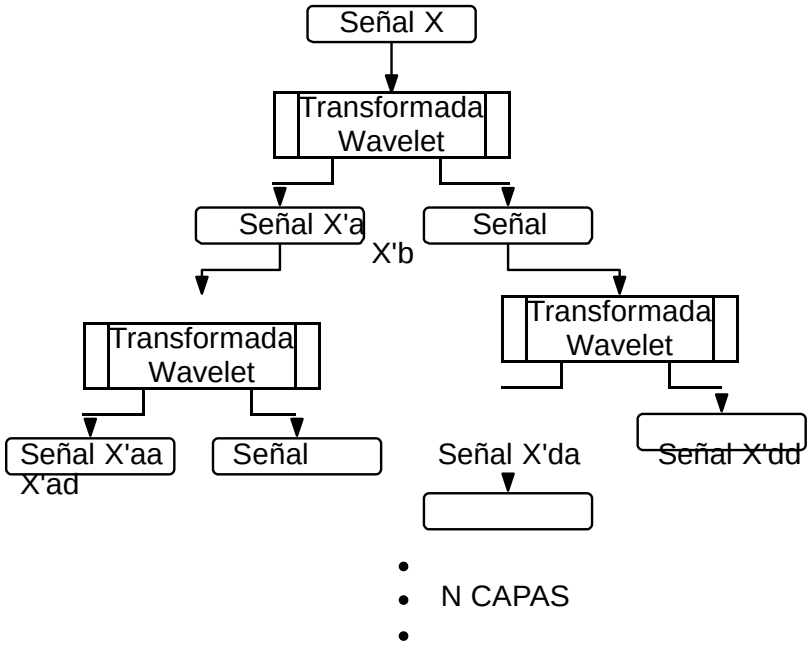

Figura 3.6: Proceso de Transformadas wavelet.

debidamente entrenado. Para que esto funcione correctamente, el modelo debe tener el mínimo error posible.

La arquitectura representada en este esquema general como un modelo F esta conformado por varias fases y componentes, estas etapas se explicaran a continuación por partes.

3.4.1 Descomposición de la señal

La primera fase consta de aplicar la transformada de wavelet a las señales o datos de entrada X y Y, de esta manera obtener los nodos de N niveles X' y Y'. Estos serán los datos que se utilizarán para entrenar el modelo. La transformada de wavelet separa en cada nivel la información de la baja frecuencia a la que se conoce como aproximación y la alta frecuencia a la que se le conoce como detalle. En la Figura 3.6 se muestra este proceso de descomposición de la señal X en donde se aplica la transformada wavelet en cada nivel así se obtiene el grupo de 2^N señales resultantes en el nivel N; con lo que se obtiene el grupo de señales descompuestas X'. El mismo proceso se realiza para la señal de potencia del dispositivo en cuestión Y para obtener el grupo de señales descompuestas Y'.

Para el desarrollo de este trabajo se descompuso la señal en 5 niveles, siguiendo el mismo procedimiento anteriormente, desde los datos originales se obtuvo el nodo “dddd”, correspondiente a la ultima descomposición de los detalles. En total se tienen 255 nodos. Se mencionó en el apartado anterior que cada batch tiene 1024 muestras de datos válidos, en cada descomposición se divide estas muestras; por ejemplo, en el primer nivel el nodo a(aproximación) la señal tiene 512 datos y en el nodo d(detalle) la señal tiene 512 datos. Siguiendo este comportamiento, en el ultimo nivel cada nodo solamente procesa señales con 32 datos.

Se debe recordar que esta misma metodología se aplica a las señales de entrada X y a la señal del dispositivo Y. Así, cada nodo tiene su subconjunto de datos a procesar, es fácil deducir entonces que el procesamiento en los niveles inferiores será mucho mas rápido que los niveles mas cercanos al nodo principal.

3.4.2 Entrenamiento de los modelos

En ésta fase se utiliza el conjunto de datos para moldear cada uno de los nodos con el fin de lograr obtener de la arquitectura una señal de potencia de un dispositivo a partir de las señales de potencia media consumidas por la casa. Las señales de entrada para cada modelo(nodo) ya se han preparado y definido anteriormente, así también ya se conoce la señal del dispositivo deseado; es decir, nuestro objetivo deseado que nos servirá para compara con lo obtenido y calcular un error de retroalimentación.

En la arquitectura se ha definido que cada nodo del árbol de descomposición tenga un modelo GAN diferente; esto significa que en este proceso se entrenarán 2^N modelos. Formalmente, lo que se busca es que, teniendo la señal del nodo X’n, se entrene un modelo que me permita generar la señal Y’n con el mínimo error posible.

La Figura 3.7 ilustra este procedimiento, explicando el proceso en el nodo a, en donde el modelo correspondiente se le nombra como modelo generativo a, se nota que utiliza como datos de entrada los valores de la señal descompuesta X’a, es importante tener en cuenta que esta potencia media en la acometida de la casa se compone de dos señales, una por cada linea principal. El modelo GAN genera una señal Y’a est que es la señal estimada de la potencia consumida por el dispositivo en su descomposición wavelet a. El objetivo es que esta señal estimada, generada por la GAN sea lo más parecido posible a la señal real descompuesta Y’a; por lo tanto, la diferencia entre la señal Y’a est y Y’a es el error utilizado para retro-alimentar el modelo en el nodo a.

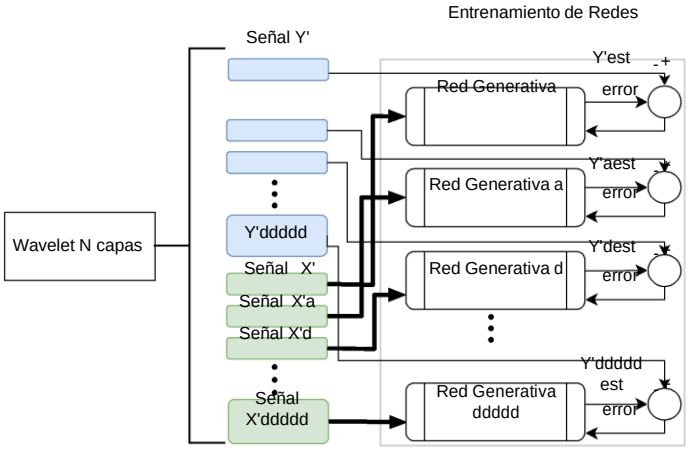

Figura 3.7: Entrenamiento de las GAN.

Este mismo proceso se realiza para cada uno de los modelos encargado de generar la señal estimada de la potencia del dispositivo en cada nodo de descomposición. Al finalizar el entrenamiento, los modelos son almacenados para las siguientes fases.

3.4.3 Reconstrucción de la señal

Con los modelos entrenados y listos para utilizarse, empieza la fase de reconstrucción de la señal Y est. Esta es la señal que la arquitectura es capaz de generar, conociendo únicamente la señal X; y se espera sea lo mas parecida posible a la señal real Y.

El proceso de reconstrucción de la señal necesita a priori la siguiente información preprocesada:

- Los modelos de cada nodo del árbol de descomposición correctamente entrenados.
- Las señales X' del árbol de descomposición para cada uno de los nodos.
- La wavelet madre para aplicar la transformada inversa.

La Figura 3.8 muestra la primera etapa de la reconstrucción; en el nivel mas bajo, es decir el nivel N del árbol de descomposición, cada modelo GAN correspondiente a un nodo n procesa la información de la señal X'n de ese nodo para generar la señal Y'n equivalente al mismo nodo.

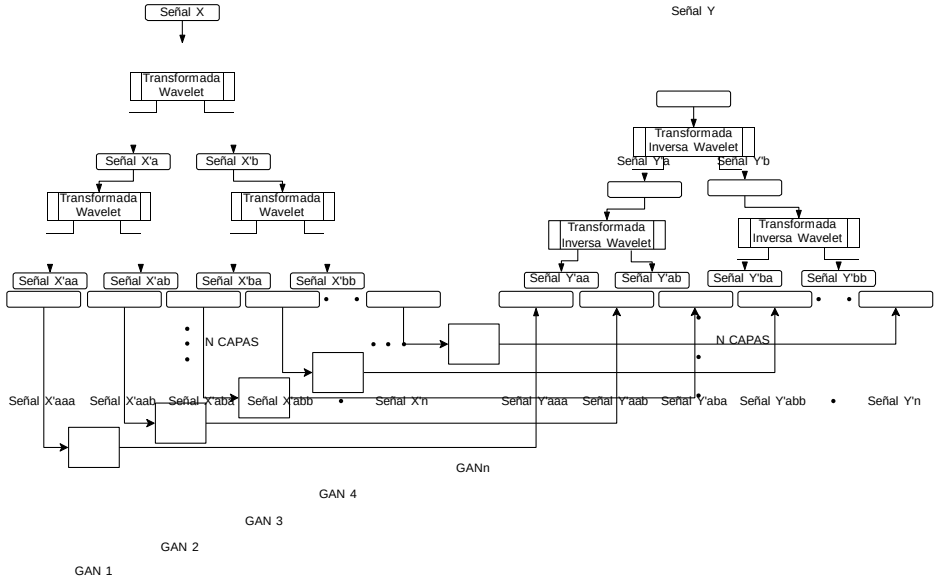

Figura 3.8: Reconstrucción de la señal Y.

Para entender de mejor manera este proceso se utiliza el ejemplo del nodo “ddddd” (ultimo nodo del nivel N=5), el modelo entrenado para este nodo procesa la información en la señal X’ddddd (señal que almacena la información de las dos lineas de potencia principal descompuesta a través del proceso de la transformada wavelet 5 veces), como resultado del modelo se obtiene la señal Y’ddddd, misma que se utilizará posteriormente para la reconstrucción de la señal.

Este proceso se aplica en todos los nodos del nivel N del árbol de descomposición; sin embargo, para los siguientes niveles la reconstrucción de la señal es un poco mas elaborada.

Para la reconstrucción de un nodo de Y’m, no perteneciente al nivel más bajo del árbol de descomposición de señales, la metodología utilizada se ilustra en la Figura 3.9, la señal resultante del modelo GAN en el nodo m al procesar los datos de entrada Y’m se denota como el residual de Y’m denotado por $\hat{Y}$m, a este valor se le debe agregar la transformada inversa de wavelet del nivel anterior para obtener el valor de Y’m. Este proceso se repite para cada nodo del árbol de reconstrucción.

Siguiendo esta metodología, finalmente en el nivel mas alto del árbol se obtiene la señal Y est, la misma que representa lo mas fielmente posible la señal de potencia consumida por el dispositivo. De esta manera se ha logrado el objetivo de resolver la tarea NILM.

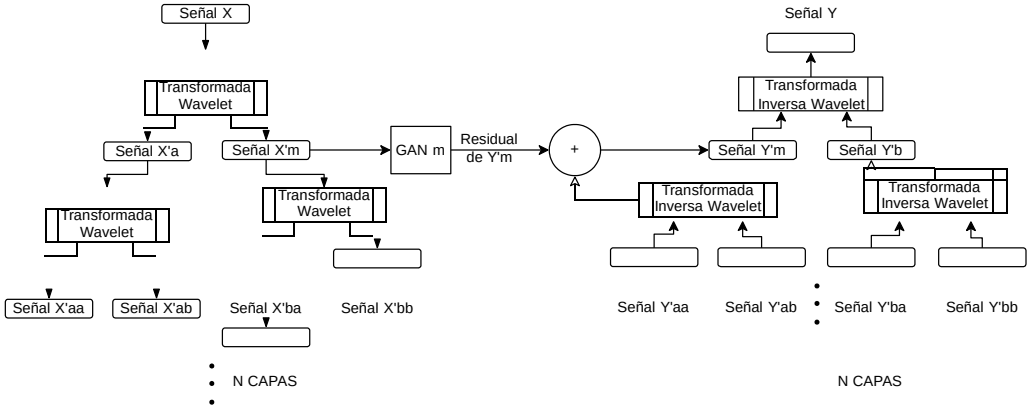

Figura 3.9: Reconstrucción de la señal Y.

3.5 Optimización

Hasta el momento, se ha logrado desagregar la señal de potencia de un dispositivo a partir de la señal de potencia consumida por la casa tomada de la acometida principal. Ciertamente en este proceso cada entrenamiento de modelo tiene una retroalimentación que busca minimizar el error continuamente, sin embargo este error solamente se obtiene comparando la señal deseada y real en ese nodo, sin involucrar una retroalimentación basada en el error entre la señal real y generada al final del proceso. Esta problemática provoca que la señal generada en el ultimo nivel no tenga la misma fidelidad que en la mayoría de los nodos, un análisis detallado de los resultados ha demostrado que existen nodos que aportan favorablemente a la reconstrucción de la señal del dispositivo mientras que existen otros nodos que no aportan o incluso desfavorecen la reconstrucción de la señal, es decir incrementan el error final.

Como solución a esto, se propone implementar un algoritmo de optimización capaz de disminuir este error final, el algoritmo de optimización por enjambre se basa en el comportamiento de los enjambres de abejas en la naturaleza, cada individuo o partícula busca aleatoriamente la configuración con mejores resultados mientras se va movimiento a través del espacio de posibles soluciones.

En esta arquitectura, la configuración de cada partícula de se compone de 2^N números reales, cada uno de los cuales darán un peso mayor o menor a cada nodo de reconstrucción, el objetivo es que el optimizador encuentre cuales son los nodos de reconstrucción con información más importante y los que tengan información menos importante, tomando en cuenta que el

objetivo es que el error al reconstruir la señal Y sea el mínimo posible.

El proceso de optimización está restringido por una cantidad definida de iteraciones, en cada iteración se almacena la configuración de la mejor partícula entre todas las pruebas que se han realizado hasta el momento.

Capítulo 4

Desagregación energética en la nube

En este capítulo se describe el proceso para el diseño e implementación del sistema de desagregación energética en la nube. Concretamente, se explican los detalles de la arquitectura usada para cargar y controlar el servicio web, así como los ajustes que se realizaron sobre el algoritmo para adaptarlo al entorno web. Además, muestra los pasos a seguir para poner en marcha el sistema y su funcionamiento.

4.1 Requerimientos del sistema

Una aplicación web se aloja en un servidor en internet que debe cumplir con las necesidades de la aplicación, caso contrario se producirán fallos que no permitirán ejecutarse al programa. Por lo tanto, el punto de partida es definir los requerimientos mínimos que la plataforma debe tener disponibles; es decir, el entorno de ejecución y las versiones de las librerías utilizadas durante el desarrollo del algoritmo.

Cuando se habla de entorno no solamente es el lenguaje de programación, también se debe definir la cantidad mínima de recursos que el servidor debe proveer como por ejemplo memorias, procesadores, tiempos de respuesta, entre otros. El algoritmo de este trabajo fue elaborado en un lenguaje de programación gratuito y procesado en un sistema operativo bajo la misma licencia libre, estas características facilitan la selección de un servidor para la preparación y montaje de la aplicación web.

A continuación se detallan estos requerimientos mínimos necesarios en la plataforma a escoger para que el sistema de desagregación funcione correctamente:

- Sistema operativo basado en Linux, escogida para el desarrollo y procesamiento del sistema por su licencia gratuita y compatibilidad.
- Entorno de ejecución de Python, lenguaje de programación de alto nivel gratuito, escogido por sus múltiples ventajas entre las que se destaca que es compatible con toda plataforma de tecnología actual y la comunidad de programadores pone a disposición mucha ayuda en librerías y grupos de trabajo. Para el desarrollo de este trabajo se utiliza la versión 3.8.
- Librería Flask, descrito como un “micro” Framework basado en Python y concebido para facilitar el desarrollo de aplicaciones web bajo un patrón que separa las paginas web, los datos y los algoritmos. Ésta librería permite de manera fácil montar aplicaciones web que se ejecuten en un entorno Python. La versión utilizada para este trabajo es la 1.1.2.
- Librería PyWavelets, utilizada para aplicar la transformada wavelet a una señal en Python, optimizada para brindar el mejor rendimiento y de simple uso. La versión utilizada en este trabajo es la 2.0.1.
- Librería matplotlib, librería de Python especializada en la creación de gráficos en dos dimensiones, usada para representar gráficamente los

resultados procesados. La versión en uso es la 3.5.1.

- Librería tensorflow, especializada en computación numérica. Permite trabajar de forma amigable con modelos de aprendizaje, para ello utiliza eficientemente las matrices de datos multidimensionales (tensores). La versión usada en este trabajo es la 2.3.0.

- Librería pandas, especializada en el manejo y análisis de estructuras de datos y utilizada para almacenar la información procesada que se necesita para futuros procesos durante la solución del algoritmo. La versión utilizada es 1.4.1.

En la lista se han detallado las versiones de cada librería, este detalle es muy importante ya que varias librerías cambian el nombre de sus funciones o la manera como se utilizan al crear una nueva versión; esto significa que si se utiliza una versión diferente el programa no va a poder concluir su proceso correctamente o no va a devolver los resultados deseados. Típicamente se piensa que utilizar la última versión, o la más actual es lo mejor; sin embargo, eso puede provocar fallos en el programa. Con estas herramientas preparadas se procede a la etapa de diseño de la aplicación.

4.2 Diseño

Una vez establecido los requerimientos para el sistema, se procede a la fase de diseño. En esta fase el objetivo es identificar el flujo que tienen los datos en la aplicación web y los posibles fallos que el sistema tendrá que tolerar. La arquitectura del sistema de desagregación energética en la nube se refleja en la Figura 4.1. En síntesis, los datos de potencia de la vivienda se envían como un archivo a través de internet a la aplicación web, la misma que procesa la información y muestra al usuario la potencia del dispositivo. La comunicación entre el cliente y el servidor es con el uso del protocolo HTTP; en el servidor, la plataforma de Google utiliza diferentes aplicaciones para el procesamiento, almacenamiento e interconexión de los datos, de entre las diversas opciones para el desarrollo de la aplicación se selecciona trabajar con App Engine, especializada en aplicaciones web en diferentes lenguajes y entornos, para el almacenamiento de datos se utilizan dos repositorios, el uno conocido como bucket y el otro un almacenamiento temporal, finalmente las aplicaciones de monitoreo y de IAM controlan el flujo de la información en la aplicación. El modelo de comunicación utilizado por los servicios de la plataforma son de tipo API REST, entre todas las ventajas que esta tecnología significa, la principal es la versatilidad de adaptarse a las plataformas de los usuarios,

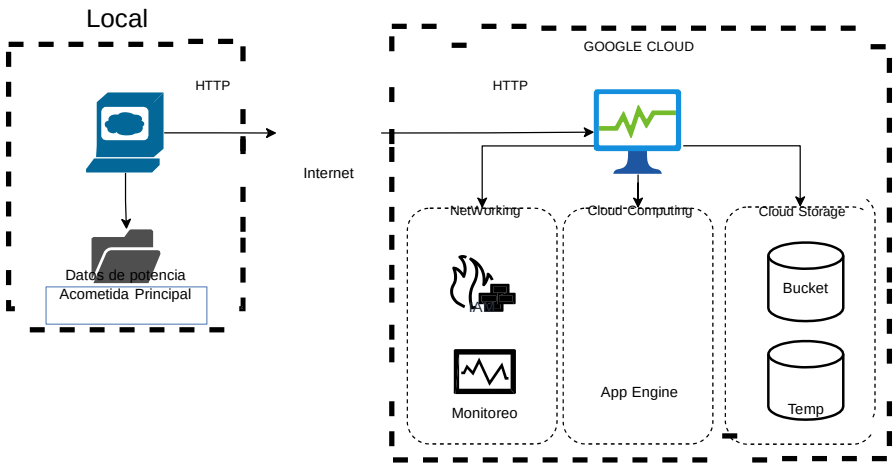

Figura 4.1: Arquitectura de la aplicación web.

quienes pueden acceder al servicio desde cualquier dispositivo o plataforma. Para una mejor organización, el diseño de la aplicación se divide en dos partes principales: front-end y back-end, a continuación se detalla cada parte.

4.2.1 Front-End

El front-end de una aplicación web hace referencia a la interfaz gráfica que el usuario va a utilizar, la información y las opciones que tiene a su disposición deben ser lo mas intuitivos y amigables posible. Para este trabajo, el diseño del front-end es simple, pues en sí la dinámica de resolver la tarea NILM en la nube es un proceso sencillo. Se compone básicamente de dos etapas, la primera pantalla muestra un entorno para cargar el archivo con los datos de la potencia principal de la casa; una vez cargado exitosamente el archivo, en la segunda etapa se muestra una pantalla con un menú que permite al usuario escoger en cual de los dispositivos se desea aplicar el algoritmo de desagregación energética.

Ambas pantallas fueron creadas bajo una estructura de diseño amigable y de fácil uso. Una vez que se selecciona alguno de los dispositivos en el menú de la segunda pantalla, la aplicación web trabaja para resolver la NILM y muestra los resultados de las potencias de entrada como el de la potencia desagregada. Esta pantalla resultante se muestra en la Figura 4.2. En este

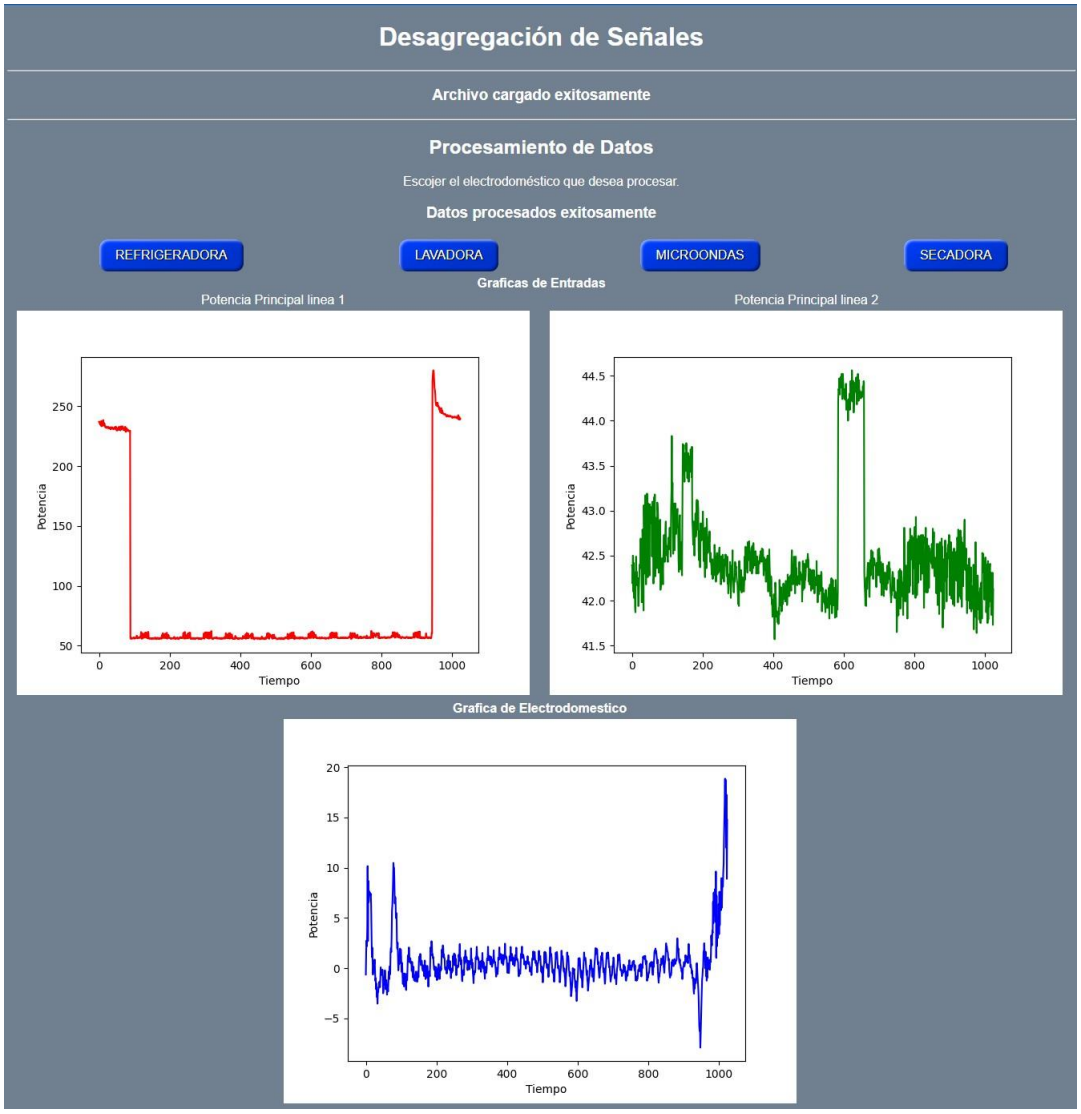

Figura 4.2: Front-End aplicación web.

entorno gráfico simple, el usuario tiene a disposición todos los resultados que solicite.

4.2.2 Back-End

Se conoce como back-end a toda la parte de una aplicación web encargada del manejo y procesamiento de datos. Típicamente, el usuario desconoce que existe o de qué se conforma esta parte. El desarrollo de la aplicación web tiene su fortaleza en esta parte; cada pantalla anteriormente mencionada tiene un proceso interno que permite cumplir con la tarea NILM. La primera pantalla tiene un algoritmo capaz de abrir un explorador de carpetas y archivos en el computador del usuario, de esta manera puede escoger el archivo que va a ser cargado con el objetivo de procesar y obtener la señal desagregada; este archivo cargado se almacena en el bucket anteriormente mencionado para que la siguiente pantalla pueda disponer fácilmente de esta información.

En la segunda pantalla, después de que se selecciona un dispositivo de entre las opciones, el proceso de desagregación energética comienza. Este algoritmo se compone de varios pasos, a continuación se describe brevemente el flujo de los datos para cumplir la tarea NILM en un dispositivo:

1. Se carga los modelos entrenados que generan las señales en cada uno de los nodos del árbol de descomposición. Estos modelos están almacenados previamente dentro del bucket.
2. Se aplica la transformada de wavelet a la señal de potencia de entrada de la casa, y se almacena estos datos en el bucket.
3. Cada modelo genera su estimación de la señal en el nodo correspondiente. Estos datos también se almacenan en el bucket.
4. Se aplica la transformada inversa de wavelet para reconstruir la señal del dispositivo utilizando los datos del anterior nivel y se gráfica. Estas imágenes se almacenan en la caché temporal.
5. Por ultimo, el proceso de front-end se encarga de tomar las imágenes almacenadas temporalmente y mostrarlas en la pantalla.

Para ajustar el programa original a una aplicación web, se han realizado varios cambios y ajustes en la estructura del programa, teniendo en cuenta que ahora la aplicación web es la encargada administrar las páginas, los procesos y la información.

4.3 Implementación

A partir de la arquitectura generada en la fase de diseño se procede a desarrollar su implementación. En esta fase se escoge una plataforma capaz

de cumplir con los requerimientos necesarios para el funcionamiento del programa. Después, se prepara el entorno de trabajo y finalmente se carga el servicio en la nube.

4.3.1 Selección de Plataforma

En base a los requerimientos necesarios para que el programa se ejecute en un servidor web, se selecciona la plataforma de Google llamada Google Cloud. Este entorno se trata de una infraestructura de servicios que esta disponible para cualquier empresa a un costo accesible. Como potencial ventaja, esta plataforma ofrece y tiene experiencia en manejo de big data y modelos de aprendizaje. Además tiene alta confiabilidad ya que es la plataforma que la misma empresa Google utiliza para su desarrollo.

La infraestructura ofrece una gran cantidad de servicios de diferentes tipos para múltiples propósitos. De entre todos estos servicios se escogen dos que permiten montar un servicio web en la nube. El adecuado para el procesamiento del algoritmo en una aplicación web se le conoce como Google App Engine. Se le considera una plataforma mas que un servicio, que permite a los desarrolladores crear y alojar aplicaciones web aprovechando su plataforma sin servidor. Entre sus principales ventajas se destaca su compatibilidad, su escalabilidad y facilidad de uso. Por otro lado se escoge Google Storage como un servicio para almacenar los objetos. Se entiende a objeto como un dato inmutable que consta de un archivo en cualquier formato. Los objetos se almacenan en contenedores llamados buckets, todos los buckets están asociados a un proyecto que, a su vez, se puede agrupar en una organización.

Además de estos dos servicios, para la implementación y pruebas se utilizan herramientas que la plataforma pone a disposición del desarrollador como el apartado de monitoreo para controlar el tráfico de datos, la consola web para depurar el procesamiento de la aplicación y el apartado de control de errores para verificar los fallos en las pruebas. Todos estos servicios se encuentran integrados por defecto dentro del mismo proyecto en la plataforma.

4.3.2 Preparación del programa

El modelo de gestión de la aplicación web separa los datos, los procesos y las páginas web manteniendo un gestor principal entre estas tres partes. Por lo tanto, para que la aplicación desarrollada en este trabajo se ejecute en la nube, se debe hacer algunos ajustes en su estructura y algoritmo. Si bien

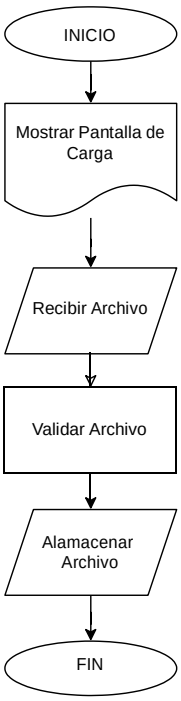

Figura 4.3: Algoritmo en pantalla de Carga.

ya se explicó brevemente los pasos que se siguen para desagregar la señal del dispositivo, en esta sección se detalla cada uno de estos pasos desde la perspectiva de la implementación. Para su entendimiento, se divide el proceso que se ejecuta en la pantalla de carga y el que se ejecuta en la pantalla de resultados.

La Figura 4.3 muestra el algoritmo que se implementa en la primera pantalla, la que sirve para cargar el archivo con extensión “PICKLE” que contiene las muestras de potencia en las lineas principales de la vivienda. Los pasos que sigue se detallan a continuación:

1. Al momento de solicitar el servicio web en la plataforma (escribir la URL en el navegador), se despliega la pagina creada en la etapa de front-end con las facilidades para que el usuario escoja el archivo que desea que se procese en la nube.

2. Recibe el archivo y lo almacena temporalmente en caché.

3. Valida que el archivo sea correcto, en este paso se verifica que la extensión del archivo sea la correcta y que el nombre del archivo no contenga caracteres inválidos que den inconvenientes al momento de almacenarlo.

4. En este paso se almacena dentro del bucket del proyecto el archivo con un nombre específico, el mismo que se utilizará en las siguientes etapas para cargarlo y procesarlo.

La segunda pantalla es la que hace el resto del procesamiento, la Figura 4.4 muestra el flujo del procesamiento de los datos desde que son cargados hasta que se resuelve la tarea NILM, a continuación se explica cada etapa de este algoritmo:

1. Una vez cargado correctamente el archivo, el programa principal muestra al usuario la segunda pantalla, un entorno en donde se puede escoger el dispositivo en el cual se desea aplicar la desagregación energética.

2. Se carga los modelos entrenados que generan las señales en cada uno de los nodos del árbol de descomposición, estos modelos están almacenados previamente dentro del bucket.

3. Inicia el bucle que recorre cada nivel. La carga de los modelos temporalmente ocupa la memoria primaria de la plataforma(RAM); a pesar que la plataforma brinda una gran escalabilidad en temas de recursos, la cantidad de modelos puede provocar que esta memoria se desborde y se convierta en un fallo o caída del servicio. Para evitar este posible problema, se procesan los datos de nivel en nivel dentro de este bucle.

4. Desde el bucket, se cargan en memoria RAM los modelos correspondientes al nivel actual del bucle, manteniéndolos listos para procesar datos y generar resultados.

5. Se adjunta los valores de las estimaciones del nivel anterior; en el caso de ser el último nivel(nivel N), este paso es despreciado pues no se dispone de más información para procesar.

6. En este paso se procesa la información de cada nodo, para esto se utiliza las señales aplicadas la transformada wavelet antes almacenadas en el repositorio. Cada modelo genera una señal estimada de la potencia del

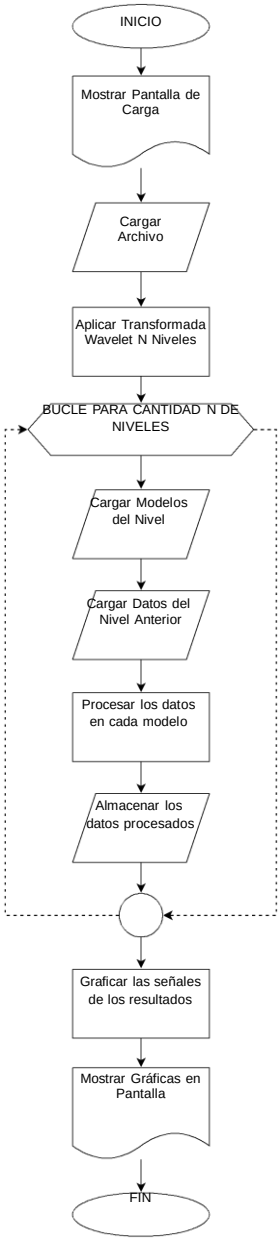

Figura 4.4: Algoritmo en pantalla de Resultados.

dispositivo a partir de la señal de potencia principal correspondiente a ese nodo.

7. Repetir este proceso N veces permite obtener la señal de potencia del dispositivo finalmente reconstruida. Es decir, la señal en el nodo más alto del árbol de descomposición. Esta señal se representa gráficamente y se almacena en un repositorio temporal, en este paso no se utiliza el bucket.

8. Finalmente, las imágenes almacenadas temporalmente son enviadas a la pantalla para que se muestren los resultados al usuario.

4.3.3 Montaje del servicio

Ya con el programa listo y acoplado a la arquitectura, se carga el sistema en la plataforma. A continuación, se detalla los pasos a seguir para que el servicio esté funcionando en la nube.

El primer paso es crear una cuenta con un usuario y una organización dentro de la plataforma. En este punto cabe denotar que no todas las plataformas son gratuitas y se requiere de un medio de pago para la creación del usuario y la organización dentro del sistema. Por otro lado, la mayoría de plataformas brindas un periodo de prueba limitado ya sea por tiempo o por capacidades máximas que se pueden usar de sus servicios, éste último es el caso de la plataforma escogida.

Una vez creado el perfil, se crea un nuevo proyecto. Todas las herramientas y servicios están disponibles para ese proyecto, estas herramientas se muestran en la Figura 4.5. Los iconos facilitan mucho el manejo y navegación entre cada servicio.

Como se mencionó anteriormente, para esta aplicación se usa App Engine. Para que un programa se convierta en una aplicación Web a través de este servicio se necesita como mínimo los siguientes elementos:

- Un archivo “Requeriments.txt” en donde se especifica todas las librerías que deben cargarse desde el repositorio pip, en caso de que no se puedan instalar desde este repositorio se deberá cargar en un bucket y referenciarlo. El servidor elimina toda las librerías anteriormente instaladas en el proyecto e instala las que estén especificadas en este archivo.

- Un archivo “main.py” que contiene el programa principal, el que se encarga de mostrar las páginas y ejecutar los programas en base a la

selección del usuario en la aplicación web. La estructura de este archivo se basa en el manejo de variables que son enviadas hacia las páginas web que se muestran, utilizan el URL de la página web para identificar el caso u opción que el usuario escogió.

- Una carpeta "templates", en donde se almacenan todas las paginas web que se utilizan, esta carpeta únicamente puede almacenar archivos con extensión HTML.
- Una carpeta "static", en donde se puede almacenar toda la información y datos que no se van a modificar pero si se van a cargar, mostrar o utilizar durante el procesamiento de la aplicación web. A diferencia de la anterior carpeta, en ésta no se tiene restricción de tipo de datos; sin embargo, no pueden ser modificados una vez que se cargan en la nube.
- el archivo "app.yaml", este es el archivo de configuración de entorno. Tiene muchas opciones dependiendo del tipo de servidor que se necesite; entre las principales características escogidas se denota la versión de Python en la que se ejecutará el programa, el tamaño de la memoria RAM que se separa para procesamiento, el tamaño de disco necesario para almacenar los datos, la cantidad de núcleos que se destinan al procesamiento, el tiempo de espera para que se procese, entre otros. Entre las configuraciones utilizadas para el desarrollo de este trabajo se escogió 4GB de RAM (lo máximo posible), dos núcleos para el procesamiento, 10 GB de disco de almacenamiento y un tiempo de respuesta de 300s.

Una vez configurados con la información correcta, estos elementos se cargan en la plataforma. el último paso para montar el servicio en la nube es compilar o hacer un "deploy" de todo el algoritmo. Para este paso se utiliza la consola que esta integrada dentro de la plataforma y se corre el comando `gcloud app deploy`. Después de varios minutos el sistema ya está en linea en una URL proporcionada por la plataforma.

4.4 Verificación

Al tener implementado todo el sistema online de desagregación energética, las pruebas y la verificación del correcto uso es el último paso.

Por ventaja, el proyecto cuenta con dos herramientas que facilitan esta fase. La primera herramienta se llama "Monitoring", la Figura 4.6 muestra una captura de pantalla de este entorno. En síntesis permite controlar el

Figura 4.5: Proyecto Google Cloud.

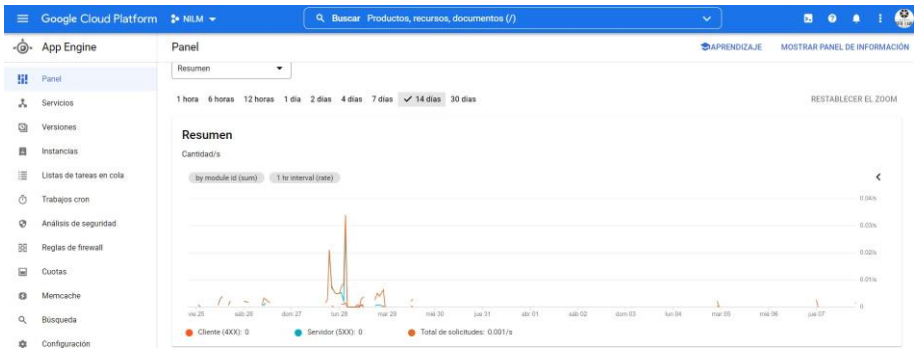

Figura 4.6: Servicio Monitoring.

trafico de red en la aplicación web, así como monitorear posibles fallas y tiempos que se mantuvo activa o en uso la aplicación web. Esta herramienta muestra también un resumen de los costos generados por la aplicación resumiendo la cantidad de solicitudes atendidas por el servidor.

La segunda herramienta utilizada para la verificación del funcionamiento del sistema en la nube se llama “Error Reporting”. Como su nombre lo indica, detalla todos los fallos y errores que ha tenido el sistema, categorizándolos por su gravedad y por la cantidad de veces que se repite el error. Al momento de investigar sobre un error en específico, brinda todos los detalles del proceso que llevó a la producción del error. Esto ayuda mucho en el proceso de depuración, encontrar e identificar las circunstancias en las que se produce un error permite al desarrollador actualizar los fallos eficientemente.

Capítulo 5

Análisis de resultados

El presente capítulo muestra el análisis de los resultados obtenidos en cada etapa del desarrollo de la metodología propuesta, así como los resultados obtenidos en la implementación y pruebas. Además, indica el cumplimiento de los objetivos relacionados con la optimización aplicada y el sistema funcionando en la nube. Toda la información recopilada en este capítulo es la base para establecer conclusiones del trabajo.

Durante el desarrollo de la metodología propuesta se obtuvieron varios resultados en cada una de las etapas planteadas. Estos resultados fueron obtenidos mediante distintos procesos de experimentación y análisis de las señales resultantes, utilizando el error como métrica principal de categorización.

Es importante denotar que las gráficas presentadas en este capitulo hacen referencia al residual de la potencia en los diferentes nodos del árbol de reconstrucción de la señal, esto justifica la razón por la cual existen valores negativos de potencia, situación no común en el consumo de energía de un dispositivo. En los nodos principales o del nivel mas alto del árbol, el residual de potencia muestra cuánto se puede sobrestima o subestimar la potencia en el transcurso del tiempo, es por esta razón que los valores son negativos en algunos puntos. Por otro lado en los siguientes niveles del árbol, la naturaleza misma de la transformada wavelet produce que el valor de la potencia residual genere también valores negativos. La metodología propuesta trabaja con este residual como objetivo de las redes WGAN-GP propuestas, por esta razón las señales generadas producen el mismo comportamiento de valores positivos y negativos. Sin embargo, se debe recalcar que esto no significa que los valores censados reales tengan potencias negativas, pues no son dispositivos que se comporten como generadores de corriente.

A continuación, se detalla cada uno de los resultados alcanzados durante cada etapa.

5.1 Resultados del proceso de entrenamiento

El entrenamiento de las redes en cada nodo de la arquitectura se realizó en 32 épocas, el tamaño de los lotes es de 1024 muestras y se utilizaron 600 lotes de datos. Al concluir el entrenamiento de cada modelo se selecciona y almacena el resultado de la mejor época; es decir, la que menos error tuvo. Con esta misma estructura se procedió para cada uno de los 4 electrodomésticos. Para la arquitectura propuesta se utilizó 5 niveles, lo que significa que se entrenaron 63 nodos:

- Nivel 5: 32 nodos con 32 muestras en cada lote.
- Nivel 4: 16 nodos con 64 muestras en cada lote.
- Nivel 3: 8 nodos con 128 muestras en cada lote.
- Nivel 2: 4 nodos con 256 muestras en cada lote.

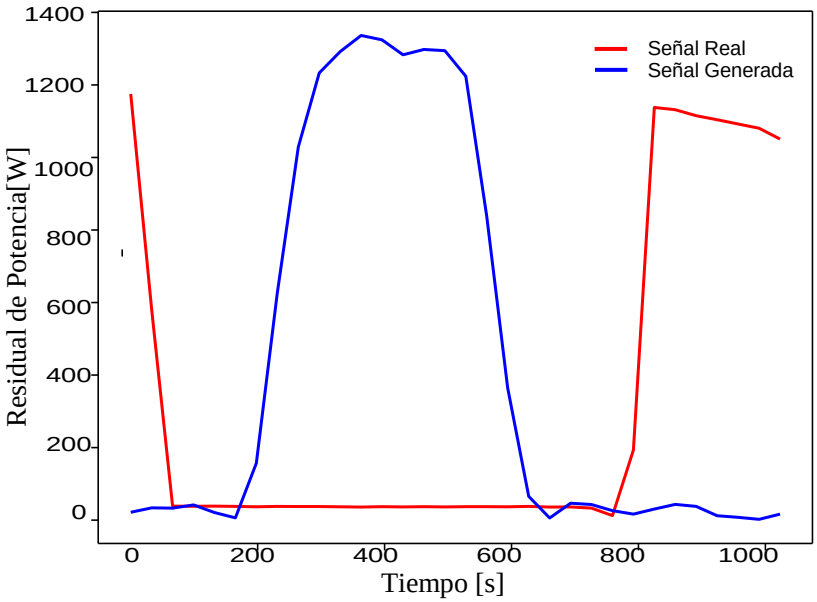

Figura 5.1: Residual de potencia en el nodo aaaaa del nivel 5.

- Nivel 1: 2 nodos con 512 muestras en cada lote.
- Nivel 0: 1 nodo con un lote de 1024 muestras.

Cada uno de los nodos tiene una WGAN-GP entrenada. En los niveles más bajos de las estructuras el error es grande debido a que la información previa es nula; por otro lado, mientras el proceso se acerca al nivel más alto el error disminuye. En la Figura 5.1 se muestra este comportamiento tomando como muestra el primer nodo de las aproximaciones en el nivel 5. En esta Figura se puede notar la cantidad de muestras de solamente 32 datos que representan los 1024 segundos del lote, la señal real pertenece a los datos censados en el refrigerador mientras que la señal generada por se obtiene del modelo WGAN-GP en este nodo de la descomposición.

En el mismo nivel se analiza el nodo de detalles ddddd. Es decir, el último de los nodos de la descomposición de wavelet. En la Figura 5.2 se puede notar que la señal en color azul (generada por el modelo) se acerca mas al comportamiento de la señal en color rojo (residual de lo censado en

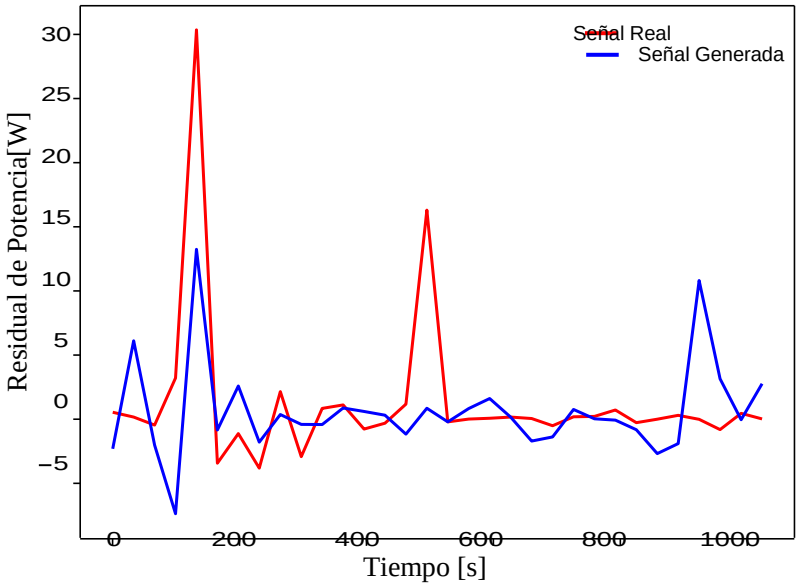

Figura 5.2: Residual de potencia en el nodo ddddd del nivel 5.

el refrigerador), incluso numéricamente la diferencia que existe en la raíz del error cuadrático medio (RMSE) es grande: el nodo aaaaa presenta un RMSE de 198.3 W en su entrenamiento y el nodo ddddd refleja un error de 19.8 W al finalizar su entrenamiento. Al analizar estos resultados se deduce que el sistema tiene muy buenos resultados en señales de alta frecuencia, sin embargo, no se obtienen los mejores resultados en las componentes de baja frecuencia.

La metodología de descomposición a través de la transformada de wavelet utilizada en este trabajo divide el lote de datos cada vez en mitades de muestras; es decir que al llegar a los niveles más bajos el lote de datos a procesar es más pequeño, por lo tanto, el tiempo de entrenamiento es menor en estos niveles y mucho más demorado en el nivel más alto.

El error disminuye conforme los datos procesados se reconstruyen en el proceso, por ejemplo en el nivel 3 el RMSE del nodo aaa es de 142.2 W y el RMSE del nodo ddd es 20.9 W, en las Figuras 5.3 y 5.4 se muestra estos resultados, es notable que la señal generada por el modelo se aproxima más

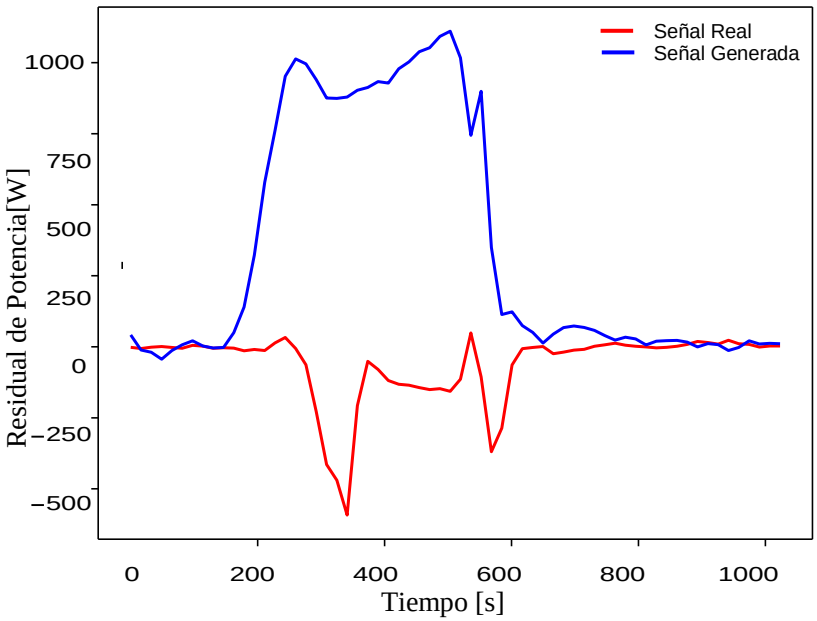

Figura 5.3: Residual de potencia en el nodo aaa del nivel 3.

al comportamiento de la señal del residual de la potencia real.

Esta misma tendencia continua manifestándose hasta el nivel más alto del árbol de recomposición, en donde el RMSE es de 52.9 W, la Figura 5.5 muestra este resultado; como se mencionó anteriormente las componentes de la señal de alta frecuencia tiene mucha similitud con la predicción del sistema, pero se tiene gran diferencia en las componentes de baja frecuencia de la señal.

5.2 Resultados del proceso de pruebas

Como es natural en los modelos de aprendizaje, en la etapa de entrenamiento se logran mejores resultados que en la fase de validación o pruebas. En el capítulo 3 se explicó que existe un conjunto de datos separados con el fin de usarlos en la etapa de pruebas; al tratarse de datos que la la red no ha procesado anteriormente, los resultados tienen en promedio un error mayor a la etapas de validación. Si embargo, estos resultados se consideran más

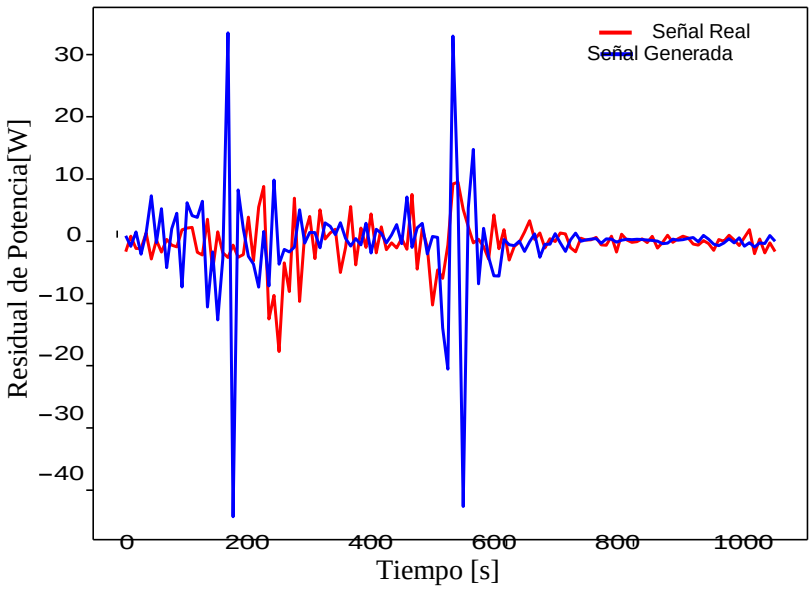

Figura 5.4: Residual de potencia en el nodo ddd del nivel 3.

importantes, pues da una perspectiva real del funcionamiento del sistema.

Al igual que en la fase de entrenamiento, se analizan los resultados desde los nodos mas bajos hasta llegar a los nodos cerca del nivel mas alto. En el nivel 5 se compara los dos extremos, el resultado mas bajo de las aproximaciones, la Figura 5.6 muestra los resultados del residual de potencia en el nodo aaaaa con un RMSE de 289.5 W.

Su error es alto cuando se compara con el resultado del residual de potencia en el nodo mas bajo de los detalles, es decir el nodo ddddd correspondiente al nivel 5. El RMSE en este nodo es 22.2 W; la Figura 5.7 muestra un ejemplo del resultado en este nodo. Esto corrobora que la arquitectura tiene mayor exactitud al generar las componentes de alta frecuencia que las componentes de baja frecuencia.

Este mismo comportamiento se muestra en los niveles más cercanos a la señal reconstruida final. Por ejemplo, en el nivel 3 se compara los nodos aaa y ddd. El primero se muestra en la Figura 5.8 con un RMSE de 208 W y el segundo en la Figura 5.9 con un RMSE de 22.3 W. A pesar de que la

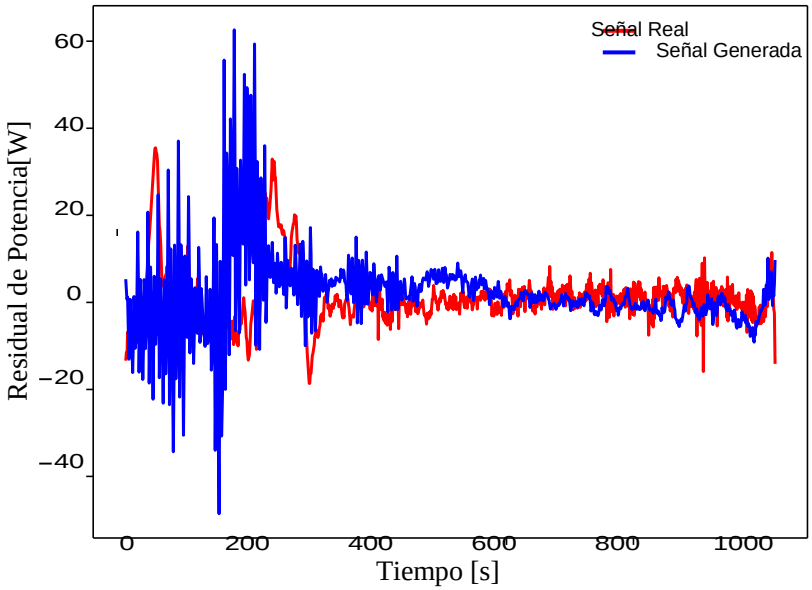

Figura 5.5: Residual de potencia en el nodo final.

diferencia entre estos dos nodos es muy alta, se puede distinguir que en las aproximaciones el error va disminuyendo considerablemente, hasta este punto casi ha disminuido a su tercera parte; mientras que el error en los detalles ha subido muy poco. Esta naturaleza muestra que el error total del modelo es menor conforme se acerca al nodo principal en el árbol de recomposición.

Finalmente se evalúa el error y gráfico de la señal generada por el modelo completo. En la Figura 5.10 se muestra el resultado, el RMSE es de 66.5 W; como se esperaba, el error es mayor al del último nodo en el proceso de entrenamiento, al analizar la Figura de la gráfica se distingue que la señal azul tiene una forma de onda similar a la roja en las altas frecuencias pero un error elevado en las bajas frecuencias.

5.3 Resultados del proceso de optimización

El proceso final para conseguir el modelo definitivo que resuelve la tarea NILM fue optimizar el resultado, en el capítulo 3 se detalló este proceso.

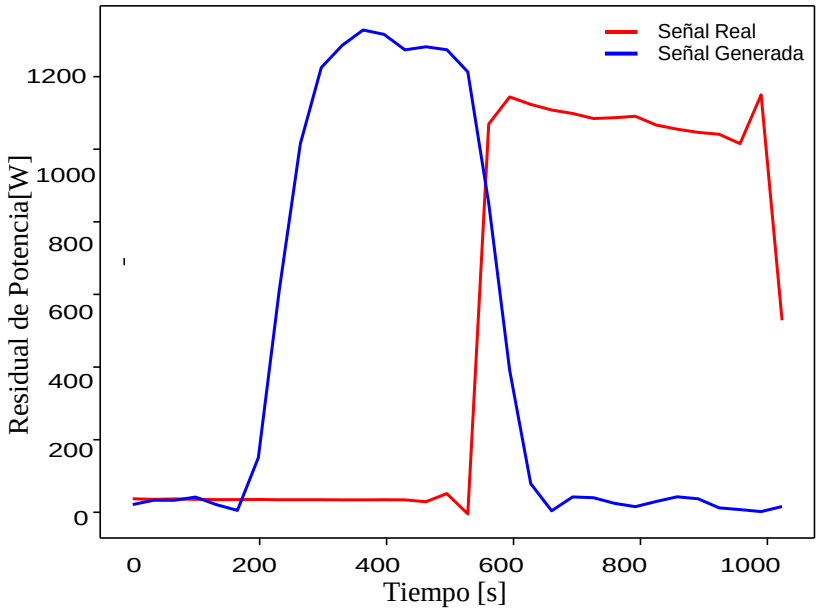

Figura 5.6: Residual de potencia en el nodo aaaaa.

Al culminar el bucle de optimización se utilizan los resultados para realizar pruebas de desagregación de la señal, nuevamente en esta etapa se utiliza el conjunto de datos separado para pruebas.

En esta sección no hace falta evaluar las gráficas en cada nivel, pues al igual que en los anteriores procesos la naturaleza se mantiene; lo importante se revela en la Tabla de resultados de ponderación a cada uno de los nodos. Aquí se distingue que los nodos con más componentes de alta frecuencia son ponderados positivamente mientras que los nodos con mas componentes de baja frecuencia (aproximaciones) son ponderados negativamente.Este comportamiento era esperado, pues como se analizó en las anteriores secciones los nodos que inyectaban mas error a la arquitectura eran los modelos de las componentes de aproximaciones.

Como resultado la Figura 5.11 muestra la señal reconstruida con el proceso de optimización, su RMSE es de 58.7 W, mas bajo que el resultado sin el proceso de optimización.

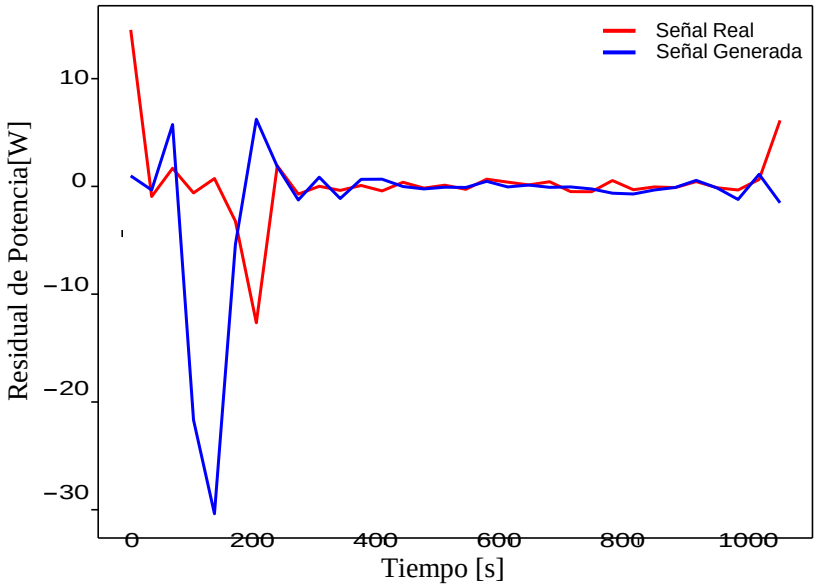

Figura 5.7: Residual de potencia en el nodo ddddd.

5.4 Análisis de desempeño en diferentes dispositivos

En algunos trabajos se aplica la tarea NILM en la refrigeradora, secadora, lavadora y microondas ([Singh and Majumdar, 2018] ,[Çavdar and Faryad, 2019]), [Sirojan et al., 2018]). Estos mismos dispositivos se seleccionaron para el desarrollo de este trabajo. En primera instancia, todos los dispositivos han conseguido un modelo con similares comportamientos; es decir, con una mayor exactitud en las componentes de alta frecuencia. Sin embargo, los resultados no son los mismos en cada dispositivo.

En la Tabla 5.1 se puede notar claramente que en todos los dispositivos el error es mas bajo en el entrenamiento que en la prueba, y el proceso de optimización disminuye notablemente el error. Sin embargo, el error de los demás dispositivos son mucho mas altos que el de la refrigeradora.

En las Tablas 5.2 y 5.3 se detallan los resultados del proceso de aprendizaje y optimización de la refrigeradora, los mismos resultados se detallan en las

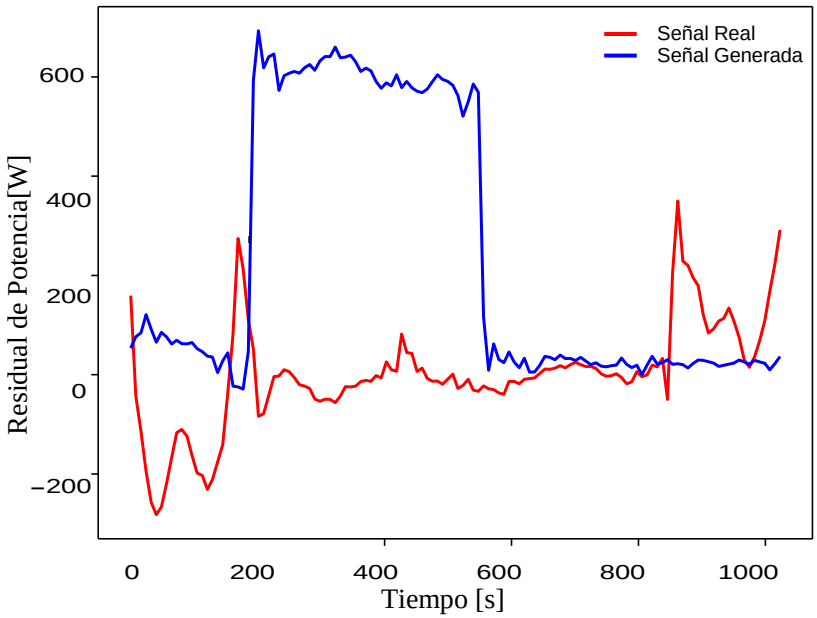

Figura 5.8: Residual de potencia en el nodo aaa.

Tablas 5.4 y 5.5 de la lavadora; por otro lado, las Tablas 5.6 y 5.7 muestran los resultados del microondas y finalmente la información de la secadora esta en las Tablas 5.8 y 5.9.

5.5 Desviación del error

Otro indicativo de la naturaleza de los resultados es la desviación del error, este valor nos muestra que tan dispersos están los resultados de RMSE en el proceso de optimización. Como se identifica en la Tabla 5.10, estos valores son altos pues el error tiene mucha diferencia de error entre los nodos de aproximación y los nodos de detalles.

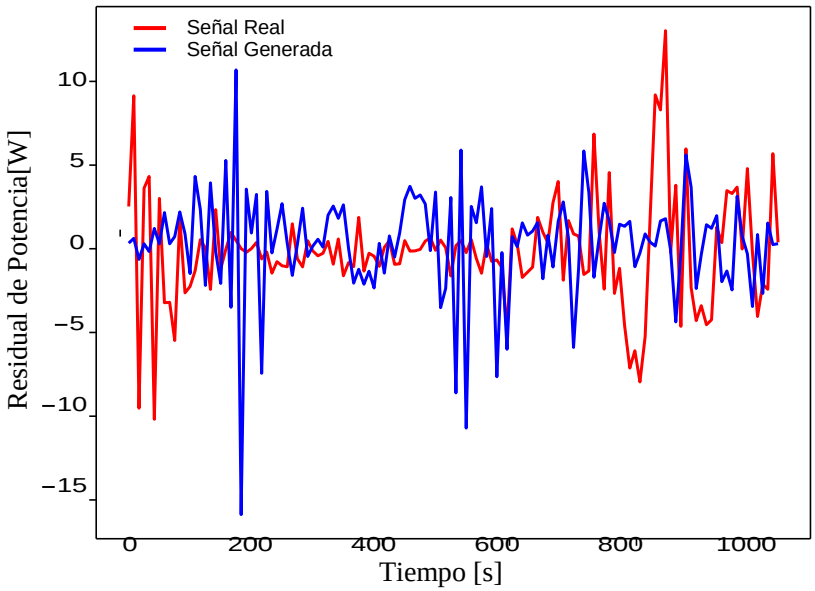

Figura 5.9: Residual de potencia en el nodo ddd.

Tabla 5.1: Resultados de error.

Dispositivo	RMSE Entrenamiento	RMSE Validación	RMSE Optimización
Refrigerador	52.91	66.51	58.73
Lavadora	140.47	158.06	134.34
Microondas	119.41	152.39	137.73
Secadora	98.52	121.83	102.14

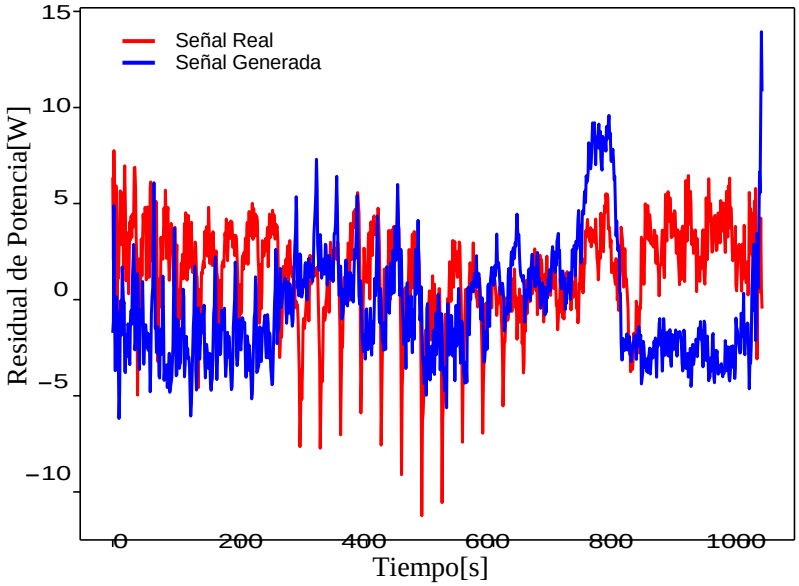

Figura 5.10: Residual de potencia de la señal generada en pruebas.

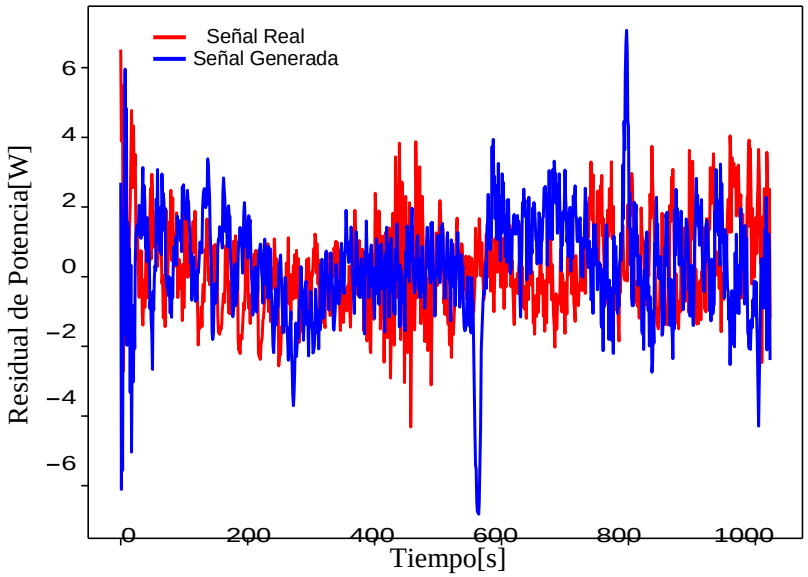

Figura 5.11: Residual de potencia de la señal generada optimizada.

Tabla 5.2: Resultados de la Refrigeradora Nivel 5 y 4.

Nodo	RMSE Entrenamiento	RMSE Validación	RMSE Optimización
aaaaa	198.31	289.46	238.98
aaaad	50.66	54.97	52.43
aaada	27.62	29.65	28.45
aaadd	35.96	39.10	37.24
aadaa	21.55	23.38	22.30
aadad	22.87	24.86	23.68
aadda	25.99	28.02	26.82
aaddd	24.43	26.19	25.15
adaaa	20.54	22.64	21.41
adaad	20.41	22.66	21.34
adada	20.51	22.71	21.42
adadd	20.63	22.86	21.55
addaa	21.18	23.03	21.94
addad	21.43	23.19	22.15
addda	20.74	22.65	21.53
adddd	21.12	22.96	21.87
daaaa	19.35	22.71	20.76
daaad	19.94	22.42	20.97
daada	21.24	23.11	22.00
daadd	20.60	22.68	21.46
dadaa	19.62	22.36	20.76
dadad	19.82	22.27	20.84
dadda	19.75	22.55	20.91
daddd	19.94	22.39	20.95
ddaaa	20.15	22.65	21.19
ddaad	20.16	22.53	21.14
ddada	20.27	22.61	21.24
ddadd	20.29	22.65	21.27
dddaa	20.16	22.58	21.16
dddad	20.08	22.25	20.97
dddda	20.15	22.31	21.04
ddddd	19.80	22.20	20.79

Tabla 5.3: Resultados de la Refrigeradora.

Nodo	RMSE Entrenamiento	RMSE Validación	RMSE Optimización
aaaa	142.18	208.01	171.57
aaad	31.98	34.72	33.10
aada	22.24	24.16	23.03
aadd	25.26	27.24	26.07
adaa	20.49	22.67	21.39
adad	20.63	22.85	21.54
adda	21.31	23.14	22.06
addd	20.95	22.83	21.72
daaa	19.64	22.62	20.88
daad	20.21	22.54	21.17
dada	19.72	22.35	20.81
dadd	19.84	22.52	20.95
ddaa	20.15	22.62	21.17
ddad	20.28	22.62	21.25
ddda	20.13	22.42	21.08
dddd	19.98	22.27	20.93
aaa	90.55	143.00	114.45
aad	23.71	25.75	24.55
ada	20.54	22.76	21.46
add	21.14	23.01	21.91
daa	19.91	22.58	21.02
dad	19.77	22.44	20.88
dda	20.20	22.62	21.20
ddd	20.07	22.36	21.02
aa	68.14	106.44	85.54
ad	20.84	22.88	21.68
da	19.82	22.51	20.94
dd	20.11	22.47	21.08
a	47.93	75.08	60.27
d	19.90	22.44	20.95
	52.91	66.51	58.73

Tabla 5.4: Resultados de la Lavadora Niveles 5 y 4.

Nodo	RMSE Entrenamiento	RMSE Validación	RMSE Optimización
aaaaa	687.91	771.12	741.81
aaaad	112.44	103.47	108.94
aaada	58.85	52.24	56.29
aaadd	73.04	70.35	72.51
aadaa	35.39	30.84	33.65
aadad	40.65	35.87	38.80
aadda	51.70	46.63	49.73
aaddd	48.32	42.02	45.91
adaaa	13.60	11.25	12.71
adaad	14.18	11.83	13.29
adada	18.67	16.20	17.74
adadd	16.67	14.24	15.74
addaa	33.21	28.67	31.48
addad	30.32	26.00	28.67
addda	21.38	18.28	20.20
adddd	26.19	22.42	24.76
daaaa	11.95	9.58	11.07
daaad	11.34	9.41	10.61
daada	11.01	9.33	10.37
daadd	11.38	9.87	10.80
dadaa	12.75	11.91	12.43
dadad	13.46	12.94	13.25
dadda	11.88	10.78	11.46
daddd	11.84	10.93	11.48
ddaaa	13.81	11.53	12.95
ddaad	14.01	12.08	13.27
ddada	14.85	13.32	14.26
ddadd	14.77	12.95	14.07
dddaa	12.45	11.59	12.11
dddad	13.98	12.99	13.59
ddddа	14.81	13.12	14.15
ddddd	13.79	12.79	13.40

Tabla 5.5: Resultados de la Lavadora Niveles 3, 2 y 1.

Nodo	RMSE Entrenamiento	RMSE Validación	RMSE Optimización
aaaa	486.11	550.03	522.77
aaad	66.24	61.99	64.88
aada	38.13	33.47	36.33
aadd	50.00	44.44	47.85
adaa	13.89	11.54	13.00
adad	17.69	15.26	16.76
adda	31.80	27.39	30.10
addd	23.90	20.46	22.59
daaa	11.65	9.50	10.84
daad	11.20	9.60	10.59
dada	13.08	12.37	12.81
dadd	11.87	10.87	11.48
ddaa	13.89	11.82	13.10
ddad	14.81	13.15	14.16
ddda	13.24	12.31	12.87
dddd	14.31	12.96	13.78
aaa	336.08	404.78	368.97
aad	44.47	39.40	42.46
ada	15.91	13.55	15.01
add	28.17	24.20	26.62
daa	11.43	9.56	10.72
dad	12.52	11.66	12.17
dda	14.36	12.50	13.65
ddd	13.79	12.65	13.34
aa	234.15	292.68	261.07
ad	22.87	19.61	21.61
da	11.99	10.67	11.48
dd	14.10	12.60	13.50
a	168.86	205.76	184.19
d	13.42	12.04	12.56
	140.47	158.06	134.34

Tabla 5.6: Resultados del Microondas Niveles 5 y 4.

Nodo	RMSE Entrenamiento	RMSE Validación	RMSE Optimización
aaaaa	509.08	659.54	640.29
aaaad	339.53	343.18	341.03
aaada	123.07	130.23	125.99
aaadd	216.41	219.51	218.31
aadaa	76.58	63.05	71.48
aadad	80.95	79.03	80.17
aadda	95.23	101.60	97.84
aaddd	93.79	95.99	94.75
adaaa	39.76	35.01	37.94
adaad	36.17	34.02	35.30
adada	39.20	39.36	39.24
adadd	36.73	35.68	36.35
addaa	69.42	56.49	64.56
addad	58.42	56.71	57.75
addda	44.46	43.98	44.29
adddd	54.64	53.95	54.34
daaaa	24.52	23.64	24.18
daaad	23.91	24.97	24.34
daada	25.83	25.56	25.72
daadd	24.27	24.29	24.24
dadaa	26.93	27.02	26.96
dadad	25.55	27.13	26.19
dadda	26.49	25.15	25.96
daddd	25.61	24.55	25.19
ddaaa	38.41	33.96	36.70
ddaad	32.32	30.93	31.77
ddada	31.15	30.66	30.96
ddadd	30.73	30.62	30.70
dddaa	27.12	27.43	27.23
dddad	27.02	30.15	28.31
ddddа	30.58	30.34	30.53
ddddd	26.82	28.23	27.39

Tabla 5.7: Resultados del Microondas niveles 3, 2 y 1.

Nodo	RMSE Entrenamiento	RMSE Validación	RMSE Optimización
aaaa	406.67	525.11	501.75
aaad	176.04	180.47	178.23
aada	78.79	71.49	75.94
aadd	94.51	98.82	96.31
adaa	37.99	34.53	36.64
adad	37.97	37.54	37.82
adda	64.18	56.62	61.25
addd	49.80	49.22	49.57
daaa	24.20	24.31	24.26
daad	25.18	25.02	25.00
dada	26.25	27.09	26.57
dadd	26.05	24.85	25.57
ddaa	35.48	32.48	34.31
ddad	30.95	30.66	30.82
ddda	27.07	28.82	27.78
dddd	28.74	29.29	28.97
aaa	313.75	390.87	378.00
aad	87.01	86.24	86.72
ada	37.98	36.07	37.23
add	57.46	53.07	55.72
daa	24.74	24.70	24.63
dad	26.15	25.99	26.08
dda	33.30	31.60	32.62
ddd	27.92	29.06	28.38
aa	224.92	295.31	269.48
ad	48.70	45.37	47.38
da	25.44	25.34	25.36
dd	30.71	30.34	30.57
a	161.40	206.63	193.96
d	28.20	27.95	28.09
	119.41	152.39	137.73

Tabla 5.8: Resultados de la Secadora Niveles 5 y 4.

Nodo	RMSE Entrenamiento	RMSE Validación	RMSE Optimización
aaaaa	198.31	289.46	238.98
aaaad	50.66	54.97	52.43
aaada	27.62	29.65	28.45
aaadd	35.96	39.10	37.24
aadaa	21.55	23.38	22.30
aadad	22.87	24.86	23.68
aadda	25.99	28.02	26.82
aaddd	24.43	26.19	25.15
adaaa	20.54	22.64	21.41
adaad	20.41	22.66	21.34
adada	20.51	22.71	21.42
adadd	20.63	22.86	21.55
addaa	21.18	23.03	21.94
addad	21.43	23.19	22.15
addda	20.74	22.65	21.53
adddd	21.12	22.96	21.87
daaaa	19.35	22.71	20.76
daaad	19.94	22.42	20.97
daada	21.24	23.11	22.00
daadd	20.60	22.68	21.46
dadaa	19.62	22.36	20.76
dadad	19.82	22.27	20.84
dadda	19.75	22.55	20.91
daddd	19.94	22.39	20.95
ddaaa	20.15	22.65	21.19
ddaad	20.16	22.53	21.14
ddada	20.27	22.61	21.24
ddadd	20.29	22.65	21.27
dddaa	20.16	22.58	21.16
dddad	20.08	22.25	20.97
ddddа	20.15	22.31	21.04
ddddd	19.80	22.20	20.79

Tabla 5.9: Resultados de la Secadora Niveles.

Nodo	RMSE Entrenamiento	RMSE Validación	RMSE Optimización
aaaa	142.18	208.01	171.57
aaad	31.98	34.72	33.10
aada	22.24	24.16	23.03
aadd	25.26	27.24	26.07
adaa	20.49	22.67	21.39
adad	20.63	22.85	21.54
adda	21.31	23.14	22.06
addd	20.95	22.83	21.72
daaa	19.64	22.62	20.88
daad	20.21	22.54	21.17
dada	19.72	22.35	20.81
dadd	19.84	22.52	20.95
ddaa	20.15	22.62	21.17
ddad	20.28	22.62	21.25
ddda	20.13	22.42	21.08
dddd	19.98	22.27	20.93
aaa	90.55	143.00	114.45
aad	23.71	25.75	24.55
ada	20.54	22.76	21.46
add	21.14	23.01	21.91
daa	19.91	22.58	21.02
dad	19.77	22.44	20.88
dda	20.20	22.62	21.20
ddd	20.07	22.36	21.02
aa	68.14	106.44	85.54
ad	20.84	22.88	21.68
da	19.82	22.51	20.94
dd	20.11	22.47	21.08
a	47.93	75.08	60.27
d	19.90	22.44	20.95
	52.91	66.51	58.73

Tabla 5.10: Desviación Estándar del error.

Dispositivo	Desviación Estándar
Refrigerador	108.90
Lavadora	291.81
Microondas	260.06
Secadora	254.80

Capítulo 6

Conclusiones y recomendaciones

En el presente capítulo se encuentran las conclusiones y recomendaciones en base a los resultados obtenidos en el anterior capítulo. Además, se sugiere los trabajos futuros que se podrían desarrollar para mejorar el sistema propuesto.

En este trabajo se desarrolló una metodología para la desagregación de las señales de potencia principal de una casa, con el objetivo de obtener las señales de potencia consumida por los electrodomésticos conectados en dicha casa. Para cumplir con ese objetivo, la arquitectura utiliza modelos WGAN-GP, obteniendo como resultado las señales de potencia consumidas por la lavadora, refrigeradora, secadora y microondas de una vivienda.

Los resultados obtenidos en cada una de las etapas de desarrollo permiten llegar a las siguientes conclusiones:

- El trabajo realizado resuelve la tarea de desagregación energética en un hogar de manera no intrusiva. Es decir, a partir de la señal de potencia consumida de la casa medida en la acometida principal, se obtiene satisfactoriamente las señales de potencia consumidas por el refrigerador, la lavadora, la secadora y el microondas de dicho hogar. Además, este modelo capaz de resolver la tarea NILM está montado en un servidor como aplicación web, disponible en todo momento en la nube con un entorno simple y amigable.

- La metodología propuesta es un modelo completamente innovador; al involucrar la transformada de wavelet en el preprocesamiento de los datos, se extraen características del tiempo y la frecuencia de la señal, necesarias por la naturaleza misma de las señales de potencia con información tanto en alta como en baja frecuencia. Además, el uso de un modelo WGAN-GP para cada uno de los nodos de descomposición de la señal aporta a que el aprendizaje del sistema no dependa del resultado de un solo algoritmo de aprendizaje artificial. Finalmente, el proceso de optimización escoge adecuadamente los mejores resultados de estos modelos y permite reconstruir la señal de potencia con la mejor fidelidad posible.

- Todas las herramientas matemáticas e informáticas utilizadas en esta arquitectura significan un coste computacional alto. El proceso de descomposición de wavelet, el proceso de entrenamiento de un modelo WGAN-GP para cada nodo, el proceso de composición de la señal utilizando la wavelet inversa y finalmente la optimización de los resultados son tareas que exigen recursos informáticos de altas capacidades. Sin embargo, una vez listo el modelo final, se puede implementar como una función matemática para la resolución de la tarea NILM de manera relativamente rápida y en cualquier plataforma de computación de datos; por ejemplo, en un servidor como aplicación web.

- Los resultados analizados en este documento muestran que el error es menor en las componentes de frecuencias altas mientras que aumenta en las componentes de baja frecuencia de la señal. Por lo tanto, al reconstruir la señal desde su árbol de descomposición, el error de las frecuencias bajas se va amortiguando al procesarlas con las componentes de alta frecuencia. Esto da como resultado que la señal de potencia reconstruida es confiable en sus componentes de cambios rápidos, sin embargo no sigue fielmente la señal medida en sus cambios lentos; por esta razón los valores de RMSE en el trabajo son grandes.

- El proceso de optimización por enjambre de partículas mejoró notablemente los resultados, el RMSE disminuyo significativamente. El algoritmo de optimización utilizado otorga una ponderación a cada uno de los nodos de reconstrucción, de esta manera utiliza solamente los nodos que más contribuyen a disminuir el error y les da menos valor a los nodos que no tienen información relevante para la reconstrucción.

- Un enfoque aplicativo

A continuación se mencionan algunas recomendaciones para el desarrollo de trabajos similares o futuros:

- Para este trabajo se ha utilizado la información de una base de datos para todas las fases. El fin aplicativo de el desarrollo propuesto es usarlo en tiempo real en cualquier vivienda.Para lograr este fin, se propone crear un dispositivo capaz de tomar las muestras medidas por los medidores digitales instalados en la acometida principal de la casa, y enviarlas a través de internet a la aplicación web cargado en la nube.

- Se propone mejoras en la aplicación web, el diseño del front-end es básico y se puede recrear un mejor entorno, más interactivo, intuitivo y con una mejor animación. Además, la metodología de como se consigue los datos de entrada no es la óptima. Aprovechando que las acometidas de la mayoría de las casas ya disponen de un medidor digital, en un trabajo futuro se puede implementar un sistema capaz de tomar los datos en tiempo real y cargarlos en el servicio web para su procesamiento.

- Un malestar actual es el tiempo de respuesta de la aplicación web, lo que no permitiría que trabaje en tiempo real verdadero, sino trabaje con un pequeño retardo. En un futuro trabajo se propone que se

utilice un servidor de mayor capacidades para solventar este tiempo de respuesta.

Bibliografía

M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan, 2017.

D. Bau, J.-Y. Zhu, J. Wulff, W. Peebles, H. Strobelt, B. Zhou, and A. Torralba. Seeing what a gan cannot generate. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 4502–4511, 2019.

C. Burrus, R. Gopinath, and H. Guo. Introduction to wavelets and wavelet transform—a primer. Recherche, 67, 2 1998.

P. Busch, T. Heinonen, and P. Lahti. Heisenberg's uncertainty principle. Physics Reports, 452(6):155–176, 2007. ISSN 0370-1573. doi: https://doi.org/10.1016/j.physrep.2007.05.006. URL https://www.sciencedirect.com/science/article/pii/S0370157307003481.

Ş. Cobzaş, R. Miculescu, and A. Nicolae. Lipschitz functions, volume 2241. Springer, 2019.

A. Cominola, M. Giuliani, D. Piga, A. Castelletti, and A. E. Rizzoli. A hybrid signature-based iterative disaggregation algorithm for non-intrusive load monitoring. Applied Energy, 185:331–344, 1 2017. ISSN 0306-2619. doi: 10.1016/J.APENERGY.2016.10.040.

J. R. Demey, L. Pla, J. L. Vicente-Villardón, J. Di Rienzo, and F. Casanoves. Medidas de distancia y similitud. Valoración y análisis de la diversidad funcional y su relación con los servicios ecosistémicos, 384:47–59, 2011.

D. Edwards. On the kantorovich–rubinstein theorem. Expositiones Mathematicae, 29(4):387–398, 2011. ISSN 0723-0869. doi: https://doi.org/10.1016/j.exmath.2011.06.005. URL https://www.sciencedirect.com/science/article/pii/S0723086911000430.

J. M. Gillis and W. G. Morsi. Non-intrusive load monitoring using semi-supervised machine learning and wavelet design. IEEE Transactions on Smart Grid, 8:2648–2655, 11 2017. ISSN 1949-3061. doi: 10.1109/TSG.2016.2532885.

I. Goodfellow. Nips 2016 tutorial: Generative adversarial networks, 2017. URL https://arxiv.org/abs/1701.00160.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks, 2014.

I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans, 2017.

S. Hitawala. Comparative study on generative adversarial networks, 2018.

C. A. Holt and A. E. Roth. The nash equilibrium: A perspective. Proceedings of the National Academy of Sciences, 101(12):3999–4002, 2004. doi: 10.1073/pnas.0308738101. URL https://www.pnas.org/doi/abs/10.1073/pnas.0308738101.

S. S. Hosseini, K. Agbossou, S. Kelouwani, and A. Cardenas. Non-intrusive load monitoring through home energy management systems: A comprehensive review. Renewable and Sustainable Energy Reviews, 79: 1266–1274, 2017. ISSN 1364-0321. doi: https://doi.org/10.1016/j.rser.2017.05.096. URL http://www.sciencedirect.com/science/article/pii/S1364032117307359.

T. Y. Ji, L. Liu, T. S. Wang, W. B. Lin, M. S. Li, and Q. H. Wu. Non-intrusive load monitoring using additive factorial approximate maximum <italic>a posteriori</italic>based on iterative fuzzy <inline-formula><tex-math notation="latex»c </tex-math></inline-formula>-means. IEEE Transactions on Smart Grid, 10:6667–6677, 2019. doi: 10.1109/TSG.2019.2909931.

M. Kahl, A. U. Haq, T. Kriechbaumer, and H.-A. Jacobsen. Whited-a worldwide household and industry transient energy data set. In 3rd International Workshop on Non-Intrusive Load Monitoring, pages 1–4, 2016.

J. Kelly and W. Knottenbelt. The uk-dale dataset, domestic appliance-level electricity demand and whole-house demand from five uk homes. Scientific

Data, 2(1):150007, Mar 2015. ISSN 2052-4463. doi: 10.1038/sdata.2015.7. URL https://doi.org/10.1038/sdata.2015.7.

J. Kim, T.-T.-H. Le, and H. Kim. Nonintrusive load monitoring based on advanced deep learning and novel signature. Computational Intelligence and Neuroscience, 2017:4216281, 2017. ISSN 1687-5265. doi: 10.1155/2017/4216281. URL https://doi.org/10.1155/2017/4216281.

J. Z. Kolter and M. J. Johnson. Redd: A public data set for energy disaggregation research. In Workshop on data mining applications in sustainability (SIGKDD), San Diego, CA, volume 25, pages 59–62, 2011.

O. Krystalakos, C. Nalmpantis, and D. Vrakas. Sliding window approach for online energy disaggregation using artificial neural networks. 2018.

A. Kumar and P. Bhattacharjee. Non-intrusive appliance identification for energy disaggregation of indian households–an use case for energy informatics. pages 239–242, 2018. doi: 10.1109/iSES.2018.00059.

T. Le, J. Kim, and H. Kim. Classification performance using gated recurrent unit recurrent neural network on energy disaggregation. volume 1, pages 105–110, 2016. doi: 10.1109/ICMLC.2016.7860885.

M. Liang, Y. Meng, N. Lu, D. Lubkeman, and A. Kling. Hvac load disaggregation using low-resolution smart meter data. pages 1–5, 2 2019. doi: 10.1109/ISGT.2019.8791578.

S. G. Mallat. A theory for multiresolution signal decomposition: the wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11:674–693, 1989. doi: 10.1109/34.192463.

M. N. Meziane, P. Ravier, G. Lamarque, J. L. Bunetel, and Y. Raingeaud. High accuracy event detection for non-intrusive load monitoring. pages 2452–2456, 2017. doi: 10.1109/ICASSP.2017.7952597.

A. Miyasawa, M. Matsumoto, Y. Fujimoto, and Y. Hayashi. Energy disaggregation based on semi-supervised matrix factorization using feedback information from consumers. pages 1–6, 2017. doi: 10.1109/ISGTEurope.2017.8260211.

L. Nambiar, V. KumarGopal, and R. Pradeep. Energy disaggregation for nilm applications using shallow and deep networks. pages 1–6, 2019. doi: 10.1109/I2CT45611.2019.9033955.

W. K. Ngui, M. S. Leong, L. M. Hee, and A. M. Abdelrhman. Wavelet analysis: mother wavelet selection methods. Applied mechanics and materials, 393:953–958, 2013.

K. Osathanunkul and K. Osathanunkul. Different sampling rates on neural nilm energy disaggregation. pages 318–321, 2019. doi: 10.1109/ECTI-NCON.2019.8692281.

Y. Pan, K. Liu, Z. Shen, X. Cai, and Z. Jia. Sequence-to-subsequence learning with conditional gan for power disaggregation. pages 3202–3206, 5 2020. doi: 10.1109/ICASSP40776.2020.9053947.

T. Picon, M. N. Meziane, P. Ravier, G. Lamarque, C. Novello, J.-C. L. Bunetel, and Y. Raingeaud. Cooll: Controlled on/off loads library, a public dataset of high-sampled electrical signals for appliance identification. arXiv preprint arXiv:1611.05803, 2016.

D. Renaux, R. Linhares, F. Pottker, A. Lazzaretti, C. Lima, A. Coelho Neto, and M. Campaner. Designing a novel dataset for non-intrusive load monitoring. In 2018 VIII Brazilian Symposium on Computing Systems Engineering (SBESC), pages 243–249, 2018. doi: 10.1109/SBESC.2018.00045.

S. Singh and A. Majumdar. Deep sparse coding for non–intrusive load monitoring. IEEE Transactions on Smart Grid, 9:4669–4678, 2018. ISSN 1949-3061. doi: 10.1109/TSG.2017.2666220.

T. Sirojan, B. T. Phung, and E. Ambikairajah. Deep neural network based energy disaggregation. pages 73–77, 2018. doi: 10.1109/SEGE.2018.8499441.

M. Sun, F. M. Nakoty, Q. Liu, X. Liu, Y. Yang, and T. Shen. Non-intrusive load monitoring system framework and load disaggregation algorithms: A survey. pages 284–288, 2019. doi: 10.1109/ICAMechS.2019.8861646.

J. Tian, Y. Wu, S. Liu, and P. Liu. Residential load disaggregation based on resident behavior learning and neural networks. pages 1–5, 2017. doi: 10.1109/EI2.2017.8245665.

A. Tongta and K. Chooruang. Long short-term memory (lstm) neural networks applied to energy disaggregation. pages 1–4, 2020. doi: 10.1109/iEECON48109.2020.229559.

S. Welikala, N. Thelasingha, M. Akram, P. B. Ekanayake, R. I. Godaliyadda, and J. B. Ekanayake. Implementation of a robust real-time non-intrusive load monitoring solution. Applied Energy, 238:1519–1529, 2019. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2019.01.167. URL http://www.sciencedirect.com/science/article/pii/S0306261919301849.

A. You, J. K. Kim, I. H. Ryu, and T. K. Yoo. Application of generative adversarial networks (gan) for ophthalmology image domains: a survey. Eye and Vision, 9(1):6, Feb 2022. ISSN 2326-0254. doi: 10.1186/s40662-022-00277-3. URL https://doi.org/10.1186/s40662-022-00277-3.

C. Zhang, M. Zhong, Z. Wang, N. Goddard, and C. Sutton. Sequence-to-point learning with neural networks for nonintrusive load monitoring. arXiv e-prints, page arXiv:1612.09106, 2016.

S. Zhang and Y. Zhang. Introduction to game theory. Chinese Science Bulletin, 48(9):841–846, 2003.

I. H. Çavdar and V. Faryad. New design of a supervised energy disaggregation model based on the deep neural network for a smart grid, 2019. ISSN 19961073.

Glosario

GAN Generative Adversarial Network .

NILM Non-Intrusive Load Monitoring .

RMSE Root Mean Squared Error .

WGAN Wasserstein Generative Adversarial Network .

WGAN-GP Wasserstein Generative Adversarial Network with Gradient Penalty .