



UNIVERSIDAD PONTIFICIA DE COMILLAS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

DESAGREGACIÓN DEL CONSUMO ELÉCTRICO E IDENTIFICACIÓN DE DISPOSITIVOS EN HOGARES EN EL CONTEXTO DE MONITORIZACIÓN NO INTRUSIVA (NILM) UTILIZANDO DEEP LEARNING

Autor: Victor Guizien Martin

Directores : Eugenio Francisco Sánchez Ubeda y Andrea Veiga Santiago

Madrid

Enero 2021

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

**DESAGREGACIÓN DEL CONSUMO ELÉCTRICO E IDENTIFICACIÓN DE
DISPOSITIVOS EN HOGARES EN EL CONTEXTO DE MONITORIZACIÓN NO
INTRUSIVA (NILM) UTILIZANDO DEEP LEARNING**

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2020/21 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.



Fdo.: Victor Guizien Martin

Fecha: 10/02/2021

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO



Fdo.: Eugenio Sánchez Úbeda

Fecha: 10/02/2021



Fdo.: Andrea Veiga Santiago

Fecha: 10/02/2021

Agradecimientos

Me gustaría agradecer ante todo a mis directores, Eugenio Sánchez Úbeda y Andrea Veiga Santiago, por el apoyo y ayuda que me han dado durante toda la realización de este trabajo. En primer lugar y sin lugar a duda por el conocimiento, ideas y soluciones que han aportado a lo largo de todos los meses de trabajo, siempre queriendo llegar un poco más lejos en el desarrollo de las soluciones y queriendo explorar hasta el final los distintos caminos que se podían tomar para resolver los retos que se nos presentaban. Gracias por ayudarme a enfocar las cosas de forma sencilla y con sentido, por ayudarme a aterrizar las ideas, a acotar tiempos, a organizarme y en definitiva, a sacar este proyecto adelante.

Pero también, y me atrevería a decir que más importante, agradecerles por la infinita paciencia que han demostrado tener conmigo, por explicarme las cosas una y otra vez cuando alguna que otra mañana estaba más espeso de la cuenta y por haber hecho tanto esfuerzo en sacar tiempo en sus ajustados horarios, para poder seguir cumpliendo con los seguimientos, para resolver dudas, debatir ideas y comentar resultados. Agradecerles también la atmósfera distendida, cómoda y cercana que se crea a su alrededor, fruto de sus personalidades humildes, amables, genuinas y sobre todo, profesionales.

Gracias a Eugenio en particular por explicarme las cosas con tanta claridad y paciencia, por ofrecer un sinfín de ideas, de perspectivas y de análisis que han sido y son realmente fascinantes, inspiradoras y motivacionales, y que en más de una ocasión disiparon las dudas que empezaban a mermar las ganas de seguir luchando con la misma intensidad.

También me gustaría agradecer a mi familia, que siempre ha estado a mi lado y han hecho todo lo posible para liberarme al máximo de tareas y responsabilidades para que pudiera centrarme en desarrollar este TFM en las mejores condiciones posibles, aunque no siempre haya sido fácil para ellos por tener que lidiar con mis caras largas, mi cansancio aparentemente infinito y mis repentinos cambios de humor.

Agradecer también a mis compañeros de carrera, por el intercambio de ideas, por el apoyo moral en tiempos de crisis, por las risas, por los piques, por la piña que hemos formado todos juntos, por cerrar una etapa entera de nuestras vidas que empezamos juntos hace ya casi 7 años. Etapa que nos hubiese gustado a todos cerrar juntos, reunidos, entre risas y abrazos, y que sin duda hubiese sumado una de esas anécdotas de la que cada persona tiene una versión, a las que ya estábamos acostumbrados y que tanto echamos de menos.



UNIVERSIDAD PONTIFICIA DE COMILLAS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

DESAGREGACIÓN DEL CONSUMO ELÉCTRICO E IDENTIFICACIÓN DE DISPOSITIVOS EN HOGARES EN EL CONTEXTO DE MONITORIZACIÓN NO INTRUSIVA (NILM) UTILIZANDO DEEP LEARNING

Autor: Victor Guizien Martin

Directores : Eugenio Francisco Sánchez Ubeda y Andrea Veiga Santiago

Madrid

Enero 2021

DESAGREGACIÓN DEL CONSUMO ELÉCTRICO E IDENTIFICACIÓN DE DISPOSITIVOS EN HOGARES EN EL CONTEXTO DE MONITORIZACIÓN NO INTRUSIVA (NILM) UTILIZANDO DEEP LEARNING

Autor: Victor Guizien Martin

Directores: Eugenio Sánchez Úbeda y Andrea Veiga Santiago

Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Este trabajo propone una metodología innovadora dentro del campo de NILM – *Non Intrusive Load Monitoring*. Dicha metodología cubre todas las actividades necesarias desde la preparación y tratamiento de datos hasta el entrenamiento y validación de los modelos de Machine Learning, siendo especialmente innovadora la primera parte del mismo. Con esta preparación de datos, y enfocando el problema como uno de clasificación, se ha diseñado una CNN que ofrece buenos resultados en el reconocimiento de electrodomésticos en el ámbito NILM.

Palabras clave: NILM, Deep Learning, MLP, CNN, Transfer Learning

1. Introducción

Este trabajo tiene como objetivo desarrollar una metodología basada en algoritmos de Machine Learning que permita identificar la utilización de un electrodoméstico en una vivienda a partir del consumo total de energía de la misma. La monitorización del consumo eléctrico tiene como objetivo principal optimizar el consumo de los distintos electrodomésticos de un hogar, reduciendo así el consumo de electricidad y el impacto medioambiental asociado [1]. Estos objetivos se pueden conseguir mostrando a los habitantes de un hogar el consumo de cada electrodoméstico, detectando posibles anomalías en dichos aparatos o incluso indicándoles cuándo sería más económico utilizarlos en función del precio de la electricidad.

En términos generales, la monitorización de energía es el proceso de identificar y adquirir medidas energéticas en un sistema dado [2]. Esta monitorización permite determinar el consumo y el estado de los aparatos de forma individual. Esta monitorización se puede hacer de forma intrusiva (Intrusive Load Monitoring – ILM) o no intrusiva (Non-Intrusive Load Monitoring – NILM).

En el campo de NILM, el problema general trata de desagregar la señal total del consumo eléctrico en los distintos dispositivos que estuvieran consumiendo en ese instante de tiempo. En este trabajo, se ha planteado el problema para tratar de indicar qué electrodomésticos estaban consumiendo (encendido o apagado) en un instante determinado, es decir, un problema de clasificación, en lugar de tratar de predecir el consumo exacto de cada electrodoméstico (problema de regresión).

2. Definición del proyecto

Este trabajo pretende avanzar en la investigación de técnicas de NILM, en este caso utilizando Deep Learning y proponiendo una idea novedosa en el tratamiento los datos de entrada de los modelos. El objetivo es conocer en cada instante de tiempo el estado de cada uno de los electrodomésticos (encendido o apagado), información que podrá utilizarse para multitud de usos como por ejemplo ofrecer recomendaciones a los consumidores y así mejorar sus hábitos de consumo o poner a su disposición productos personalizados por parte de las comercializadoras de electricidad.

Se realiza en un primer lugar un estudio de las bases de datos disponibles para después realizar el análisis exploratorio de los datos e investigar posibles incoherencias, errores o inconsistencias. Se eligen los electrodomésticos de los cuales se quieren predecir su estado (encendido/apagado) con los modelos y se elegirá una casa de referencia en base a la cual se realizarán las demás etapas del proyecto. Con estos datos definidos, se empieza la preparación y planteamiento del problema, incorporando una idea novedosa basada en imágenes para tratar con los datos de consumo eléctrico. Finalmente, se evalúan los resultados obtenidos con un modelo MLP de referencia y con una CNN.

3. Descripción del modelo

La idea principal en torno a la que se centra este trabajo es una representación de los datos de consumo eléctrico especialmente diseñada para alimentar una CNN, a la que se ha denominado ‘vizgrama’. Un ejemplo de este se puede ver en la Figura 1.

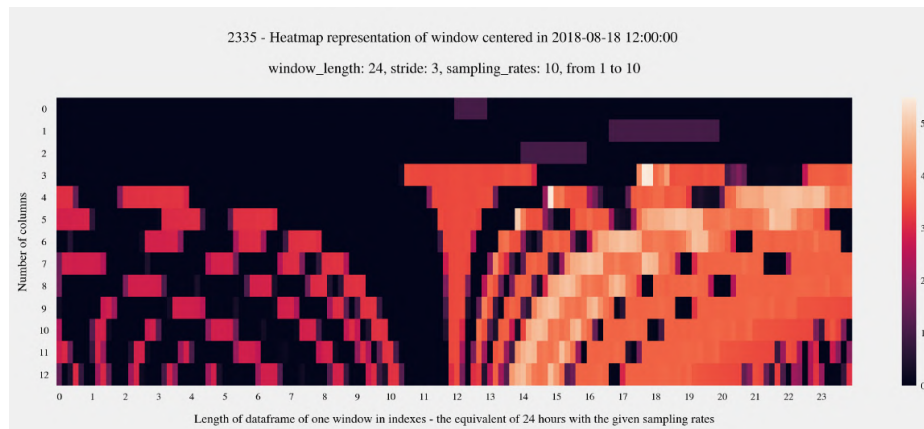


Figura 1 – Ejemplo de vizgrama

Esta imagen representa una matriz que contiene en las 3 primeras filas información temporal (hora del día, día de la semana y mes del año) y en las 10 siguientes filas la información sobre el consumo eléctrico total que ha ocurrido un determinado instante antes y después (en este caso, 12h antes y 12h después) muestreada a distintos intervalos (en este caso, de 1 minuto a 10 minutos). La figura 2 y figura 3 muestran la equivalencia del vizgrama con la señal tradicional de consumo eléctrico.

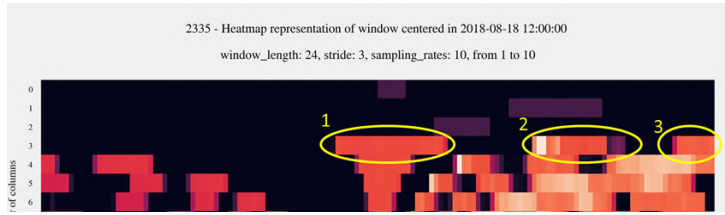


Figura 2 – Identificación de picos de potencia en un vizgrama en el primer muestreo

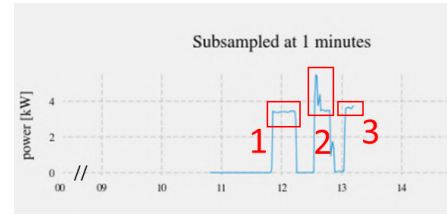


Figura 3 – Identificación de picos de potencia en la señal del total en el primer muestreo

Se construye un vizgrama por cada instante de tiempo que se quiere muestrear y del que se quiere predecir qué electrodomésticos estaban activados.

Una vez se tienen los vizgramas contruidos, se entrenan dos modelos, un MLP básico de referencia y un modelo CNN que aprovecha esta representación visual.

4. Resultados

Se ha estudiado la activación de 5 electrodomésticos: el aire acondicionado (air1), la lavadora (clotheswasher1), el lavaplatos (dishwasher1), microondas (microwave1) y el circuito de la cocina al que van enchufados pequeños electrodomésticos (kitchenapp1).

La tabla 1 resume los resultados obtenidos por los mejores modelos de ambas arquitecturas, así como el ratio de no-información (proporción de datos según la clase mayoritaria), a modo de referencia.

Tabla 1 – Resumen de resultados obtenidos (accuracy) MLP vs. CNN vs. REF

		MLP (%)	CNN (%)	REF (%)
air1	Train accuracy	99.95	99.83	91.0
	Test accuracy	99.71	99.8	
clotheswasher1	Train accuracy	99.98	99.44	98.68
	Test accuracy	98.44	98.83	
dishwasher1	Train accuracy	99.83	99.76	96.854
	Test accuracy	98.31	99.45	
kitchenapp1	Train accuracy	99.98	99.86	99.647
	Test accuracy	99.54	99.43	
microwave1	Train accuracy	99.95	99.82	99.665
	Test accuracy	99.71	99.75	

Como se puede ver en esta tabla, los resultados obtenidos por ambos modelos son bastante similares y con resultados mejores al valor de REF. En el caso de los electrodomésticos de utilización más larga (aire acondicionado, lavadora y lavaplatos), los resultados obtenidos con ambos modelos son parecidos, teniendo en cuenta que la CNN tiene menor cantidad de parámetros, se puede decir que es el modelo superior para estos electrodomésticos.

Sin embargo, en el caso de los electrodomésticos de menor tiempo de utilización (microondas y circuito de cocina), la CNN tiene peores resultados que el MLP.

5. Conclusiones

Este trabajo presenta una forma novedosa de tratar y procesar los datos de consumo eléctrico que se usan en la resolución de problemas de NILM. Se ha construido el denominado vizgrama, una matriz que añade información sobre el consumo eléctrico que se ha realizado antes y después del instante en el que se quiere desagregar la señal del total. El vizgrama se construye a través de un conjunto de parámetros que lo hacen modulable y escalable, y, como se ha visto en los resultados, esta forma de procesar los datos permite a los modelos CNN obtener buenos resultados.

6. Referencias

- [1] Darby, S.: The effectiveness of feedback on energy consumption a review for DEFRA of the literature on metering, billing and direct displays. In: Environmental Change Institute University of Oxford, 22 April, pp. 1–21 (2006)
- [2] Abubakar, I., et al.: Application of load monitoring in appliances energy management a review. *Renew. Sustain. Energy Rev.* 67, 235–245 (2017)

ENERGY DISAGGREGATION AND APPLIANCE DETECTION IN THE CONTEXT OF NILM USING DEEP LEARNING

Author: Victor Guizien Martin

Supervisor: Eugenio Sánchez Úbeda y Andrea Veiga Santiago

Collaborating Entity: ICAI – Universidad Pontificia Comillas)

ABSTRACT

Research in the field of NILM (non-intrusive load monitoring) has seen an unprecedented surge in the past years mainly due to the increase in domestic household energy monitoring, finally making this alternative viable. Over these years, researchers have tried and implemented multiple techniques and algorithms that managed, with a certain degree of success, to perform energy disaggregation. However, these techniques are often very niche, do not generalize very well because of their complexity, or they simply do not perform all that great. This paper proposes the vizgram, a new approach on how to process and interpret energy data that serves as the model input, transforming it into an image that can later be fed to a Convolutional Neural Network, taking advantage of the already proven usefulness and remarkable performance of these types of architectures in image classification. The proposed approach has been applied with success to a real case consisting of five different appliances.

Keywords: NILM, Deep Learning, MLP, CNN, Transfer Learning

1. Introduction

This paper proposes a new methodology based in Machine Learning algorithms that is able to identify the activation of a given appliance in a domestic household based on the signal of the total electrical consumption. The main purpose of energy load monitoring is to optimize the electrical consumption of the different appliances being used in a house and thus, reducing said consumption and the associated greenhouse gas emissions [1]. These goals can be achieved by informing the household's occupants of how much energy each appliance consumes, detecting possible anomalies in the appliances' operations or by recommending timeframes where energy is cheaper.

Generally speaking, load monitoring is the process of identifying and acquiring electrical measures in a given system [2]. This load monitoring is able to determine electrical consumption and the individual appliances' state, and it can be done intrusively (Intrusive Load Monitoring or ILM) or non-intrusively (Non-Intrusive Load Monitoring or NILM).

In the field of NILM, the problem is trying to break down a households' aggregate electricity consumption into the individual appliances that were consuming in that instant of time. In this paper, instead of trying to predict the exact consumption of all appliances at a given instant (regression problem), we will try to detect if the appliances were on/off (classification problem).

2. Project definition

This work aims to advance in the investigation of NILM techniques, in this case using Deep Learning and proposing a novel idea in the treatment of the models' input data. The objective is to know, at a given instant, the status of each of appliance (on/off), information that can later be used for various purposes, such as offering recommendations to consumers and thus improving their consumption habits or offering them targeted products and electricity plans.

In the first place, we study the available databases and perform an exploratory analysis of the data to look for possible inconsistencies, errors or invalid data. We then select which appliances are relevant and adequate for this work, and choose a reference house, upon which will be carried out the next stages of the project. With this data defined, we can start preparing the problem. We propose a novel way of treating and processing the input data, which is based on the use of images that represent the electrical consumption of data. Finally, we evaluate the results obtained with our reference MLP model and a CNN.

3. Model description

The main idea behind this project is what we called a 'vizgram'. It is a visual representation of electrical consumption specifically designed to be fed to a CNN. An example of a vizgram can be seen in Figure 1.

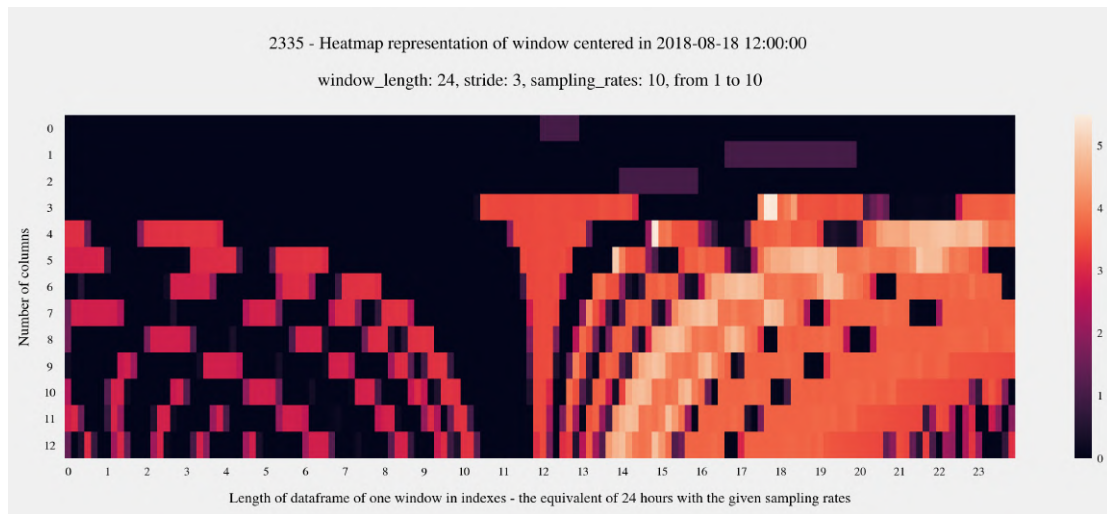


Figure 1 – Example of vizgram

This image is but a representation of a matrix that contains time-related information in the first three rows (hour of day, day of week and month of year) and in the next 10 rows information about the electrical consumption that occurred in a certain time *before* and *after* the sampled instant (in this example, 12 hours before and 12 hours after). This information is sampled at different intervals (in this example, 1 minute to 10 minutes). Figure 2 and Figure 3 show the equivalence between the traditional electrical consumption signal and the vizgram.



Figure 2 – Power peak identification in a vizgram in the first sampling rate

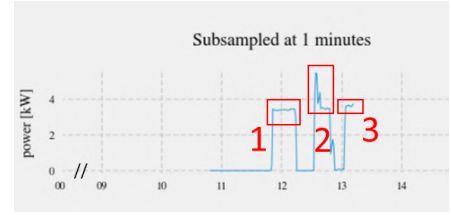


Figure 3 – Power peak identification in signal representation

One vizgram is constructed for each sampled instant where we want to identify what appliances were on. Once we have all the vizgrams built, we develop and train two models. First, a basic MLP that will act as a reference model, and a CNN model that takes advantage of this visual representation.

4. Results

We studied the activation of 5 different appliances: the air conditioning (air1), the clotheswasher (clotheswasher1), the dishwasher (dishwasher1), the microwave (microwave1) and the kitchen circuit, where smaller appliances are connected (kitchenapp1).

Table 1 sums up the results obtained by the best models obtained in both architectures, as well as the no information rate (data proportion of the most common class).

Table 1 – Summary of accuracy obtained in MLP and CNN vs REF value

		MLP (%)	CNN (%)	REF (%)
air1	Train accuracy	99.95	99.83	91.0
	Test accuracy	99.71	99.8	
clotheswasher1	Train accuracy	99.98	99.44	98.68
	Test accuracy	98.44	98.83	
dishwasher1	Train accuracy	99.83	99.76	96.854
	Test accuracy	98.31	99.45	
kitchenapp1	Train accuracy	99.98	99.86	99.647
	Test accuracy	99.54	99.43	
microwave1	Train accuracy	99.95	99.82	99.665
	Test accuracy	99.71	99.75	

As we can see in this table, results obtained by both models are similar and offer great performance when compared to the REF value. In the case of bigger and longer activated appliances (AC, clotheswasher and dishwasher), results obtained by both models are comparable, and, given that the CNN architecture has less trained parameters, we can say that it is superior in this case.

However, in the case of smaller appliances (microwave and kitchen circuit), the results of the CNN are worse than those of the MLP.

5. Conclusions

This paper offers a novel way of treating and processing the data used for the resolution of problems in the field of NILM. We constructed the vizgram, a matrix that adds information about the electrical consumption that happened before and after the instant where we want to predict what appliances were on or off. The vizgram is constructed with a set of parameters that make it modulable and scalable, and, as we saw in the results, this way of processing the data gives good results when fed to CNN models.

6. References

- [1] Darby, S.: The effectiveness of feedback on energy consumption a review for DEFRA of the literature on metering, billing and direct displays. In: Environmental Change Institute University of Oxford, 22 April, pp. 1–21 (2006)
- [2] Abubakar, I., et al.: Application of load monitoring in appliances energy management a review. *Renew. Sustain. Energy Rev.* 67, 235–245 (2017)

Índice de la memoria

Capítulo 1. Introducción.....	10
1.1 Estado del arte.....	12
1.2 Motivación.....	14
1.3 Objetivos del proyecto.....	15
1.3.1 Elección de base datos.....	15
1.3.2 Preparación, procesado y análisis de datos.....	15
1.3.3 Planteamiento del problema.....	15
1.3.4 Evaluación del problema con modelo de referencia.....	16
1.3.5 Evaluación del problema con Deep Learning.....	16
1.4 Alienación con los objetivos de desarrollo sostenible.....	16
1.5 Metodología de trabajo y recursos.....	17
Capítulo 2. Descripción del problema.....	18
2.1 Ejemplo ilustrativo del problema.....	19
Capítulo 3. Metodología.....	25
3.1 Esquema general del proceso.....	25
3.2 Preparación del problema.....	26
3.2.1 Elección del Dataset.....	26
3.3 Dataset seleccionado.....	32
3.3.1 Análisis exploratorio.....	32
3.3.2 Exploración de una casa de Texas.....	39
3.4 Vizgrama.....	43
3.4.1 Concepto.....	44
3.4.2 Parámetros.....	44
3.5 Variable de salida.....	52
3.6 Procesado de datos.....	54
3.7 Preparación de conjuntos de entrenamiento.....	60
3.8 Identificación y ajuste de modelo MLP.....	60
3.9 Identificación y ajuste de modelo CNN.....	63

Capítulo 4. Casos de uso	67
4.1 Resultados MLP	67
4.1.1 Evaluación.....	68
4.2 Resultados CNN.....	72
4.2.1 Evaluación.....	74
4.2.2 Transfer Learning	78
4.2.3 Feature maps y filtros	82
4.3 Comparativa entre modelos	85
4.3.1 Cantidad de minutos encendidos	86
4.3.2 Cantidad de activaciones de electrodomésticos	91
4.3.3 Tiempo de activación medio.....	95
Capítulo 5. Conclusiones y trabajos futuros	98
Capítulo 6. Referencias.....	100
ANEXO A – Resultados MLP.....	103
ANEXO B – Resultados CNN.....	107
ANEXO C – Resultados CNN segunda vivienda	111
ANEXO D – Resultados CNN segunda vivienda y transfer learning	115
ANEXO E – Comparativa entre modelos.....	119
ANEXO F – Iteración de modelos CNN	135
6.1 CNN - 3.2.1	135
6.2 CNN - 3.2.2.....	136
6.3 CNN - 3.2.3.....	137
6.4 CNN - 3.2.4.....	138
6.5 CNN - 3.2.5.....	139
6.6 CNN - 3.2.6.....	140
6.7 CNN - 3.2.7.....	141
6.8 CNN - 3.2.8.....	142

6.9 CNN - 3.2.9.....	143
6.10 CNN - 3.2.10.....	144
6.11 CNN - 3.2.11.....	145
6.12 CNN - 3.2.12.....	146
6.13 CNN - 3.2.13.....	147
6.14 CNN - 3.2.14.....	148
6.15 CNN - 3.2.15.....	149
6.16 CNN - 3.2.16.....	150
6.17 CNN - 3.2.17.....	151
6.18 CNN - 3.2.18.....	152
6.19 CNN - 3.2.19.....	153
6.20 CNN - 3.2.20.....	154
6.21 CNN - 3.2.21.....	155
6.22 CNN - 3.2.22.....	156
Anexo G – Feature Maps.....	157
air1.....	157
clotheswaasher1.....	160
dishwasher1.....	163
kitchenapp1.....	166
microwave1.....	169

Índice de figuras

Figura 1 – Ejemplo de consumo de un hogar	19
Figura 2 – Ejemplo de consumo de un hogar con electrodomésticos 'típicos'	20
Figura 3 – Ejemplo de consumo de un hogar con electrodomésticos 'típicos' + señal del total	21
Figura 4 – Ejemplo de consumo de un hogar con electrodomésticos 'típicos' + señal del total creada	22
Figura 5 – Ejemplo de consumo de un hogar con electrodomésticos 'típicos', solo la señal del total	22
Figura 6 – Visualización e interpretación de la variable de entrada en un problema NILM	23
Figura 7 – Regresión vs. Clasificación	24
Figura 8 – Esquema general de la metodología de trabajo	25
Figura 9 – Resumen de datos disponibles del dataset Dataport [30]	27
Figura 10 – Resumen de datos disponibles del dataset Dataport [30]	32
Figura 11 – Nueva York – Número de hogares en función de todos los electrodomésticos	34
Figura 12 – Nueva York – Número de hogares en función de los electrodomésticos y conteniendo mínimo <code>clotheswasher1</code> , <code>drye1</code> , <code>kitchenapp1</code>	35
Figura 13 – Nueva York – Número de hogares en función de los electrodomésticos y conteniendo mínimo <code>clotheswasher1</code> , <code>drye1</code> , <code>air1</code> y <code>kitchenapp1</code>	36
Figura 14 – Austin – Número de hogares en función de todos los electrodomésticos	37
Figura 15 – Austin – Número de hogares en función de los electrodomésticos y conteniendo mínimo <code>dishwasher1</code> , <code>clotheswasher1</code> , <code>microwave1</code> , <code>furnace1</code> , <code>air1</code> , <code>kitchenapp1</code>	38
Figura 16 – Consumo del hogar 2335 de Austin en todo el año	39
Figura 17 – Consumo del hogar 2335 de Austin en un día del año	39
Figura 18 – Consumo del hogar 2335 de Austin en varios días	40
Figura 19 – Consumo del hogar 2335 de Austin con los electrodomésticos de interés	41

Figura 20 – Serie temporal del lavaplatos en todo el año	41
Figura 21 – Serie temporal de la lavadora en todo el año	42
Figura 22 – Serie temporal del microondas en todo el año	42
Figura 23 – Serie temporal del sistema de calefacción en todo el año.....	42
Figura 24 – Serie temporal del aire acondicionado en todo el año	43
Figura 25 – Serie temporal del circuito de la cocina en todo el año	43
Figura 26 – Señal del total de consumo en un día cualquiera	45
Figura 27 – Muestreo de la señal del total a diferentes frecuencias.....	48
Figura 28 – Ejemplo numérico de Vizgrama	49
Figura 29 – Ejemplo gráfico de información de hora del día, día de la semana y mes del año	50
Figura 30 – Vizgrama.....	50
Figura 31 – Identificación de picos de potencia en un	51
Figura 32 – Identificación de picos de potencia en la señal del total en el primer muestro	51
Figura 33 – Identificación de picos de potencia en un	51
Figura 34 – Identificación de picos de potencia en un vizgrama en el segundo muestreo .	51
Figura 35 – Vizgrama.....	51
Figura 36 – Muestreos a diferentes frecuencias para construcción del vizgrama	52
Figura 37 – Umbrales de activación en los distintos electrodomésticos.....	53
Figura 38 – Variable de salida.....	53
Figura 39 – Comprobación de activaciones definidas por umbrales – vizgramas cada 3h.	54
Figura 40 – Comprobación de activaciones definidas por umbrales – vizgramas cada minuto	55
Figura 41 – Representación temporal de la salida Y – <code>air1</code>	55
Figura 42 – Representación temporal de la salida Y – <code>clotheswasher1</code>	56
Figura 43 – Representación temporal de la salida Y – <code>dishwasher1</code>	56
Figura 44 – Representación temporal de la salida Y – <code>kitchenapp1</code>	56
Figura 45 – Representación temporal de la salida Y – <code>microwave1</code>	57

Figura 46 – Representación temporal de la salida Y – Umbral redefinido – clotheswasher1	57
Figura 47 – Representación temporal de la salida Y – Umbral redefinido – dishwasher1	58
Figura 48 – Algoritmo de filtrado – dishwasher1	58
Figura 49 – Algoritmo de relleno – dishwasher1	59
Figura 50 – Esquema genérico de un MLP [38]	61
Figura 51 – Arquitectura AlexNet [41]	63
Figura 52 – Ejemplo de capa convolucional de 10 filtros de tamaño 5x5, stride de 1 y padding de 1 [43]	64
Figura 53 – Ejemplo de capa de max pool de tamaño 2x2, stride de 2 [43]	65
Figura 54 – Proporción de valores en la variable de salida.....	67
Figura 55 – Representación gráfica de los resultados de la matriz de confusión.....	68
Figura 56 – Resultados mejor modelo MLP – dishwasher1 – train.....	69
Figura 57 – Resultados mejor modelo MLP – dishwasher1 – test	70
Figura 58 – Zoom en días 110-160 de la evaluación del mejor modelo MLP – test	70
Figura 59 – Arquitectura del primer modelo CNN	72
Figura 60 – Accuracy por epoch – Primer modelo CNN	72
Figura 61 – Arquitectura de modelo CNN	73
Figura 62 – Accuracy por epoch – Modelo CNN con overfitting.....	73
Figura 63 – Arquitectura del mejor modelo CNN.....	74
Figura 64 – Accuracy por epoch – Mejor modelo CNN	74
Figura 65 – Estructura de la mejor CNN – Estilo AlexNet.....	74
Figura 66 – Resultados mejor modelo CNN – dishwasher1 – train	75
Figura 67 – Resultados mejor modelo CNN – dishwasher1 – test	75
Figura 68 – Evaluación del mejor modelo CNN sobre los datos de otro hogar – dishwasher1 – train	77
Figura 69 – Evaluación del mejor modelo CNN sobre los datos de otro hogar – dishwasher1 – test.....	78

Figura 70 – Evaluación del modelo con transfer Learning sobre los datos de otro hogar – dishwasher1 – train.....	78
Figura 71– Evaluación del modelo con transfer Learning sobre los datos de otro hogar – clotheswasher1 – test.....	79
Figura 72 – Comparativa entre evaluación de CNN y CNN con transfer learning sobre dishwasher1 – train.....	79
Figura 73 – Comparativa entre evaluación de CNN y CNN con transfer learning sobre clotheswasher1 – train.....	80
Figura 74 – Comparativa zoom entre evaluación de CNN y CNN con transfer learning sobre clotheswasher1 – train.....	81
Figura 75 – Comparativa entre evaluación de CNN y CNN con transfer learning sobre kitchenapp1 – train	81
Figura 76 – Arquitectura del mejor modelo CNN.....	82
Figura 77 – Vizgrama con solo dishwasher1 activado	83
Figura 78 – Feature maps de la primera capa convolucional – vizgrama con dishwasher1 como input.....	83
Figura 79 – Feature maps de la segunda capa convolucional (filtro 1 a 8) – vizgrama con dishwasher1 como input	84
Figura 80– Feature maps de la segunda capa convolucional (filtro 9 a 16) – vizgrama con dishwasher1 como input	84
Figura 81 – Feature maps de la tercera capa convolucional (filtro 1 a 16) – vizgrama con dishwasher1 como input	85
Figura 82 – Feature maps de la tercera capa convolucional (filtro 17 a 32) – vizgrama con dishwasher1 como input	85
Figura 83 – Cantidad de minutos ‘on’ por mes del año – air1	86
Figura 84 – Cantidad de minutos ‘on’ por mes del año – clotheswasher1	87
Figura 85 – Cantidad de minutos ‘on’ por mes del año – dishwasher1	87
Figura 86 – Cantidad de minutos ‘on’ por mes del año – kitchenapp1	88

Figura 87 – Cantidad de minutos ‘on’ por mes del año – microwave1	88
Figura 88 – Cantidad de minutos ‘on’ por día de la semana – kitchenapp1	89
Figura 89 – Cantidad de minutos 'on' por día de la semana – microwave1	89
Figura 90 – Comparativa CNN (izquierda) y MLP (derecha) en meses de verano – kitchenapp1	90
Figura 91 – Cantidad de minutos ‘on’ por hora del día	90
Figura 92 – Representación visual de detección de activaciones basada en datos reales – dishwasher1	91
Figura 93 – Representación visual de detección de activaciones basada en predicciones – MLP – dishwasher1	92
Figura 94 – Representación visual de detección de activaciones basada en predicciones – CNN – dishwasher1	92
Figura 95 – Número de activaciones por hora del día – clotheswasher1	93
Figura 96 – Número de activaciones por día de la semana – clotheswasher1	93
Figura 97 – Número de activaciones por mes del año – clotheswasher1	94
Figura 98 – Tiempo medio de activación por día de la semana – clotheswasher1	95
Figura 99 – Tiempo medio de activación por mes del año – clotheswasher1	95
Figura 100 – Tiempo mediano de activación por mes del año – clotheswasher1	96
Figura 101 – Tiempo ‘on’ por hora del día – todos los electrodomésticos	97

Índice de tablas

Tabla 1 – Comparativa de los distintos datasets disponibles	30
Tabla 2 – Resultados accuracy MLP vs REF	71
Tabla 3 – Resultados accuracy MLP vs CNN vs REF	76

Capítulo 1. INTRODUCCIÓN

Este trabajo tiene como objetivo desarrollar un algoritmo que permita identificar la utilización de un electrodoméstico a partir del consumo total de energía. El siguiente trabajo pone en contexto el uso e importancia de la monitorización del consumo energético en el sector doméstico, explicando por qué es una necesidad creciente y las distintas técnicas que existen para llevarla a cabo. A continuación, expone la justificación del proyecto, define sus objetivos y justifica su alineación con los objetivos de desarrollo sostenible. Finalmente, el documento recoge los distintos procedimientos y técnicas que se utilizarán para resolver el problema.

La monitorización del consumo energético tiene como objetivo principal optimizar el gasto energético de los distintos electrodomésticos de un hogar, reduciendo así el consumo de electricidad y el impacto medioambiental asociado [4]. Estos objetivos se pueden conseguir mostrando a los habitantes de un hogar el consumo de cada electrodoméstico, detectando posibles anomalías en dichos aparatos o incluso indicándoles cuándo sería más económico utilizarlos, en función del precio de la electricidad. Existe división de opinión en cuanto a la efectividad de conocer y mostrar el consumo al usuario final sobre la reducción de dicho consumo [1, 2], pero es una realidad que en los últimos años hay una demanda creciente de instalaciones de energía renovable y *Smart Grids*, por lo que la monitorización de esa carga energética es crucial para automatizar, optimizar y controlar esos sistemas.

En términos generales, la monitorización de energía es el proceso de identificar y adquirir medidas energéticas en un sistema dado [3]. Esta monitorización permite determinar el consumo y el estado de los aparatos de forma individual. Esta monitorización se puede hacer de forma intrusiva (*Intrusive Load Monitoring – ILM*) o no intrusiva (*Non-Intrusive Load Monitoring – NILM*).

- o **Intrusive Load Monitoring:** Este término recoge aquellas técnicas que necesitan un dispositivo individual de medida por cada aparato o punto de medida que se quiera medir. Esta necesidad hace que esta solución sea poco escalable, costosa de desplegar, que requiera una planificación muy detallada y gestión de posibles riesgos asociados a la instalación. La terminología ‘intrusiva’ proviene de que el dispositivo de medida tiene que estar físicamente colocado en el aparato que se quiera medir, casi siempre a poca distancia. En [4] se detallan las distintas categorías de ‘intrusividad’:
 - o **ILM 1** se basa en *sub-meters* que suelen medir el consumo de una zona de la casa, colocándose en un sub-circuito del cuadro eléctrico (e.g. el sub-circuito ‘cocina’).
 - o **ILM 2** utiliza dispositivos de medida a nivel de enchufe (*Smart plugs*), pudiendo medir uno o varios aparatos al mismo tiempo (e.g. poniendo este dispositivo en una regleta con varios aparatos conectados).
 - o **ILM 3** utiliza esos *Smart plugs* en un solo aparato.

Por las razones evocadas anteriormente, nace la necesidad de un tipo de monitorización **no intrusiva**.

- o **Non-Intrusive Load monitoring:** Son aquellas técnicas que consisten en estudiar el dato agregado - el consumo total de un hogar, típicamente mediante la instalación de un contador inteligente en el cuadro eléctrico. Al tratarse de un solo dispositivo de medida, esta opción es mucho más barata y fácil de desplegar y escalar.

En la literatura también se encuentra un sinónimo de NILM que es ‘desagregación de energía’, que consiste en aplicar técnicas computacionales y algorítmicas para estimar el consumo de los electrodomésticos de forma individual, a partir de la señal del total.

Las motivaciones principales el estudio y aplicación de NILM son: (1) identificación detallada del uso de un electrodoméstico, (2) control y administración de electrodomésticos,

(3) detección de robo de energía, (4) detección ocupación en el hogar y (5) menor coste frente a ILM.

1.1 ESTADO DEL ARTE

Como se ha comentado anteriormente, la mayoría de las aproximaciones NILM utilizan el dato del contador inteligente que proporciona el consumo total, aunque en la práctica y para entrenar los modelos se necesitan también los datos de consumo individual.

Existe una gran cantidad de técnicas basadas en NILM, tal y como ha recogido J. Zico Kolter et al. de forma concisa, resumida y con detalle técnico [6]. En este documento se explican sus características principales para poder poner en contexto la utilidad y novedad de este trabajo.

El primer factor diferencial en estas técnicas es la frecuencia con la que se recogen esos datos:

- o **Alta frecuencia:** Los datos se recogen con una frecuencia de muestro de 1 Hz o más. Este tipo de datos permite extraer características en el consumo que solo se aprecian con este nivel de detalle. En el caso de frecuencias muy altas, se necesita hardware especial.
- o **Baja frecuencia:** Los datos se recogen con frecuencias de muestro inferiores a 1 Hz, es decir, de minutos o incluso horas. Es el tipo de muestreo más común.

Estos datos se recogen y almacenan en bases de datos para ser procesados y analizados en fases posteriores del trabajo. Existen distintas bases de datos que contengan la información de consumo desagregada, como son REDD [6], UK-DALE [7], AMPds [8] o incluso Dataport [9]. Esta última fue la que se eligió finalmente para este trabajo.

Una vez se tienen los datos, se pueden aplicar distintas técnicas y algoritmos para tratar de explotarlos y sacar conclusiones. Las categorías principales que engloban estas técnicas son las supervisadas y las semi-supervisadas o no supervisadas. La desagregación

mediante técnicas supervisadas necesita el dato de consumo individual para entrenar el modelo y clasificar correctamente si un aparato está funcionando o no en un determinado momento. Las técnicas semi-supervisadas solo necesitan entrenar unos pocos datos al principio del proceso para realizar la clasificación, y las no supervisadas pueden aprender de los datos sin necesidad de tener el histórico.

Técnicas supervisadas: se pueden dividir a su vez en dos:

- **Problema de optimización:** con las características importantes extraídas de los datos, estos algoritmos buscan la combinación de electrodomésticos que representan la realidad con una mayor probabilidad. Algunas de estas técnicas son *Integer Programming* [10] o algoritmos genéticos [11].
- **Reconocimiento de patrones:** es la técnica más empleada en los últimos trabajos publicados por los investigadores de este campo. Las técnicas van desde simples algoritmos de clusterización [12], inferencias Bayesianas [27], SVM para clasificar armónicos [13], y otros acercamientos como Modelos ocultos de Markov (*Hidden Markov Models – HMM*) o Redes Neuronales Artificiales [14], que han demostrado tener un buen rendimiento por la capacidad de introducir una componente temporal y de cambio de estado.

Técnicas semi y no supervisadas: estas técnicas están siendo exploradas con mucho detalle hoy en día ya que necesitan muy pocos datos para obtener buenos resultados. Algunas de estas técnicas son: utilizar las características de P (potencia activa) y Q (potencia reactiva) de la señal para realizar el clustering [15], utilización de una aproximación de *Motif Mining* [16] o por ejemplo una variación de HMM llamada *Factorial Markov Models (FHMM)* [17]. En [18], los autores detallan una metodología completa y funcional para realizar la desagregación de forma no supervisada. Por lo general, estas técnicas suelen tener peores resultados que las supervisadas.

Finalmente, entre las técnicas más novedosas y punteras se encuentran los *autoencoders* [19], las HMM [20, 21], *Deep Learning* y CNN (*Convolutional Neural*

Network). En el caso de este trabajo, se ha decidido utilizar de Deep Learning y CNN por los buenos resultados que han proporcionado en estudios similares, por lo que se explicará su funcionamiento con más detalle en el apartado 3.9.

1.2 MOTIVACIÓN

La contribución global por parte de los edificios al consumo de energía lleva en aumento desde hace años, situándose entre un 20 y 40% en países desarrollados [22]. Además, el sector energético es responsable en gran medida (entre 65 y 70% a nivel global) de las emisiones de gases de efecto invernadero [23]. La predicción y clasificación del consumo de energía en edificios son necesarios para mejorar el rendimiento energético de dichos edificios y reducir el impacto medioambiental [24].

El conocimiento del consumo energético detallado permitirá al usuario final conocer con detalle qué electrodomésticos están consumiendo, con qué frecuencia, si se ha averiado o si existe alguna anomalía (enganche a la red ilegal, por ejemplo). Aunque no está claro, como se ha comentado antes, si este conocimiento está directamente ligado a una reducción del consumo, las empresas que desarrollan este tipo de tecnologías sí que pueden tomar la iniciativa para optimizar este consumo sin que afecte de forma directa al usuario. Por ejemplo, algunos enchufes inteligentes cortan el suministro de electricidad al aparato al que estén enchufados, reduciendo así el consumo en ‘stand-by’, que se corresponde entre el 3 y 10% del consumo de hogares y oficinas [25]. Otras empresas se toman la libertad de enviar recomendaciones u ofertas personalizadas, como por ejemplo si una persona es muy consciente de lo que consume y de cuánto gasta, se le enviarán comparativas de su consumo anual o mensual. Finalmente, algunos proyectos proponen que una forma de incentivar el consumo responsable puede ser mostrando y comparando el consumo de un usuario con los usuarios de su alrededor [26].

Por último y al menos en España, este conocimiento es de utilidad para las comercializadoras, ya que conociendo más y mejor a los clientes, pueden lanzar ofertas personalizadas que favorecen el cumplimiento de esos objetivos.

Este trabajo pretende avanzar en la investigación de técnicas de NILM, en este caso utilizando *Deep Learning*, para poder conocer ese consumo energético de forma sencilla y poder poner en uso ese conocimiento.

1.3 OBJETIVOS DEL PROYECTO

1.3.1 ELECCIÓN DE BASE DATOS

Se estudiará en un primer lugar las distintas bases de datos disponibles. Se elegirá la que cumpla con los requisitos técnicos de la tecnología seleccionada (*Deep Learning*). Estos requisitos incluyen frecuencia de muestro baja, una cantidad de electrodomésticos medidos aceptable (5-15) por si hubiera que descartar alguno en un futuro, y al menos unas pocas casas en las que se dispongan de los mismos datos (mínimo 2 o 3).

1.3.2 PREPARACIÓN, PROCESADO Y ANÁLISIS DE DATOS

Se realizará el análisis exploratorio de los datos, para investigar posibles incoherencias, errores o inconsistencias. Se elegirán los electrodomésticos de los cuales se quieren predecir su estado (encendido/apagado) con los modelos y se elegirá una casa de referencia en base a la cual se realizarán las demás etapas del proyecto. Al trabajar según un proceso iterativo e incremental, puede que los electrodomésticos elegidos en un principio, la preparación de los datos de entrada o de salida se cambie en puntos determinados del trabajo.

1.3.3 PLANTEAMIENTO DEL PROBLEMA

Se plantea una idea novedosa para analizar y representar la serie temporal de consumo eléctrico en un hogar que sirve de entrada para los modelos de regresión o

clasificación, a partir de la cual se espera obtener mejores resultados que de la forma tradicional.

1.3.4 EVALUACIÓN DEL PROBLEMA CON MODELO DE REFERENCIA

Se evaluarán los resultados con un modelo de base MLP (*Multi Layer Perceptron*) que servirá como modelo de referencia con respecto al cual se compararán los resultados de la metodología propuesta en este trabajo obtenidos con la CNN.

1.3.5 EVALUACIÓN DEL PROBLEMA CON DEEP LEARNING

Se desarrollarán distintos modelos de Deep Learning siguiendo una metodología incremental e iterativa para evaluar los resultados y compararlos con el modelo de referencia.

1.4 ALIENACIÓN CON LOS OBJETIVOS DE DESARROLLO SOSTENIBLE

Como se ha comentado en el capítulo 3, el sector energético es responsable en gran medida de las emisiones de gases de efecto invernadero. Estas emisiones están causando ya, y seguirán causando en los próximos años, un cambio sustancial en el clima global, como establece el Panel Intergubernamental del Cambio Climático (IPCC) en sus informes [23].

Este cambio del clima tendrá repercusiones importantes en la economía y sociedad, y la única solución es una reducción significativa de estas emisiones. Las principales estrategias para lograr este objetivo son:

- o Mejorar la eficiencia energética
- o Abandonar los combustibles fósiles y sustituirlos por energías limpias

Este trabajo se alinea así con los objetivos del desarrollo sostenible, puesto que la solución que se propone permite mejorar el conocimiento del consumo energético de un individuo, y por lo tanto permite tomar acciones para optimizarlo y finalmente reducirlo.

1.5 METODOLOGÍA DE TRABAJO Y RECURSOS

El proyecto se abordará según las metodologías ágiles, realizando pocos cambios de una vez y asegurando la escalabilidad del proyecto, validando los resultados a medida que se van obteniendo.

Los recursos principales que se utilizarán serán los datos proporcionados por la base de datos Dataport [9], la utilización de herramientas de visualización derivadas de Python y la ejecución del código en un entorno dinámico, Jupyter Notebook.

Capítulo 2. DESCRIPCIÓN DEL PROBLEMA

Monitorizar el consumo de cada electrodoméstico en un hogar es una tarea costosa, difícil de desplegar y poco escalable. Ante esta problemática y con la creciente necesidad de conocer mejor los hábitos de consumo de los hogares (y así poder reducirlos), la investigación en el contexto de NILM ha experimentado un fuerte crecimiento en los últimos años.

Como se ha explicado en la introducción, NILM trata de monitorizar el consumo de cada electrodoméstico de manera individual, pero conociendo únicamente el consumo total de la energía. En los países de la Unión Europea a fecha de 2018, 34% de los puntos de medida de electricidad estaban equipados con un contador inteligente (99 millones), y para 2020 se prevé que esta cifra alcance 72%, aunque el objetivo establecido era del 80% [28].

Además, un estudio publicado por ACER en 2017 detalla que en España, Italia, Suecia, Finlandia y Estonia se alcanzó el objetivo marcado por la UE de tener instalados contadores inteligentes en el 80% de los hogares domésticos [29].

Si bien es cierto que la mayoría de los contadores instalados en los inicios del despliegue solo proporcionan datos cada hora, los más recientemente instalados sí que proporcionan datos de consumo con mayor granularidad, abriendo las puertas a que las técnicas NILM se puedan utilizar mucho más frecuentemente.

En la sección siguiente, a través de una serie de gráficas ilustrativas, se pone en contexto el problema, se explican las distintas características que deben tener los datos y se explican las distintas decisiones tomadas en cuanto al planteamiento del problema.

2.1 EJEMPLO ILUSTRATIVO DEL PROBLEMA

Para poder utilizar técnicas de modelado de datos supervisadas (aquellas en las que se conoce el resultado), se necesita el dato de consumo eléctrico **total** del hogar, y el consumo de forma **individual** de la mayor cantidad de electrodomésticos, aparatos y dispositivos que hayan contribuido a ese total de consumo.

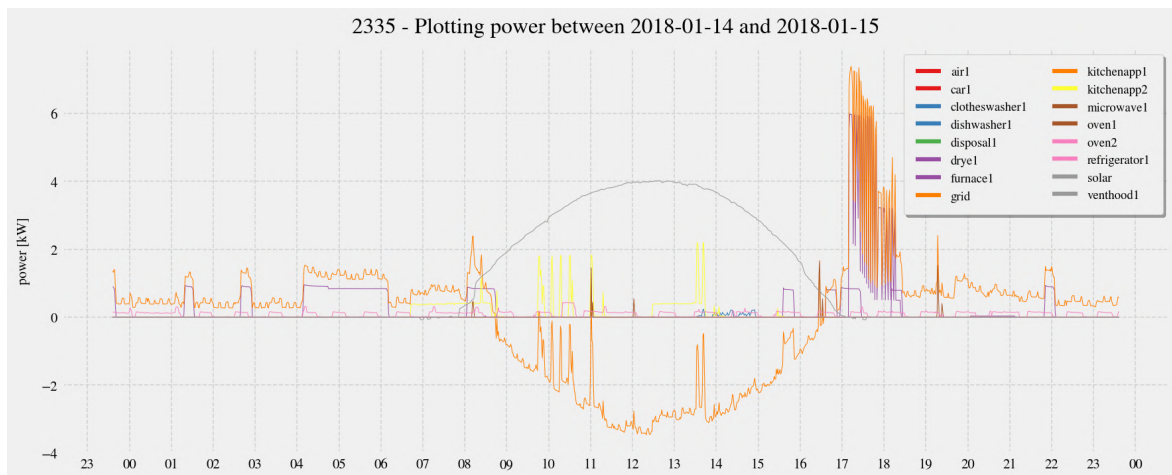


Figura 1 – Ejemplo de consumo de un hogar

La Figura muestra un ejemplo de consumo de un hogar (este consumo es un ejemplo de los datos que se han utilizado en este trabajo). Se trata de un hogar de Austin, Texas, en Estados Unidos, donde la presencia de paneles solares o puntos de carga para los coches eléctricos es más común.

Una gráfica de consumo de electrodomésticos típicos sería la de la Figura 2:

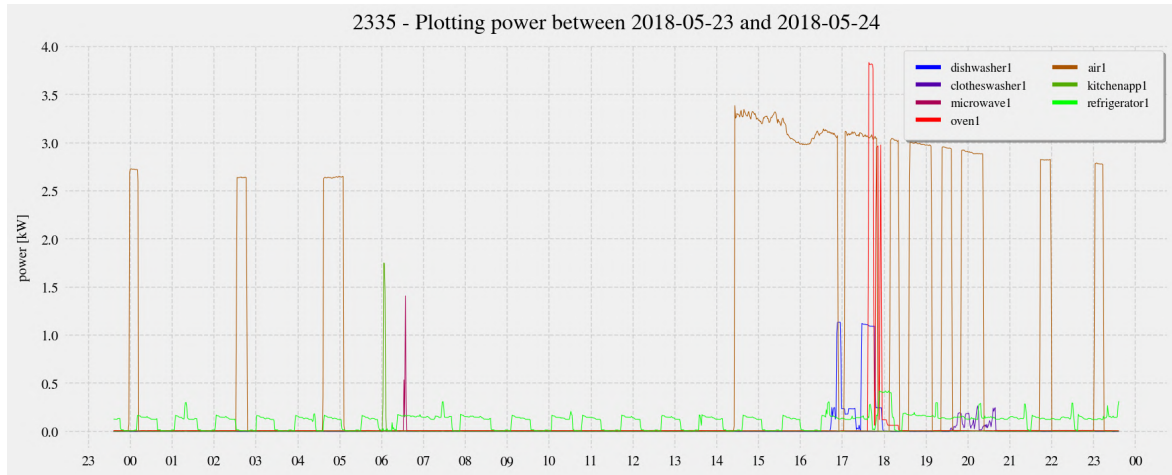


Figura 2 – Ejemplo de consumo de un hogar con electrodomésticos 'típicos'

En este caso, se está representando:

- O Dishwasher1: lavaplatos
- O Clotheswasher1: lavadora
- O Microwave1: microondas
- O Oven1: horno
- O Air1: aire acondicionado
- O Kitchenappl: circuito eléctrico de cocina, que puede tener pequeños electrodomésticos como tostadoras, batidoras, picadoras, etc.
- O Refrigerator1: frigorífico

Si añadimos la señal del total medida (`grid1`), se obtiene la siguiente figura:

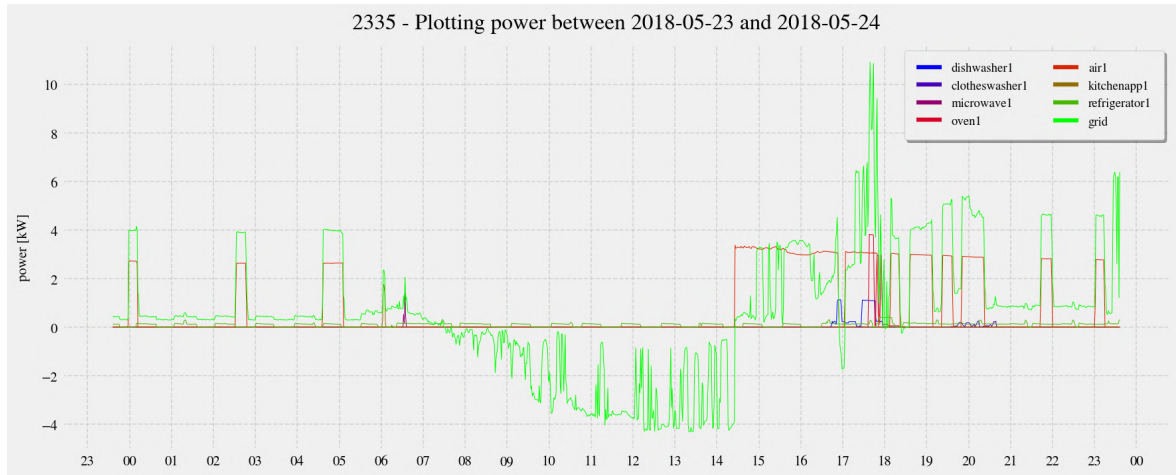


Figura 3 – Ejemplo de consumo de un hogar con electrodomésticos 'típicos' + señal del total

En la Figura 3 se puede observar que la suma del consumo de los electrodomésticos medidos no es igual a la curva del consumo total (grid1) ya que solo se representando una selección de electrodomésticos. Es por ello que, en el caso de querer identificar solo un cierto determinado número de electrodomésticos, esta medida del total puede estar introduciendo un cierto ruido (más o menos significativo en función de cuantos electrodomésticos se descarten) que, de nuevo, dependiendo del estudio y de cómo se quiera plantear el problema, puede ser o no interesante.

En el caso de este trabajo, por el número de electrodomésticos descartados, se ha decidido construir la señal del total a partir de los electrodomésticos seleccionados y no del total medido en el hogar.

Esta decisión se fundamenta principalmente en el hecho de que muchos de los electrodomésticos medidos no son muy representativos de los que se encuentran en un hogar en Europa - y en España en particular, como coches eléctricos, paneles solares, trituradoras de basura domésticas o secadoras, y otros electrodomésticos no presentan mucho interés en ser medidos por su forma de utilización (el frigorífico siempre está conectado y el usuario no decide encenderlo o apagarlo, no hay 'utilizaciones' que predecir, aunque el hecho de medir su consumo puede dar indicaciones sobre posibles averías, por ejemplo).

En la Figura 4 se muestra un ejemplo de construcción de señal del total a partir de la suma de las demás señales elegidas.

Llegado a este punto, se recuerda que el problema trata de identificar la utilización de los electrodomésticos a partir de la señal del **total**, únicamente.

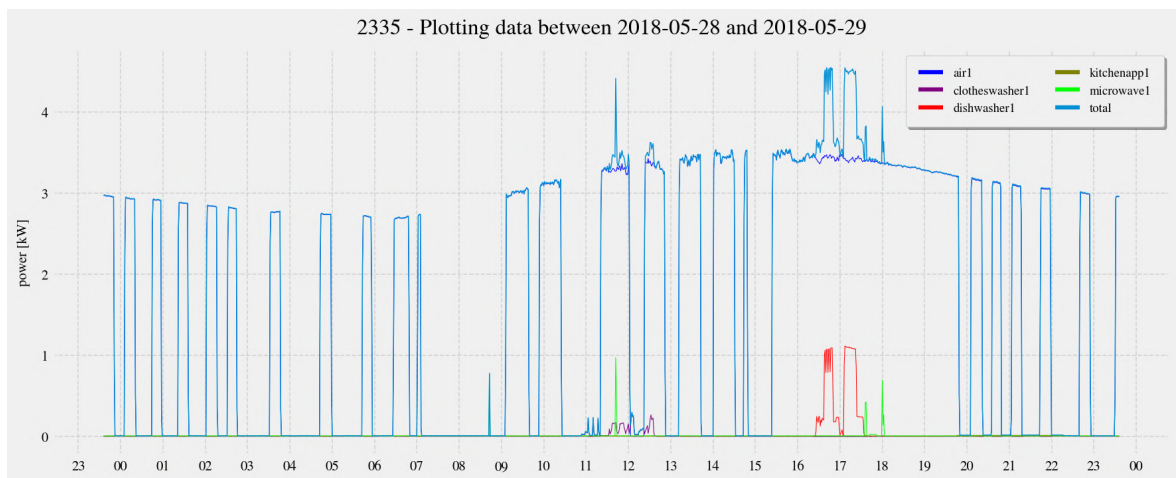


Figura 4 – Ejemplo de consumo de un hogar con electrodomésticos 'típicos' + señal del total creada

Es decir, la variable de entrada al modelo es únicamente la serie temporal del total, como se ve en la Figura 5, y el modelo tiene que ser capaz de identificar, en un instante dado, los electrodomésticos que se han usado en ese instante.

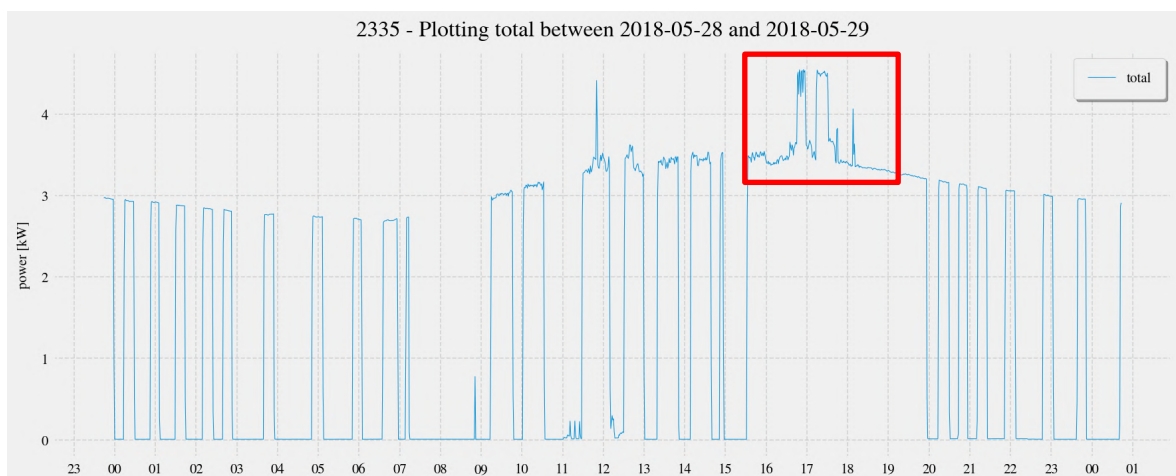


Figura 5 – Ejemplo de consumo de un hogar con electrodomésticos 'típicos', solo la señal del total

La siguiente figura es un zoom de la zona marcada en rojo en la Figura 5, a la izquierda con todas las señales y a la derecha solo la del total:

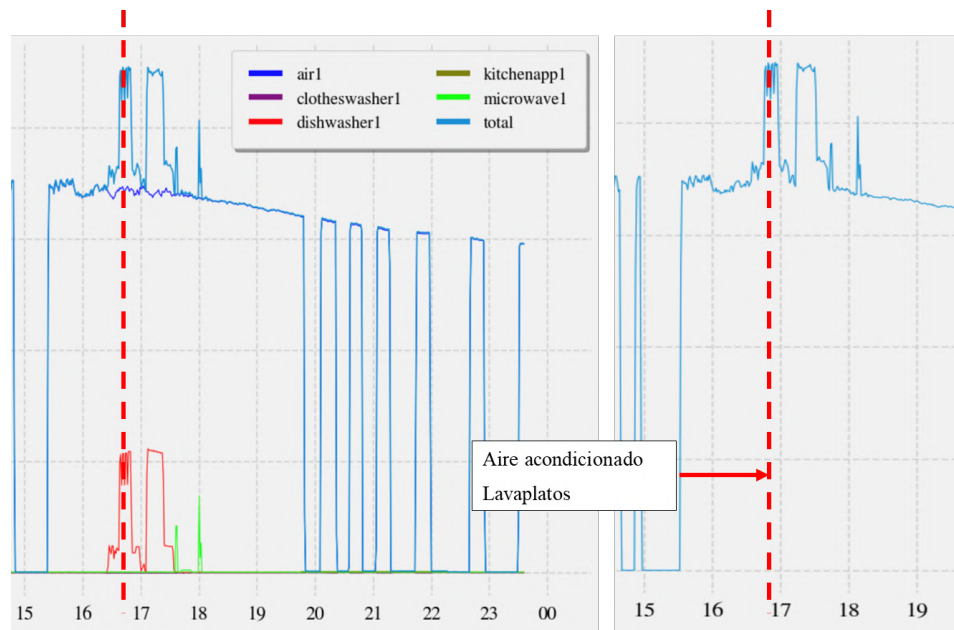


Figura 6 – Visualización e interpretación de la variable de entrada en un problema NILM

Con esta variable de entrada, se presentan dos opciones para plantear el problema:

- Problema de regresión, donde el modelo tiene que ser capaz de predecir la **potencia** de cada electrodoméstico en cada instante
- Problema de clasificación, donde el modelo tiene que ser capaz de distinguir si un electrodoméstico estaba encendido o apagado en cada instante

La Figura 7 ilustra estas dos opciones.

En este trabajo se ha decidido plantear el problema como uno de clasificación.

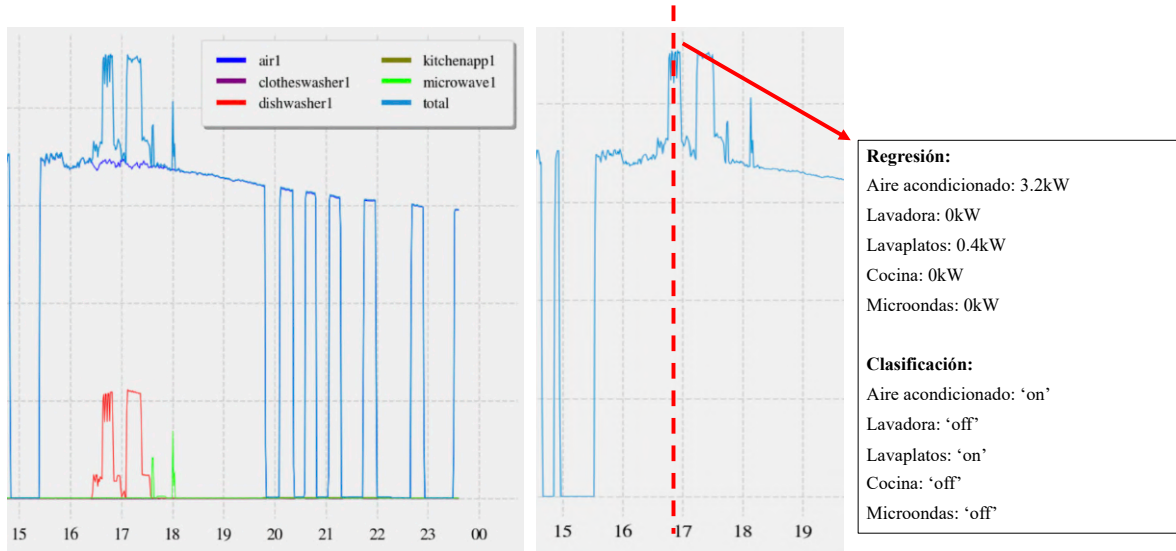


Figura 7 – Regresión vs. Clasificación

Capítulo 3. METODOLOGÍA

3.1 ESQUEMA GENERAL DEL PROCESO

La Figura 8 presenta el esquema general que se va a seguir a lo largo de todo el trabajo.

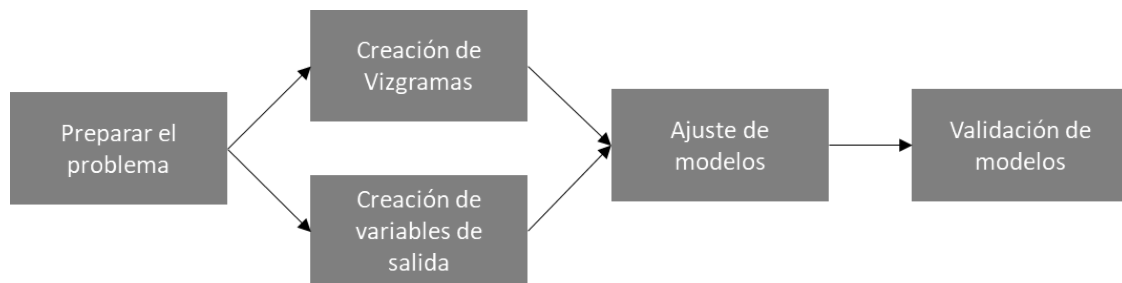


Figura 8 – Esquema general de la metodología de trabajo

La preparación del problema consiste en preparar los datos de forma que puedan ser utilizados por un modelo de Machine Learning. Esto significa construir una entrada X y la salida deseada Y . Para ello, se tiene primero que elegir unos datos adecuados entre las distintas fuentes de datos disponibles y existentes de manera gratuita, realizar una breve exploración de estos y dejar los datos limpios para poder trabajar con ellos.

Se destaca que a lo largo de todo el proyecto se ha seguido una metodología iterativa y que, en muchas ocasiones, se han tenido que modificar parámetros que correspondían con etapas anteriores, para tratar de mejorar los resultados.

En este problema la entrada es la señal de consumo total, ya sea la que se construye a través de otras o la que venga medida en el propio dataset. Típicamente, esta variable simplemente tomará el valor del total en un instante dado. En el ejemplo de la Figura 7, su valor sería de $3.6kW$. Pero en este trabajo, para aprovechar las arquitecturas de las CNN, se ha decidido construir la variable de entrada X introduciendo un concepto novedoso que se ha denominado ‘vizgrama’ y que se explicará en detalle en el apartado 3.4.

La salida Y será, por cada instante de tiempo, un vector binario representando el estado encendido o apagado del electrodoméstico en cuestión, también se detallará su construcción en detalle en el apartado 3.5.

Por ejemplo, retomando la Figura 7, la variable de salida en ese instante será:

```
[1, 0, 1, 0, 0]
```

A continuación, se procederá al ajuste de los modelos. En esta sección del trabajo se ha creado un primer modelo de referencia que consiste en una MLP con unas pocas capas ocultas. El segundo modelo que se realiza es utilizando arquitecturas propias de las CNN, dado que se ha planteado el problema y construido una variable de entrada específicamente para ello.

Por último, se validan los modelos, obteniendo unos resultados preliminares con el primer modelo MLP y los resultados definitivos con el modelo CNN que serán comparados con los primeros.

3.2 PREPARACIÓN DEL PROBLEMA

3.2.1 ELECCIÓN DEL DATASET

Existen una variedad de datasets disponibles para el estudio de NILM. En los últimos años, se han publicado un gran número de nuevos datasets para atacar este problema, que contienen tanto el consumo energético total del hogar como el consumo individual de dispositivos.

Sin embargo, estos datasets suelen cubrir apenas entre 5 y 10 casas debido al coste financiero asociado al despliegue de hardware necesario para medir el consumo de cada dispositivo, y no suelen estar publicados en formatos ‘amigables’ para que los investigadores de este campo puedan utilizarlos con facilidad, además de que varios de estos tampoco son gratuitos.

A continuación, se detallan algunos de los datasets que se han considerado como candidatos para este estudio.

3.2.1.1 Dataport Dataset

Dataport es un dataset creado por Pecan Street Inc, que contiene datos de consumo a nivel de dispositivo (consumo desagregado) como de hogar (consumo agregado), hasta una granularidad de 1 segundo, teniendo una versión de 1 minuto y de 15 minutos [30]. Además, contiene información de 722 hogares y por períodos de hasta más de 3 años para ciertos hogares, superando con creces el alcance y variedad de los demás datasets creados hasta la fecha.

Los electrodomésticos medidos cambian en cada hogar, aunque los principales suelen repetirse en todos los hogares.

La información a la totalidad de los datos no es gratuita, pero existe una versión gratuita para investigadores y universitarios que contiene 25 hogares de tres zonas geográficas de Estados Unidos: Nueva York, California y Austin (Texas).

La Figura 9 se muestra una tabla resumen de los datos ofrecidos de forma gratuita:

	New York	California	Austin
1-second	New York 1-second file 1 (3.2 GB) New York 1-second file 2 (1.1 GB) New York 1-second file 3 (1.0 GB) New York 1-second file 4 (1.0 GB)	NA	Austin 1-second file 1 (3.1 GB) Austin 1-second file 2 (3.3 GB) Austin 1-second file 3 (3.3 GB) Austin 1-second file 4 (3.2 GB)
1-minute	New York 1-min (121 MB)	California 1-min (164 MB)	Austin 1-min (251 MB)
15-minute	New York 15-min (11.4 MB)	California 15-min (3.3 MB)	Austin 15-min (22.4 MB)

Figura 9 – Resumen de datos disponibles del dataset Dataport [30]

3.2.1.2 ECO Dataset

Este dataset [31] es open-source y contiene información de consumo eléctrico de 6 hogares suizos durante un período de 8 meses y con una granularidad de 1Hz (1 segundo). Contiene el dato agregado (corriente, tensión y cada fase) y desagregado a nivel de electrodoméstico.

Cada hogar contiene un número determinado de electrodomésticos medidos, aunque los principales suelen estar cubiertos en todos los hogares.

3.2.1.3 UK-DALE Dataset

En este caso, el conjunto de datos [32] contiene información de consumo eléctrico de 5 casas domésticas de Inglaterra, durante un período variable en función de los hogares. Por ejemplo, en la casa #1, se tiene más de 2 años de medidas, mientras que en la casa #3 apenas se tienen 39 días.

La granularidad de la medida es de 6 segundos tanto para el dato de consumo agregado como desagregado a nivel de dispositivos.

Existe también una gran discrepancia entre la cantidad de dispositivos medidos en cada casa. En la casa #1 existen hasta 54 dispositivos de medida colocados, mientras que en la casa #3 hay tan solo 5 y en la #4 6.

3.2.1.4 REFIT Dataset

REFIT [33] es un conjunto de datos open-source y contiene información de consumo eléctrico de 20 hogares de Inglaterra, durante un período de 2 años.

La granularidad de la medida es de 8 segundos para el dato agregado y desagregado, y se miden 8 electrodomésticos.

3.2.1.5 iAWE

Este dataset [34] tiene información de consumo eléctrico de 1 sola casa en India, durante un período de 73 días.

La granularidad de la medida es de 1 segundo para el dato agregado y desagregado, y se miden hasta 33 dispositivos.

3.2.1.6 REDD

Este dataset [35] tiene información de consumo eléctrico de 6 casas de Massachussets, Estados-Unidos, durante un período variable de entre 3 y 19 días en función del hogar.

La granularidad de la medida es de 1 segundo para el dato agregado, y de 3 segundos para el dato desagregado. También mide el consumo agregado con una frecuencia de 15kHz (típicamente esto es para hacer el estudio de la *load signature*, que son patrones en las señales eléctricas de los dispositivos durante los primeros momentos de su utilización y que solo son identificables con una frecuencia de muestro mucho mayor a las típicas.

Además, el acceso al dataset es un poco más restrictivo que otras alternativas.

3.2.1.7 BLUED

Ese dataset [36] tiene información de consumo eléctrico de 1 hogar en Pennsylvania, Estados-Unidos, durante un período de 8 días y con una granularidad de medida de 12kHz para el dato agregado y desagregado. Existen hasta 55 dispositivos de medida distintos.

De la misma forma que el anterior, este dataset está más dirigido al estudio de la *load signature*.

3.2.1.8 RAE

Este dataset [37] tiene información de consumo eléctrico de 2 hogares de Canadá, durante un período de 72 días y con una granularidad de 1 segundo para el dato agregado y desagregado. Existen hasta 24 dispositivos de medida distintos por hogar.

3.2.1.9 ENERTALK dataset

Este dataset [38] tiene información de consumo eléctrico de 22 hogares de Corea del Sur, durante un período de entre 29 y 122 días en función del hogar.

La granularidad de la medida es de 15 Hz (o lo que es lo mismo, 4 veces por segundo) tanto para el dato agregado como desagregado, y se miden entre 1 y 7 dispositivos por hogar.

Tabla 1 – Comparativa de los distintos datasets disponibles

Dataset	Location	Duration	No. Of houses	No. Of appliance instances	Sampling rate
REDD	MA, USA	3-19 days	6 houses	10-24 per house	15 kHz (aggregate), 1/3 Hz (appliance)
BLUED	PA, USA	8 days	1 house	43 (on-off tag)	12 kHz (aggregate only)
Smart	MA, USA	3 months	3 houses	21-26 per house	1 Hz (aggregate and appliance)
Tracebase	Germany	1 day	N/A	122	1 Hz (appliance only)
BERDS	USA	1 year	1 house	4	20 sec. (aggregate and appliance)
AMPds	Canada	1 year	1 house	19	1 min. (aggregate and appliance)
iAWE	India	73 days	1 house	33	1 min. (aggregate), 1 Hz or 6 sec. (appliance)
GREEND	Austria/Italy	1 year	9 houses	9 per house	1 Hz (aggregate and appliance)
ECO	Switzerland	8 months	6 houses	7-19 per house	1 Hz (aggregate and appliance)
PLAID	USA	6 seconds	N/A	1074	30 kHz (appliance only)
COMBED	India	1 month	6 buildings	200	30 sec. (aggregate and appliance)
DRED	Holland	6 months	1 house	12	1 Hz (aggregate and appliance)
Dataport	USA	4+ years	1200+ houses	~70 per house	1 Hz to 1 min. (aggregated and appliance)
UK-DALE	UK	2.5 years	5 houses	5-54 per house	16 kHz (aggregate), 6 sec. (appliance)
AMPds2	Canada	2 years	1 house	21	1 min. (aggregate and appliance)
REFIT	UK	2 years	20 houses	9 per house	8 sec. (aggregate and appliance)
RAE	Canada	72 days	1 house	24	1 Hz (aggregate and appliance)
I-BLEND	India	52 months	7 buildings	N/A	1 min. (aggregate only)
ENERTALK	Korea	29-122 days	22 houses	1~7 per house	15 Hz (aggregate and appliance)

Existen otros cuantos datasets de los cuales simplemente se va a incluir la información relevante y resumida en la Tabla .

Hay varios datasets candidatos al estudio que se quiere realizar. Como se comentó anteriormente, se requiere:

- o Una granularidad de medida de entre 1s y 1min tanto para el dato agregado como desagregado
- o Un número de electrodomésticos medidos lo mayor posible (mínimo 5 idénticos por hogar)
- o Un período de estudio que permita obtener conclusiones relevantes (6-12 meses, al menos para tener todas las estaciones y poder observar los distintos comportamientos asociados a los cambios de temperatura)
- o Un número de hogares que nos asegure que el modelo generaliza correctamente (más de 5)

Los datasets que cumplen estos requisitos son:

- o Dataport
- o REFIT
- o GREEND
- o ENERTALK

Entre estos 4 candidatos, se escogió Dataport, principalmente por tener un número de hogares estudiados mucho mayor al resto, lo que permite reducir el riesgo de experimentar problemas por la calidad del dato.

3.3 DATASET SELECCIONADO

Como se ha comentado en el apartado anterior, el dataset seleccionado para este trabajo es Dataport Dataset. La Figura 10 resume los datos disponibles de este dataset:

	New York	California	Austin
1-second	New York 1-second file 1 (3.2 GB) New York 1-second file 2 (1.1 GB) New York 1-second file 3 (1.0 GB) New York 1-second file 4 (1.0 GB)	NA	Austin 1-second file 1 (3.1 GB) Austin 1-second file 2 (3.3 GB) Austin 1-second file 3 (3.3 GB) Austin 1-second file 4 (3.2 GB)
1-minute	New York 1-min (121 MB)	California 1-min (164 MB)	Austin 1-min (251 MB)
15-minute	New York 15-min (11.4 MB)	California 15-min (3.3 MB)	Austin 15-min (22.4 MB)

Figura 10 – Resumen de datos disponibles del dataset Dataport [30]

Como se puede ver en la figura, los datos a nivel de segundos están disponibles solo para las casas de Nueva York y de Austin (en el momento de realizar el estudio, los datos de 1 minuto de California no estaban disponibles todavía, por lo que solo se analizaron los de Nueva York y Austin) mientras que los de minuto (tanto 1 como 15) están disponibles en las 3 zonas geográficas.

3.3.1 ANÁLISIS EXPLORATORIO

Lo primero que hay que destacar es que este trabajo se ha realizado en un PC personal, con una potencia de computación moderada, haciendo que la realización del trabajo con los datos de 1 segundo sea imposible debido a su volumen. Además, para el objetivo de este estudio, la resolución de 1 minuto debería ser suficiente.

Dataport ofrece unos documentos que contienen la información relativa a qué dispositivos se han medido en cada hogar. Se ha decidido coger los 5 electrodomésticos más comunes de un hogar: la lavadora, el lavaplatos, el aire acondicionado, microondas y pequeños electrodomésticos de la cocina que van todos conectados a un mismo circuito eléctrico. Esto supone un compromiso entre tener una representación suficientemente buena de la casa y que el modelo sea escalable.

Con estos requisitos iniciales definidos, se puede empezar a explorar el dataset.

3.3.1.1 Nueva York

Los 25 hogares de Nueva York contienen datos del 1 de mayo de 2019 al 31 de octubre de 2019, es decir, 6 meses de datos cada minuto.

```
In [91]: print(ny_lmin.loc[ny_lmin['dataid']==5997].sort_values(by='localminute').iloc[0]['localminute'])
print(ny_lmin.loc[ny_lmin['dataid']==5997].sort_values(by='localminute').iloc[-1]['localminute'])

2019-05-01 00:00:00-05
2019-10-31 23:59:00-05
```

Los electrodomésticos medidos entre todos los hogares son los siguientes:

```
In [6]: ny_lmin.columns

Out[6]: Index(['dataid', 'localminute', 'air1', 'air2', 'air3', 'airwindowunit1',
              'aquarium1', 'bathroom1', 'bathroom2', 'bedroom1', 'bedroom2',
              'bedroom3', 'bedroom4', 'bedroom5', 'battery1', 'car1', 'car2',
              'circpump1', 'clotheswasher1', 'clotheswasher_drygl', 'diningroom1',
              'diningroom2', 'dishwasher1', 'disposal1', 'dryel', 'drygl', 'freezer1',
              'furnacel', 'furnace2', 'garage1', 'garage2', 'grid', 'heater1',
              'heater2', 'heater3', 'housefan1', 'icemaker1', 'jacuzzi1', 'kitchen1',
              'kitchen2', 'kitchenappl1', 'kitchenapp2', 'lights_plugs1',
              'lights_plugs2', 'lights_plugs3', 'lights_plugs4', 'lights_plugs5',
              'lights_plugs6', 'livingroom1', 'livingroom2', 'microwavel', 'office1',
              'outsidelights_plugs1', 'outsidelights_plugs2', 'oven1', 'oven2',
              'pool1', 'pool2', 'poollight1', 'poolpump1', 'pump1', 'rangel',
              'refrigerator1', 'refrigerator2', 'security1', 'sewerpump1', 'shed1',
              'solar', 'solar2', 'sprinkler1', 'sumppump1', 'utilityroom1',
              'venthood1', 'waterheater1', 'waterheater2', 'wellpump1', 'winecooler1',
              'leg1v', 'leg2v'],
              dtype='object')

In [7]: len(ny_lmin.columns)

Out[7]: 79
```

Hay un total de 79 variables.

Como se puede observar en la Figura 11, los campos `dataid`, `localminute` y `grid` están presentes en las 25 casas, lo cual es buena señal, puesto que `dataid` es simplemente el id de la casa, `localminute` es el registro temporal de la medida y `grid` es la señal del consumo eléctrico total. Estas 3 variables deberían estar en todas las filas del dataset.

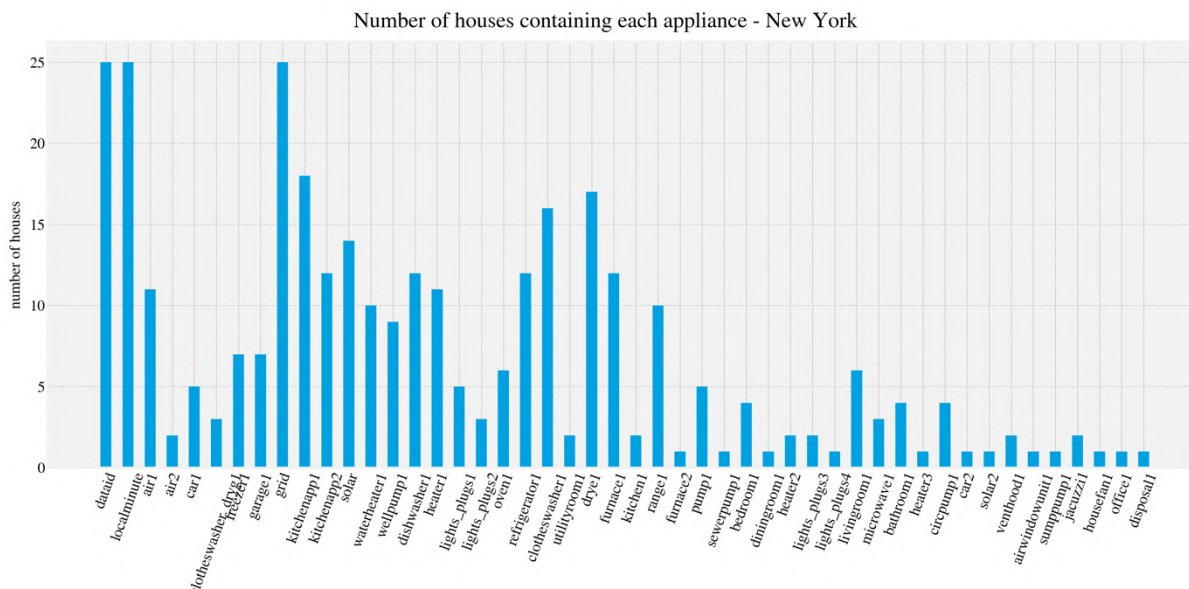


Figura 11 – Nueva York – Número de hogares en función de todos los electrodomésticos

Algo que llama la atención es que en apenas 12 hogares se está midiendo el frigorífico, un electrodoméstico que probablemente esté presente en todos.

Con esta lista de electrodomésticos y siguiendo con la idea que se ha comentado anteriormente para limitar el alcance del proyecto, hay que definir qué electrodomésticos se quiere estudiar para averiguar en cuántos hogares están todos presentes. Estos electrodomésticos serán los de uso más común y significativo en un hogar ‘típico’, como puede ser la lavadora, el lavaplatos o el aire acondicionado.

En la Figura 12, se representan el número de casas en función de si contienen o no los siguientes electrodomésticos **como mínimo**: clotheswasher1 (lavadora), drye1 (secadora), kitchenapp1 (circuito de cocina):

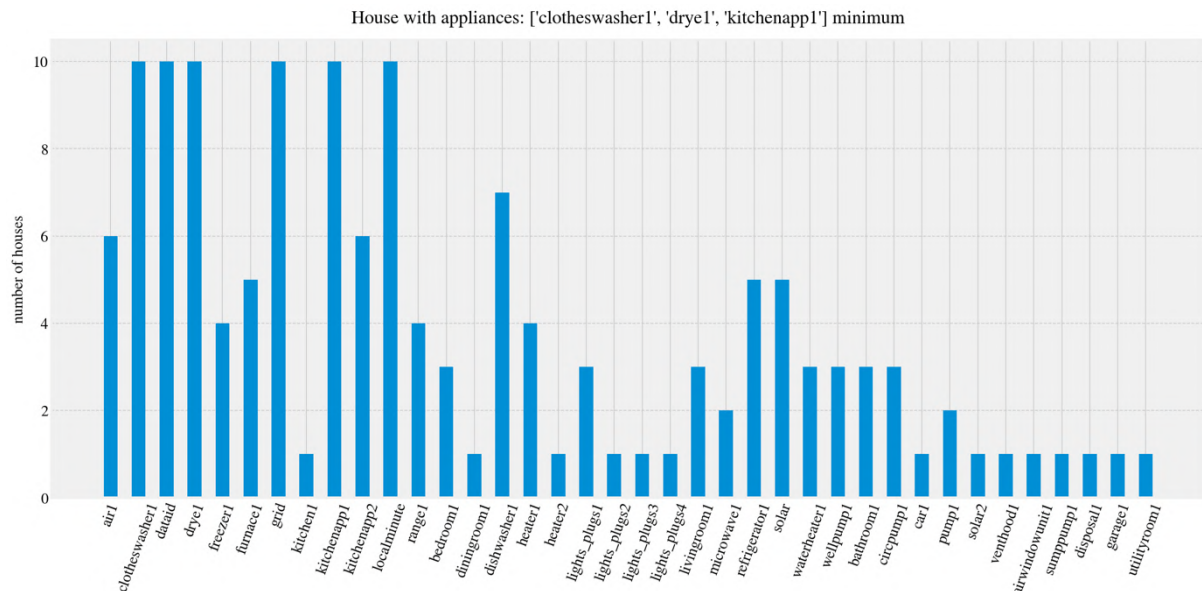


Figura 12 – Nueva York – Número de hogares en función de los electrodomésticos y conteniendo mínimo clotheswasher1, drye1, kitchenapp1

Se puede ver que hay hasta 10 hogares que comparten dichos electrodomésticos como mínimo, siendo el siguiente umbral añadiendo dishwasher1 (lavaplatos) con 7 hogares, y luego 6 hogares añadiendo air1 (aire acondicionado), y 5 hogares incorporando furnace1 (sistema de calefacción), refrigerator1 (frigorífico) y solar (paneles solares).

Afinando un poco la búsqueda con estos datos, se repite el ejercicio buscando como mínimo:

clotheswasher1, drye1, kitchenapp1, air1 y furnace1

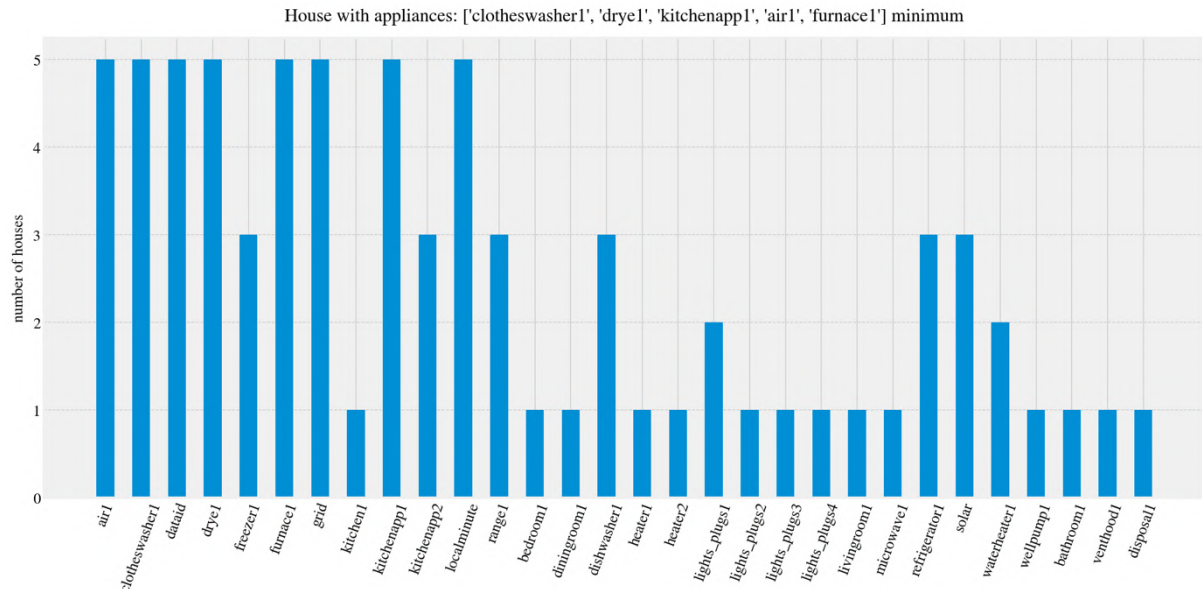


Figura 13 – Nueva York – Número de hogares en función de los electrodomésticos y conteniendo mínimo clotheswasher1, drye1, air1 y kitchenapp1

Como se puede ver en la Figura 13, 5 hogares contienen medidas de estos aparatos, lo cual es un buen punto de partida.

3.3.1.2 Austin

Los 25 hogares de Austin contienen datos del 1 de enero de 2018 al 31 de diciembre de 2018, es decir, un año de datos cada minuto.

```
In [104]: print(aus_lmin.loc[aus_lmin['dataid']==661].sort_values(by='localminute').iloc[0]['localminute'])
print(aus_lmin.loc[aus_lmin['dataid']==661].sort_values(by='localminute').iloc[-1]['localminute'])

2018-01-01 00:00:00-06
2018-12-31 23:59:00-06
```

Los electrodomésticos medidos entre todos los hogares son los siguientes:


```
In [51]: aus_lmin.columns

Out[51]: Index(['dataid', 'localminute', 'air1', 'air2', 'air3', 'airwindowunit1',
               'aquarium1', 'bathroom1', 'bathroom2', 'bedroom1', 'bedroom2',
               'bedroom3', 'bedroom4', 'bedroom5', 'battery1', 'car1', 'car2',
               'circpump1', 'clotheswasher1', 'clotheswasher_dryg1', 'diningroom1',
               'diningroom2', 'dishwasher1', 'disposal1', 'drye1', 'dryg1', 'freezer1',
               'furnace1', 'furnace2', 'garage1', 'garage2', 'grid', 'heater1',
               'heater2', 'heater3', 'housefan1', 'icemaker1', 'jacuzzi1', 'kitchen1',
               'kitchen2', 'kitchenapp1', 'kitchenapp2', 'lights_plugs1',
               'lights_plugs2', 'lights_plugs3', 'lights_plugs4', 'lights_plugs5',
               'lights_plugs6', 'livingroom1', 'livingroom2', 'microwavel', 'officel',
               'outsidelights_plugs1', 'outsidelights_plugs2', 'oven1', 'oven2',
               'pool1', 'pool2', 'poollight1', 'poolpump1', 'pump1', 'rangel',
               'refrigerator1', 'refrigerator2', 'security1', 'sewerpump1', 'shed1',
               'solar', 'solar2', 'sprinkler1', 'sumppump1', 'utilityroom1',
               'venthood1', 'waterheater1', 'waterheater2', 'wellpump1', 'winecooler1',
               'leg1v', 'leg2v'],
              dtype='object')
```

```
In [105]: len(aus_lmin.columns)
```

```
Out[105]: 79
```

De la misma forma que para el dataset de Nueva York, en el total de hogares se miden un total de 79 electrodomésticos.

Al igual que en Nueva York, en los 25 hogares están presentes los campos `dataid`, `localminute` y `grid`.

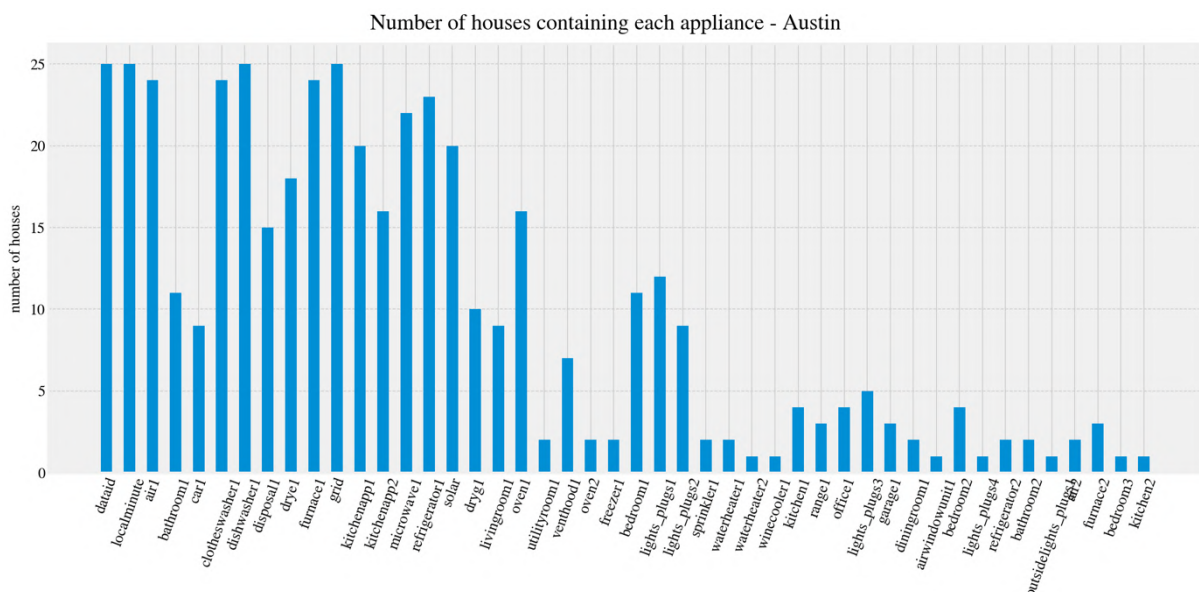


Figura 14 – Austin – Número de hogares en función de todos los electrodomésticos

Realizando el mismo ejercicio que antes, se buscan el número de casas que tengan como mínimo: `dishwasher1`, `clotheswasher1`, `microwave1`, `furnace1`, `air1`, `kitchenapp1`:

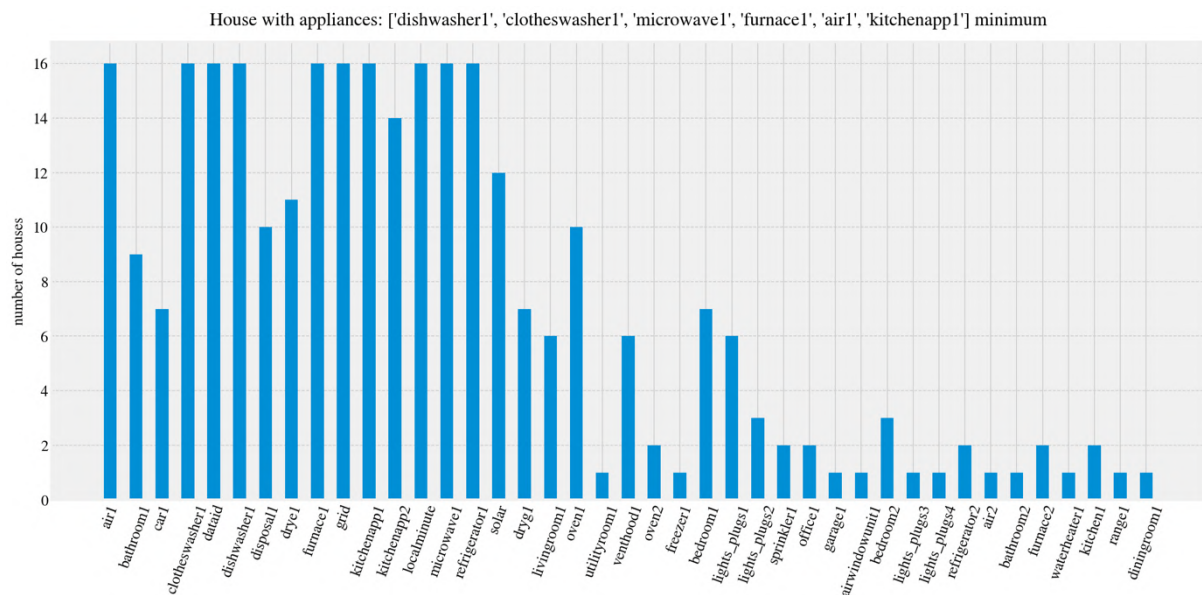


Figura 15 – Austin – Número de hogares en función de los electrodomésticos y conteniendo mínimo `dishwasher1`, `clotheswasher1`, `microwave1`, `furnace1`, `air1`, `kitchenapp1`

Con esta búsqueda inicial (que incluye más electrodomésticos mínimos que la primera que se hizo para Nueva York, por lo que a priori debería ser más restrictiva), se puede ver que se obtienen 16 hogares que contienen medidas de dichos electrodomésticos, un número también superior al obtenido en el caso de Nueva York en la primera búsqueda.

Al tener más datos (1 año frente a 6 meses) y mejor homogeneidad en cuanto a electrodomésticos, se ha decidido trabajar con hogares del dataset de Austin para el resto del trabajo.

En una primera iteración, los electrodomésticos elegidos para empezar con la preparación del problema serán el lavaplatos, la lavadora, el microondas, el sistema de calefacción, el aire acondicionado y el circuito de cocina, respectivamente:

`dishwasher1`, `clotheswasher1`, `microwave1`, `furnace1`, `air1`, `kitchenapp1`

3.3.2 EXPLORACIÓN DE UNA CASA DE TEXAS

3.3.2.1 Casa 2335

La Figura 16 y la Figura 17 muestran la serie temporal de consumo de todos los electrodomésticos medidos en el hogar con id 2335 durante todo el período y un día, respectivamente, mientras que la Figura 18 muestra el consumo durante 4 días distintos del año.

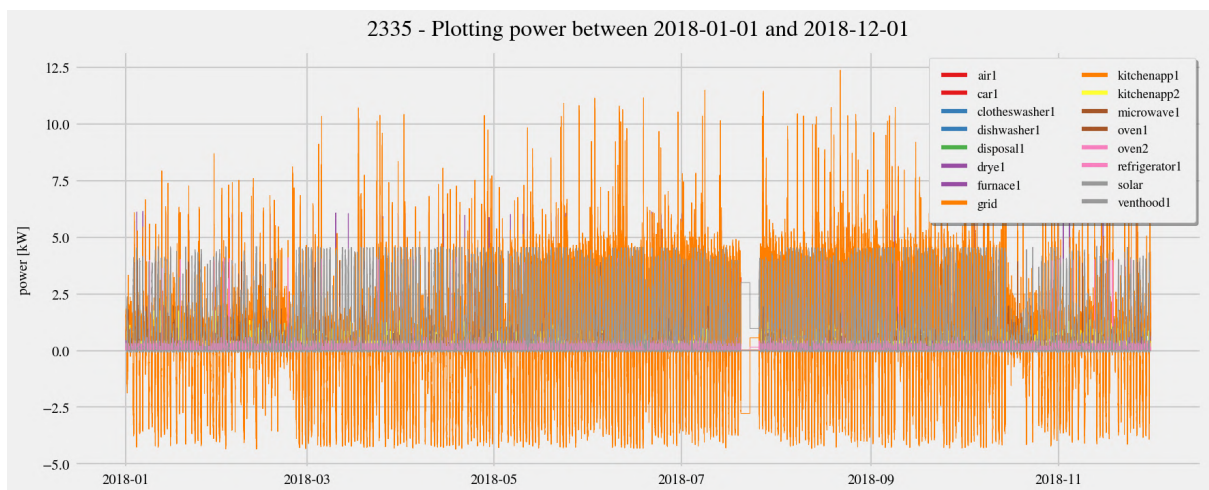


Figura 16 – Consumo del hogar 2335 de Austin en todo el año

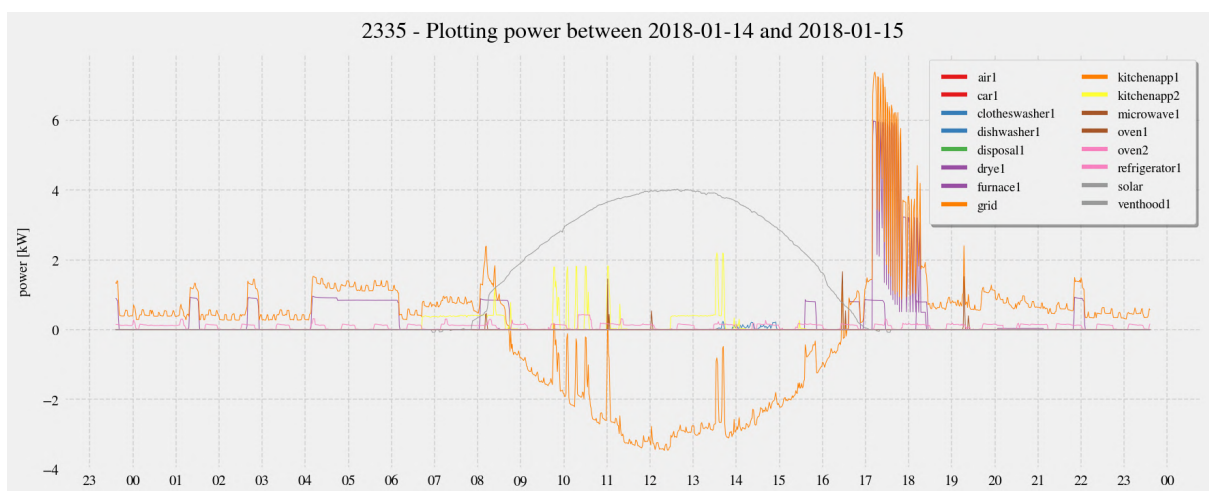


Figura 17 – Consumo del hogar 2335 de Austin en un día del año

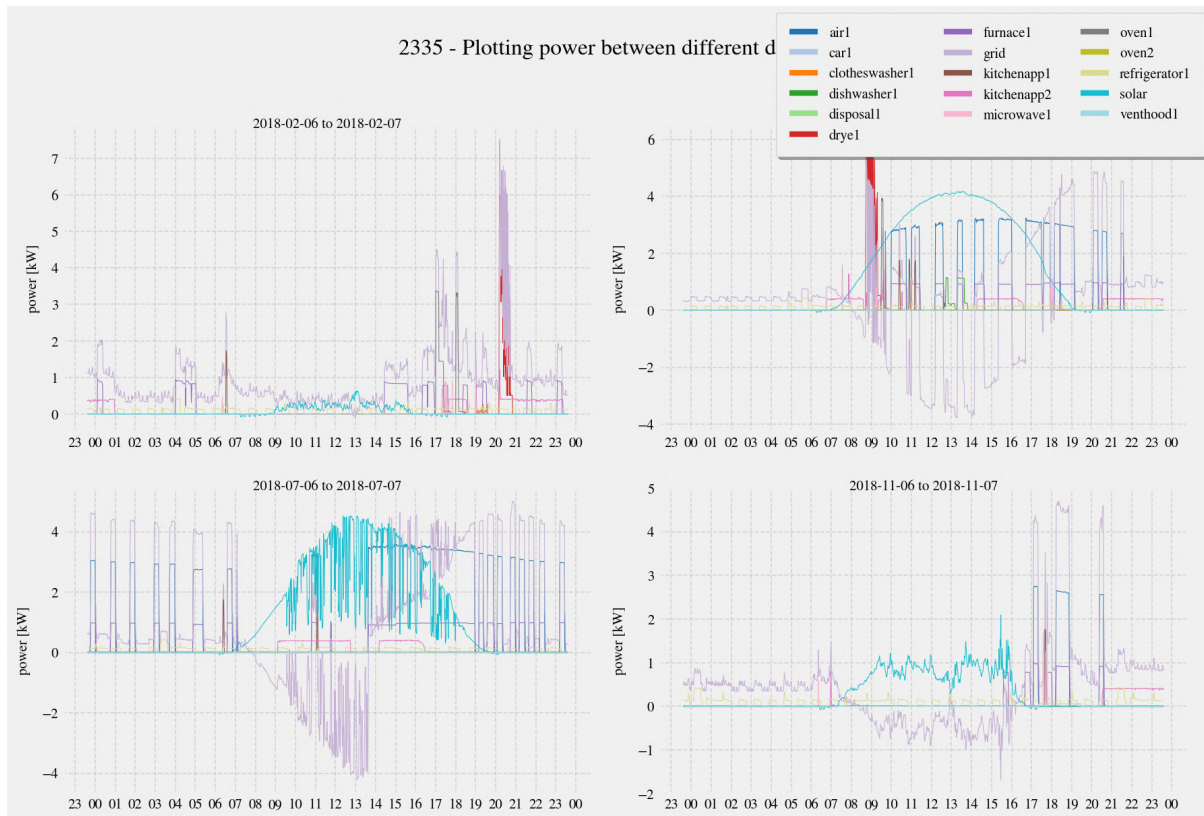


Figura 18 – Consumo del hogar 2335 de Austin en varios días

En estas gráficas se puede ver claramente la influencia de los paneles solares instalados en el hogar (consumo negativo), y de otros consumos de mayor magnitud como el punto de carga del coche eléctrico car1.

Esto también ayuda a entender por qué se ha decidido trabajar con un número reducido de variables y centrarse en los electrodomésticos principales (se entiende que disponer de un coche eléctrico no es lo normal en un hogar).

En el apartado anterior se ha comentado cuáles son los electrodomésticos de mayor interés:

- Dishwasher1
- Clotheswasher1
- Microwave1
- Furnace1

- Air1
- Kitchenapp1

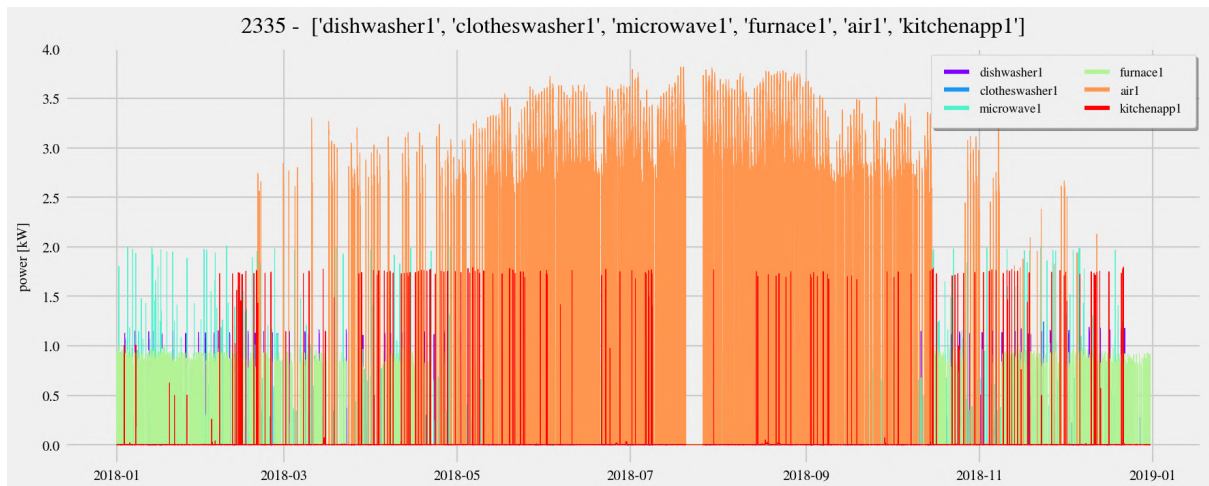


Figura 19 – Consumo del hogar 2335 de Austin con los electrodomésticos de interés

A continuación, se muestran las distintas series temporales de dichos electrodomésticos:

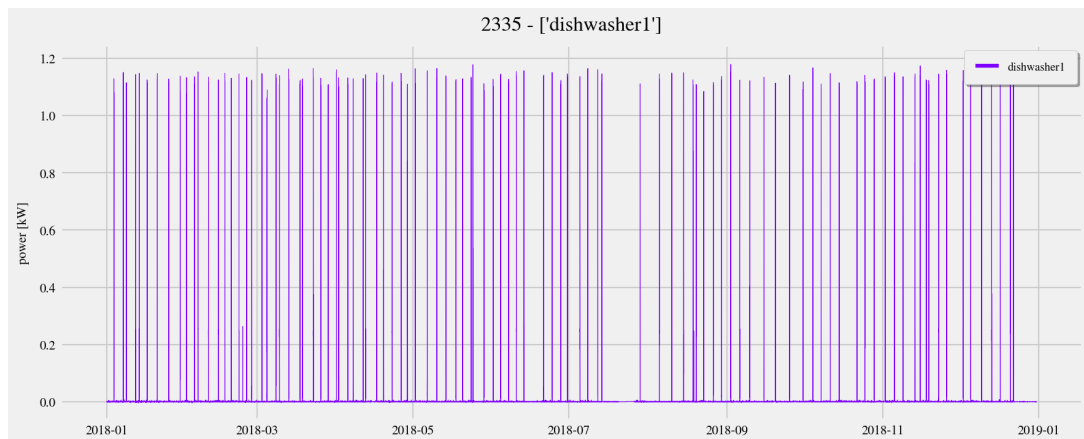


Figura 20 – Serie temporal del lavaplatos en todo el año

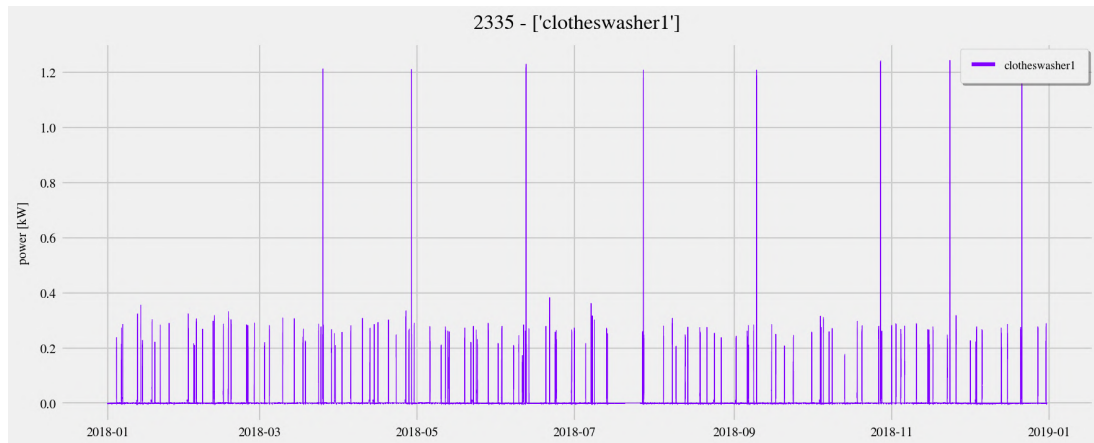


Figura 21 – Serie temporal de la lavadora en todo el año

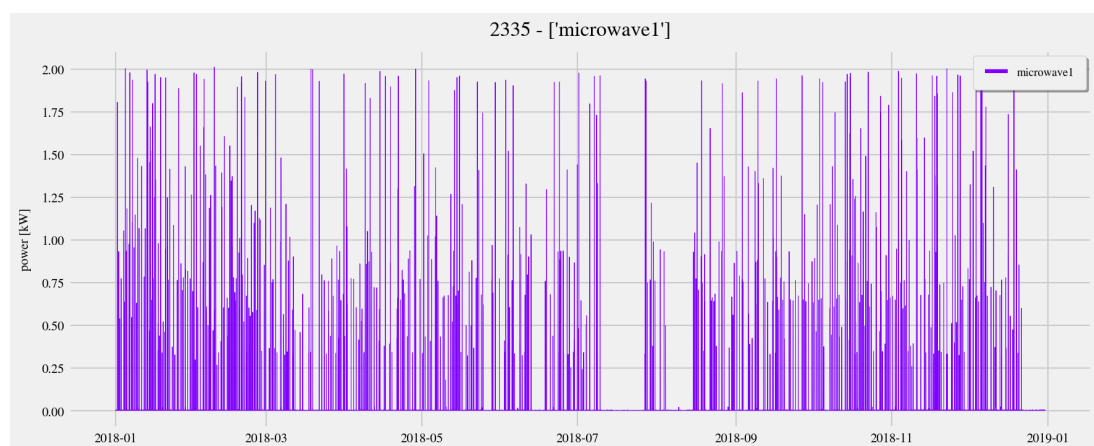


Figura 22 – Serie temporal del microondas en todo el año

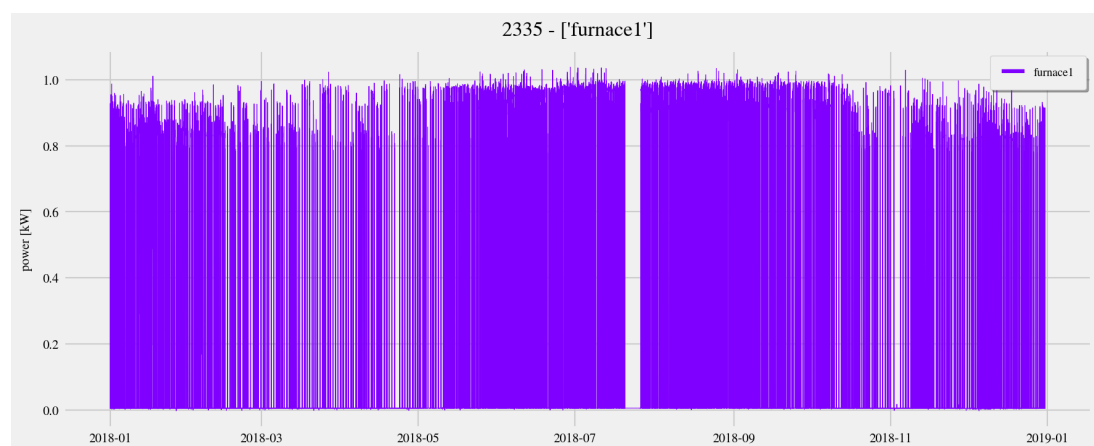


Figura 23 – Serie temporal del sistema de calefacción en todo el año

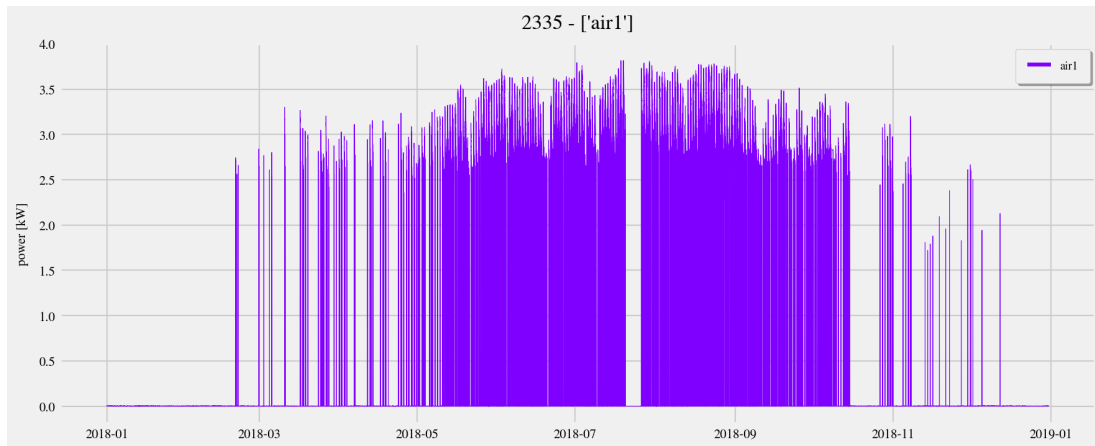


Figura 24 – Serie temporal del aire acondicionado en todo el año

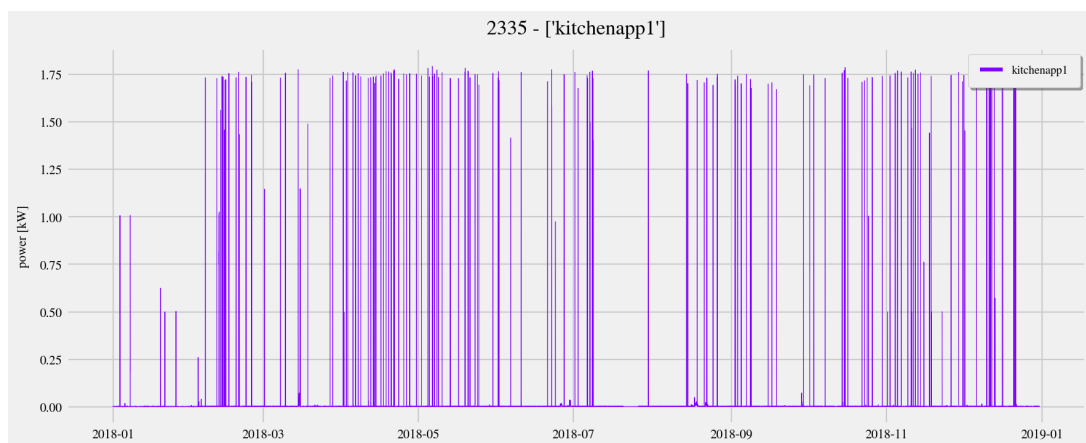


Figura 25 – Serie temporal del circuito de la cocina en todo el año

Llama la atención la serie temporal del `furnace1`, que es prácticamente constante e invariante durante todo un año entero. Buscando un poco en detalle qué tipo de electrodoméstico es, parece tratarse de una bomba de calor o caldera. Mirando el consumo de este aparato, no parece que aporte mucho al estudio puesto que parece estar siempre activado, por lo que se ha descartado.

3.4 VIZGRAMA

El objetivo de este proyecto es aprovecharse de la robustez de las arquitecturas de las CNN, que han mostrado tener una eficacia muy buena siempre y cuando se plantee el problema

correctamente. Una CNN, y más en general, los modelos de Deep Learning, son capaces de aprenderse patrones y de identificar relaciones donde parece que no los hay, haciendo que estos sean los modelos cada vez más utilizados para todo tipo de problemas y en todo tipo de industrias (ANN para datos tabulares, RNN para datos secuenciales, CNN para imágenes y vídeos, etc.).

Si se pretende utilizar CNNs, ¿cómo se ha planteado entonces el problema para trabajar con *imágenes*?

3.4.1 CONCEPTO

En lugar de construir la variable de entrada X como un simple vector de datos del total, en el que cada valor es simplemente el valor del total de consumo en ese instante, se ha creado lo que se ha denominado *vizgrama*.

El vizgrama es una matriz que recoge información de la señal del total de consumo de forma temporal, así como información sobre la hora del día, día de la semana y mes del año, y se construye para recoger esa información de forma gráfica.

La idea principal es aportarle de forma indirecta a la CNN información de antes y de después del instante en el que se mide el total de la señal, para facilitarle la (difícil) tarea de identificar qué electrodomésticos estaban o no encendidos en ese instante dado.

3.4.2 PARÁMETROS

Para construir el vizgrama, es necesario decidir una serie de parámetros. Como se ha comentado antes, el vizgrama es una representación matricial de lo que está ocurriendo alrededor del instante t_0 que se quiere medir.

Retomemos el ejemplo anterior:

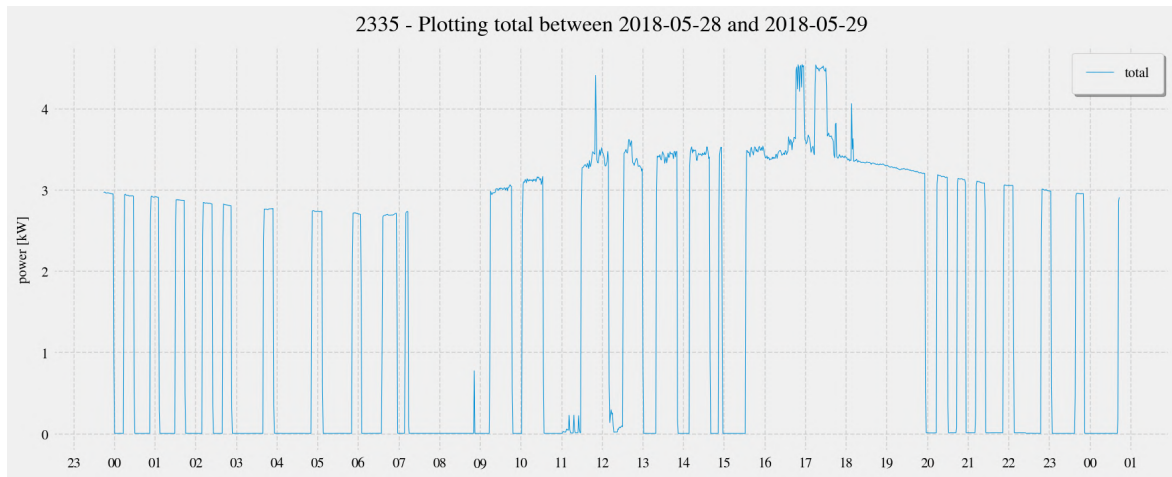


Figura 26 – Señal del total de consumo en un día cualquiera

Para un instante dado, por ejemplo, a las 12:00, el consumo total es de aproximadamente $3.3kW$. Este valor se puede dar en muchísimos otros instantes, sin que realmente haya un patrón (si solo se fija uno en la hora). De forma intuitiva, nuestro cerebro mira una señal como la que se ve en la Figura 26, y en seguida es capaz de adivinar que a las 17, estaba puesto el aire acondicionado (por el patrón que viene arrastrando la señal) y que en ese mismo instante se encendió otro aparato que, durando aproximadamente media hora, y de entre los electrodomésticos elegidos restantes (microondas, aparatos pequeños de cocina, lavaplatos y lavadora), difícilmente puede tratarse de otro electrodoméstico que no sea el lavaplatos o la lavadora.

Este es el razonamiento y la lógica que se pretende enseñar a la CNN. Para ello se tiene que dar entonces la información no solo de ese instante t_0 , sino la de $t_0 + n$ y $t_0 - n$.

3.4.2.1 Ventana

Para añadir esa información del antes y después, y como se quiere construir una matriz (y por lo tanto todas las filas tienen que ser del mismo tamaño), hay que decidir primero el tamaño de nuestra ventana, hasta dónde se quiere mirar hacia adelante y hacia atrás.

Se ha planteado un valor inicial de ventana de 24h, es decir, 12h hacia adelante y 12h hacia atrás para un instante dado. Se entiende que en un espacio temporal mayor a 12h ninguna acción que se realice tendrá mucha relación con el instante t_0 .

Si se pone una lavadora a las 17h, es poco probable que se ponga otra a las 20h, pero es algo más probable que se ponga a las 10h del próximo día, es decir 15h más tarde. De la misma forma, si se detecta un pico de x kW, y se sabe que ese valor se mantiene durante 2 horas, por ejemplo, se puede suponer que se trata de una lavadora.

3.4.2.2 Frecuencias de muestreo

El siguiente parámetro que se tiene que elegir para la construcción del vizgrama son las frecuencias de muestreo:

- ¿Cuántas frecuencias de muestreo distintas se quieren?
- ¿Qué intervalo se quiere entre dichas frecuencias?

Por ejemplo, dado un instante t_0 , si se muestrea cada 10 minutos hacia adelante y hacia atrás, el límite de la ventana en:

$$\frac{24 \cdot 60^1}{10} = 144 \quad (1)$$

144 muestras son las que se obtendrían muestreando cada 10 minutos. Esto implica que se pierde cierto detalle, puesto que se solo se está midiendo la señal cada 10 minutos, picos de potencia puntuales como al usar el microondas (cuyo uso es típicamente de menos de 5 minutos) se podrían perder fácilmente. Por esto es necesario añadir más frecuencias de muestreo.

¹ Se recuerda que el dataset dispone de datos cada minuto.

Si se desea por ejemplo tener 5 frecuencias de muestro distintas, es necesario determinar primero cuál va a ser la mayor, puesto que es la que impone el tamaño de la matriz.

Supongamos que se quieren tener 4 frecuencias de muestro distintas y que la máxima será de 10 minutos, ya que se estima que una frecuencia mayor es demasiado alta y se perdería mucho detalle. Las demás frecuencias de muestro serán 1, 2 y 5 minutos.

El tamaño máximo de las filas será de 144 (1), por lo que, el alcance de los muestreos será distinto en función de su frecuencia:

- 1 minuto: $1 \cdot 144 = 144 \text{ min}$, es decir 2h24, o 1h12 de cada lado del instante t_0
- 2 minutos: $2 \cdot 144 = 288 \text{ min}$, es decir 4h48, o 2h24 de cada lado del instante t_0
- 5 minutos: $5 \cdot 144 = 720 \text{ min}$, es decir 12h, o 6h de cada lado del instante t_0

En el problema que se plantea se ha decidido escoger 10 frecuencias de muestreo, equiespaciadas de 1 minuto, de 1 minuto a 10 minutos. La Figura 27 presenta el resultado de lo que se acaba de explicar.

Como se puede ver en dicha figura, las frecuencias de muestreo más bajas tienen menor alcance, puesto que ‘rellenan’ las muestras máximas (definidas por la mayor frecuencia de muestreo) mucho antes. También se observa muy bien como algunos picos detectados en frecuencias más bajas, van desapareciendo completamente (marcados en rojo).

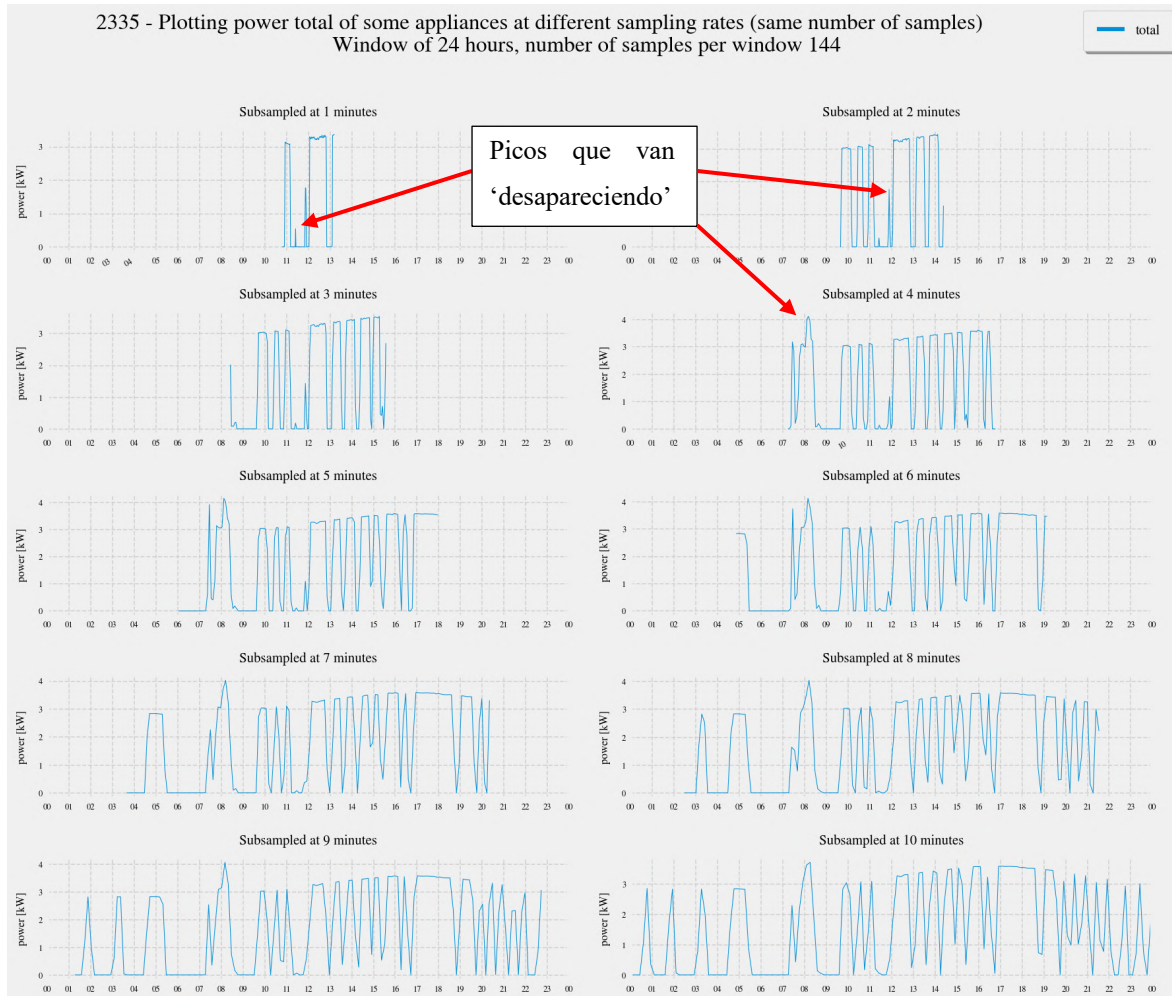


Figura 27 – Muestreo de la señal del total a diferentes frecuencias

3.4.2.3 Stride

El último parámetro que se tiene que elegir para la construcción del vizgrama es el ‘stride’, el salto temporal que existe entre cada instante t_0 que se desea medir.

Se recuerda que los datos escogidos contienen medidas cada minuto durante un año entero. Además, se ha descartado el primer y último día, ya que para la construcción del vizgrama en un instante t_0 se necesitan las 12 horas previas y posteriores, no se puede empezar en la hora 0 del primer día, y se ha decidido empezar en la hora 0 del segundo día y terminar en la hora 0 del penúltimo día. Lo que significa que se tiene un total de 363 días de datos, pudiendo recoger las 12 horas previas al primer día y las 12 horas posteriores al último.

Entonces, si por ejemplo se decide calcular un vizgrama cada 3 horas, se obtienen:

$$\frac{363 \cdot 24}{stride} = \frac{363 \cdot 24}{3} = 2904 \text{ Vizgramas}$$

Cuanto más pequeño sea este stride, más información redundante existirá entre vizgramas.

El resultado matricial de un vizgrama tiene la siguiente forma:

$$\begin{matrix} t_{0-n_{samp_1}} & t_{0-(n-1)_{samp_1}} & \dots & t_{0_{samp_1}} & \dots & t_{0+(n-1)_{samp_1}} & t_{0+n_{samp_1}} \\ t_{0-n_{samp_2}} & \dots & & & & & \\ \vdots & & & & & & \vdots \\ t_{0-n_{samp_i}} & t_{0-(n-1)_{samp_i}} & \dots & t_{0_{samp_i}} & \dots & t_{0+(n-1)_{samp_i}} & t_{0+n_{samp_i}} \end{matrix}$$

Siendo n el número de muestras obtenido dividido por 2 (ver (1)).

La Figura 28 muestra un ejemplo numérico de un vizgrama:

```
In [168]: print(vizgrama_t.shape)
          vizgrama_t
          (10, 144)

Out[168]: array([[0.01 , 0.01 , 0.009, ..., 0.008, 0.008, 0.007],
                  [0.005, 0.007, 0.007, ..., 0.005, 0.007, 0.007],
                  [0.008, 0.01 , 0.01 , ..., 0.008, 0.007, 0.006],
                  ...,
                  [0.005, 0.006, 0.008, ..., 0.005, 0.007, 0.006],
                  [0.004, 0.004, 0.004, ..., 0.008, 0.008, 0.007],
                  [0.007, 0.007, 0.005, ..., 0.005, 0.006, 0.006]])
```

Figura 28 – Ejemplo numérico de Vizgrama

Como se puede ver, el vizgrama tiene tamaño (10,144), ya que se han elegido 10 frecuencias de muestreo (de 1 min a 10 min), y el tamaño de las filas viene dado por (1).

Además, tal y como se ha comentado anteriormente que el vizgrama incluiría información sobre la hora del día, día de la semana y mes del año. Para representar esto de forma gráfica,

se ha optado por crear un vector de tamaño n muestras y se rellena la parte proporcional para representar dicha medida.

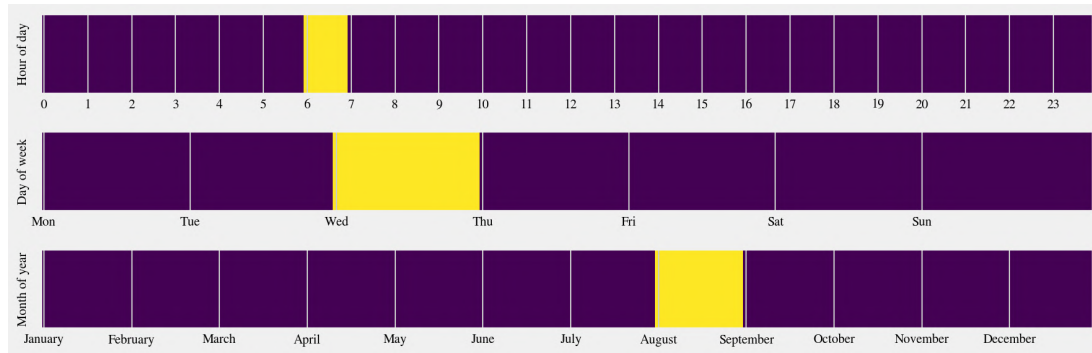


Figura 29 – Ejemplo gráfico de información de hora del día, día de la semana y mes del año

Finalmente, la Figura 30 es un ejemplo de vizgrama del 18 de agosto de 2018 centrado a las 12:00:00.

Se puede ver como las 3 primeras filas representan las 3 variables temporales como se deseaba. La primera fila indica la hora. Como se está representando un vizgrama centrado a la hora 12, la casilla que se rellena es la del medio. El 18 de agosto de 2018 fue un sábado, por lo que en la segunda fila aparece rellena la penúltima casilla. Finalmente se trata del mes de agosto, por lo que en la tercera fila aparece de color la octava casilla.

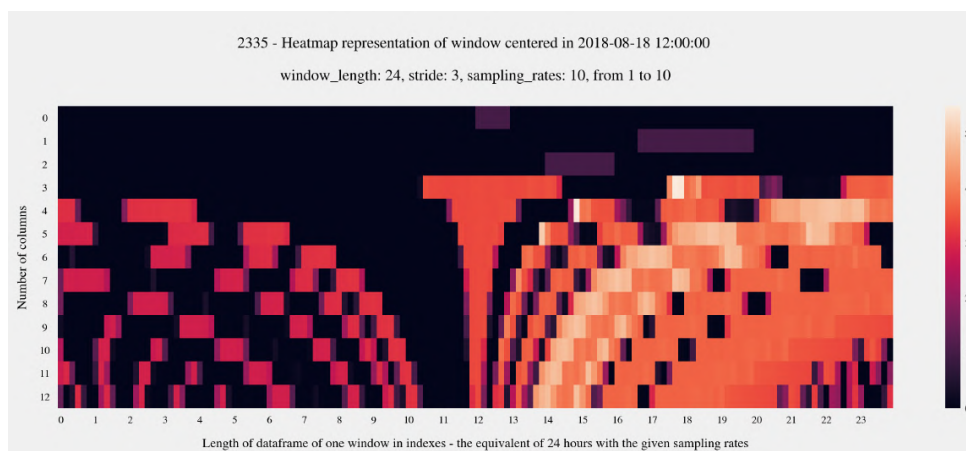


Figura 30 – Vizgrama

Para entender un poco mejor la figura y poder interpretarla mejor, se van a identificar los picos de consumo (colores más rojizos/blancos) en el vizgrama y en la gráfica de muestreos a distintas frecuencias:

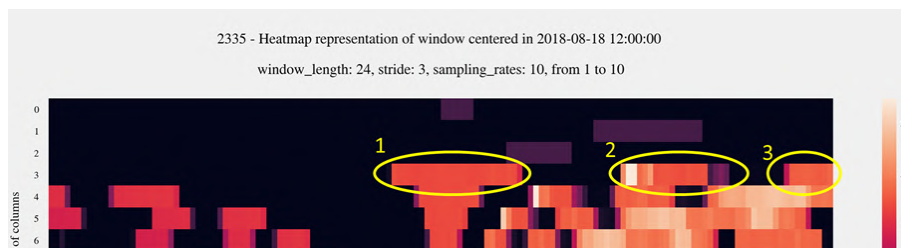


Figura 31 – Identificación de picos de potencia en un vizgrama en el primer muestreo

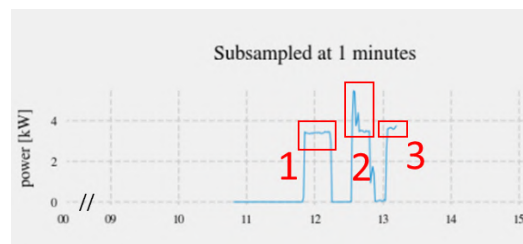


Figura 32 – Identificación de picos de potencia en la señal del total en el primer muestreo

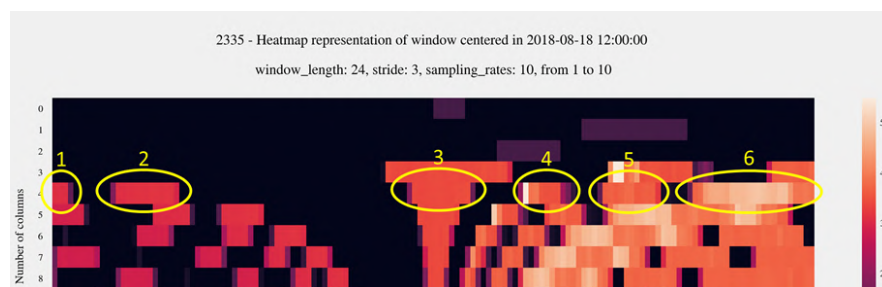


Figura 33 – Identificación de picos de potencia en un vizgrama en el segundo muestreo

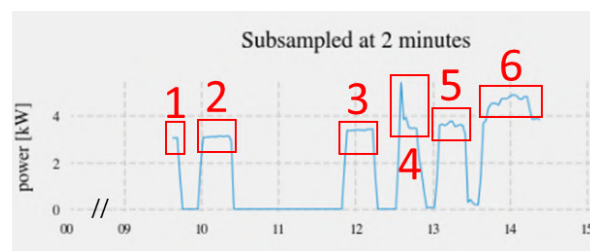


Figura 34 – Identificación de picos de potencia en un vizgrama en el segundo muestreo

La visión completa de este ejemplo de vizgrama y de sus distintas frecuencias de muestreo se muestran en las siguientes figuras (Figura 35 y Figura 36):

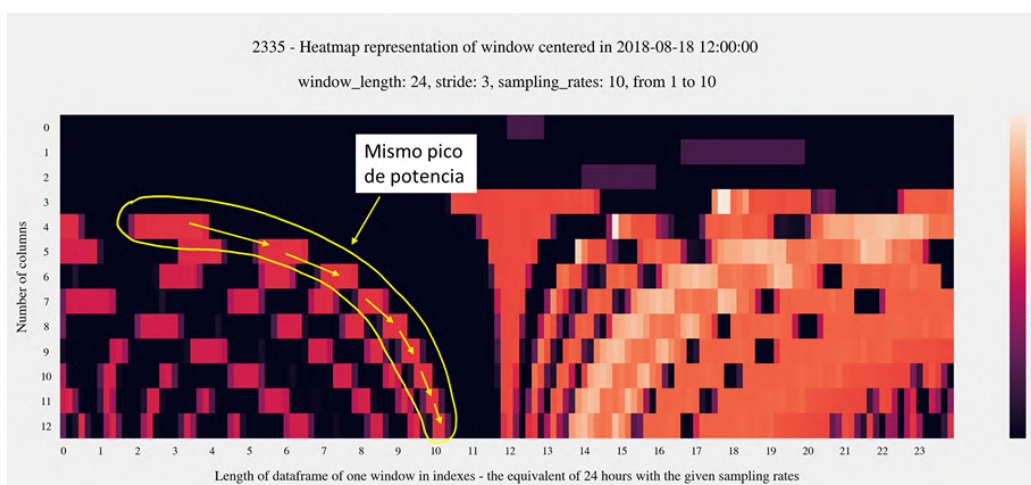


Figura 35 – Vizgrama

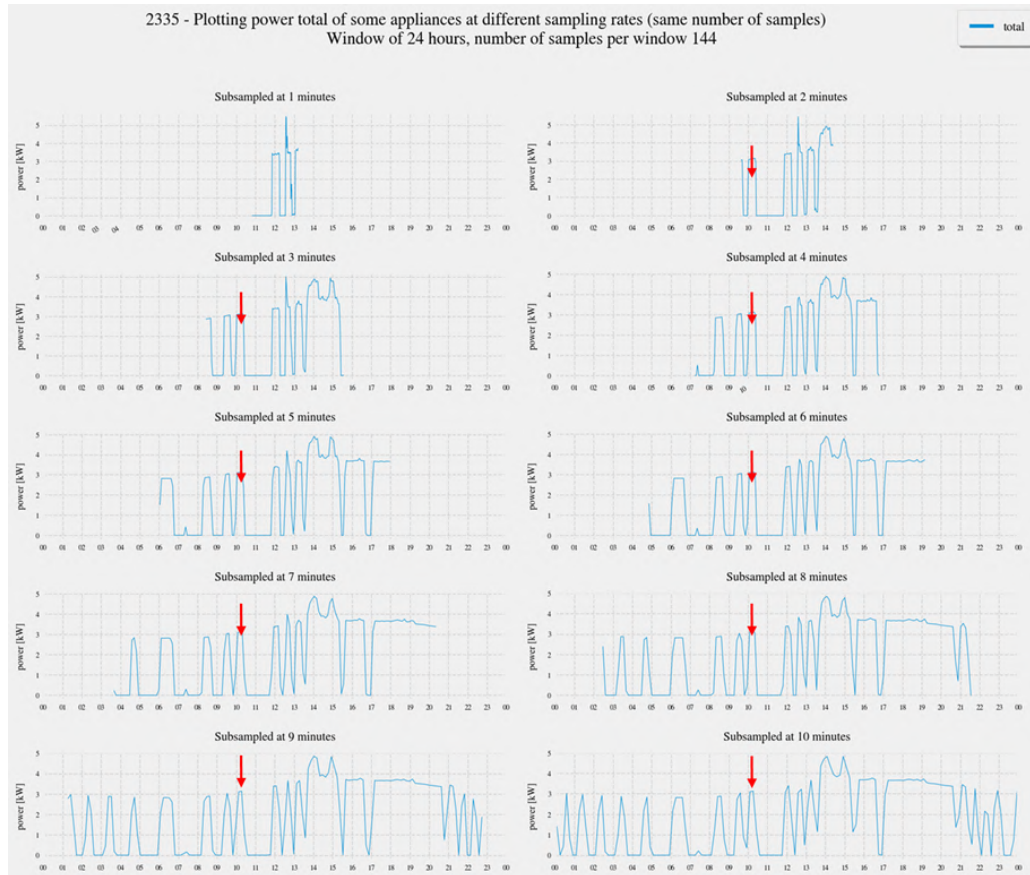


Figura 36 – Muestreos a diferentes frecuencias para construcción del vizgrama

Se observa como un patrón de ‘des-zoom’. A medida que se muestrea a una frecuencia mayor, se alcanza mayor horizonte temporal, y, relativo a una frecuencia de muestreo dada, lo observado en un instante se compacta sobre el eje central.

3.5 VARIABLE DE SALIDA

Llegados a este punto, solo queda construir la variable de salida. Como se ha comentado anteriormente, la variable de salida en este problema será un vector binario indicando la activación de cada electrodoméstico en cada instante.

Para ello hay que definir qué se considera activado y qué no, es decir, hay que definir un umbral por cada aparato.

Se ha definido un primer umbral en base a las series temporales de los electrodomésticos elegidos, tal y como se puede ver en la Figura 37:

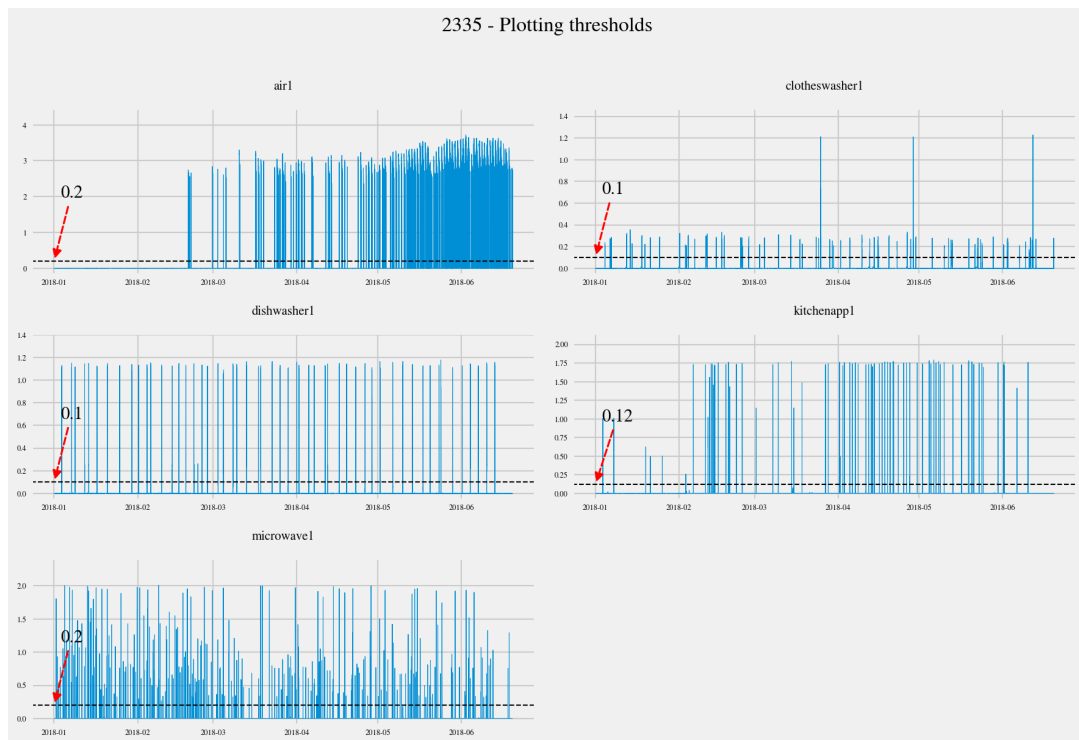


Figura 37 – Umbrales de activación en los distintos electrodomésticos

Con estos umbrales ya se puede construir la variable de salida:

In [88]: output_matrix_w_time

Out[88]:

	air1	clotheswasher1	dishwasher1	kitchenapp1	microwave1	localminute
0	0	0	0	0	0	2018-01-02 00:00:00
1	0	0	0	0	0	2018-01-02 03:00:00
2	0	0	0	0	0	2018-01-02 06:00:00
3	0	0	0	0	0	2018-01-02 09:00:00
4	0	0	0	0	0	2018-01-02 12:00:00
...	...	...	...	...	...	...
2899	0	0	0	0	0	2018-12-30 09:00:00
2900	0	0	0	0	0	2018-12-30 12:00:00
2901	0	0	0	0	0	2018-12-30 15:00:00
2902	0	1	0	0	0	2018-12-30 18:00:00
2903	0	0	0	0	0	2018-12-30 21:00:00

2904 rows × 6 columns

Figura 38 – Variable de salida

3.6 PROCESADO DE DATOS

Con los parámetros definidos en los apartados anteriores, se ha creado un vizgrama y una variable de salida asociada para cada instante medido (cada 3h). La siguiente figura muestra la variable de salida construida con los umbrales definidos.

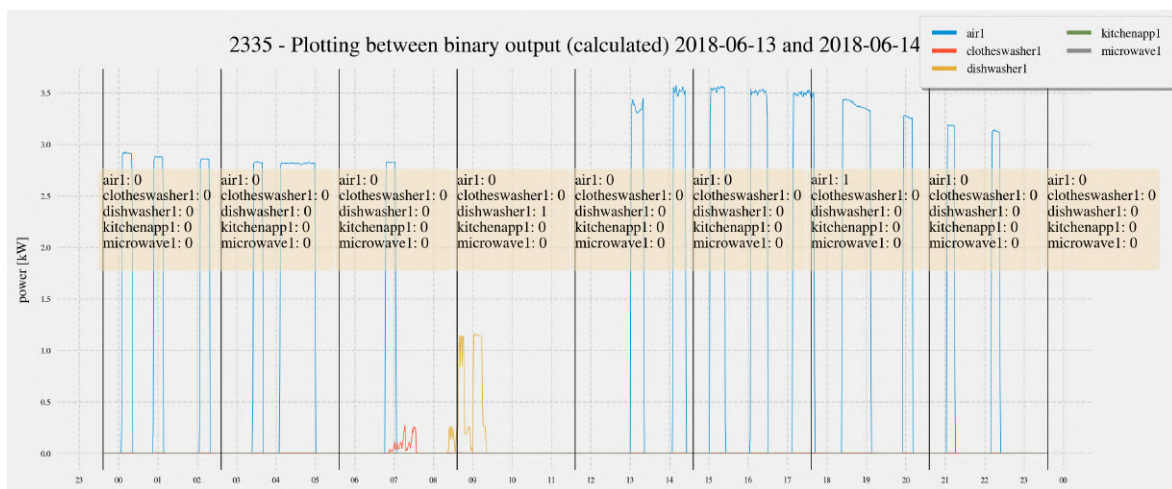


Figura 39 – Comprobación de activaciones definidas por umbrales – vizgramas cada 3h

En esta se puede ver por un lado cómo los umbrales elegidos parecen ser adecuados. Por ejemplo, justo antes de la hora 9, se puede ver que la variable de salida para ese instante y para el electrodoméstico `dishwasher1` vale correctamente '1', y para la hora 17h30 aproximadamente, el valor para `air1` también es correcto. Sin embargo, el stride elegido de 3h es inadecuado, es demasiado alto, haciendo que no se recoja información con suficiente frecuencia y que por lo tanto se pierda demasiada información entre cada medición.

Por ello se han vuelto a calcular los vizgramas y la salida correspondiente a cada vizgrama, pero con un stride de 1 minuto, es decir, por cada dato que se tenía originalmente.

Esto significa que se generan un total de $363 \cdot 24 \cdot 60 = 522.720$ vizgramas

En la Figura 40 se puede ver un ejemplo en el que se detecta perfectamente la rampa de activación de la lavadora y se ve como los valores creados en la salida toman correctamente el valor 'on' al pasar el umbral definido.

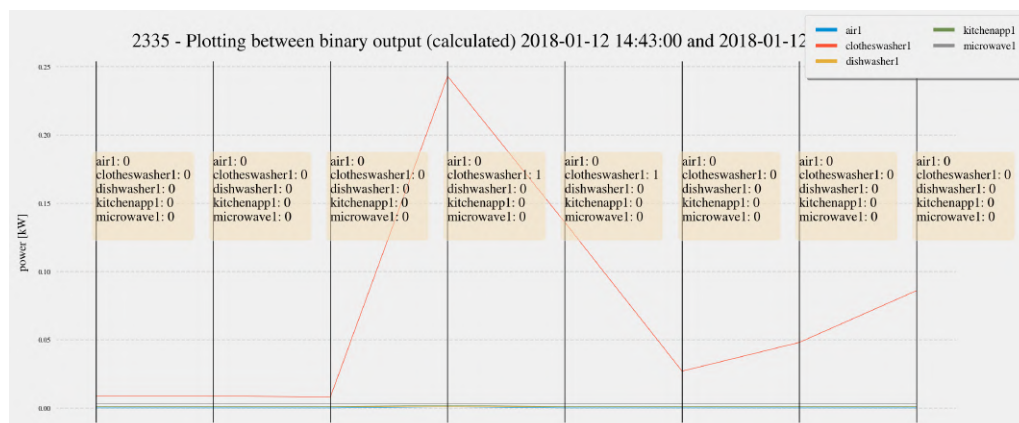


Figura 40 – Comprobación de activaciones definidas por umbrales – vizgramas cada minuto

Como última comprobación de los datos, se han representado las activaciones binarias (la salida que se ha construido) en una matriz, cuyas filas son las 24 horas del día y las columnas son los 363 días de datos (Figura 41 a Figura 45)².

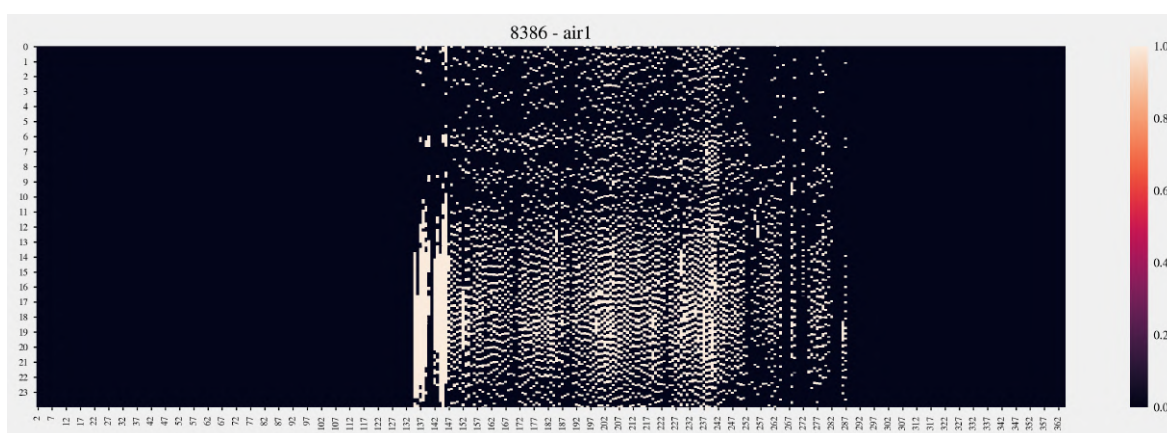


Figura 41 – Representación temporal de la salida Y – air1

² Llegados a este punto del trabajo, se decidió cambiar de hogar, de la casa 2335 a 8386. Todo lo realizado anteriormente es válido también para este hogar.

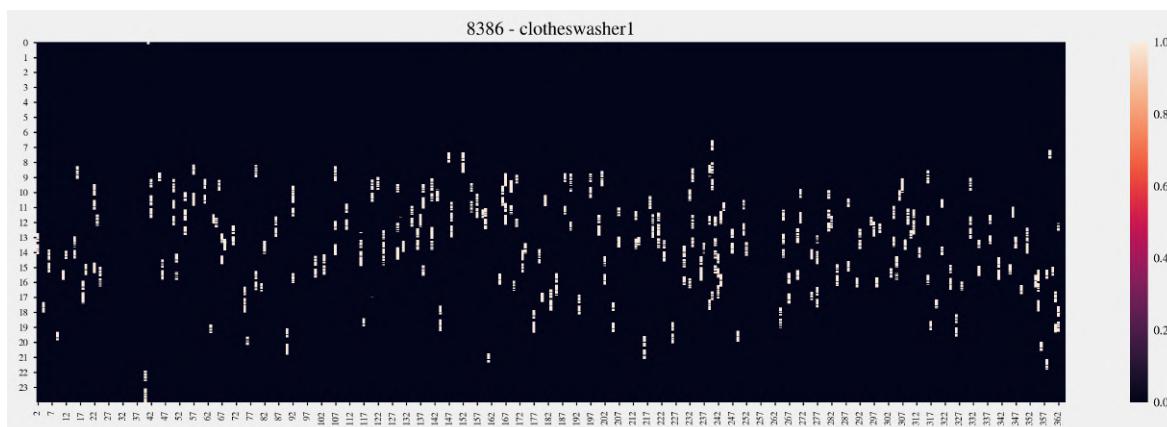


Figura 42 – Representación temporal de la salida Y – clotheswasher1

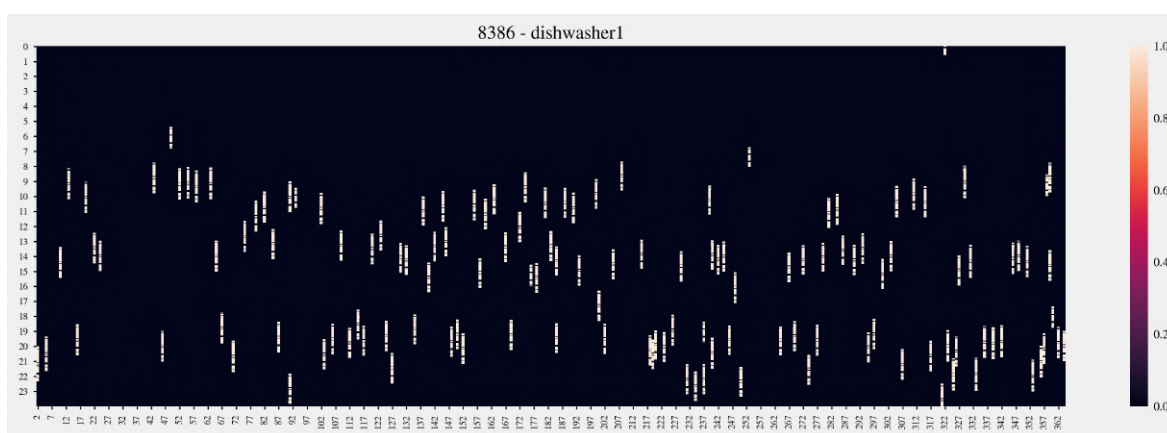


Figura 43 – Representación temporal de la salida Y – dishwasher1

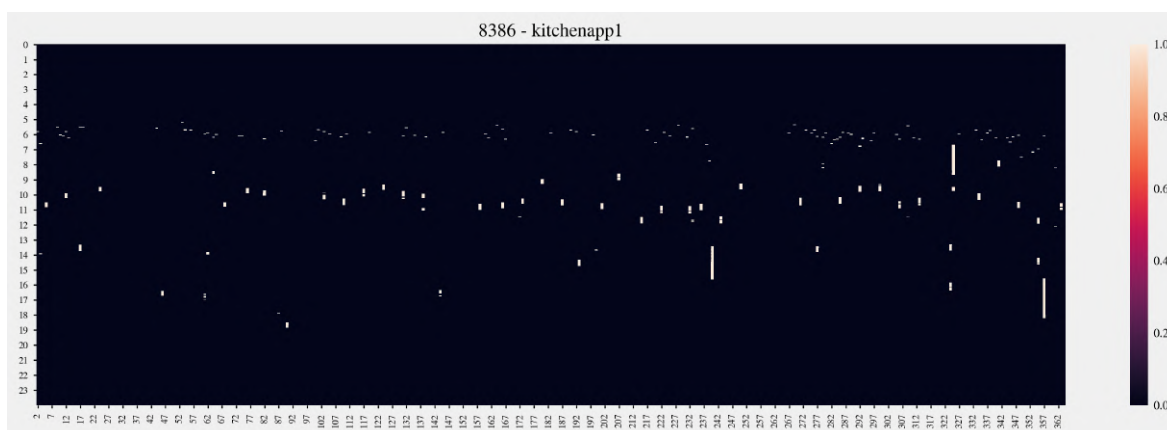


Figura 44 – Representación temporal de la salida Y – kitchenapp1

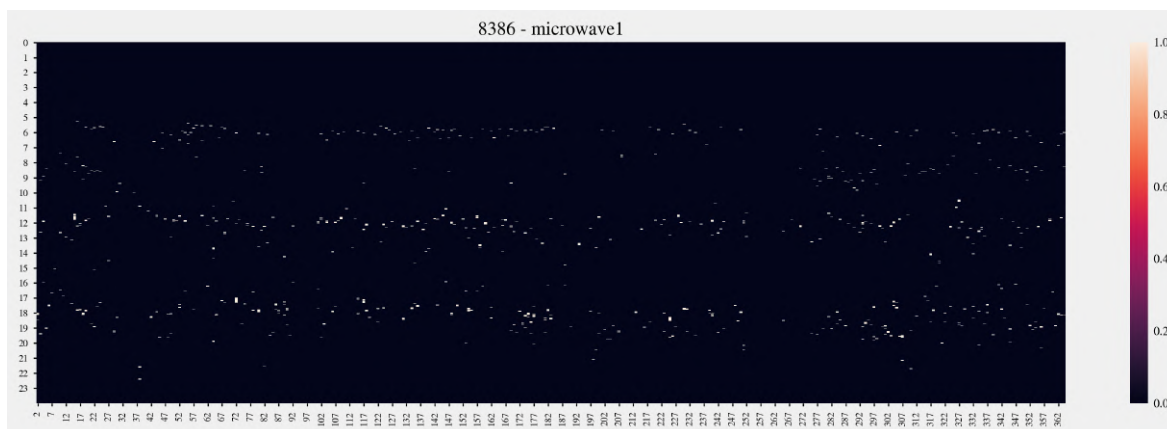


Figura 45 – Representación temporal de la salida Y – microwave1

Llama la atención el caso de la lavadora y del lavaplatos, ya que se pueden ver que muchos bloques de activaciones están segmentados, mientras que lo deseable sería tener una activación completa (es decir, un bloque continuo) puesto que no tiene sentido poner 3 lavadoras seguidas, o casi todos los días 2 lavadoras seguidas (lo mismo para el lavaplatos). Esto parece indicar que quizás si se bajase un poco el umbral de activación para estos electrodomésticos, se pueda conseguir que salgan esas activaciones enteras.

Se han bajado los umbrales de 0.1 a 0.01 y de 0.1 a 0.001 para la lavadora y el lavaplatos respectivamente. Estos son los resultados con estos cambios:

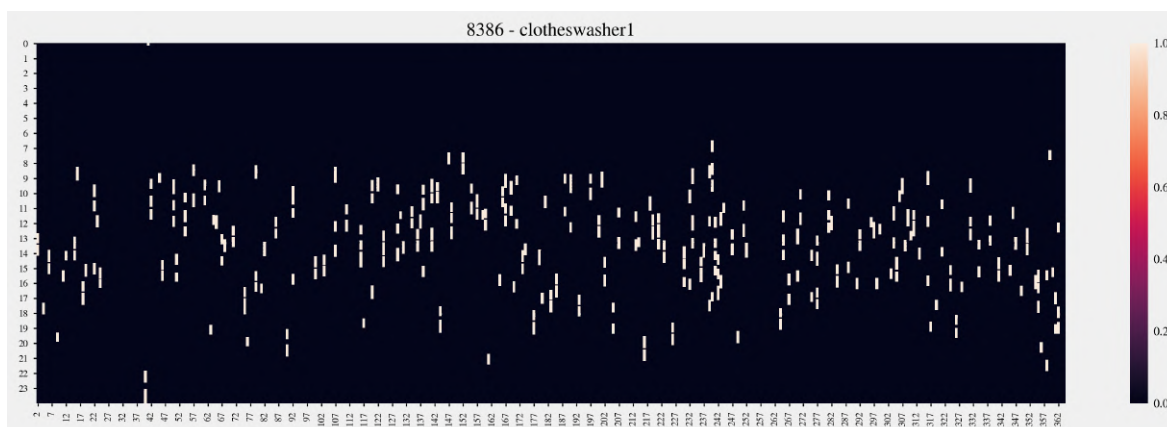


Figura 46 – Representación temporal de la salida Y – Umbral redefinido – clotheswasher1

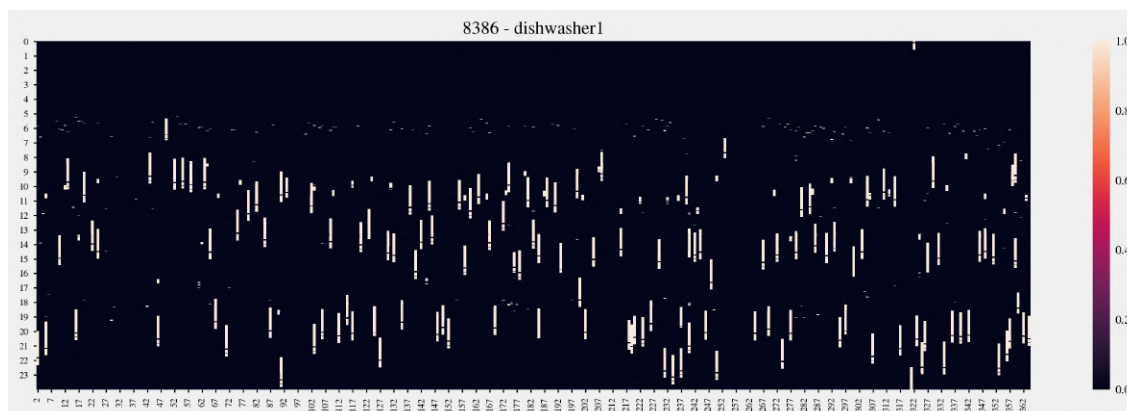


Figura 47 – Representación temporal de la salida Y – Umbral redefinido – dishwasher1

Se puede observar cómo las activaciones son mucho más continuas y que aparecen más las formas de bloque que se comentaban anteriormente. Sin embargo, también se observa que aparece ruido no deseado. Esto es relativamente sencillo de filtrar. Se ha diseñado un pequeño algoritmo de limpieza que detecta activaciones inferiores a un umbral razonable para estos electrodomésticos y que filtra ese ruido. Las figuras a continuación muestran los resultados de dicho filtrado:

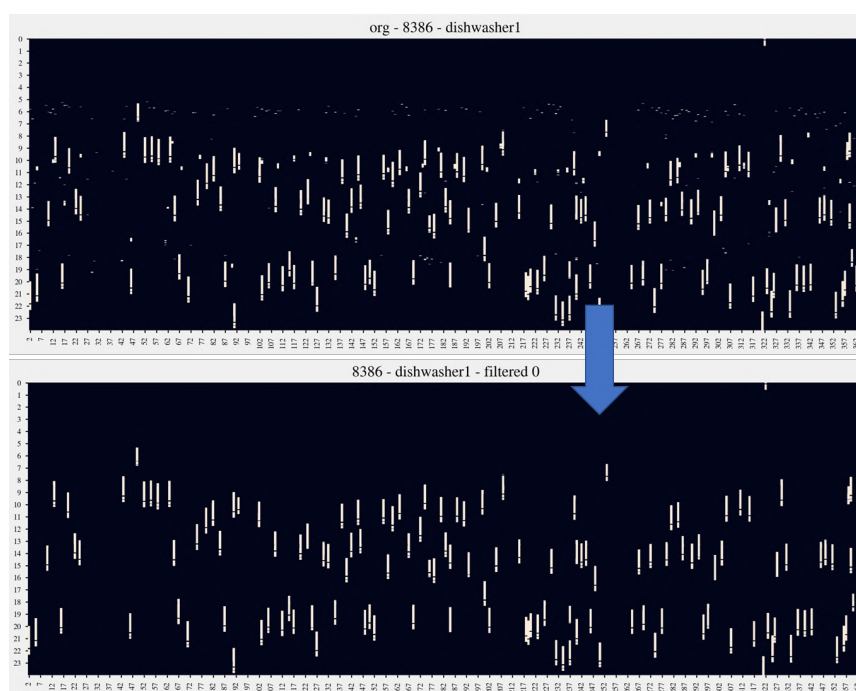


Figura 48 – Algoritmo de filtrado – dishwasher1

Se puede ver cómo todavía quedan algunos huecos en las activaciones binarias que sería deseable que fueran continuas. Para ello, se implementó un algoritmo similar al de filtrado. En esta ocasión, el algoritmo rellena los huecos y su funcionalidad es parecida al del algoritmo de filtrado. Para cada electrodoméstico, se detectan los números de ‘0’ seguidos que hay, y si es inferior a un cierto umbral, se rellenan con ‘1’. Los resultados con este algoritmo de relleno se muestran en la Figura 49.

Se realiza lo mismo para la lavadora.

Tras aplicar la metodología explicada, se obtiene una salida binaria representativa de la realidad, y se tienen preparados los vizgramas y la salida para poder aplicarlos en modelos de Machine Learning.

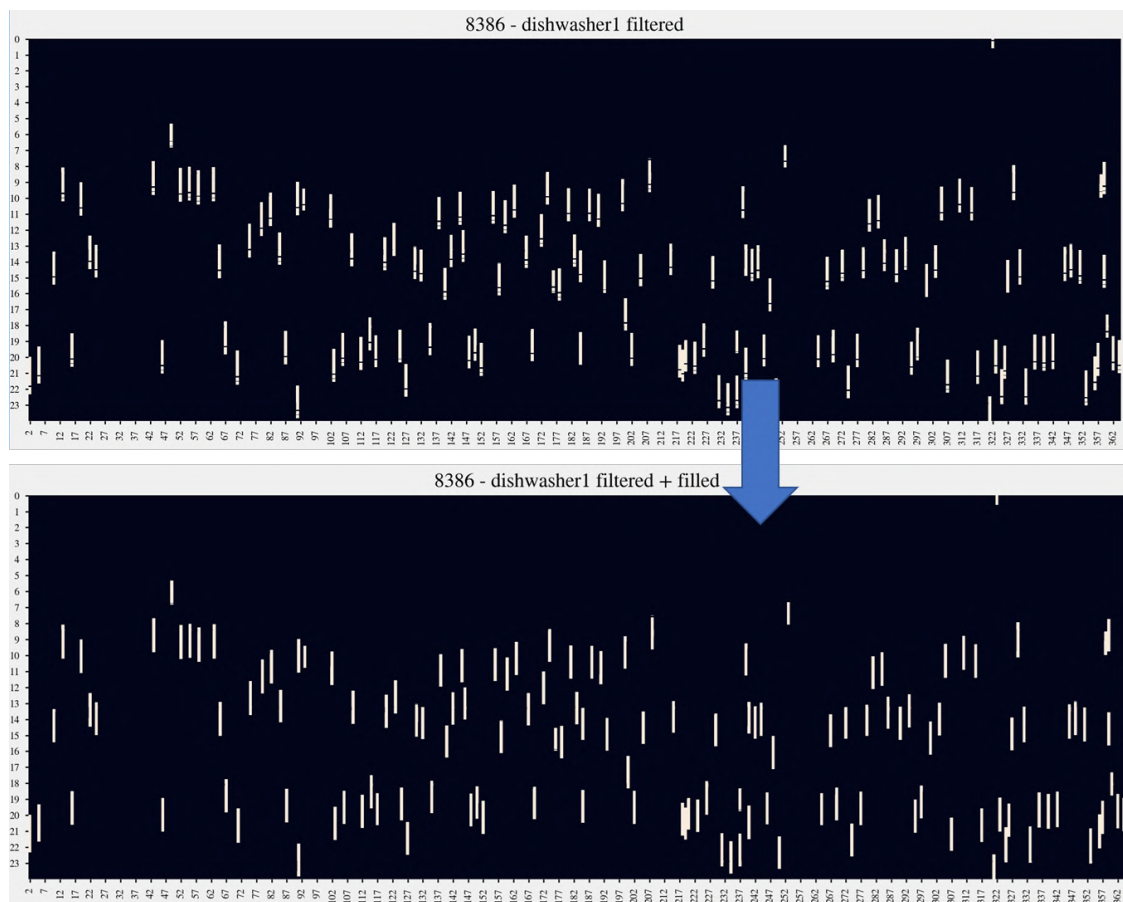


Figura 49 – Algoritmo de relleno – *dishwasher1*

3.7 PREPARACIÓN DE CONJUNTOS DE ENTRENAMIENTO

Observando las representaciones anteriores de la salida Y que se ha construido, se puede ver que la matriz que se está representando tiene por columnas el número de datos correspondientes a un día, para que sea interpretable gráficamente. De cara a la preparación de los conjuntos de datos de train y test esto puede plantear un problema.

Como para cualquier problema de Machine Learning, es necesario mezclar los datos de forma aleatoria, entrenar un modelo sobre una parte de ese conjunto mezclado y validar sobre el resto. Sin embargo, en este caso, si se mezclan los datos de forma aleatoria se perdería la temporalidad de las activaciones por lo que el modelo no podría identificar si el perfil de consumo es de un determinado electrodoméstico u otro. Por ejemplo, si en un determinado instante un electrodoméstico está 80 minutos encendido (y por lo tanto la salida contiene 80 '1' seguidos), romper esta continuidad parece contraproducente para el modelo, haciendo más difícil una tarea ya de por sí complicada.

Por lo anterior, se decidió mantener la temporalidad horaria dentro de un día y mezclar días completos (en el caso de que una activación se prolongue de las 23h59 a las 00h30 del día siguiente, por ejemplo, se seguiría perdiendo esa trazabilidad de la activación, pero se asume que son muy pocos casos). Se ha decidido dividir los datos (días) en un tradicional 80-20 para conjuntos de train y test respectivamente.

3.8 IDENTIFICACIÓN Y AJUSTE DE MODELO MLP

Con las variables de entrada X y de salida Y construidas y procesadas, y con los conjuntos de train y test definidos, ya se puede empezar a plantear un primer modelo de clasificación. Se ha decidido trabajar primero con MLP (*MultiLayer Perceptron*), cuyo esquema general se puede ver en la Figura 50.

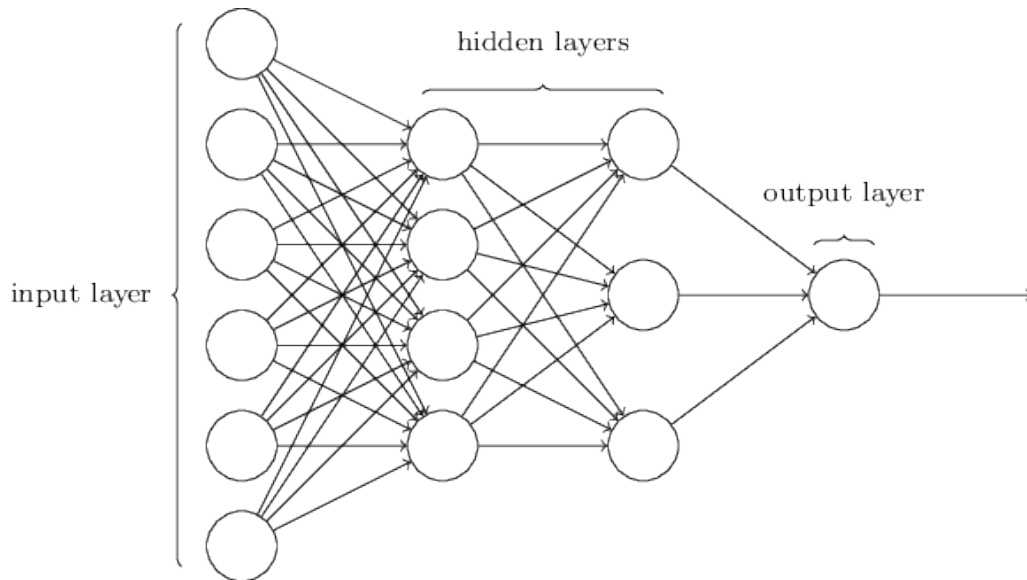


Figura 50 – Esquema genérico de un MLP [38]

Esta decisión se fundamenta principalmente por el hecho de que la construcción de los vizgramas y de las salidas también sirve como entrada para un MLP, incluso aunque este último no se beneficie de la idea visual que hay detrás del vizgrama. Al fin y al cabo, el vizgrama no es más que una representación visual de una matriz, cuyos datos se pueden utilizar de forma ‘tradicional’, y en este caso, sin necesidad de realizar ningún cambio adicional. Además, un MLP clásico (con una capa oculta) con suficientes neuronas en la capa oculta es un aproximador universal [40].

En cuanto a las librerías disponibles para construir un MLP, se ha decidido trabajar con la librería **sklearn**, y no con **Tensorflow**. **Sklearn** es una librería de más alto nivel que permite construir modelos de forma sencilla sin tener que adentrarse en el detalle del modelo. Esto permite asegurarse de que el MLP que se construya valga de modelo de referencia, ya que, si se realiza construye un MLP con Tensorflow y se le va añadiendo capas de regularización y optimización, deja ser un modelo sencillo y básico y empieza a entrar en el campo de las DNN, en cuyo caso la comparativa con la CNN no aporta tanto valor.

Por ello se ha creado un modelo de estructura simple con dicha librería para tener una referencia a la hora de evaluar los resultados obtenidos con las CNNs y poder compararlos.

En teoría, si se ha planteado correctamente el problema, las CNN deberían aportar mejores resultados, y en el caso de no serlos, se debería conseguir reducir el número de parámetros considerablemente sin comprometer los resultados, convirtiéndolas en una opción viable e interesante.

El primer modelo que se ha realizado es un MLP de 3 capas: 1 de entrada del tamaño de un vizgrama, 1 capa oculta con 15 neuronas y una capa de salida de 5 neuronas. La última capa es de tamaño 5 ya que se tienen 5 dispositivos que se quieren detectar. Las activaciones en las neuronas de la capa oculta como en las de la salida son sigmoidales, y el resto de los parámetros utilizados son los que vienen por defecto en la librería sklearn.

Además, se ha decidido realizar un modelo por electrodoméstico en lugar de un modelo predictivo para todos. Esto permite que el entrenamiento individual sea más liviano (ya que no se necesita el mismo número de parámetros para una red que intente detectar un solo aparato) y por lo tanto facilita también el reentrenamiento de los modelos en el caso de tener que recalcular la entrada o la salida (por ejemplo, si se retoca el algoritmo de lavado y relleno de datos que se ha comentado anteriormente, y se vuelve a aplicar a un solo electrodoméstico, solo se tendría que reentrenar el modelo específico a ese electrodoméstico).

Por último, se destaca que, con esta librería, los tiempos de entrenamiento son mucho mayores que con Tensorflow (16h frente a 2-3h). Esto se debe principalmente a que sklearn es una librería de más alto nivel, ofreciendo paquetes para utilizar algoritmos sin necesitar un conocimiento profundo (SVMs, Random Forest, Regresiones Logísticas, etc) mientras que Tensorflow es una librería de más bajo nivel que permite construir un modelo con mayor control y que por lo tanto está mucho mejor optimizada. Por otro lado, como se ha intentado utilizar la versión más básica de MLP que ofrece sklearn, las activaciones de la capa oculta son sigmoidales, lo que hace que el entrenamiento de la red sea mucho más lenta e ineficiente (sobre todo en la fase de backpropagation).

En cuanto al proceso de entrenamiento, se ha seguido un acercamiento incremental, partiendo del diseño y creación de un modelo sencillo, midiendo los errores en train y test, y realizando los cambios necesarios para reducir dichos errores al máximo (ajustando los hiperparámetros adecuados, complicando el modelo o cambiando la variable de salida).

Los resultados de este modelo se presentan en el capítulo 4.

3.9 IDENTIFICACIÓN Y AJUSTE DE MODELO CNN

El vizgrama se ha diseñado con la idea de resolver este problema con una CNN, aprovechando que este tipo de redes buscan y extraen automáticamente características de una imagen.

La Figura 51 representa la arquitectura de la red AlexNet, desarrollada en 2012 para la competición ImageNet.

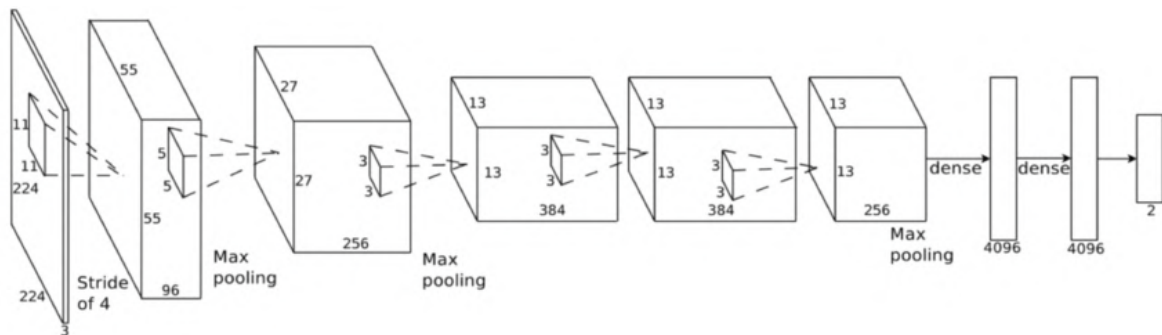


Figura 51 – Arquitectura AlexNet [41]

El primer parámetro relevante de una CNN es el tamaño de los filtros de las capas convolucionales. Este tamaño tiene que ser elegido en función del tamaño de la imagen de entrada y de cuánto detalle es necesario identificar en dicha imagen. Por ejemplo, si el tamaño de la imagen de entrada es muy grande y es necesario identificar elementos pequeños, es necesario elegir filtros más pequeños (3x3 o 5x5), mientras que de lo contrario se elegirían filtros más grandes (11x11 o 9x9). En nuestro caso, como la imagen es pequeña,

se pueden utilizar filtros más pequeños y adecuados para el tamaño de imagen del vizgrama (13x144), no siendo necesario emplear filtros de mayor tamaño que 5x5 o incluso 3x3.

Cada capa de convolución añade dimensionalidad a los datos (tantas dimensiones por cantidad de filtros que tenga la capa de convolución, como se puede ver en la Figura 52), por lo que a medida que se avanza en la arquitectura, es necesario reducir esa dimensionalidad para que no aumente exponencialmente el número de parámetros que optimizar.

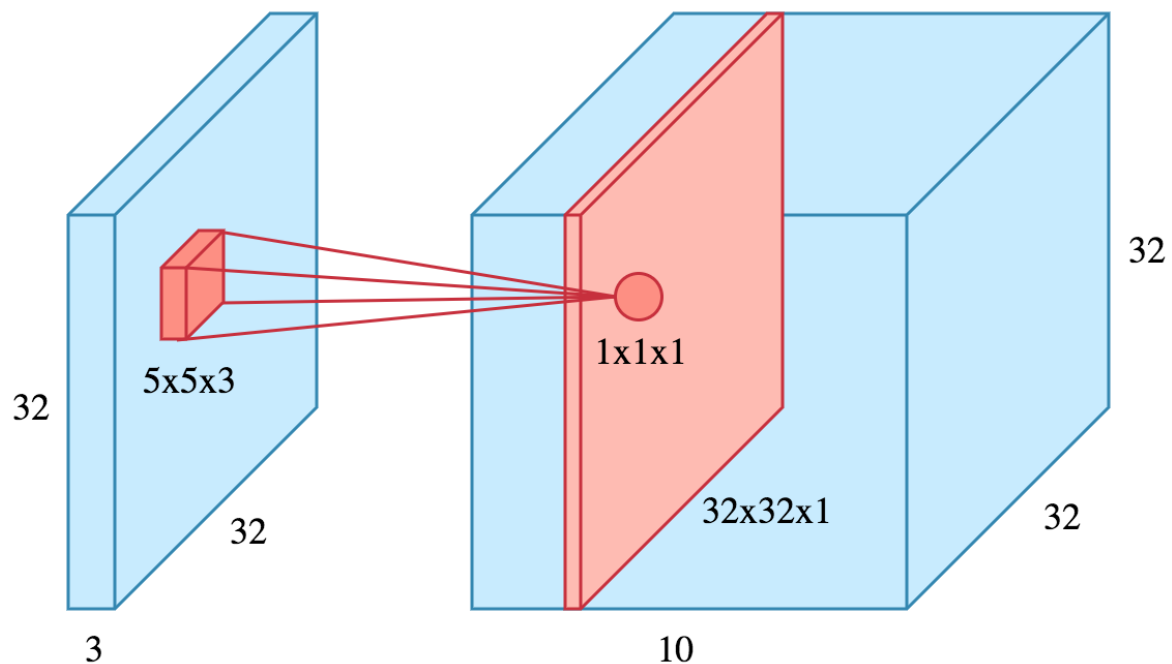


Figura 52 – Ejemplo de capa convolucional de 10 filtros de tamaño 5x5, stride de 1 y padding de 1 [43]

Para reducir esta dimensionalidad, tradicionalmente se han utilizado capas de pooling (y en particular la de max pool), como se puede ver en la Figura 53.

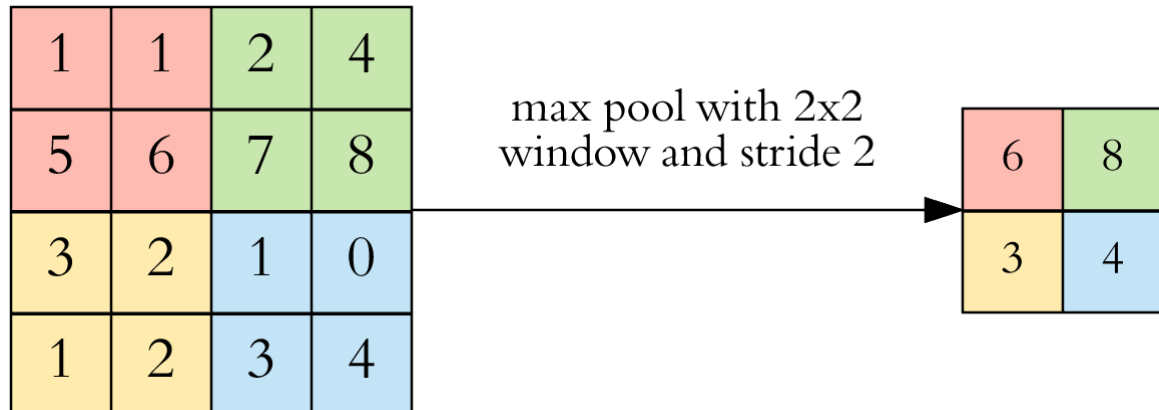


Figura 53 – Ejemplo de capa de max pool de tamaño 2x2, stride de 2 [43]

Peroen 2014, Sprigenber et al. publicaron un artículo científico demostrando que reemplazar las capas de pooling por convoluciones con stride podía mejorar el rendimiento del modelo disminuyendo al mismo tiempo su complejidad [42].

Otro aspecto de vital importancia en redes CNN (extrapolable a cualquier DNN realmente), son las distintas técnicas de regularización [39]. Entre ellas podríamos destacar las más utilizadas:

- Las capas de dropout
- Las capas de batch normalization
- Regularización en la capa de salida (típicamente L1 o L2)

Dropout es sin duda la más común de las técnicas de regularización para DNN [46]. El dropout se utiliza para evitar que haya overfitting, y su funcionamiento es relativamente sencillo. Durante el entrenamiento de la red, en cada iteración se ‘tira’ (drop) una neurona (es decir, se deshabilita) con una cierta probabilidad p . Esto implica que todas las entradas y salidas de esa neurona quedan desactivadas en la iteración que esté en marcha. Las neuronas desactivadas se recalculan con probabilidad p en cada paso del entrenamiento, por lo que una neurona desactivada en un determinado paso puede volver a estar activa en el siguiente. El hiperparámetro p se llama dropout rate es suele estar en torno a 0,5.

Batch normalization es una técnica que estandariza la entrada de una capa para cada mini-batch, con el objetivo de estabilizar el proceso de aprendizaje y de reducir drásticamente el número de épocas necesarias para entrenar DNN [44] y [45].

La regularización L1 y L2 tienen como objetivo penalizar la complejidad del modelo en la etapa de back propagation, añadiendo un término que penaliza el uso de pesos grandes.

Después de elegir el número de capas convolucionales con sus respectivos parámetros, se necesitan capas densas (fully connected layer) que permitan clasificar todas características extraídas.

En cuanto al proceso de entrenamiento, se ha seguido un proceso iterativo e incremental, partiendo del diseño y creación de un modelo sencillo, midiendo los errores en train y test, y realizando los cambios necesarios para reducir dichos errores al máximo (ajustando los hiperparámetros adecuados, añadiendo complejidad al modelo o cambiando la variable de salida).

Los resultados de este modelo se presentan en el capítulo 4.

Capítulo 4. CASOS DE USO

4.1 RESULTADOS MLP

Tal y como se ha explicado anteriormente, el número total de vizgramas es de 522720, cada uno de un tamaño $13 \cdot 144 = 1872$, por lo que, por cada aparato, se tiene el mismo número de valores en la salida.

El número de parámetros de este modelo MLP es de 29000, muy superior al número de variables de entrada. A pesar de que esto pueda parecer indicativo de que el modelo vaya a hacer overfitting, las distintas técnicas de regularización pueden prevenir en gran medida este problema [39], aunque este tipo de técnicas se suele incorporar más con librerías de más bajo nivel (como Tensorflow) que ofrecen más personalización a la hora de construir modelos y en modelos más complejos, razón por la cual el modelo MLP construido con sklearn no tiene métodos de regularización.

Además, este problema es un caso de trabajo donde los datos de salida no están equilibrados, ya que la mayoría del tiempo no están los aparatos encendidos, haciendo que las métricas típicas de evaluación de modelos (accuracy) sean poco interpretables. La siguiente figura indica la proporción de 0 y 1 que existen en las variables de salida:

```
Number of 0 air1: 475676 - prop: 91.0%
Number of 1 air1: 47044 - prop: 9.0%

Number of 0 clotheswasher1: 515820 - prop: 98.68%
Number of 1 clotheswasher1: 6900 - prop: 1.32%

Number of 0 dishwasher1: 506273 - prop: 96.854%
Number of 1 dishwasher1: 16447 - prop: 3.146%

Number of 0 kitchenapp1: 520875 - prop: 99.647%
Number of 1 kitchenapp1: 1845 - prop: 0.353%

Number of 0 microwave1: 520967 - prop: 99.665%
Number of 1 microwave1: 1753 - prop: 0.335%
```

Figura 54 – Proporción de valores en la variable de salida

Como se puede observar, si un modelo determinado simplemente previera todos los valores a 0, estaría acertando porcentajes altísimos, del orden del 99%. Por este motivo, en este trabajo no vamos a utilizar la medida de accuracy de forma aislada, sino que nuestro objetivo será batir este porcentaje de acierto. Es decir, se buscarán porcentajes de acierto superiores a los de la Figura 54.

Finalmente, se destaca que en el apartado 4.3 se realizará una comparativa entre los datos reales y los obtenidos con ambos modelos, en base a una serie de métricas que son más representativas de un caso de uso real.

4.1.1 EVALUACIÓN

Para poder entender mejor los resultados de los distintos modelos que se van a realizar, además de utilizar la métrica tradicional del accuracy del modelo (principalmente para ir haciendo un barrido de hiperparámetros y elegir así el mejor modelo MLP y el mejor modelo CNN), se ha creado una representación gráfica de los aciertos y fallos de un modelo, equivalente a una matriz de confusión.

La Figura 55 muestra un ejemplo de dicha visualización.

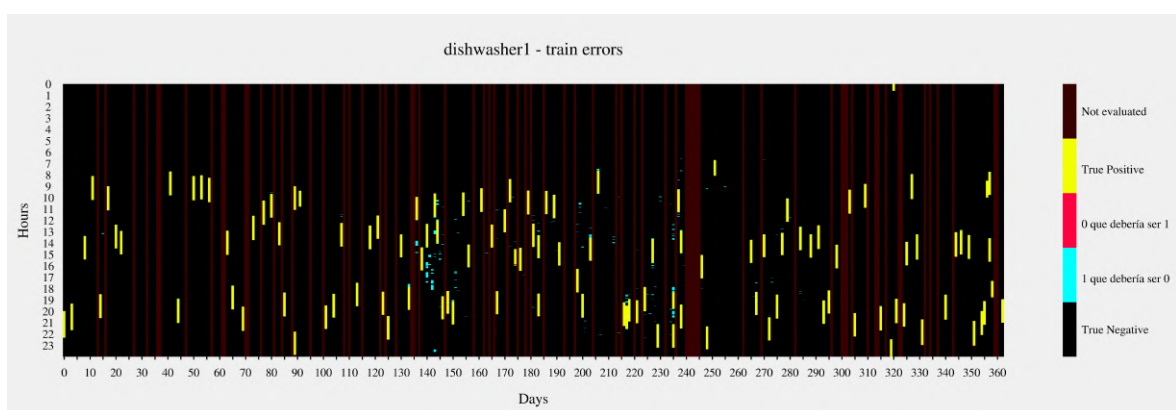


Figura 55 – Representación gráfica de los resultados de la matriz de confusión

En esta, se pueden ver en un color rojo semitransparente los días que no han sido evaluados. Por ejemplo, en el caso de representar las predicciones del set de train, los días utilizados para el set de test se representarán en dicho color.

Los días que sí han sido evaluados, se han representado los 4 tipos de aciertos/fallos con distintos colores.

- *True Positive (TP): un 1 bien acertado, en verde*
- *False Negative (FN): un 0 que debería ser 1, en rojo claro*
- *False Positive (FP): un 1 que debería ser 0, en azul claro*
- *True Negative (TN): un 0 bien acertado, en negro*

Para que este documento no resulte excesivamente denso, solo se van a mostrar los resultados de un electrodoméstico, el lavavajillas, así como una tabla resumen con los porcentajes de acierto de los distintos electrodomésticos. El resultado del resto de electrodomésticos podrá consultarse en el ANEXO A – Resultados MLP.

En el caso del dishwasher1:

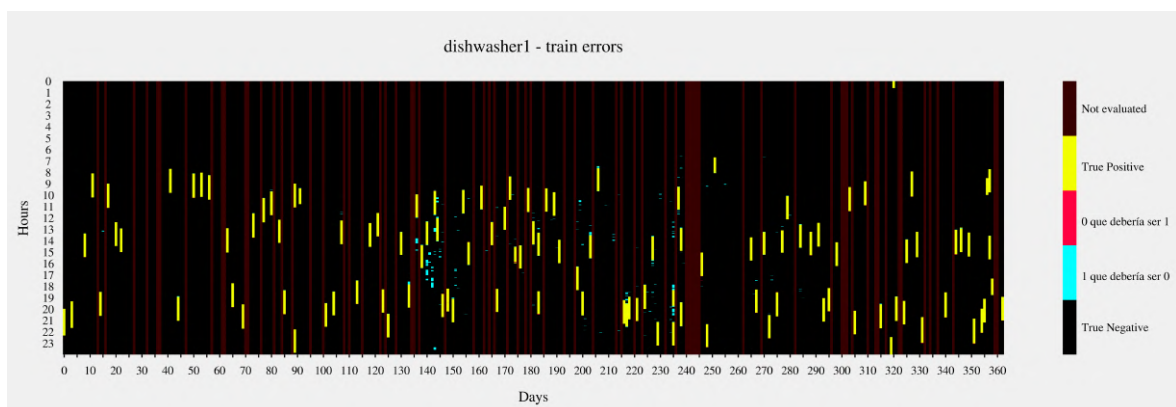


Figura 56 – Resultados mejor modelo MLP – dishwasher1 – train



Figura 57 – Resultados mejor modelo MLP – *dishwasher1* – test

Como se puede observar en la Figura 56, los resultados en train son buenos, comprobando de un vistazo que hay pocos FP o FN. De forma similar, en el set de test (Figura 57), los resultados son razonablemente buenos para un primer modelo, apreciándose sin embargo más falsos positivos entre los días 130 y 140, y más falsos negativos entre los 210 y 240.

La representa un zoom de algunos de esos errores, entre los días 110 y 160.

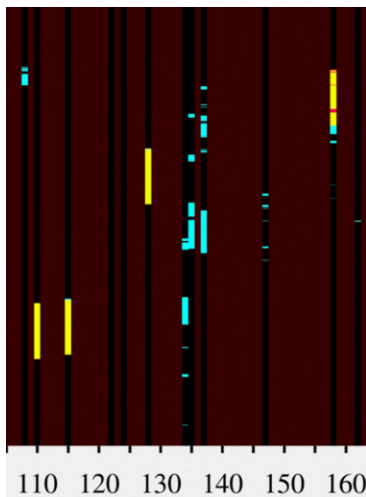


Figura 58 – Zoom en días 110-160 de la evaluación del mejor modelo MLP – test

En esta figura se puede observar cómo algunos de los errores son intermitentes. Por ejemplo, en el día 137 por la mañana (aprox.), se puede ver que no es un error prolongado, sino que el modelo no consigue adivinar que esa activación debería prolongarse en el tiempo. No es representativo de la realidad que un lavaplatos esté encendido y funcionando 2 minutos, luego apagado otro minuto, luego encendido de nuevo otro, etc. Esto se debe probablemente a que en ese mismo instante otro electrodoméstico estuviera consumiendo con una potencia parecida y el modelo no haya sido capaz de distinguirlos.

Hay otros errores más continuos (en el mismo día 137, la segunda raya continua), que indican simplemente que el modelo no generaliza demasiado

bien de train a test (posiblemente y como ya se comentó anteriormente, porque haya overfitting, debido al tamaño de la red).

A modo de resumen, también se ha realizado una tabla que recoge los resultados numéricos del modelo MLP. En esta, se resumen los porcentajes de acierto obtenidos para los distintos electrodomésticos frente al valor de referencia comentado en la Figura 54. El valor más alto se denomina *no information rate*, o ratio de no información. Es el porcentaje de acierto más alto que se obtendría si se hiciera un modelo que simplemente da como salida la clase mayoritaria.

Tabla 2 – Resultados accuracy MLP vs REF

		MLP (%)	REF (%)
air1	Train accuracy	99.95	91.0
	Test accuracy	99.71	
clotheswasher1	Train accuracy	99.98	98.68
	Test accuracy	98.44	
dishwasher1	Train accuracy	99.83	96.854
	Test accuracy	98.31	
kitchenapp1	Train accuracy	99.98	99.647
	Test accuracy	99.54	
microwave1	Train accuracy	99.95	99.665
	Test accuracy	99.71	

En la Tabla 2 se puede comprobar como en todos los casos el modelo es superior a esta ratio, lo cual demuestra una vez que el modelo es de calidad.

Globalmente, los resultados obtenidos por este MLP son prometedores. Se recuerda que lo que se ha observado en las figuras anteriores es la desagregación, a partir de la **señal del total** del consumo eléctrico, en los distintos electrodomésticos, y que el modelo está siendo

capaz de identificar, a partir de esta señal, cuando está encendido un determinado electrodoméstico y cuando no.

4.2 RESULTADOS CNN

La metodología que se ha seguido para la construcción de la CNN ha sido iterativa, empezando por modelos simples y pequeños, para después ir complicándolos, añadiendo capas convolucionales o aumentando la cantidad de filtros en dichas capas. Se han desarrollado un total de 23 modelos, ajustando en cada uno de ellos el o los hiperparámetros necesarios para conseguir dicha mejora.

Un detalle importante para el desarrollo de las CNN es que debido a los tiempos de entrenamiento y con el fin de agilizar la optimización de los modelos, el ajuste de parámetros se ha realizado entrenando un modelo para todos los electrodomésticos, por lo que se tiene un solo modelo para clasificar 5 aparatos, mientras que para el MLP se entrenó un modelo por cada electrodoméstico.

Los primeros modelos solo contenían una o dos capas convolucionales de muy poco tamaño, alguna capa de dropout para evitar overfitting y finalmente las capas densas. La arquitectura de este primer modelo se puede ver en la Figura 59. Estos primeros modelos no conseguían resultados muy buenos, como se puede ver en la Figura 60.

Layer (type)	Output Shape	Param #
conv2d_30 (Conv2D)	(None, 144, 13, 8)	80
max_pooling2d_22 (MaxPooling)	(None, 36, 3, 8)	0
dropout_10 (Dropout)	(None, 36, 3, 8)	0
flatten_10 (Flatten)	(None, 864)	0
dense_20 (Dense)	(None, 40)	34600
dense_21 (Dense)	(None, 5)	205
Total params: 34,885		
Trainable params: 34,885		
Non-trainable params: 0		

Figura 59 – Arquitectura del primer modelo CNN

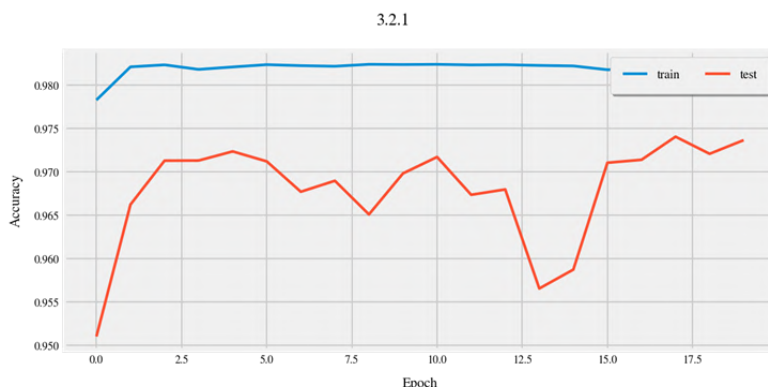


Figura 60 – Accuracy por epoch – Primer modelo CNN

A medida que se fueron añadiendo capas convolucionales, o aumentando la cantidad de filtros de éstas, se fueron consiguiendo mejores resultados, pero se observó la aparición de overfitting, como se puede ver en la Figura 61 y Figura 62.

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 144, 13, 8)	208
max_pooling2d_7 (MaxPooling2D)	(None, 72, 6, 8)	0
conv2d_10 (Conv2D)	(None, 72, 6, 16)	1168
max_pooling2d_8 (MaxPooling2D)	(None, 36, 3, 16)	0
conv2d_11 (Conv2D)	(None, 36, 3, 16)	2320
max_pooling2d_9 (MaxPooling2D)	(None, 12, 1, 16)	0
conv2d_12 (Conv2D)	(None, 12, 1, 64)	9280
flatten_3 (Flatten)	(None, 768)	0
dropout (Dropout)	(None, 768)	0
dense_6 (Dense)	(None, 40)	30760
dense_7 (Dense)	(None, 5)	205
Total params: 43,941		
Trainable params: 43,941		
Non-trainable params: 0		

Figura 61 – Arquitectura de modelo CNN con overfitting

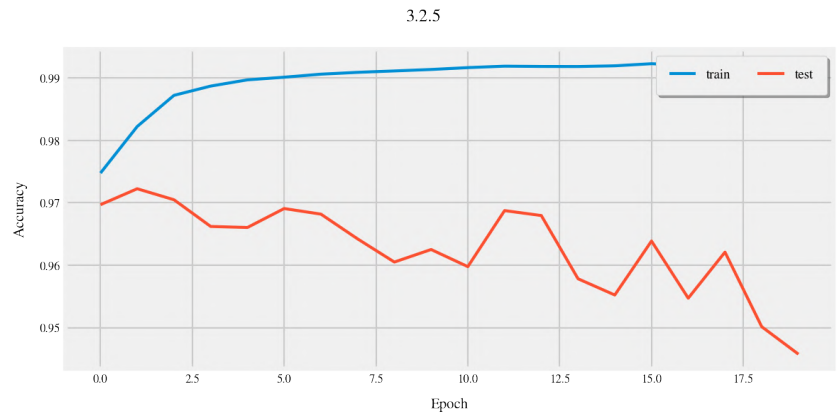


Figura 62 – Accuracy por epoch – Modelo CNN con overfitting

Tras muchas iteraciones en la arquitectura de la CNN, intentado alcanzar un equilibrio entre resultados y la complejidad del modelo (manteniendo un número de parámetros relativamente bajo (menos de 30.000), haciendo uso de técnicas de regularización y cambiando la reducción de dimensionalidad tradicional que se hace con las capas de MaxPooling por una convolución con stride) se consiguió un modelo que arroja buenos resultados en train y test, cuya arquitectura no era demasiado compleja y que a priori no parece hacer overfitting. Las siguientes figuras muestran su arquitectura y el accuracy del modelo (Figura 63 y Figura 64) y la Figura 65 muestra la representación 3D al estilo AlexNet de esta CNN.

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 72, 7, 8)	88
dropout_1 (Dropout)	(None, 72, 7, 8)	0
batch_normalization (Batch Normalization)	(None, 72, 7, 8)	32
conv2d_4 (Conv2D)	(None, 36, 4, 16)	1168
batch_normalization_1 (Batch Normalization)	(None, 36, 4, 16)	64
conv2d_5 (Conv2D)	(None, 17, 1, 32)	4640
batch_normalization_2 (Batch Normalization)	(None, 17, 1, 32)	128
flatten_1 (Flatten)	(None, 544)	0
dropout_2 (Dropout)	(None, 544)	0
dense_2 (Dense)	(None, 40)	21800
batch_normalization_3 (Batch Normalization)	(None, 40)	160
dense_3 (Dense)	(None, 5)	205
Total params: 28,277		
Trainable params: 28,085		
Non-trainable params: 192		

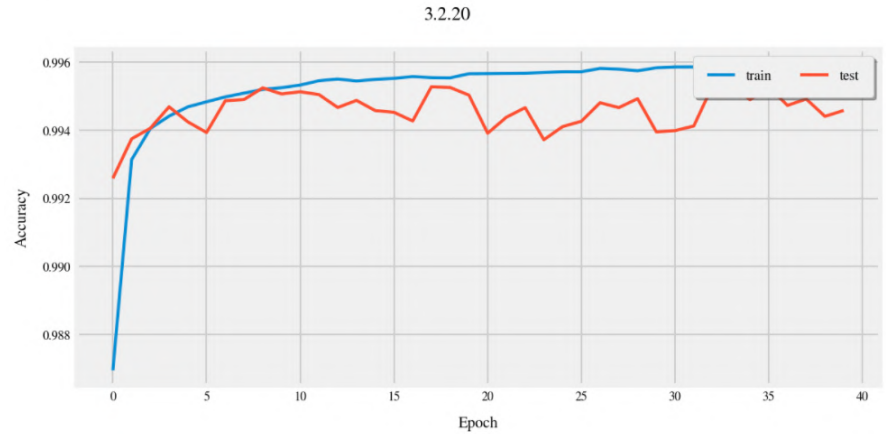


Figura 63 – Arquitectura del mejor modelo CNN

Figura 64 – Accuracy por epoch – Mejor modelo CNN

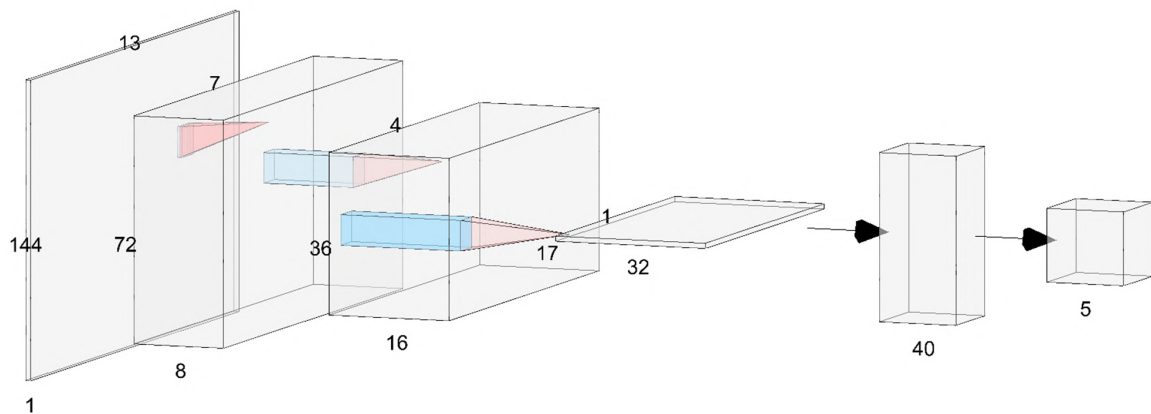


Figura 65 – Estructura de la mejor CNN – Estilo AlexNet

Los resultados de los todos los modelos realizados se pueden ver en el ANEXO F.

4.2.1 EVALUACIÓN

De forma similar a la evaluación de resultados del MLP, solo se van a representar las gráficas de un electrodoméstico para no saturar la lectura del documento, así como una tabla resumen con los porcentajes de acierto. Las gráficas de los demás resultados se pueden consultar en el ANEXO B.

En el caso de la lavadora, los resultados del mejor modelo CNN se pueden ver en la Figura 66 y Figura 67, para el set de train y de test respectivamente.

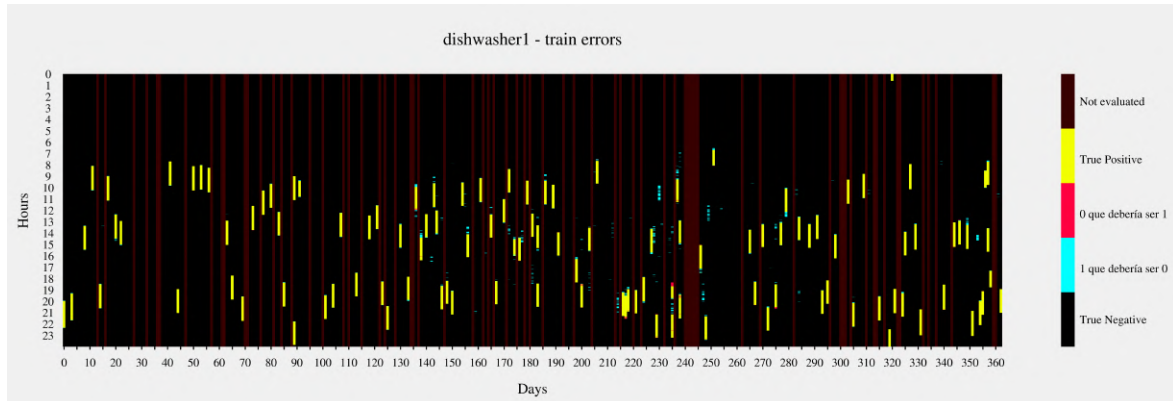


Figura 66 – Resultados mejor modelo CNN – *dishwasher1* – train

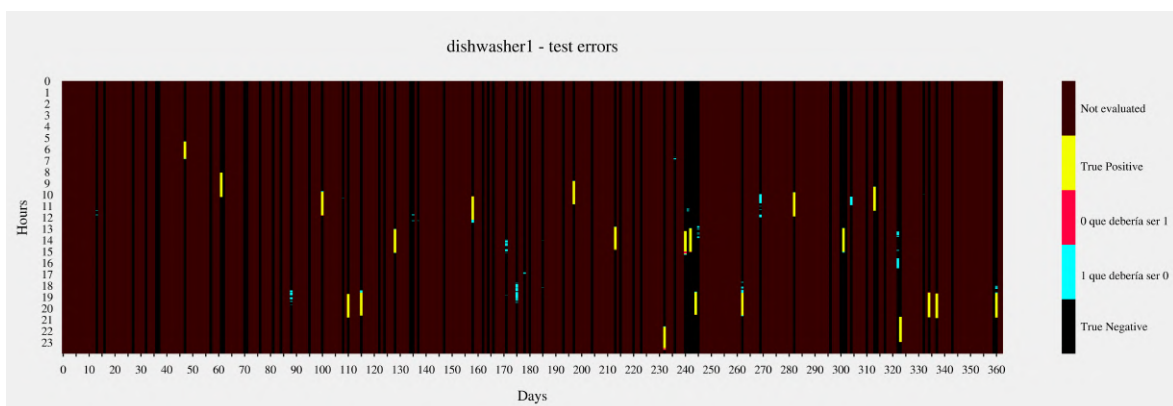


Figura 67 – Resultados mejor modelo CNN – *dishwasher1* – test

Comparando estos resultados con los obtenidos con el modelo MLP, se puede ver que son mejores, al menos para el caso del lavaplatos. Sin embargo, y como se verá a continuación, para los electrodomésticos más pequeños como el microondas o el circuito de la cocina, se obtienen resultados ligeramente peores con la CNN.

Para la CNN también se ha realizado una tabla que recoge los resultados numéricos del accuracy para los distintos electrodomésticos. Se ha dejado la columna de los porcentajes del MLP para realizar la comparativa directamente.

Tabla 3 – Resultados accuracy MLP vs CNN vs REF

		MLP (%)	CNN (%)	REF (%)
air1	Train accuracy	99.95	99.83	91.0
	Test accuracy	99.71	99.8	
clotheswasher1	Train accuracy	99.98	99.44	98.68
	Test accuracy	98.44	98.83	
dishwasher1	Train accuracy	99.83	99.76	96.854
	Test accuracy	98.31	99.45	
kitchenapp1	Train accuracy	99.98	99.86	99.647
	Test accuracy	99.54	99.43	
microwave1	Train accuracy	99.95	99.82	99.665
	Test accuracy	99.71	99.75	

Como se puede ver en la Tabla 3, los resultados confirman lo que se ha comentado a través de los resultados visuales vistos hasta el momento. La CNN ofrece resultados similares al MLP para los electrodomésticos de mayor duración de utilización (aire acondicionado, lavadora, lavaplatos) y peores resultados en los de menor duración (microondas y circuito de cocina).

Este trabajo se ha centrado en los electrodomésticos más grandes y de más larga duración, por una parte, porque con los datos de minuto del dataset, los de menor duración aparecen como activaciones esporádicas en la serie temporal (habría que utilizar el dataset de segundos), y por otra porque son los electrodomésticos de mayor impacto energético y por lo tanto medioambiental.

Además, como se comentó anteriormente, se verán en detalle más resultados numéricos en el apartado 4.3 donde se comparan una serie de métricas en los datos reales, los resultados del MLP y los de la CNN.

Globalmente se puede decir que la CNN arroja unos resultados muy buenos, teniendo en cuenta que se trata de un solo modelo para todos los electrodomésticos y no de un modelo para cada aparato como se realizó para el MLP.

Con el objetivo de comprobar la robustez de la metodología y el modelo propuesto, se ha evaluado este modelo con los datos de una segunda vivienda. Los resultados del lavaplatos se pueden ver en la Figura 68 y Figura 69, pudiéndose consultar el resto en el ANEXO C.

Como se puede observar en ambas figuras, los resultados no son buenos. Esto era de esperar principalmente porque el modelo se ha entrenado solo con los datos de una casa. Para tener un modelo robusto se tendrían que mezclar los datos de distintas casas, cuantas más, mejor. Esto hace que, si los hábitos de consumo son muy diferentes de un hogar a otro, o los electrodomésticos tienen un consumo muy distinto, es imposible que el modelo generalice bien de una casa a otra.

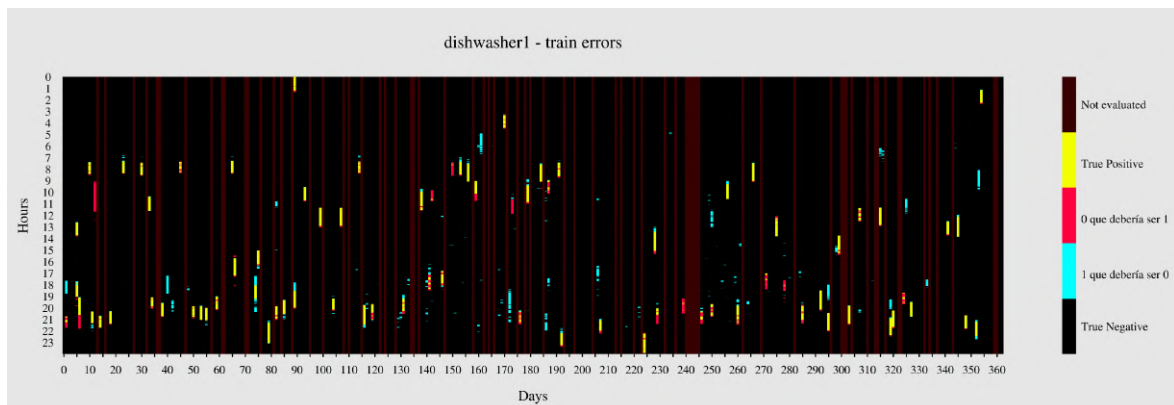


Figura 68 – Evaluación del mejor modelo CNN sobre los datos de otro hogar – *dishwasher1* – train

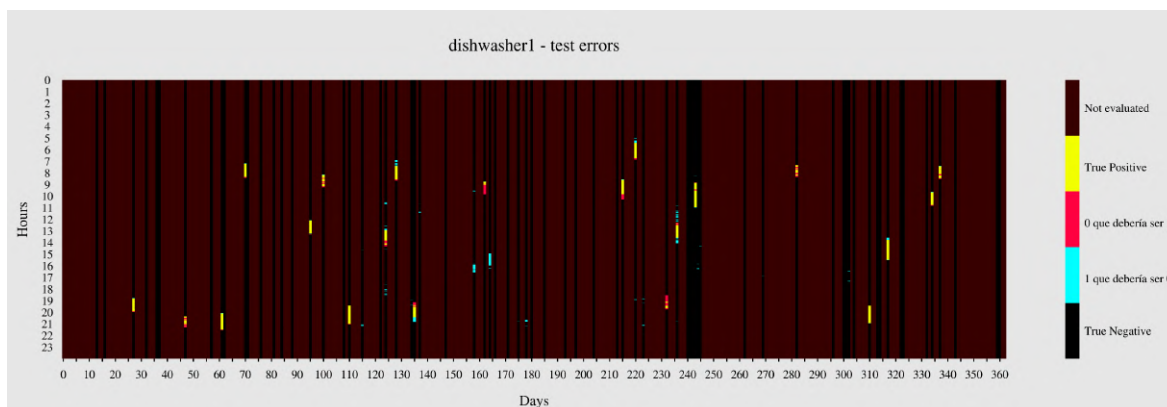


Figura 69 – Evaluación del mejor modelo CNN sobre los datos de otro hogar – *dishwasher1* – test

4.2.2 TRANSFER LEARNING

Para comprobar si el modelo entrenado puede generalizarse para modelar otras casas, se ha realizado transfer Learning utilizando el modelo entrenando, congelando las capas convolucionales y reentrenando solo las últimas capas densas.

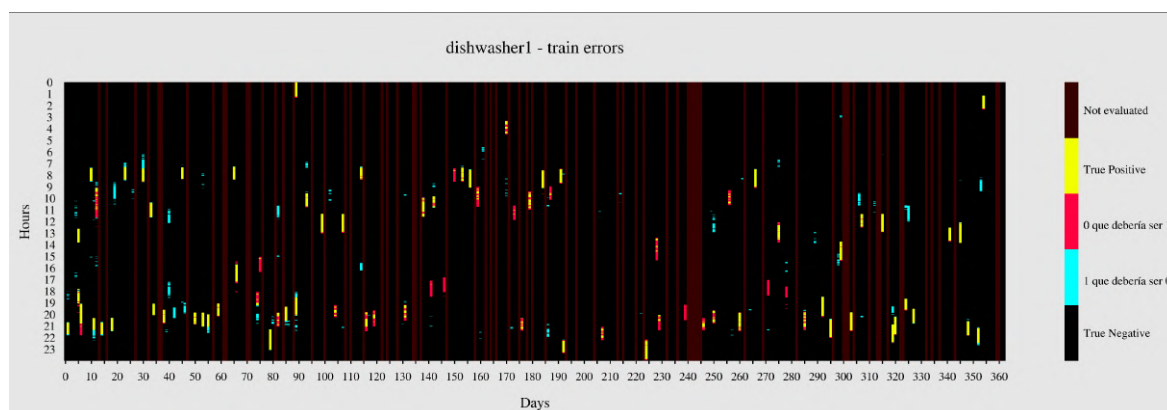


Figura 70 – Evaluación del modelo con transfer Learning sobre los datos de otro hogar – *dishwasher1* – train

La Figura 70 y la Figura 71 muestran el resultado de la evaluación de este modelo reentrenado sobre los datos de la segunda casa, y de la misma forma que en los apartados anteriores, solo se va a representar un electrodoméstico (en este caso el lavaplatos), pudiéndose consultar el resto en el ANEXO D.

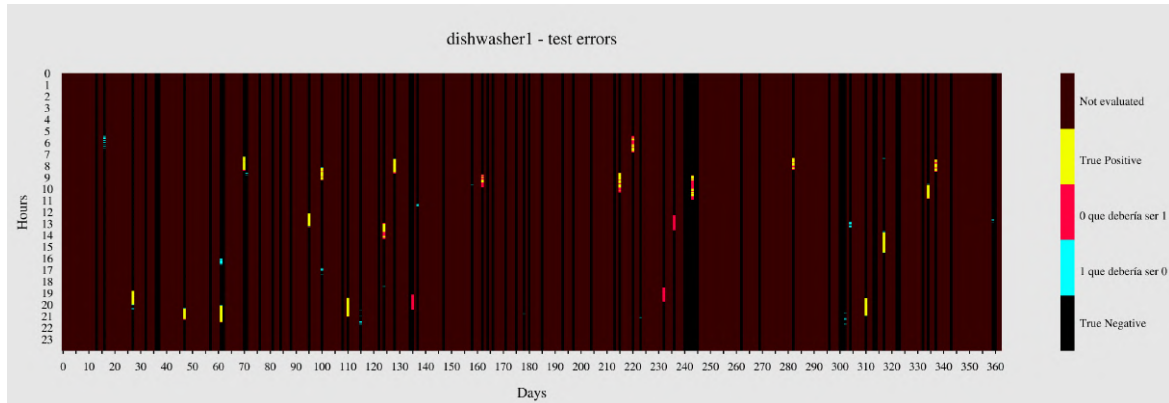


Figura 71– Evaluación del modelo con transfer Learning sobre los datos de otro hogar – *clotheswasher1* – test

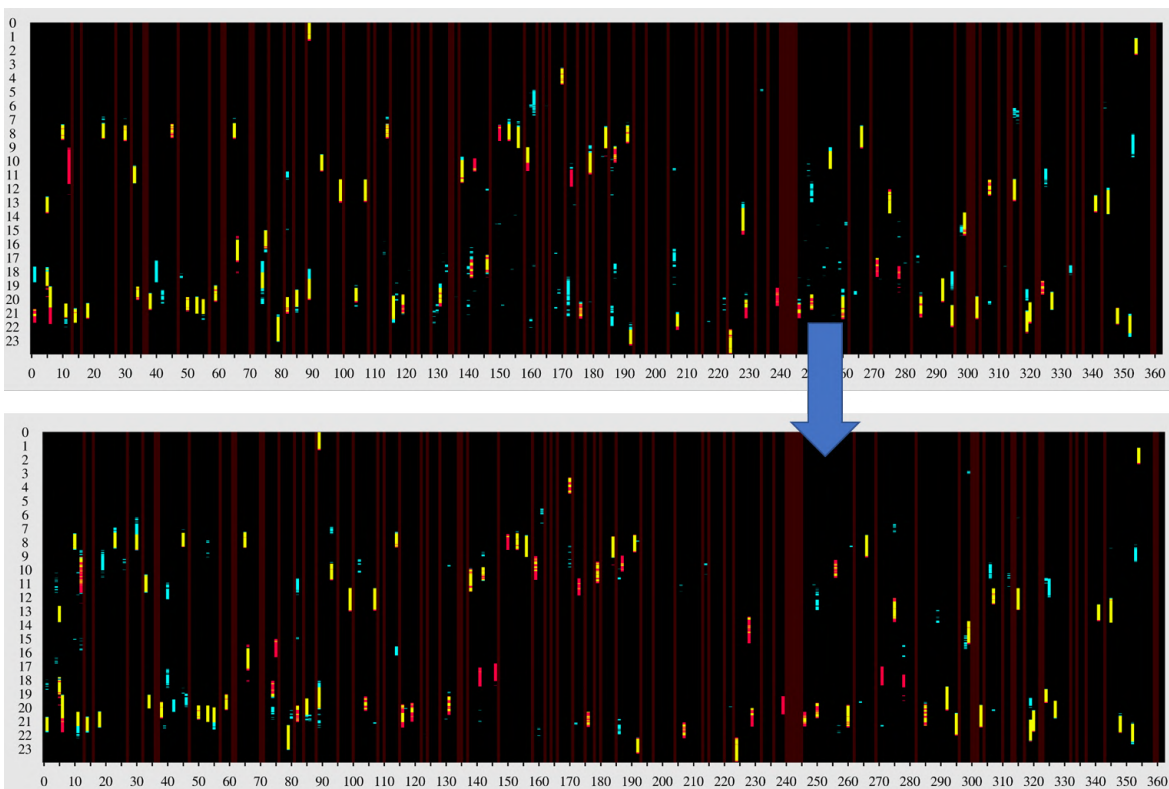


Figura 72 – Comparativa entre evaluación de CNN y CNN con transfer learning sobre *dishwasher1* – train

La Figura 72 muestra la comparativa en el set de train para el caso del lavaplatos. En esta se puede ver que los resultados no parecen mejorar mucho al haber hecho transfer Learning en

el caso del lavaplatos, ya que se siguen viendo muchos falsos positivos y muchos falsos negativos.

Por otro lado, en el caso la lavadora (Figura 73) es más difícil apreciar a simple vista los posibles cambios, pero parece que existe alguna mejora, por lo que en la Figura 74 se ha hecho un zoom de unos cuantos días para poder apreciar esas diferencias.

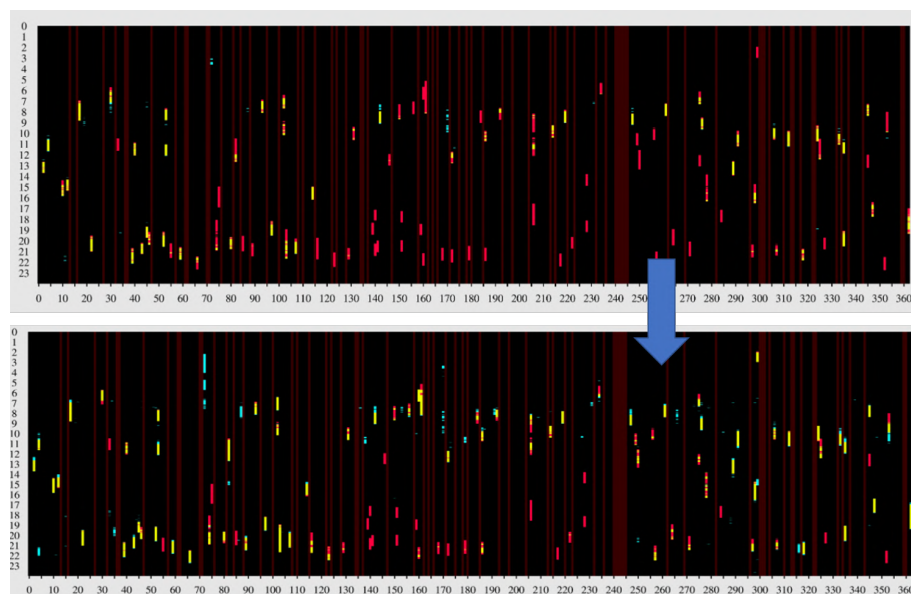


Figura 73 – Comparativa entre evaluación de CNN y CNN con transfer learning sobre clotheswasher1 – train

En Figura 74 se puede ver como en algunos casos, por ejemplo, los días 103-104, el modelo con transfer learning saca resultados mucho mejores, o por ejemplo en ellos días 164 donde el modelo original no conseguía detectar ninguna activación, el modelo con transfer Learning sí. Sin embargo, también aparecen más falsos positivos, notablemente en los días 71 y 177.

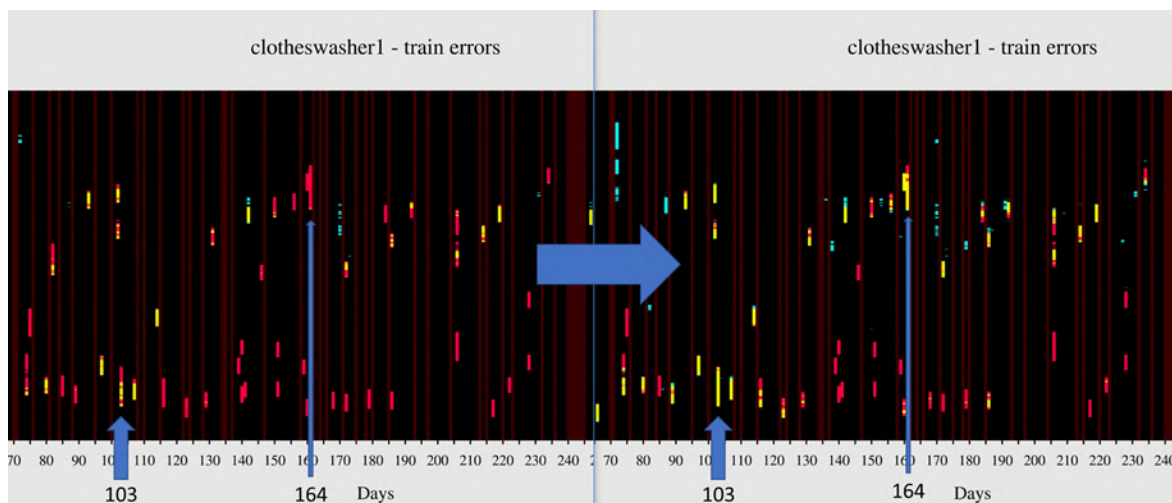


Figura 74 – Comparativa zoom entre evaluación de CNN y CNN con transfer learning sobre clotheswasher1 – train

Donde parece que el modelo sigue sin poder sacar buenos resultados es en los electrodomésticos pequeños, ya que tanto para el microondas (Anexo C – Figura 5 y Anexo D – Figura 5) y el circuito de la cocina (Anexo C – Figura 7 y Anexo D – Figura 7) las mejoras son poco apreciables, como se puede ver en la siguiente figura, que presenta un zoom en la zona de mejora principal en el caso del circuito de la cocina.

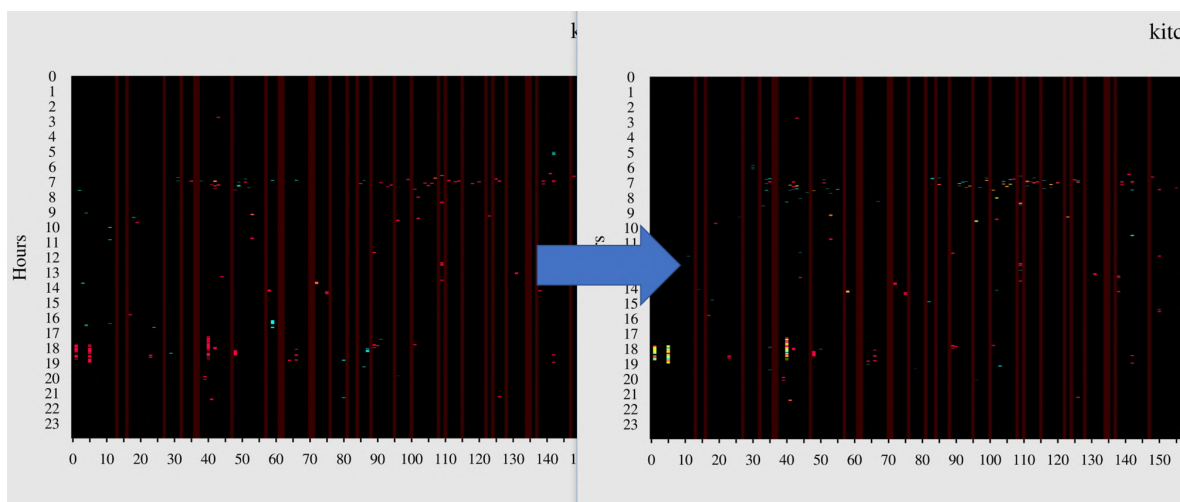


Figura 75 – Comparativa entre evaluación de CNN y CNN con transfer learning sobre kitchenapp1 – train

En particular, se aprecian mejoras en los días 2, 6 y 40, observándose también alguna mejora puntual en algún día posterior alrededor del día 95 y 105.

4.2.3 FEATURE MAPS Y FILTROS

Para asegurarse de que la CNN realmente aporta algo y está aprovechando la forma de plantear el problema (el vizgrama), se pueden pintar los denominados *feature maps*, que representan lo que está buscando y detectando cada filtro de capa convolucional cuando recibe una imagen.

La arquitectura de la mejor CNN se puede ver en la Figura 76. En esta, se puede ver que este modelo dispone de 3 capas convolucionales:

- o La primera capa consta de 8 filtros de 3x3
- o La segunda capa se compone de 16 filtros de 3x3
- o La tercera capa contiene 32 filtros de 3x3

La Figura 77 muestra un vizgrama en el que solo estaba activado el lavaplatos, y la Figura 78 muestra los feature maps de la primera capa convolucional al recibir como input este vizgrama.

```
In [71]: model_summary_info
         executed in 127ms, finished 20:28:08 2021-01-30

Out[71]: ['Model: "sequential_1"',
          'Layer (type)                Output Shape          Param #',
          '-----',
          'conv2d_3 (Conv2D)              (None, 72, 7, 8)      80',
          'dropout_1 (Dropout)              (None, 72, 7, 8)      0',
          'batch_normalization (Batch Normalization) (None, 72, 7, 8)      32',
          'conv2d_4 (Conv2D)              (None, 36, 4, 16)     1168',
          'batch_normalization_1 (Batch Normalization) (None, 36, 4, 16)     64',
          'conv2d_5 (Conv2D)              (None, 17, 1, 32)     4640',
          'batch_normalization_2 (Batch Normalization) (None, 17, 1, 32)     128',
          'flatten_1 (Flatten)           (None, 544)           0',
          'dropout_2 (Dropout)           (None, 544)           0',
          'dense_2 (Dense)                (None, 40)            21800',
          'batch_normalization_3 (Batch Normalization) (None, 40)            160',
          'dense_3 (Dense)                (None, 5)             205',
          '-----',
          'Total params: 28,277',
          'Trainable params: 28,085',
          'Non-trainable params: 192',
          '']
```

Figura 76 – Arquitectura del mejor modelo CNN

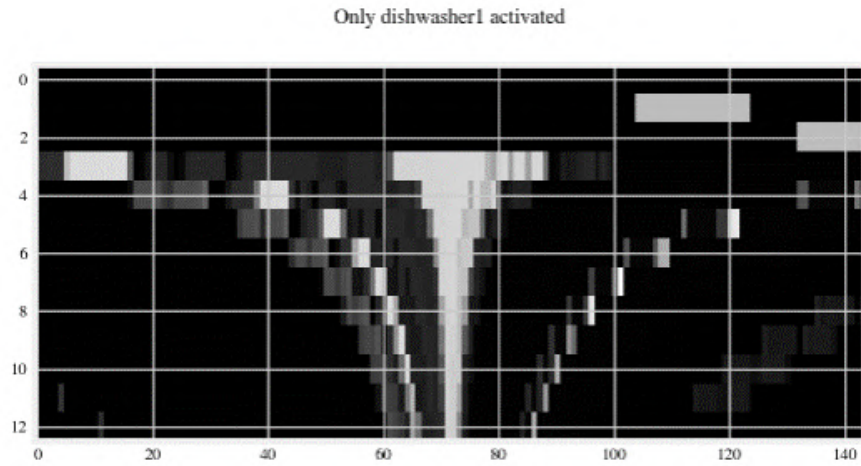


Figura 77 – Vizgrama con solo `dishwasher1` activado

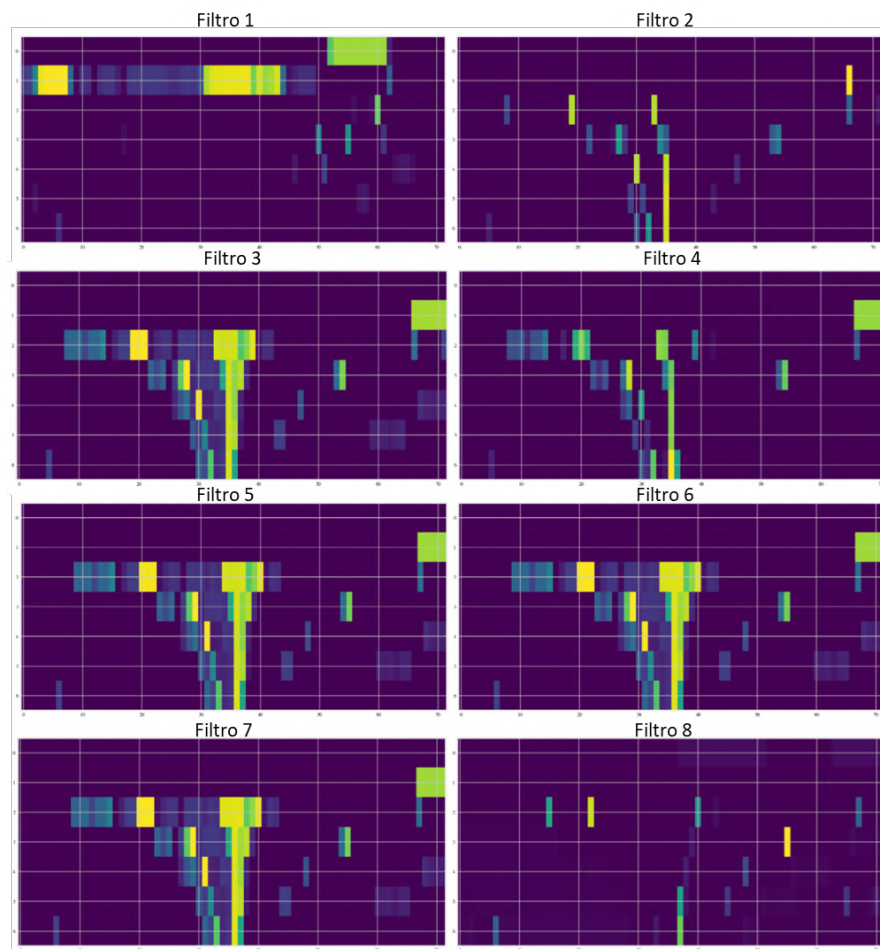


Figura 78 – Feature maps de la primera capa convolucional – vizgrama con `dishwasher1` como input

Observando la Figura 78, se puede ver como los filtros buscan distintos elementos en el vizgrama. Por ejemplo, el primer filtro se centra claramente más en las primeras filas del vizgrama, que representan la hora del día, día de la semana y mes del año en el cual se capturó dicho vizgrama, mientras que los filtros siguientes detectan la propia estructura del vizgrama, detectando tanto en la parte central (filtros 2, 4 y 8) o en los diferentes picos (filtros 3, 5, 6 y 7).

A continuación, en la Figura 79 y Figura 80 se pueden visualizar los feature maps de la segunda capa convolucional:

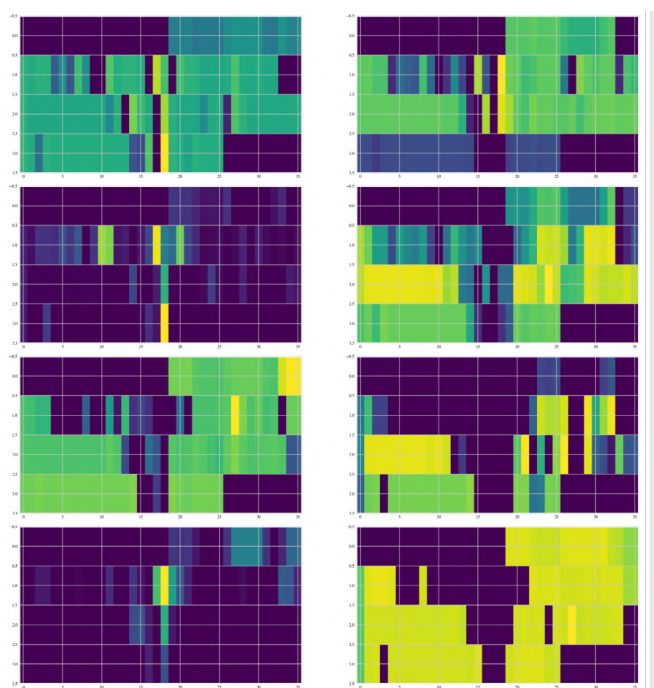


Figura 79 – Feature maps de la segunda capa convolucional (filtro 1 a 8) – vizgrama con *dishwasher1* como input

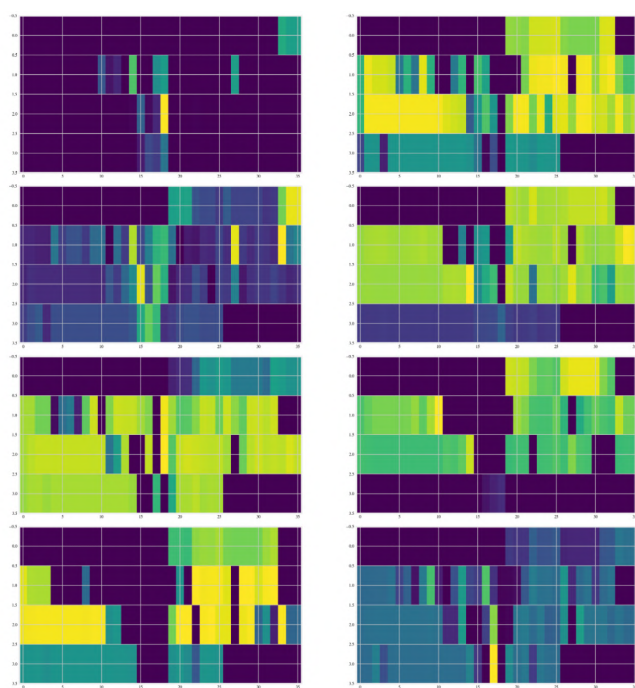


Figura 80– Feature maps de la segunda capa convolucional (filtro 9 a 16) – vizgrama con *dishwasher1* como input

A medida que se profundiza en la arquitectura del modelo, los filtros buscan extraer características cada vez más abstractas de la imagen. Llegados a este punto todavía se puede apreciar la forma de un vizgrama, y se puede ver cómo ciertos filtros se fijan en las partes oscuras (de consumo bajo o nulo) y otros en las partes claras (de consumo alto).

Finalmente, se representan los feature maps de la última capa convolucional en la Figura 81 y Figura 82, donde se pueden ver los 32 filtros de dicha capa. Cada filtro tiene como salida una matriz de 1x17, lo que significa que la salida de nuestra CNN se resume en 17x32 variables (17x32 características). Estas variables son las entradas del clasificador, y es lo que sirve para decidir si el vizgrama de entrada era una activación o no y de qué electrodoméstico. Es curioso como esta entrada es parecida a un código de barras, y que simplemente gracias a esto se pueda detectar si la activación de un electrodoméstico.

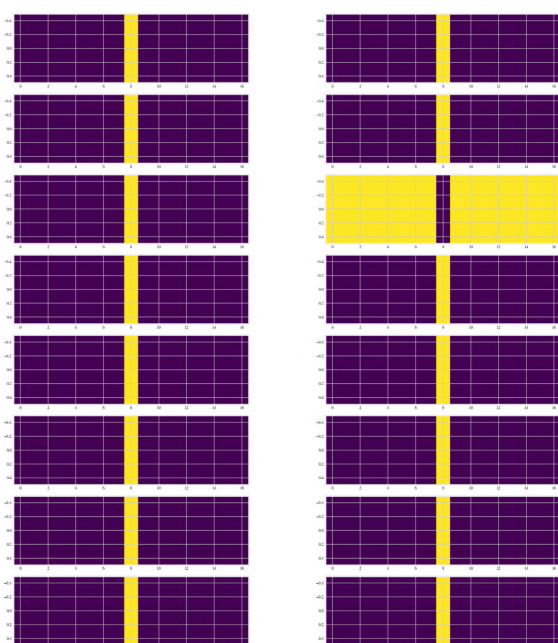


Figura 81 – Feature maps de la tercera capa convolucional (filtro 1 a 16) – vizgrama con *dishwasher1* como input

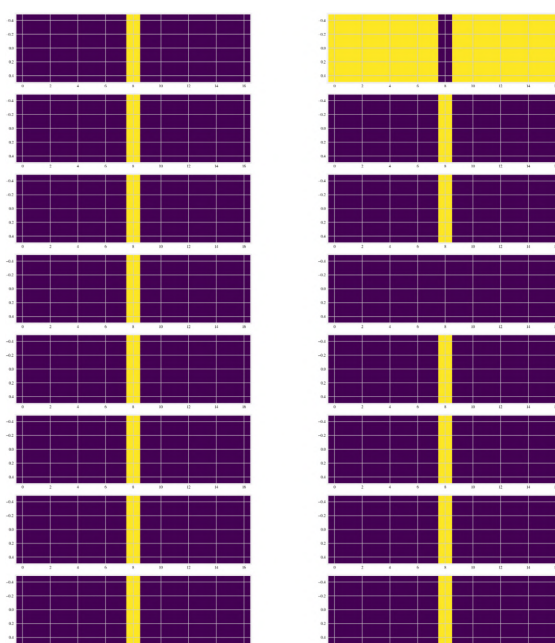


Figura 82 – Feature maps de la tercera capa convolucional (filtro 17 a 32) – vizgrama con *dishwasher1* como input

Los feature maps de las 3 capas convolucional del modelo para los demás electrodomésticos se pueden consultar en el Anexo G – Feature Maps.

4.3 COMPARATIVA ENTRE MODELOS

Como análisis final de los modelos y para darle más interpretabilidad a los resultados, se han calculado una serie de métricas que proporcionan información adicional, enriquecen la

funcionalidad del modelo y permiten entender una de las posibles aplicaciones y utilidades de la monitorización del consumo, y en este caso, de NILM.

4.3.1 CANTIDAD DE MINUTOS ENCENDIDOS

En un primer lugar, se han calculado los minutos en los que están los electrodomésticos encendidos y apagados por hora del día, por mes del año y por día de la semana. Esta serie de gráficos permite sacar conclusiones sobre la robustez y rendimiento del modelo comparando los resultados entre lo real y lo predicho, así como conocer mejor los hábitos de consumo de las personas que vivan en el hogar en cuestión. Un ejemplo de este tipo de gráfica se ve en la Figura 83.

En dicha figura se puede comprobar que tanto el modelo MLP como el modelo CNN obtienen resultados similares en comparación a los datos reales. En ambos casos, se puede comprobar que la previsión es muy cercana a la realidad.

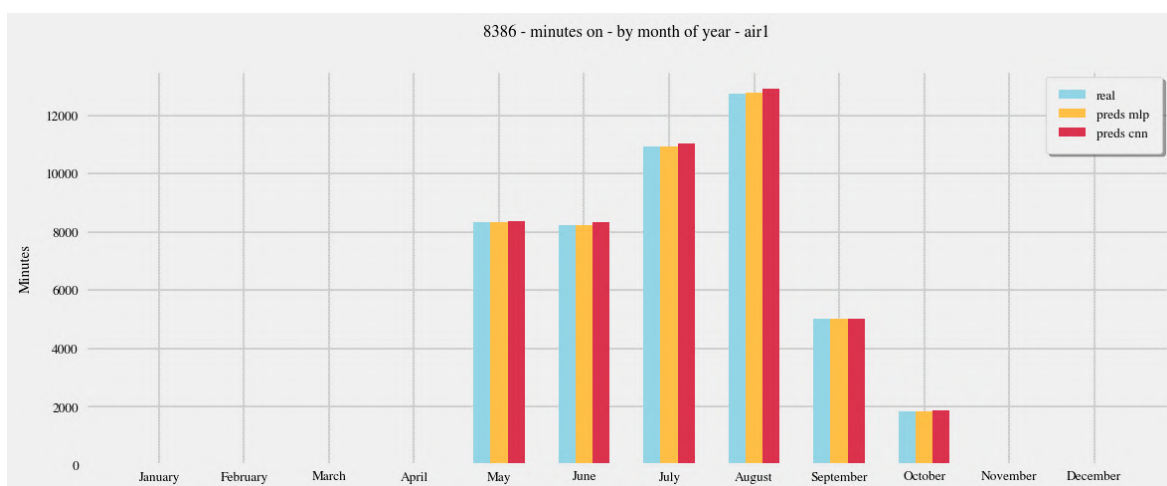


Figura 83 – Cantidad de minutos 'on' por mes del año – *air1*

Observando las mismas gráficas para los demás electrodomésticos (Figura 84 a Figura 87), se puede observar que los electrodomésticos más grandes y de mayor duración de utilización (lavadora en la Figura 84 y lavaplatos en la Figura 85) ambos modelos son comparables, pero que, sin embargo, para los aparatos de menor duración de utilización (circuito de cocina en la Figura 86 y microondas en la Figura 87) el modelo MLP parece ofrecer mejores

resultados. Esto se debe posiblemente a que este modelo esté sobre entrenado, por la cantidad de parámetros que tiene, por haber realizado un modelo por electrodoméstico y por no incorporar métodos de regularización al haberse utilizado una librería básica (sklearn).

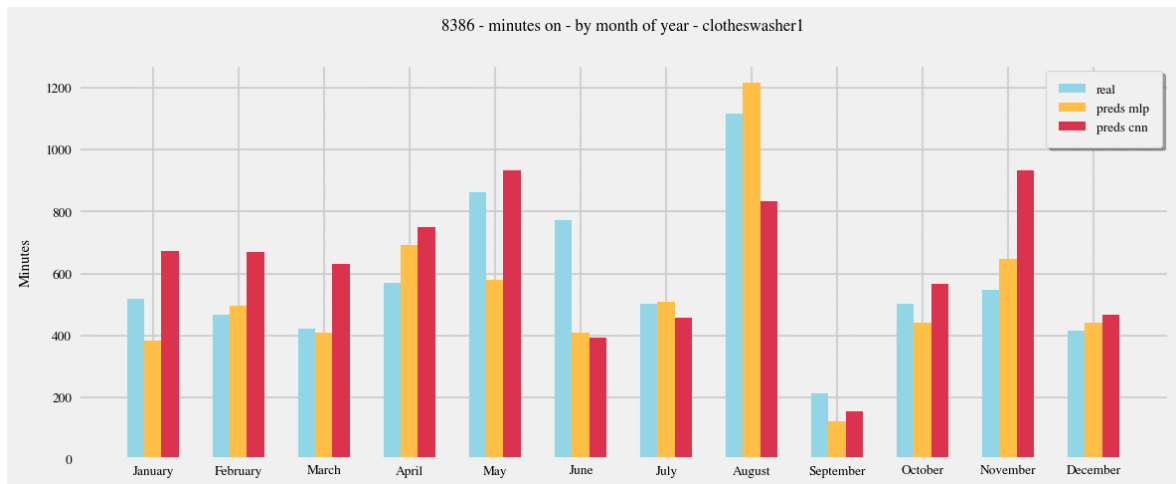


Figura 84 – Cantidad de minutos 'on' por mes del año – clotheswasher1

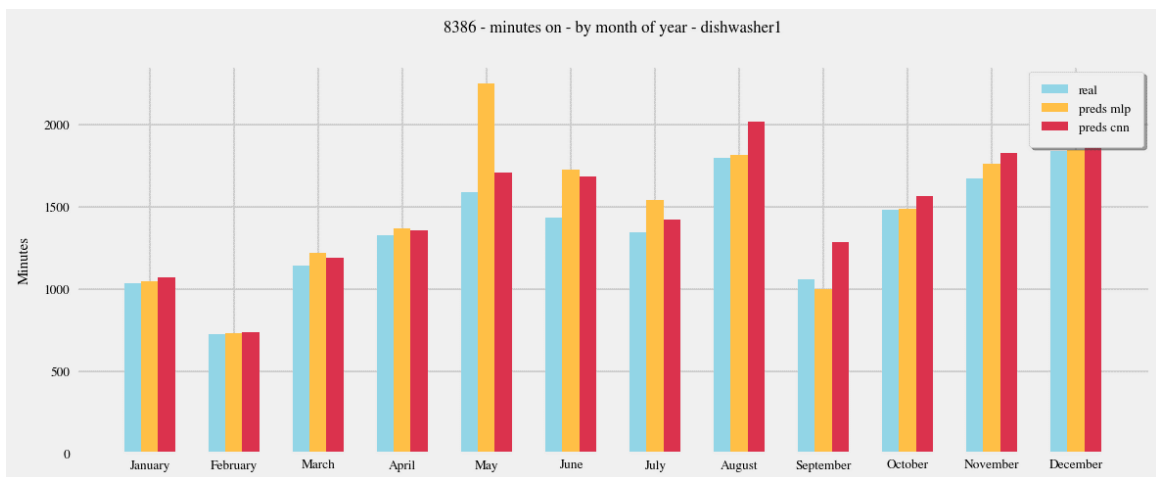


Figura 85 – Cantidad de minutos 'on' por mes del año – dishwasher1

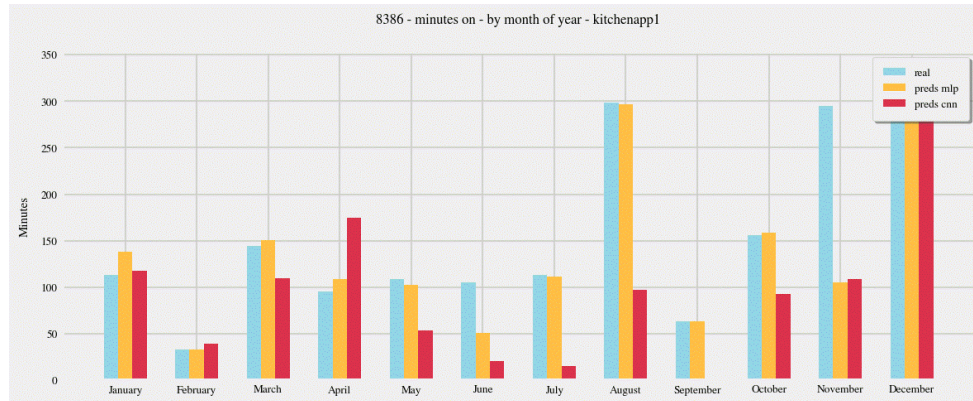


Figura 86 – Cantidad de minutos ‘on’ por mes del año – *kitchenapp1*

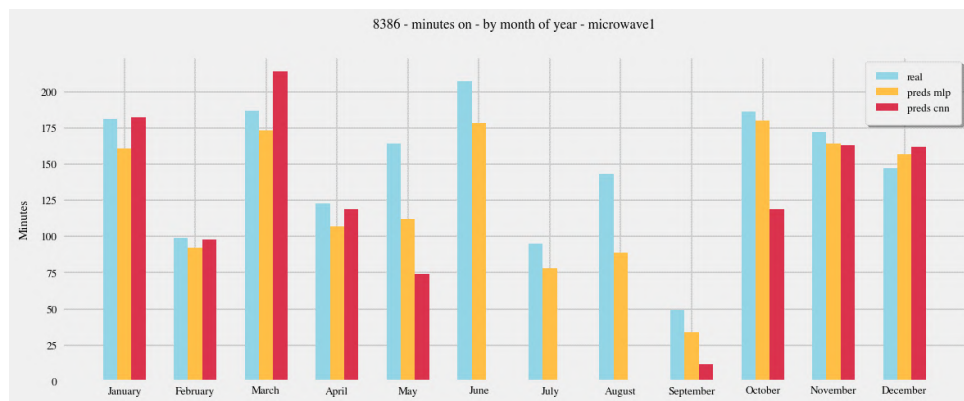


Figura 87 – Cantidad de minutos ‘on’ por mes del año – *microwave1*

Es curioso ver cómo al agrupar estos minutos de utilización por día de la semana los resultados son más homogéneos entre los dos modelos (por ejemplo, en la Figura 88). Pero sorprende aún más ver como en el caso del circuito de cocina y microondas , donde la CNN no daba buenos resultados en los meses de verano en la agrupación por meses (Figura 86 y Figura 87 respectivamente), la agrupación por día de la semana sí que da buenos resultados (Figura 88 y Figura 89).

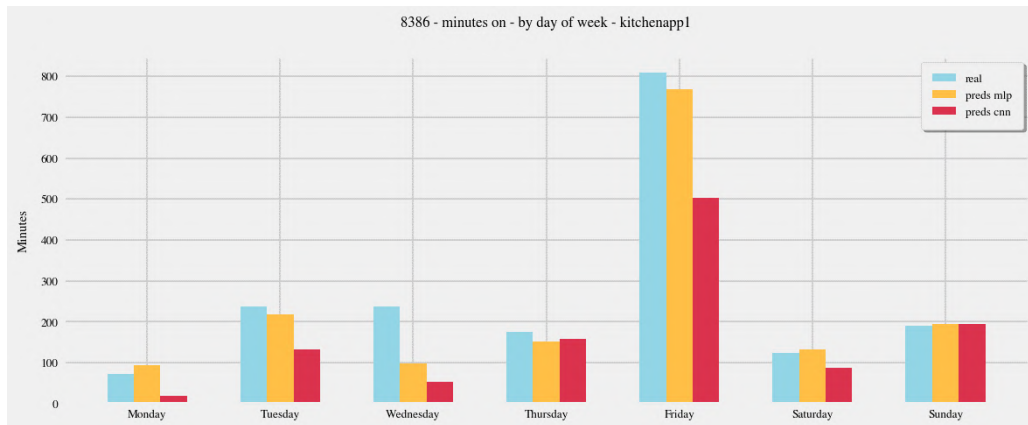


Figura 88 – Cantidad de minutos 'on' por día de la semana – *kitchenapp1*

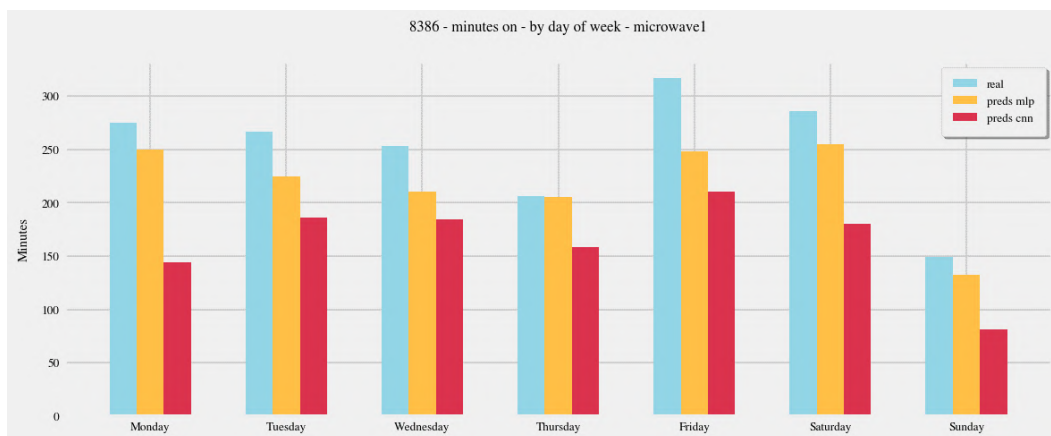


Figura 89 – Cantidad de minutos 'on' por día de la semana – *microwave1*

Para entender estas diferencias hay que volver atrás y analizar el rendimiento de ambos modelos en los meses de verano (por ejemplo, para el circuito de la cocina).

En la Figura 90 se puede claramente como el MLP (derecha) acierta, mientras que la CNN (izquierda) no acierta nada. Fijándose en el eje x de dicha figura, esta comparativa se ha hecho en los meses de verano, explicando el porqué del bajo rendimiento de la CNN al agrupar por meses del daño y el por qué este rendimiento es mejor al agrupar por día de la semana.

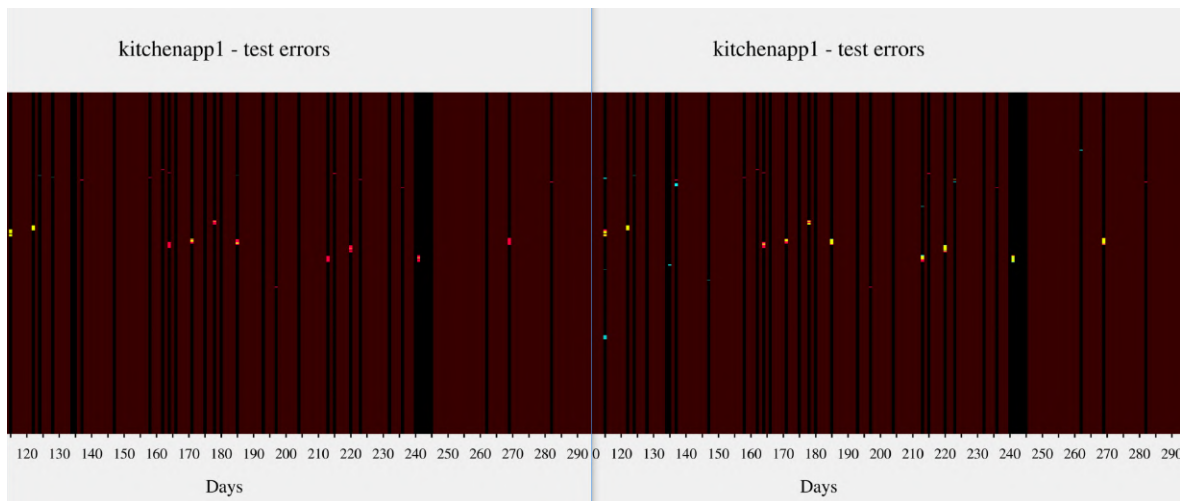


Figura 90 – Comparativa CNN (izquierda) y MLP (derecha) en meses de verano – *kitchenapp1*

Finalmente, este cálculo se ha hecho también por horas del día, para el cual se representa un ejemplo a continuación y se pueden consultar los resultados de los demás electrodomésticos desde Anexo E – Figura 11 a Anexo E – Figura 15.

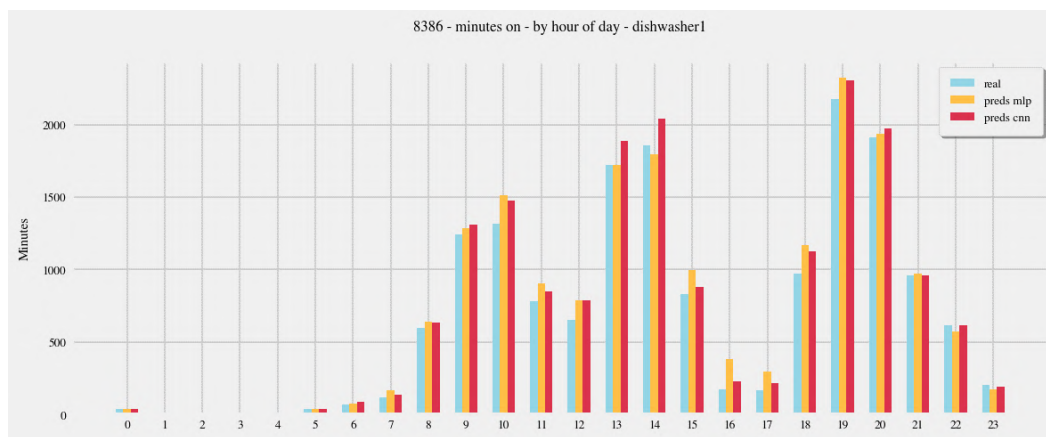


Figura 91 – Cantidad de minutos 'on' por hora del día

En esta agrupación se pueden observar resultados similares a los que se han comentado para las dos anteriores: los resultados de ambos modelos son muy buenos para los electrodomésticos de larga duración y peores para los de corta, siendo el MLP mejor que la CNN en este segundo caso.

4.3.2 CANTIDAD DE ACTIVACIONES DE ELECTRODOMÉSTICOS

Adicionalmente, se han calculado las activaciones de los electrodomésticos para aquellos que tengan un tiempo de utilización largo y que sea el usuario el que tome la decisión de encender o apagar el dispositivo. Por ejemplo, el aire acondicionado se enciende y apaga automáticamente cada x minutos si se pone una temperatura objetivo, por lo que no sería de mucho interés saber que en un día se ha activado 35 veces y sería de mayor utilidad quedarse simplemente en la cantidad de minutos que se ha puesto (resultados del apartado anterior). Es por ello que solo se ha calculado este número de utilizaciones para el lavaplatos y la lavadora.

Se ha contabilizado una utilización cuando uno de estos dos electrodomésticos está activado de principio a fin de manera ininterrumpida, aplicando un filtro similar al que se utilizó para filtrar y rellenar.

Estas utilizaciones se pueden calcular en base a la señal real, a las predicciones hechas con el MLP y con la CNN, tal y como muestran la Figura 92 y la Figura 93:

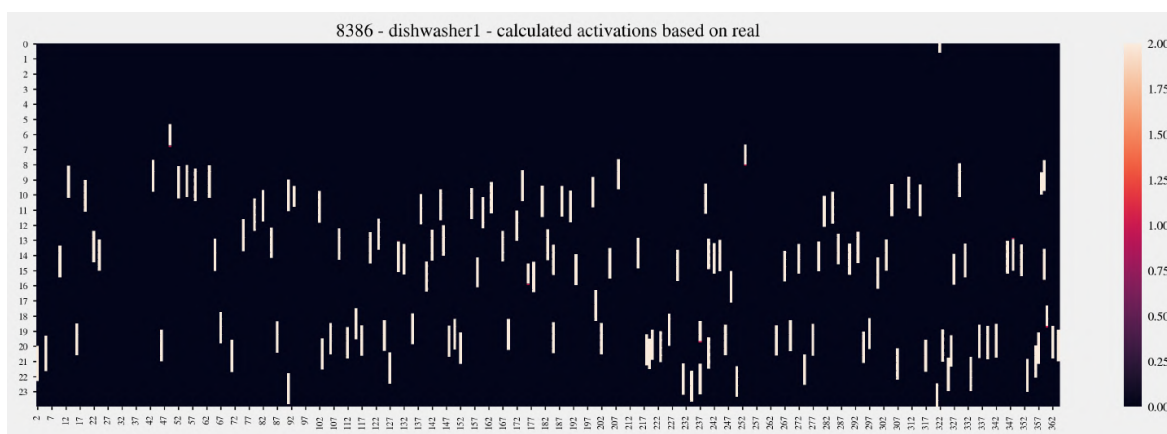


Figura 92 – Representación visual de detección de activaciones basada en datos reales – *dishwasher1*

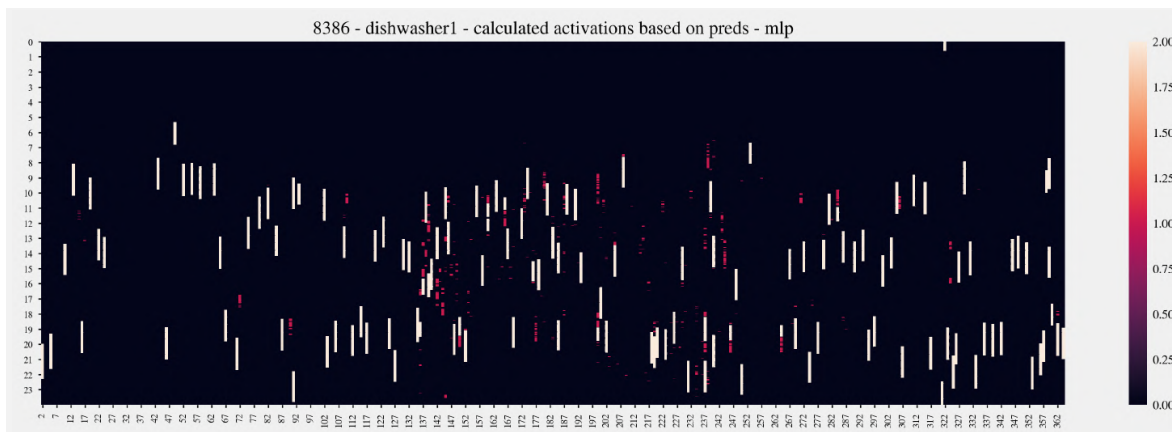


Figura 93 – Representación visual de detección de activaciones basada en predicciones – MLP – dishwasher1

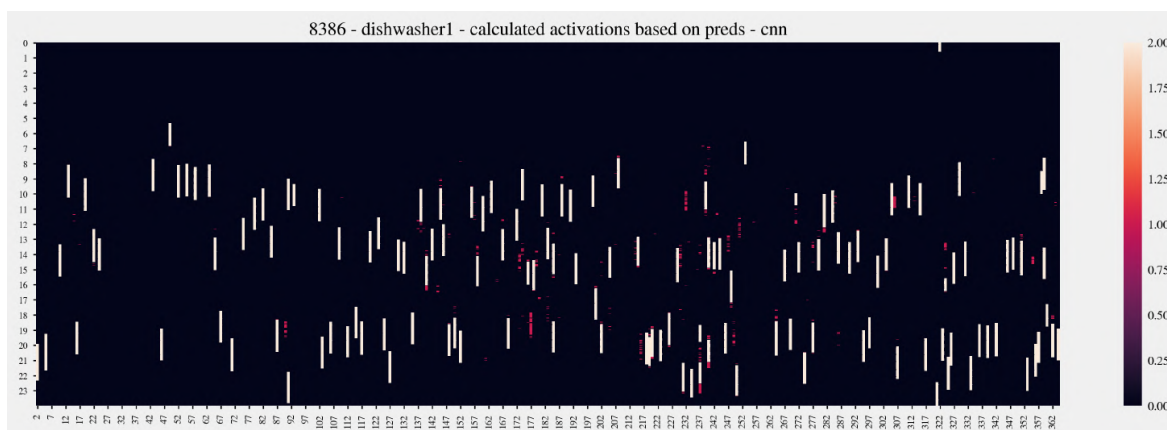


Figura 94 – Representación visual de detección de activaciones basada en predicciones – CNN – dishwasher1

Observando la Figura 93 y la Figura 94, se puede apreciar que al calcular el número de activaciones de los electrodomésticos y no simplemente la cantidad de minutos, se reduce considerablemente el margen de error, puesto que se puede contabilizar una activación aunque la duración de esta sea inferior a la real o aunque las predicciones del modelo no hayan sido del todo acertadas por presentar huecos, caso en el que un simple algoritmo de relleno permite contabilizar la utilización del electrodoméstico. En estas figuras, el color rojo indica las predicciones del modelo que no se han contabilizado como activación continua del electrodoméstico.

Las figuras equivalentes para la lavadora se pueden consultar desde el Anexo E – Figura 21 hasta el Anexo E – Figura 23.

Con este tipo de métrica, se puede contabilizar cuantas veces ha sido puesto el lavaplatos por mes del año, día de la semana y hora del día, aportando, de nuevo, información sobre los hábitos de consumo de un hogar.

La Figura 66, Figura 96 y Figura 97 muestran el número de activaciones calculadas en base a los datos de salida reales, de las predicciones con el MLP y de las predicciones con la CNN, agrupadas por hora del día, día de la semana y mes del año respectivamente, de la lavadora.

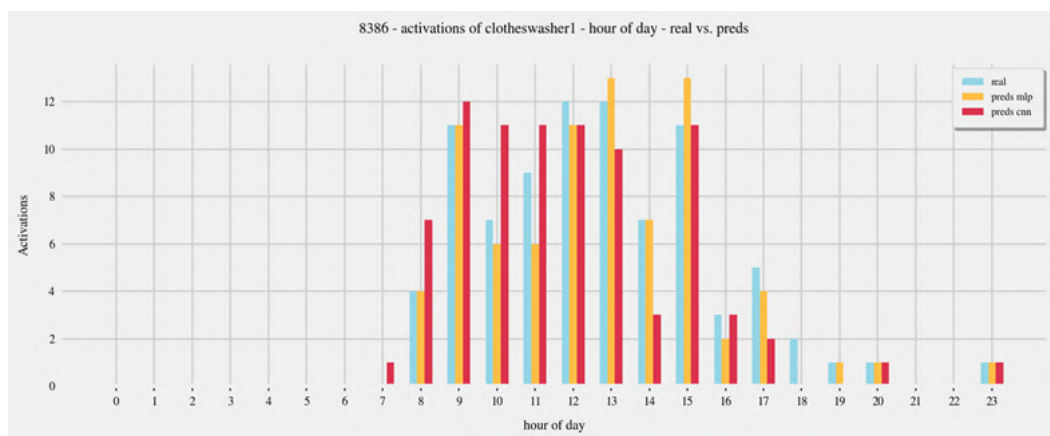


Figura 95 – Número de activaciones por hora del día – clotheswasher1

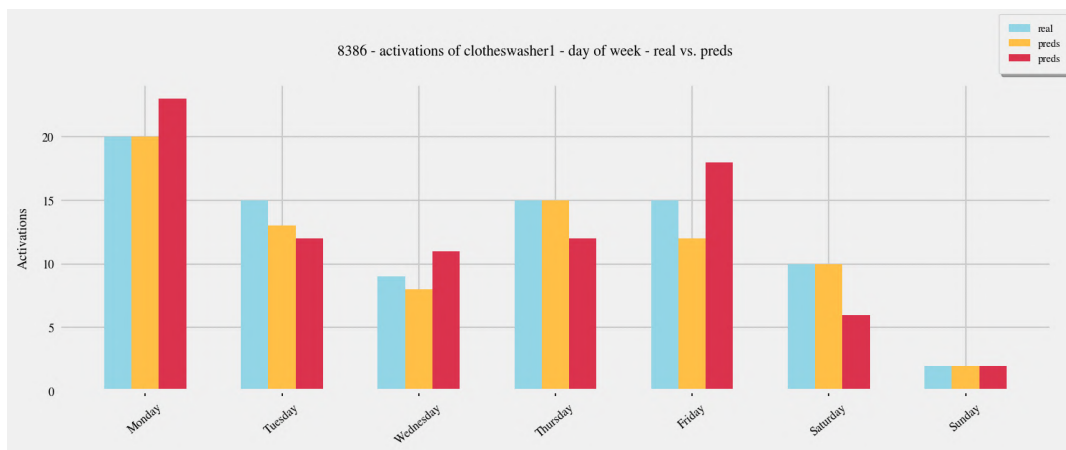


Figura 96 – Número de activaciones por día de la semana – clotheswasher1

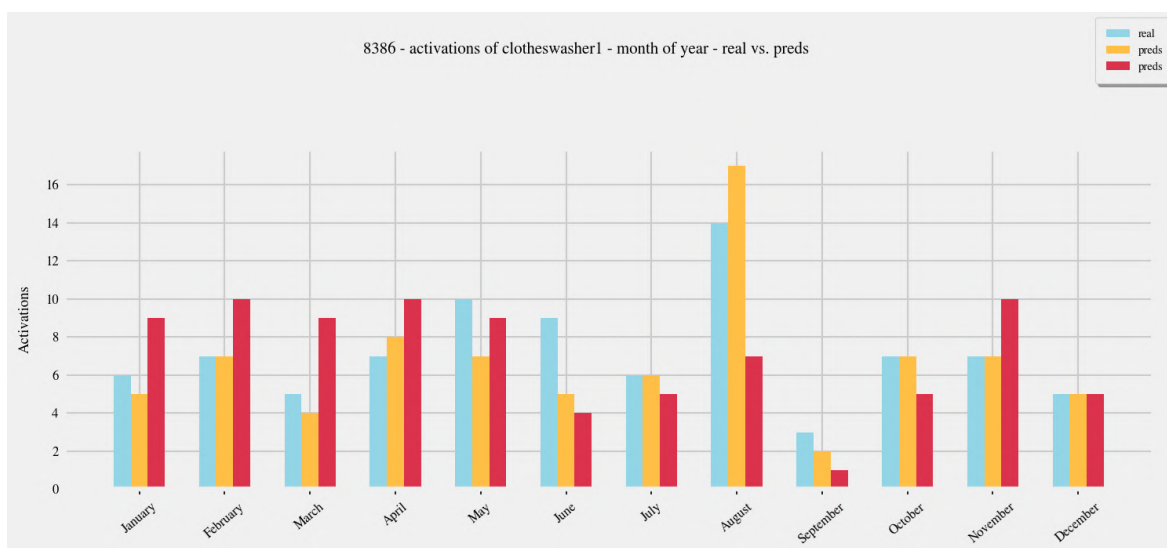


Figura 97 – Número de activaciones por mes del año – clotheswasher1

En estas figuras se puede observar que con esta métrica se obtienen resultados más homogéneos entre ambos modelos, y que los resultados son parecidos a los datos reales. En el caso de agrupar por día, se puede observar que la CNN detecta demasiadas activaciones, perdiéndose la visión del patrón de utilización de este electrodoméstico, que, observando los datos reales, se forma en torno a las 9, las 12-13 y las 15.

Observando la agrupación por día de la semana y mes del año, los resultados muestran un patrón bastante común de comportamiento, ya que se ve que los domingos es el día en el que menos se pone la lavadora y el lunes el que más. Lo mismo se puede observar en la agrupación por mes del año con el mes de septiembre, cuya actividad es claramente inferior a los demás probablemente debido a vacaciones.

En el caso del lavaplatos, el análisis es muy parecido, y las gráficas de las distintas agrupaciones se pueden consultar en el Anexo E – Figura 25, Anexo E – Figura 27 y Anexo E – Figura 29.

4.3.3 TIEMPO DE ACTIVACIÓN MEDIO

Como etapa adicional de validación del modelo, se ha calculado también el tiempo medio de activación de los electrodomésticos para los que se había calculado el número de activaciones, es decir, el lavaplatos y la lavadora. La Figura 69 y Figura 99 muestran el resultado de este cálculo para la lavadora, agrupando por día de la semana y por mes del año respectivamente.

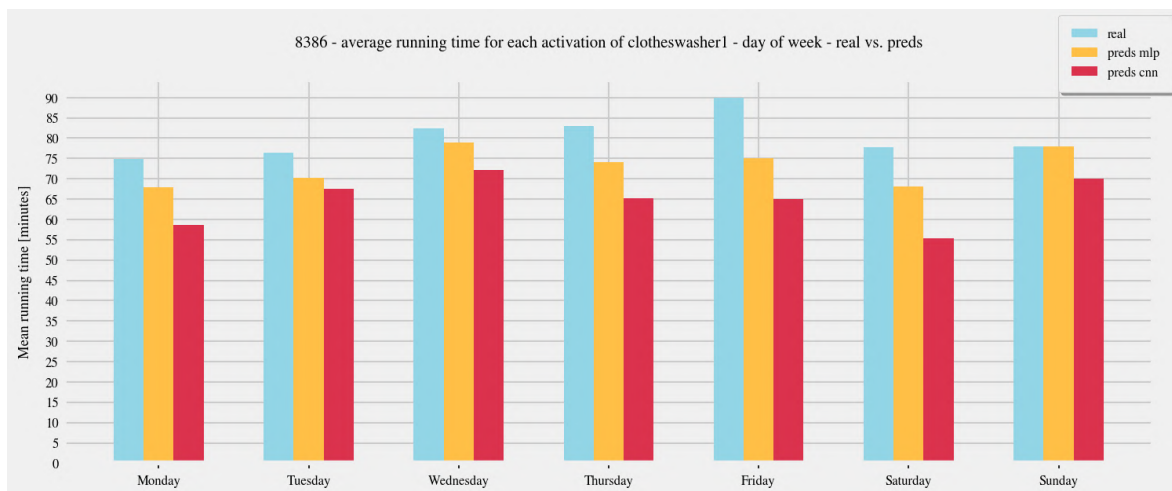


Figura 98 – Tiempo medio de activación por día de la semana – clotheswasher1

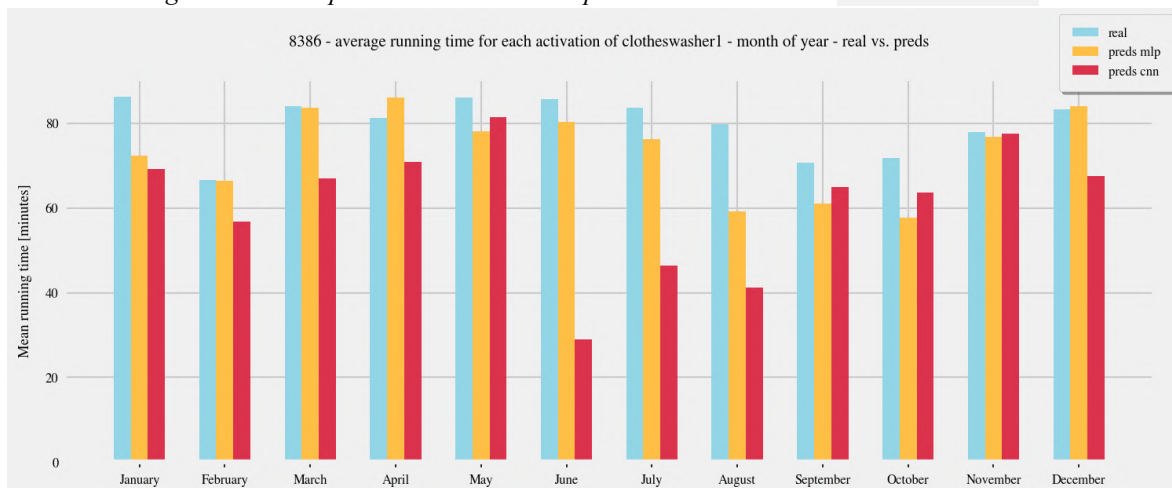


Figura 99 – Tiempo medio de activación por mes del año – clotheswasher1

Como se puede observar en estas figuras, el tiempo medio de utilización entre los datos reales y los datos predichos por ambos modelos es muy parecido, salvo en el caso de los

meses verano para la CNN, en la agrupación por meses del año. La diferencia entre este resultado y el de la Figura 97 se explica por lo comentado en el apartado 4.3.2, y es que una activación se puede contabilizar, aunque ésta haya sido de la mitad de los minutos que en el caso de los datos reales. Por ello se ha representado también el tiempo mediano en la Figura 100 (solo para la agrupación por día de la semana), menos sensible en general a los valores extremos:

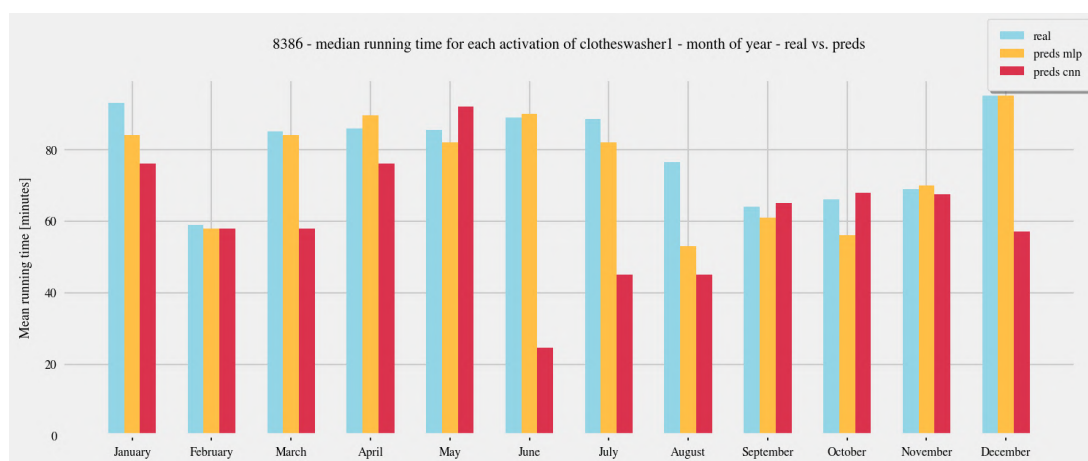


Figura 100 – Tiempo mediano de activación por mes del año – clotheswasher1

En esta gráfica se puede ver como los resultados mejoran para ciertos meses (febrero o noviembre), pero que no presentan mejoras en los meses de verano, indicando simplemente que la CNN ha detectado con menos precisión la cantidad de minutos que ha estado activado este electrodoméstico en particular.

Las gráficas equivalentes para el lavaplatos se encuentran en el Anexo E – Figura 31, Anexo E – Figura 33 y Anexo E – Figura 35.

Finalmente, y a modo de resumen, se han representado los minutos ‘on’ por hora del día para todos los electrodomésticos en la Figura 101, en la cual se puede observar de nuevo que los resultados obtenidos por el modelo son muy parecidos a los datos reales. En esta figura se pueden ver claramente los patrones de consumo comentados anteriormente. Por ejemplo, en el caso del lavaplatos, se puede ver que se pone principalmente en 3 ocasiones del día: entre

las 9-10, las 13-14 o las 18-20. En el caso de la lavadora no parece que exista ninguna franja clara en la que se concentren las utilizaciones, pero sí que se puede observar que se pone únicamente entre las 09:00 y 18:00.

En el caso del aire acondicionado, se puede observar que la mayoría de minutos encendidos se concentran en las horas de más calor, lo cual era de esperar.

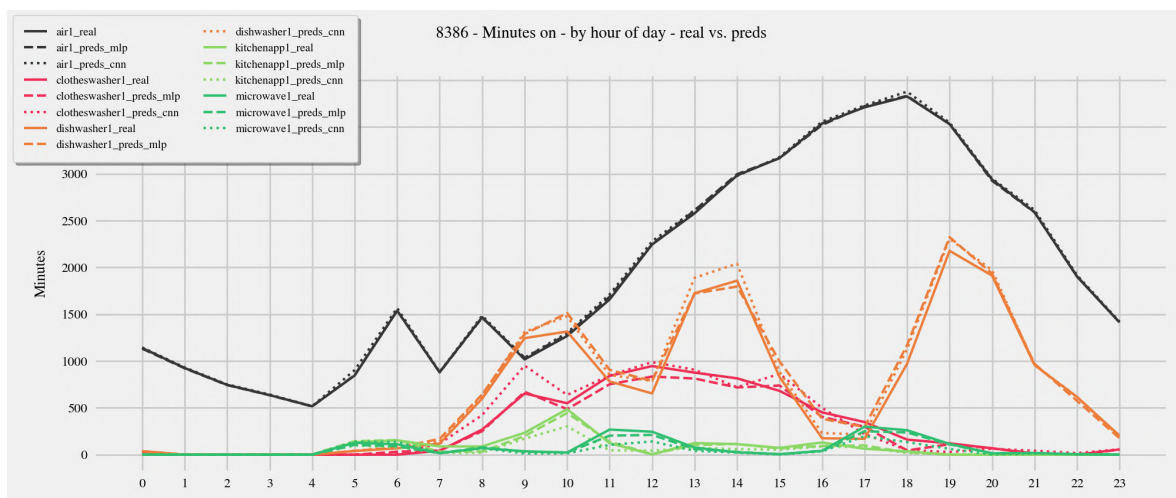


Figura 101 – Tiempo ‘on’ por hora del día – todos los electrodomésticos

Finalmente, en cuanto al microondas y circuito de cocina, se pueden sacar menos conclusiones puesto que el análisis es menos sensible a sus utilizaciones, que son más esporádicas y que con la resolución minutal del dataset, no permiten capturar su uso real. Sin embargo se pueden observar picos a las 10:00 en el caso del circuito de la cocina, o a las 11-12 y 17:00 para el microondas.

Capítulo 5. CONCLUSIONES Y TRABAJOS FUTUROS

En el contexto de NILM, los investigadores de este campo han utilizado todo tipo de modelos y algoritmos con el fin de intentar batir los resultados que se obtienen en una monitorización tradicional (invasiva, costosa y poco escalable) y así poder hacer de NILM la monitorización del futuro.

En este trabajo fin de máster se ha trabajado el diseño, desarrollo e implantación de una metodología alternativa y funcional en este campo, basado en modelos tipo MLP (para el modelo de referencia) y CNN debido a la robustez de su arquitectura y a los buenos resultados que ofrecen en los problemas de reconocimiento de imágenes. Para ello, y con el objetivo de identificar si un electrodoméstico está encendido o apagado en un determinado instante, se ha preparado un conjunto de variables derivadas del consumo agregado de todos los aparatos, organizadas matricialmente y a partir de la cual se realiza una representación gráfica a la que se ha denominado ‘vizgrama’. Un vizgrama no solo contiene información sobre el consumo eléctrico en un instante dado, sino que también contiene información del consumo pasado y futuro en diferentes ventanas de tiempo. A partir de esta codificación, se ha creado un primer modelo MLP de referencia y un modelo CNN con el que comparar los resultados.

En el capítulo 4 se presentan los resultados de la aplicación de esta metodología a un caso de uso de una vivienda en Texas. Los resultados obtenidos con los datos minutales reales de una vivienda son buenos y cumplen las expectativas con creces, llegándose a detectar las activaciones de los electrodomésticos en un alto porcentaje de casos. En el caso del MLP se ha ajustado un modelo por electrodoméstico, y los resultados son muy buenos para todos ellos. En el caso de la CNN, se ha trabajado con un único modelo para todos los electrodomésticos, obteniendo buenos resultados para los electrodomésticos de más larga duración (aire acondicionado, lavaplatos y lavadora) y resultados mejorables en los electrodomésticos más pequeños (microondas y circuito de cocina). También hay que

destacar que el vizgrama es muy escalable, pudiéndose cambiar cualquier parámetro que interviene en su construcción y que podría generar alguna mejora.

A pesar de los buenos resultados globales, el modelo, al haber sido solo entrenado con los datos de una única vivienda, no generaliza del todo bien cuando se ejecuta con los datos de una segunda vivienda y tendría que entrenarse con los datos de más viviendas para que generalice mejor. Al realizar transfer learning sobre este modelo con los datos de la segunda vivienda, se ha comprobado que para los electrodomésticos de uso más largo (lavadora o lavaplatos), el modelo conseguía resultados correctos pero que para los electrodomésticos más pequeños los resultados no mejoran.

Como trabajo futuro, entre las distintas alternativas que existen para tratar de mejorar el rendimiento del modelo obtenido destacan el procesamiento de datos, que, aunque haya sido de gran extensión en este trabajo, se podría mejorar y profundizar. Por ejemplo, se podría investigar la posibilidad de normalizar los datos de entrada en función del consumo máximo de la vivienda o en función de la potencia contratada. También se podrían realizar más pruebas con longitudes distintas de ventana de vizgrama, o distintas frecuencias de muestreo, a intervalos más separados. Además, la aplicación de la metodología propuesta sobre un conjunto mayor de datos, tanto en periodo de tiempo como en número de viviendas, sería sin duda la manera más efectiva de mejorar los resultados obtenidos. En el ámbito del machine learning, disponer de suficientes datos para que el modelo entrenado sea generalizable y estable es clave.

Capítulo 6. REFERENCIAS

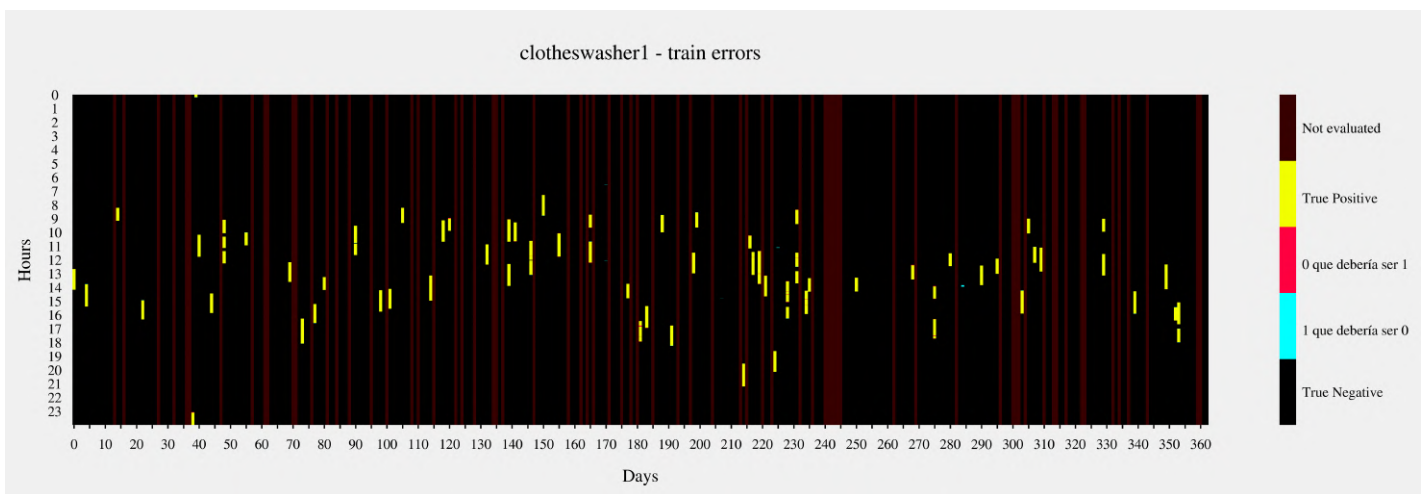
- [1] Knottenbelt, W.: Does disaggregated electricity feedback reduce domestic electricity consumption? A systematic review of the literature (2016)
- [2] The use of apps to promote energy saving: a study of smart meter-related feedback in the Netherlands Daphne Geelen et al. (2017)
- [3] Abubakar, I., et al.: Application of load monitoring in appliances energy management a review. *Renew. Sustain. Energy Rev.* 67, 235–245 (2017)
- [4] Darby, S.: The effectiveness of feedback on energy consumption a review for DEFRA of the literature on metering, billing and direct displays. In: Environmental Change Institute University of Oxford, 22 April, pp. 1–21 (2006)
- [5] A review of data-driven approaches for prediction and classification of building energy consumption. Yixuan Wei et al. (2017)
- [6] Kolter, J.Z., Johnson, M.J.: REDD : a public data set for energy disaggregation research. In: *SustKDD Workshop*, vol. 1, pp. 1–6 (2011)
- [7] Kelly, J., Knottenbelt, W.: The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. *Sci. Data* 2, 150007 (2015)
- [8] Makonin, S., et al.: Electricity, water, and natural gas consumption of a residential house in Canada from 2012 to 2014. *Sci. Data* 3(160037), 1–12 (2016)
- [9] Pecan Street Data – Pecan Street Inc.
- [10] Bhotto, M.Z.A., Makonin, S., Bajic, I.: Load disaggregation based on aided linear integer programming. *IEEE Trans. Circuits Syst. II Express Briefs* 1 (2016)
- [11] Baranski, M., Voss, J.: Genetic algorithm for pattern detection in NIALM systems. In: *IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, pp. 3462–3468. IEEE (2004)
- [12] Hart, G.W.: Nonintrusive appliance load monitoring. *Proc. IEEE* 80(12), 1870– 1891 (1992)
- [13] Kato, T., Cho, H.S., Lee, D., Toyomura, T., Yamazaki, T.: Appliance recognition from electric current signals for information-energy integrated network in home environments. In: Mokhtari, M., Khalil, I., Bauchet, J., Zhang, D., Nugent, C. (eds.) *Ambient Assistive Health*

- and Wellness Management in the Heart of the City, ICOST 2009. LNCS, vol. 5597, pp. 150–157. Springer, Heidelberg (2009)
- [14] Srinivasan, D., Ng, W.S., Liew, A.C.: Neural-network-based signature recognition for harmonic source identification. *IEEE Trans. Power Deliv.* 21(1), 398–405 (2006)
 - [15] Gonçalves, H., et al.: Unsupervised disaggregation of appliances using aggregated consumption data. In: *Environmental Engineering* (2011)
 - [16] Shao, H., Tech, V., Marwah, M.: A temporal motif mining approach to unsupervised energy disaggregation. In: *1st International Workshop on Non-Intrusive Load Monitoring*, pp. 1–2 (2012)
 - [17] Kim, H., Marwah, M., Arlitt, M., Lyon, G., Han, J.: Unsupervised disaggregation of low frequency power measurements. In: *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 747–758 (2011)
 - [18] Jia, R., Gao, Y., Spanos, C.J.: A fully unsupervised nonintrusive load monitoring framework. In: *2015 IEEE International Conference on Smart Grid Communications, SmartGridComm 2015*, pp. 872–878 (2016)
 - [19] Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.-A.: Extracting and composing robust features with denoising autoencoders. In: *Proceedings of the 25th International Conference on Machine Learning – ICML 2008*, pp. 1096–1103 (2008)
 - [20] Kim, H., Marwah, M., Arlitt, M., Lyon, G., Han, J.: Unsupervised disaggregation of low frequency power measurements. In: *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 747–758 (2011)
 - [21] Makonin, S.: Real-time embedded low-frequency load disaggregation. Pp. 1–121, August 2014
 - [22] A review on buildings energy consumption information, Luis Pérez Lombard et al. (2008)
 - [23] Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, R.K. Pachauri and L.A. Meyer (eds.)]. IPCC, Geneva, Switzerland, 151 pp
 - [24] A review on the prediction of building energy consumption. Hai-xiangZhao et al. 2012
 - [25] Standby Power Use: How Big Is the Problem? What Policies and Technical Solutions Can Address It? Paolo Bertoldi et al.
 - [26] Green home energy management system through comparison of energy usage between the same kinds of home appliances. Jinsoo Ham et al. 2011

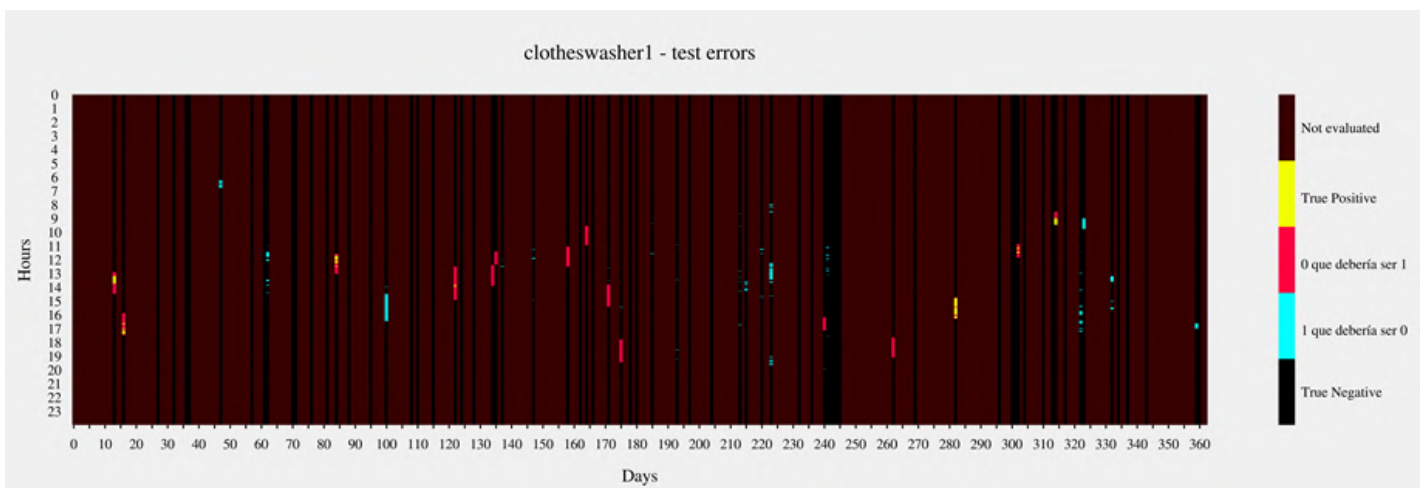
- [27] Srinivasarengan, K., et al.: A framework for non intrusive load monitoring using Bayesian inference
- [28] Benchmarking smart metering deployment in the EU-28. December – 2019
- [29] ACER/CEER – Annual Report on the Results of Monitoring the Internal Electricity and Natural Gas Markets in 2017 – Consumer Empowerment Volume
- [30] <https://dataport.pecanstreet.org/academic>
- [31] https://rossa-prod-ap21.ethz.ch:8443/delivery/DeliveryManagerServlet?dps_pid=IE594964
- [32] Jack Kelly and William Knottenbelt. The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. Scientific Data 2, Article number:150007, 2015
- [33] <https://www.nature.com/articles/sdata2016122>
- [34] <https://iawe.github.io/>
- [35] <http://redd.csail.mit.edu/>
- [36] <https://tokhub.github.io/dbecd/links/Blued.html>
- [37] <https://arxiv.org/abs/1705.05767>
- [38] https://www.peculiar-coding-endeavours.com/2019/mlp_vs_cnn/
- [39] Regularization for Deep Learning: A Taxonomy. Jan Kukačka, Vladimir Golkov, and Daniel Cremers
- [40] Hanin and M. Sellke. Approximating Continuous Functions by ReLU Nets of Minimal Width. arXiv:1710.11278, 2017.
- [41] <https://www.jeremyjordan.me/convnet-architectures/>
- [42] Striving for Simplicity: The All Convolutional Net, Jost Tobias Springenberg et. Al, 2014. arXiv:1412.6806
- [43] <https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2>
- [44] Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, Sergey Ioffe and Christian Szegedy, 2015, arXiv:1502.03167
- [45] Deep Learning (Adaptive Computation and Machine Learning series)
- [46] Srivastava, N.; Geoffrey, H.; Alex, K.; Ilya, S.; Ruslan, S. Dropout: A simple way to prevent neural networks from over-fitting. J. Mach. Learn. Res. 2014, 15, 1929–1958.

ANEXO A – RESULTADOS MLP

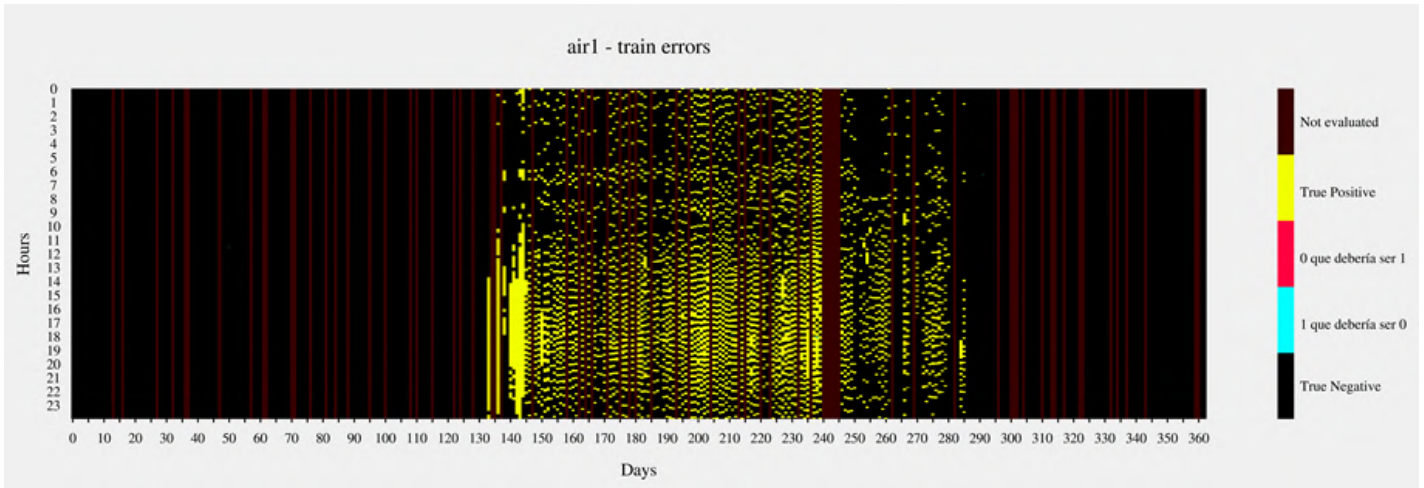
En este anexo se presentan los resultados equivalentes a la matriz de confusión de forma gráfica de los distintos electrodomésticos, evaluados con el modelo MLP de referencia.



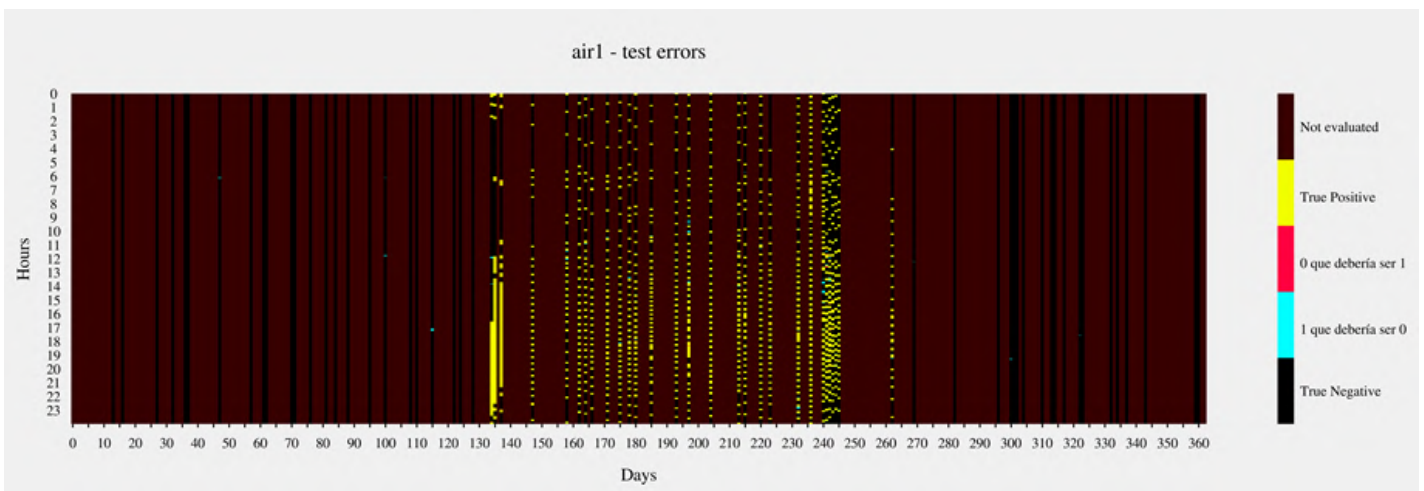
Anexo A – Figura 1– Resultados mejor modelo MLP – clotheswasher1 – train



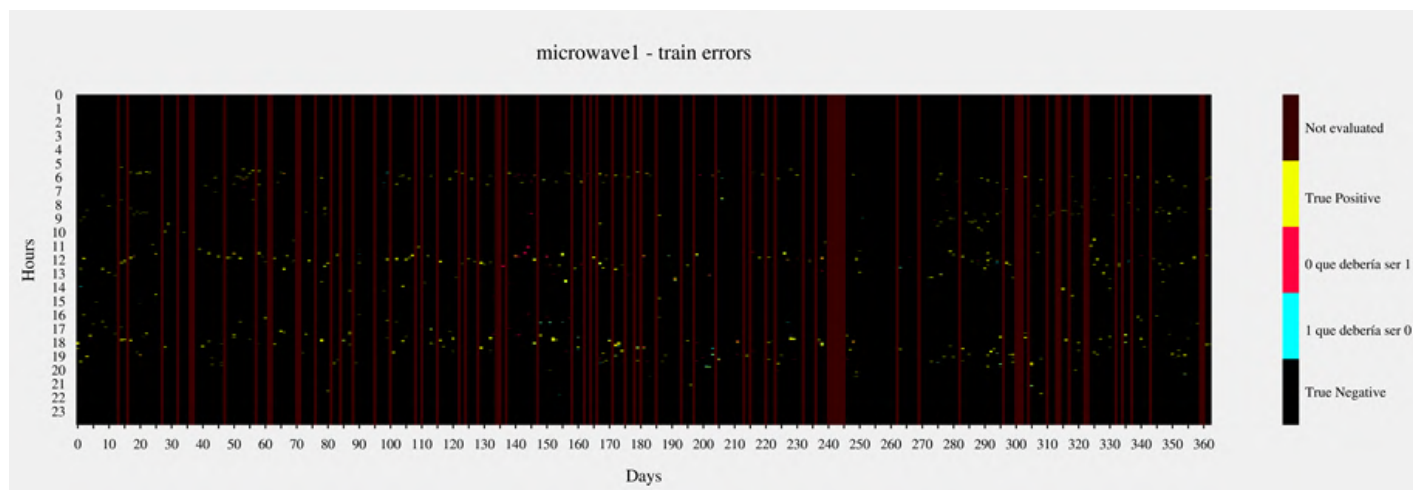
Anexo A – Figura 2– Resultados mejor modelo MLP – clotheswasher1 – test



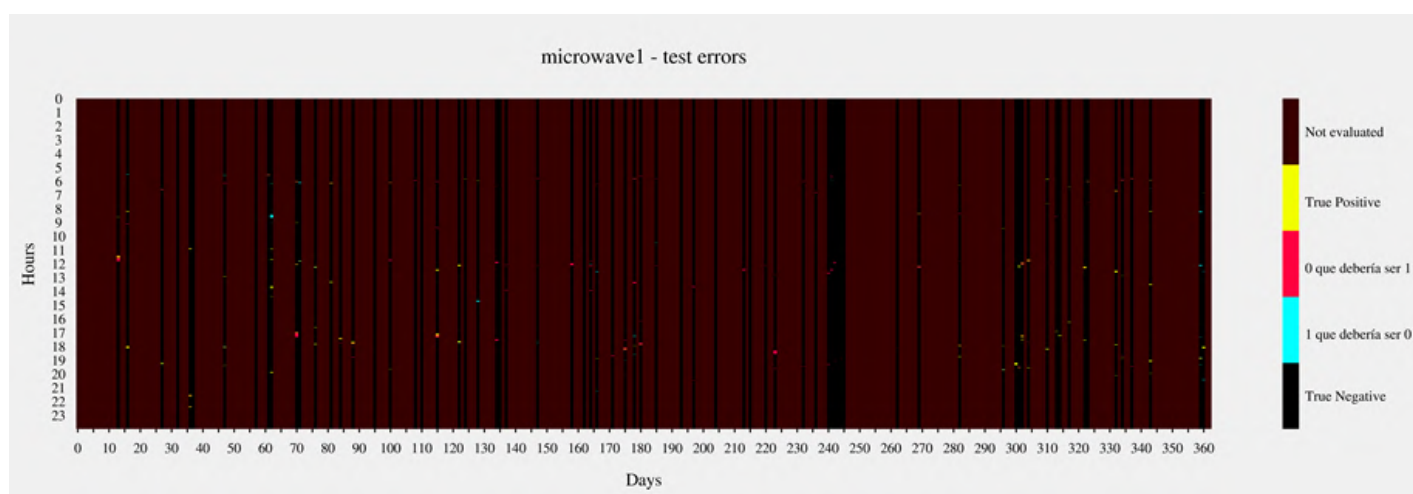
Anexo A – Figura 3– Resultados mejor modelo MLP – air1 – train



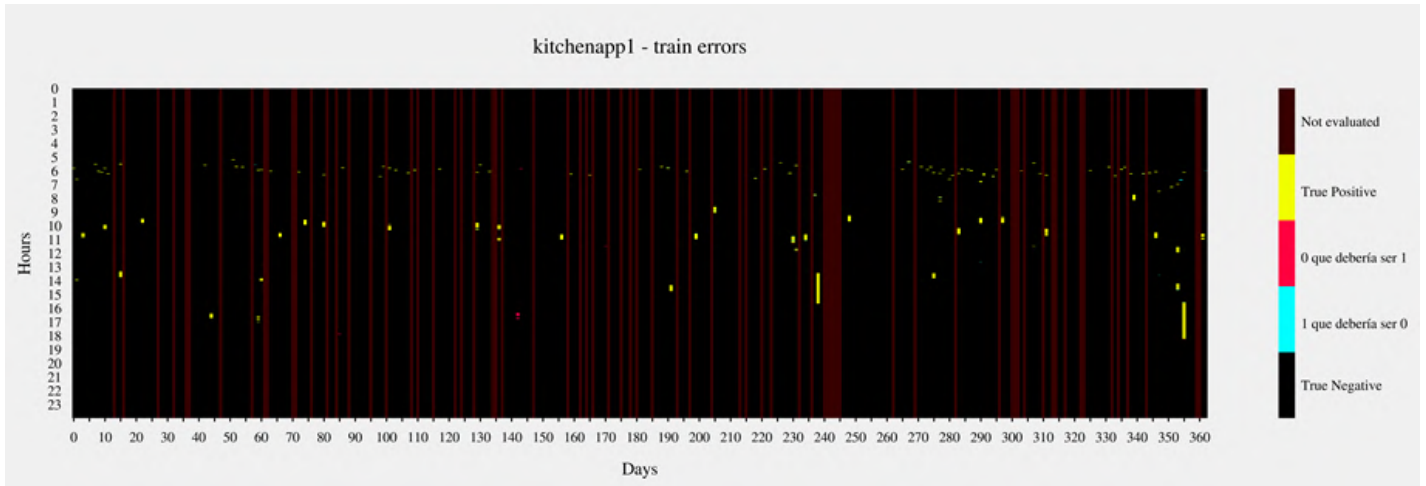
Anexo A – Figura 4– Resultados mejor modelo MLP – air1 – test



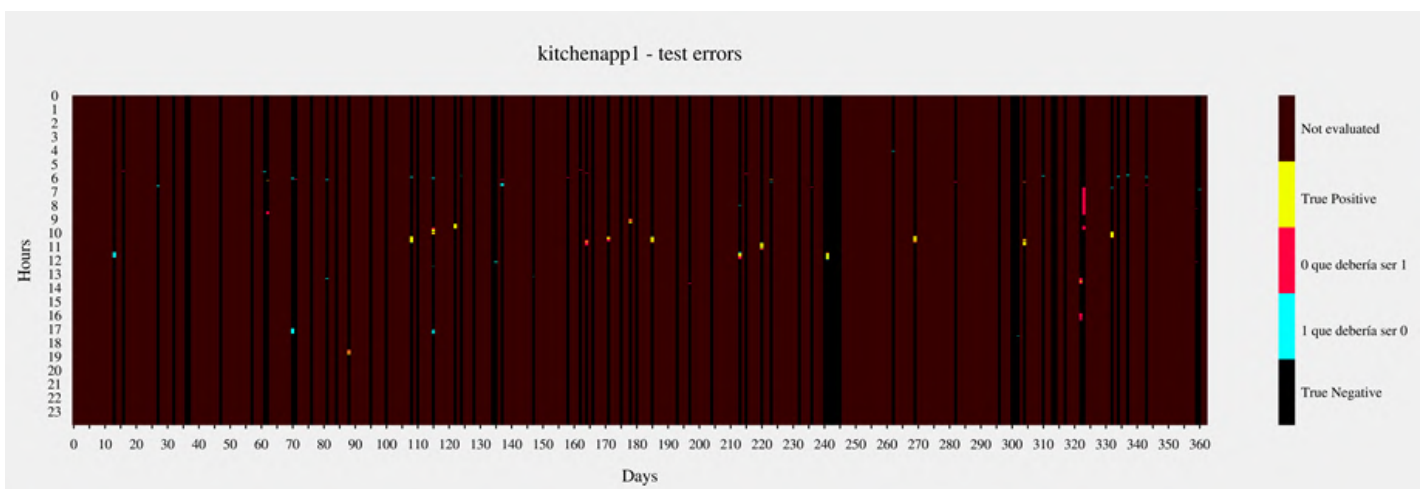
Anexo A – Figura 5– Resultados mejor modelo MLP – microwave1 – train



Anexo A – Figura 6– Resultados mejor modelo MLP – microwave1 – test



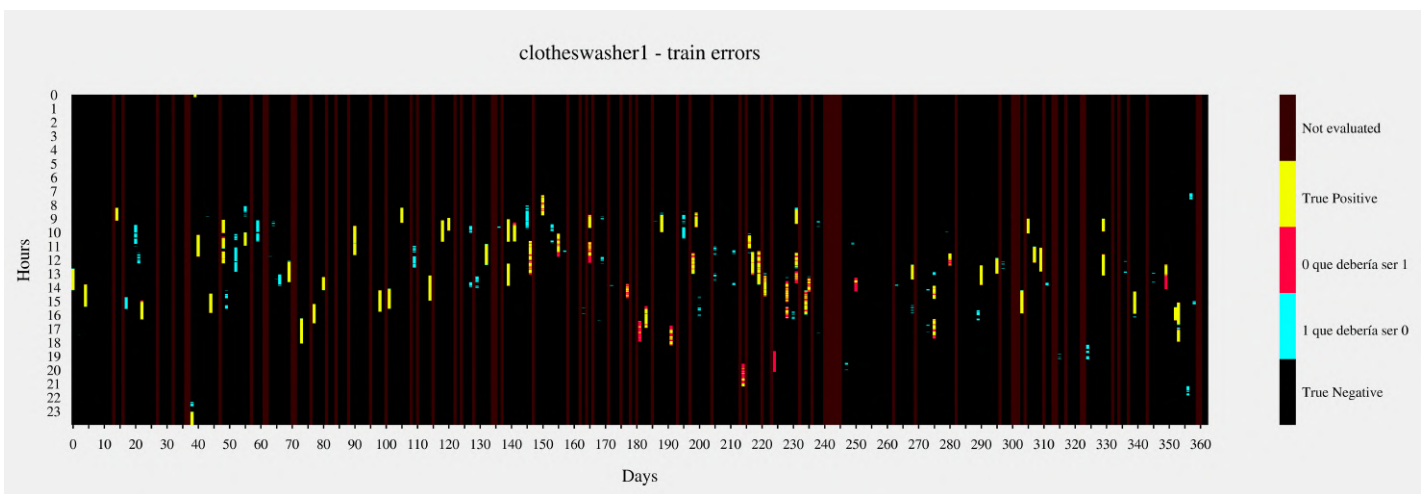
Anexo A – Figura 7– Resultados mejor modelo MLP – kitchenapp1 – train



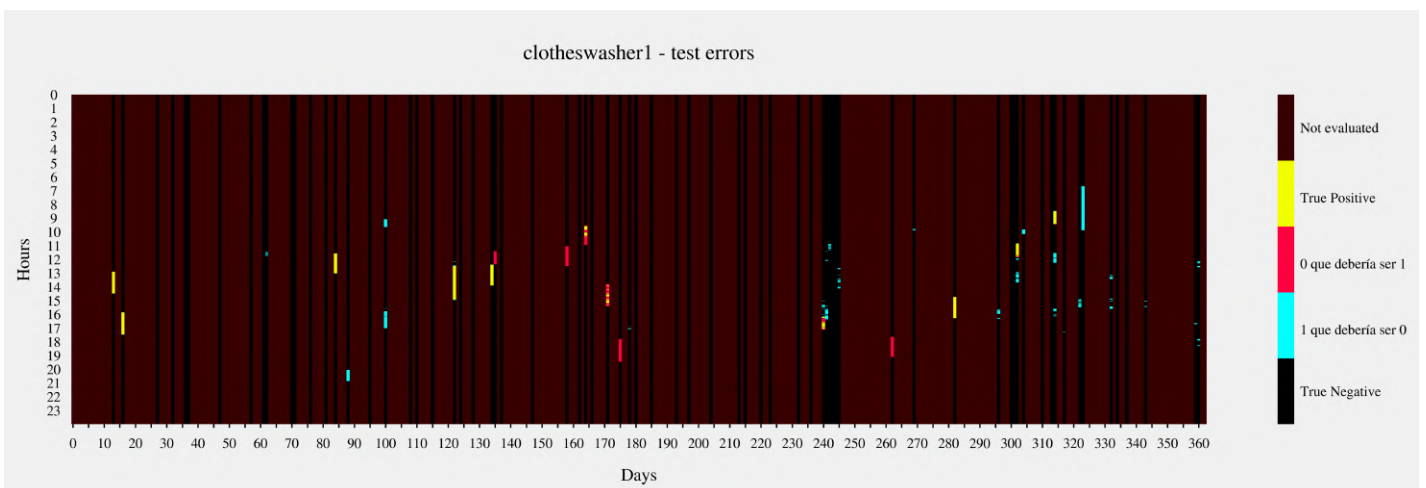
Anexo A – Figura 8– Resultados mejor modelo MLP – kitchenapp1 – test

ANEXO B – RESULTADOS CNN

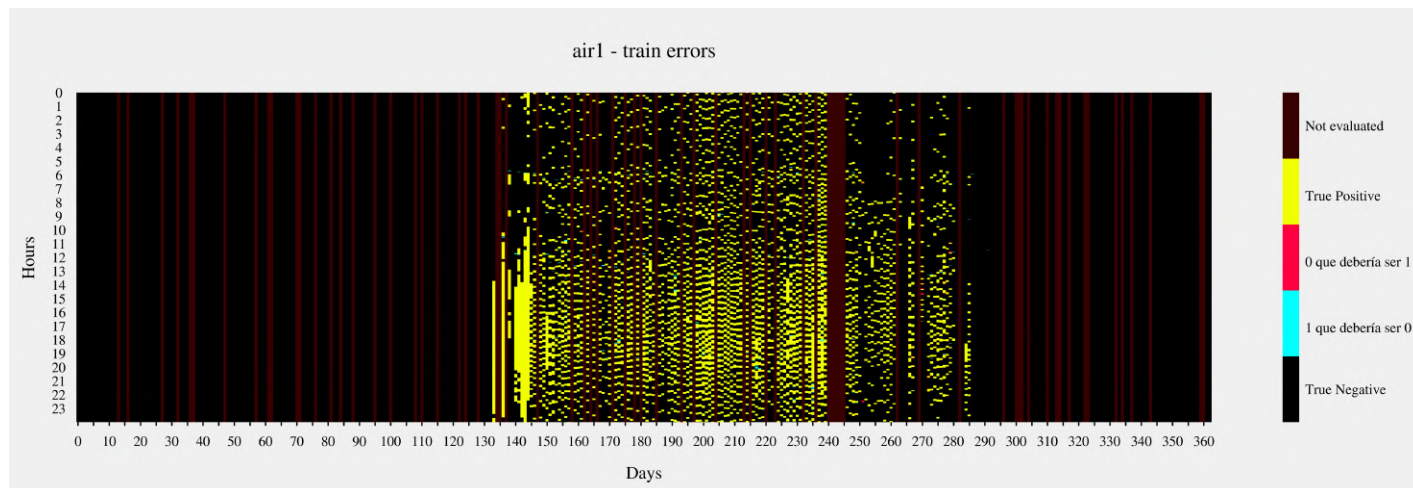
En este anexo se presentan los resultados equivalentes a la matriz de confusión de forma gráfica de los distintos electrodomésticos, evaluados con el mejor modelo CNN obtenido.



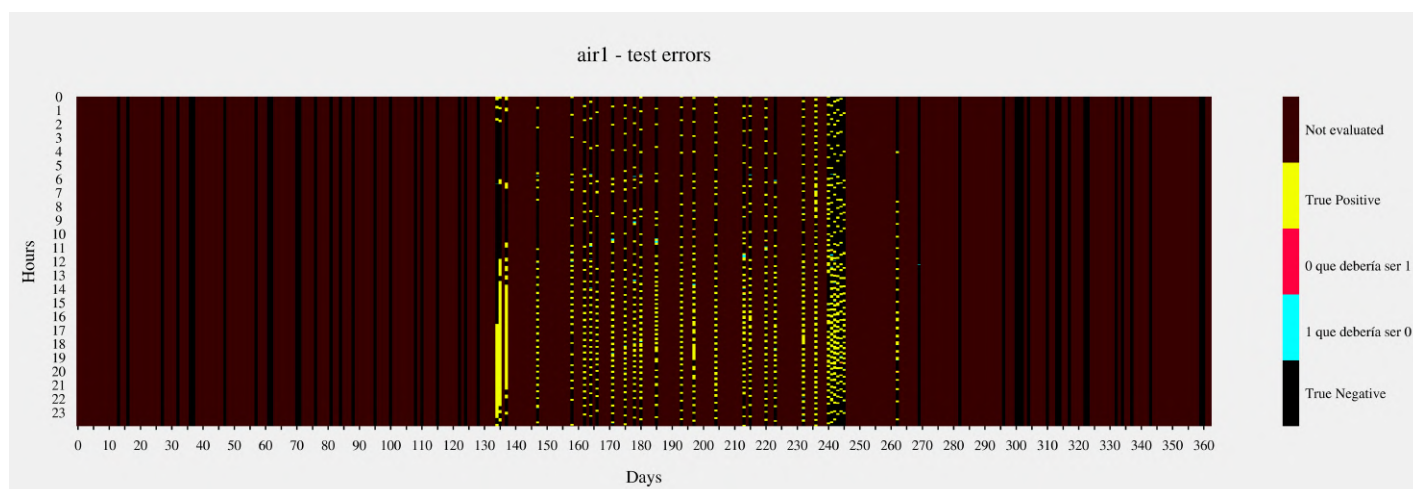
Anexo B – Figura 1 – Resultados mejor modelo CNN – clotheswasher1 – train



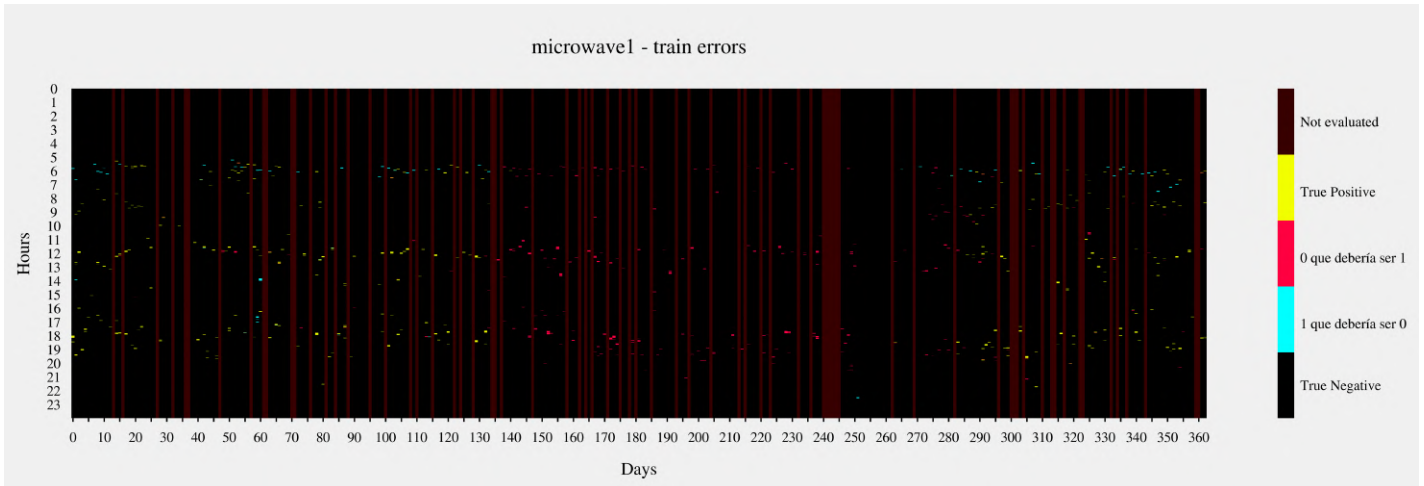
Anexo B – Figura 2 – Resultados mejor modelo CNN – clotheswasher1 – test



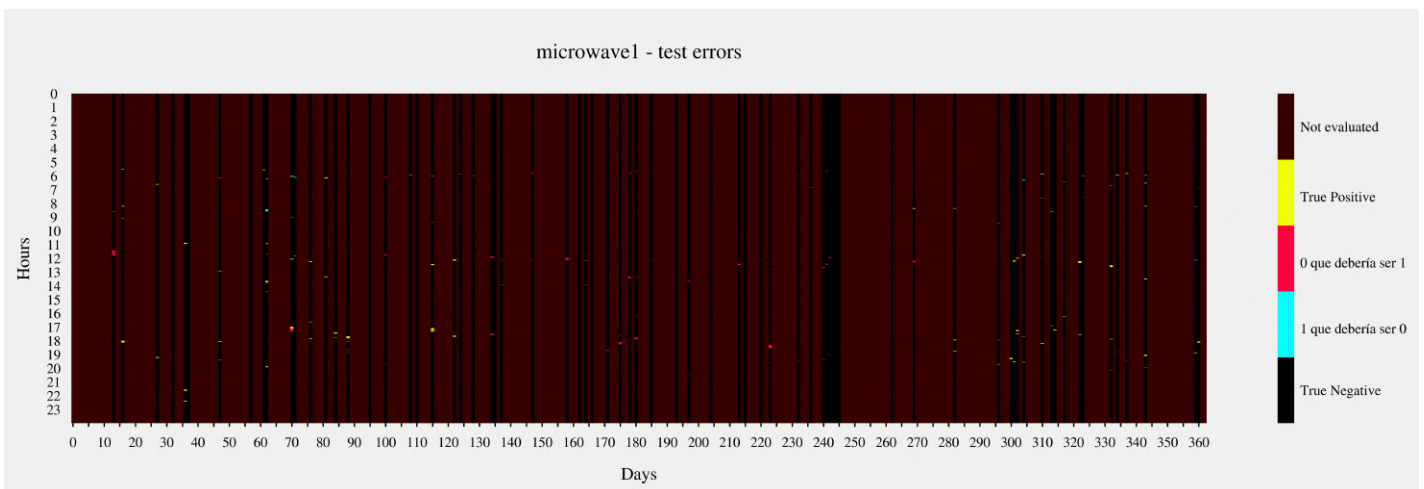
Anexo B – Figura 3 – Resultados mejor modelo CNN – air1 – train



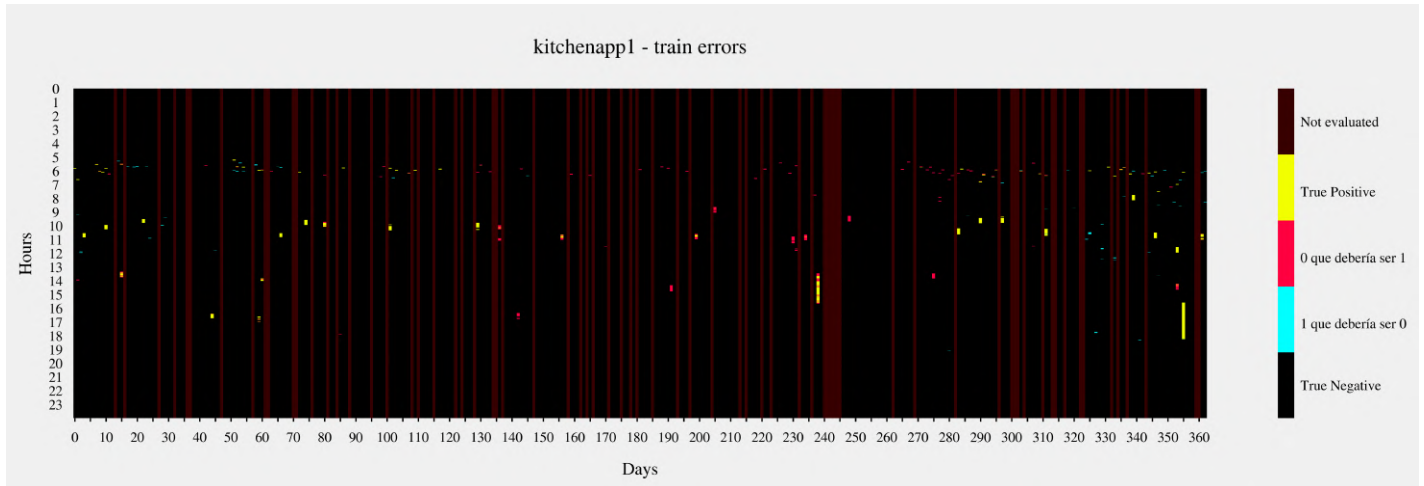
Anexo B – Figura 4 – Resultados mejor modelo CNN – air1 – test



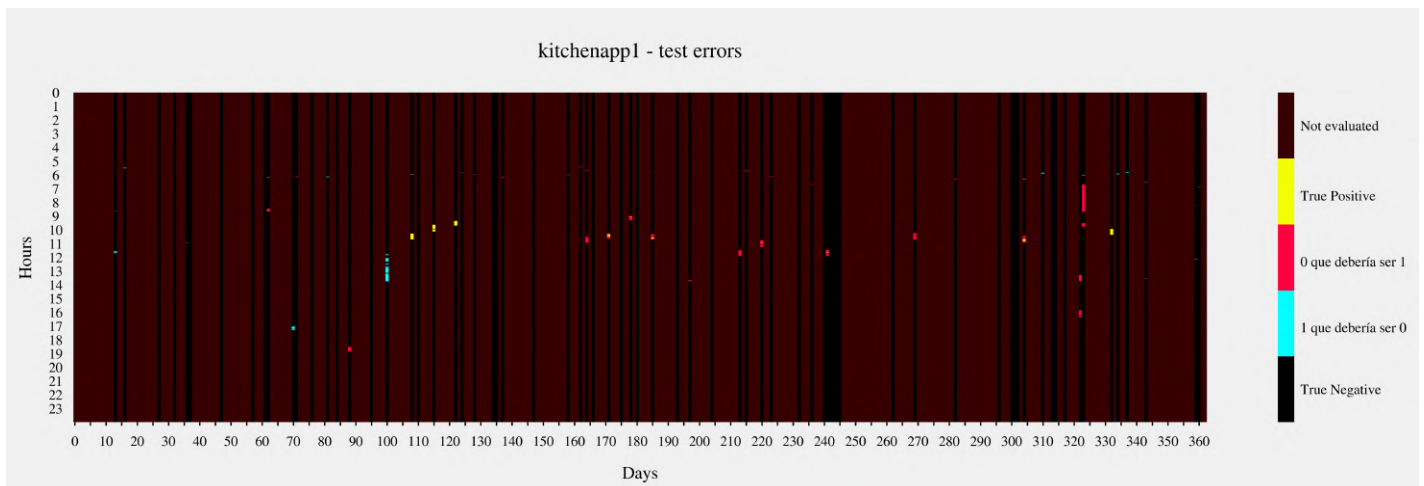
Anexo B – Figura 5 – Resultados mejor modelo CNN – microwave1 – train



Anexo B – Figura 6 – Resultados mejor modelo CNN – microwave1 – test



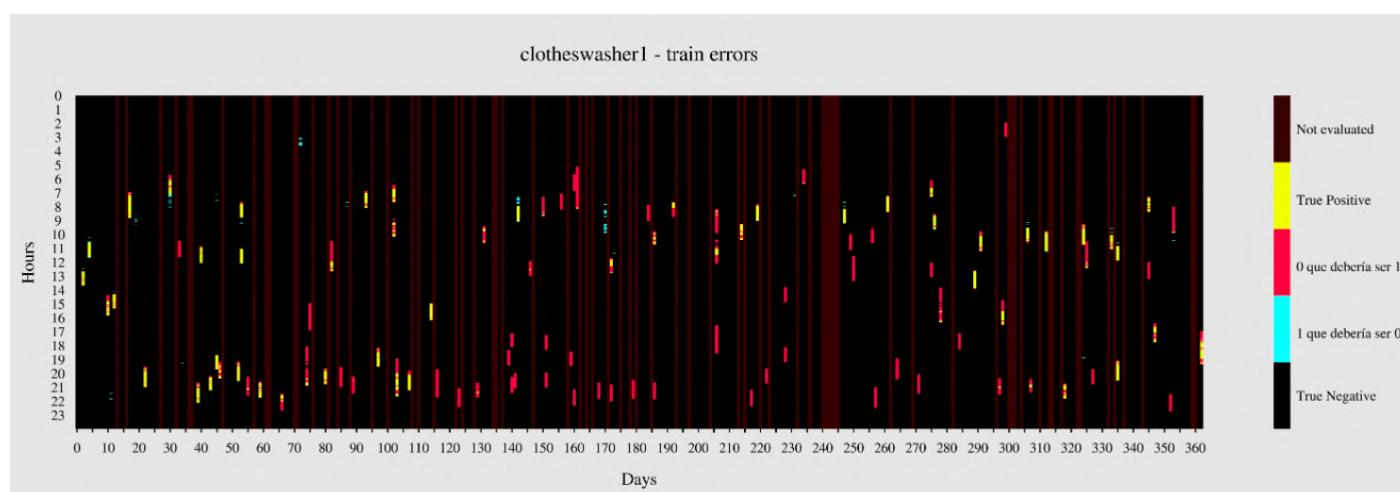
Anexo B – Figura 7 – Resultados mejor modelo CNN – kitchenapp1 – train



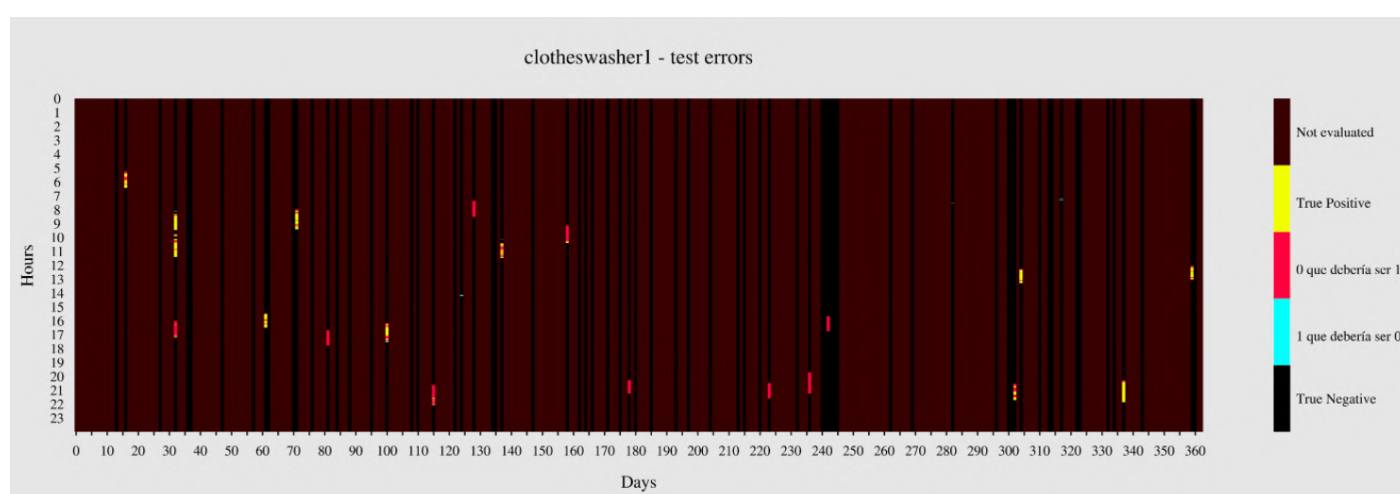
Anexo B – Figura 8 – Resultados mejor modelo CNN – kitchenapp1 – test

ANEXO C – RESULTADOS CNN SEGUNDA VIVIENDA

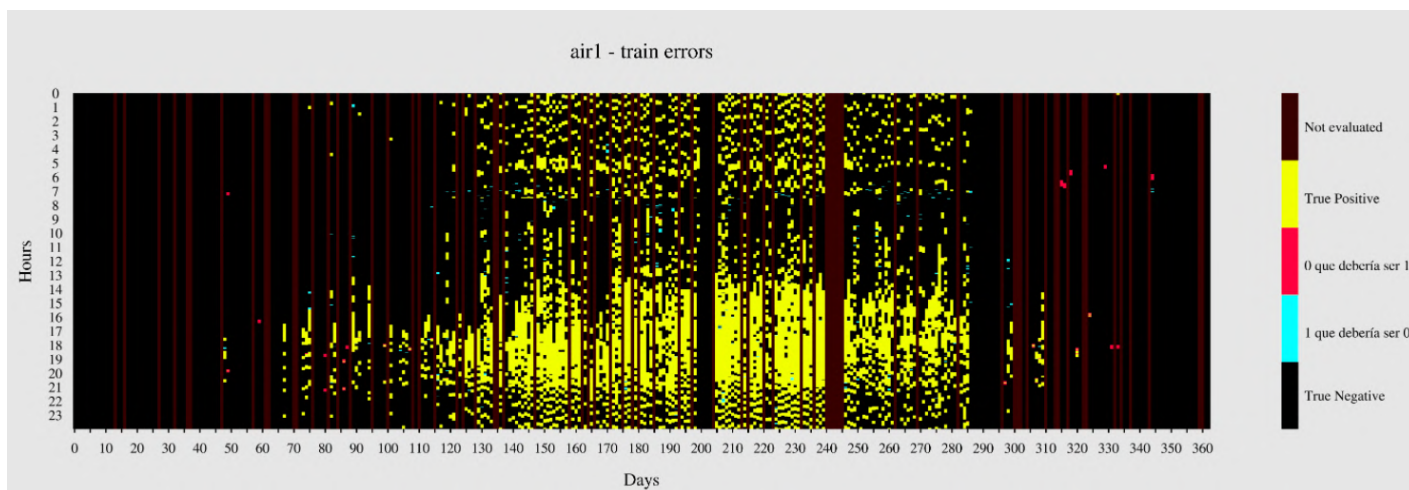
En este anexo se presentan los resultados equivalentes a la matriz de confusión de forma gráfica de los distintos electrodomésticos, evaluados con el mejor modelo CNN obtenido, pero sobre los datos de una segunda vivienda.



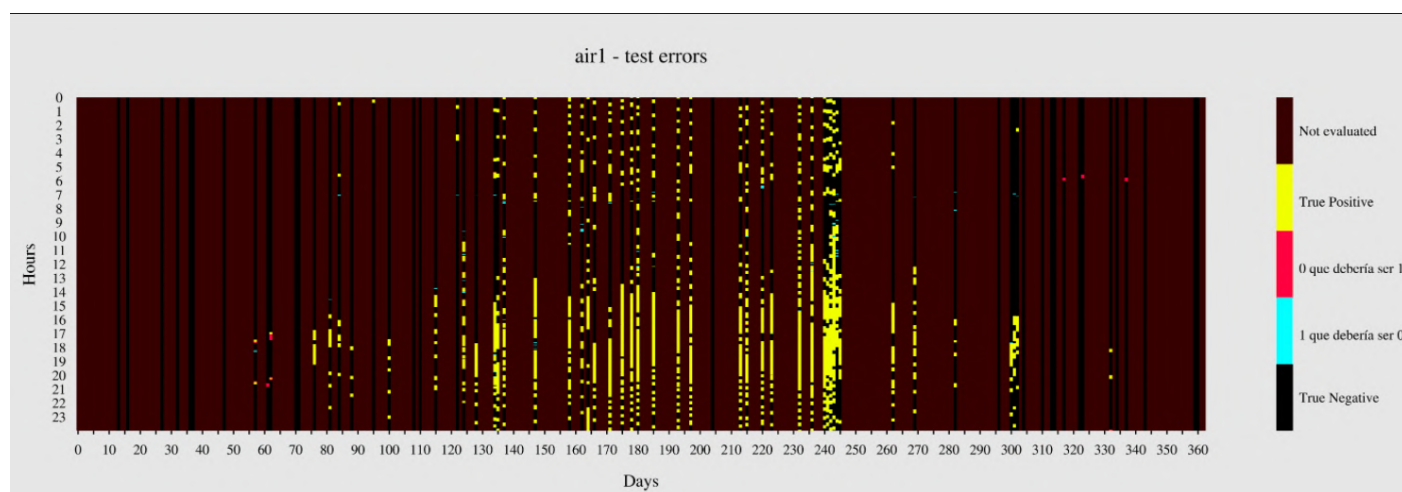
Anexo C – Figura 1 – Resultados CNN sobre otro hogar – `clotheswasher1` – train



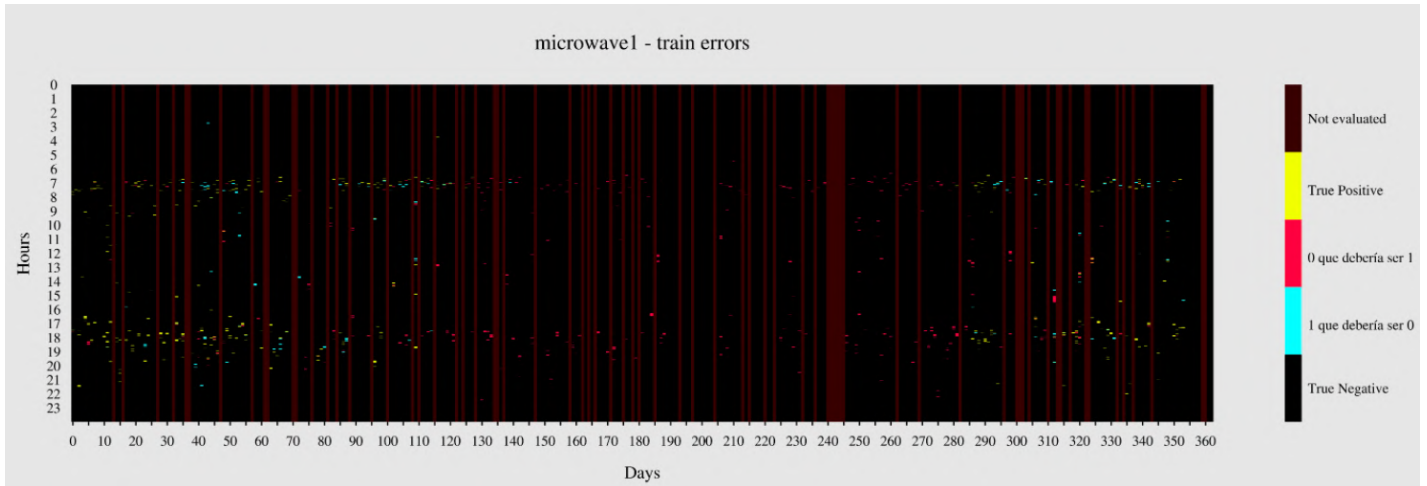
Anexo C – Figura 2 – Resultados CNN sobre otro hogar – `clotheswasher1` – test



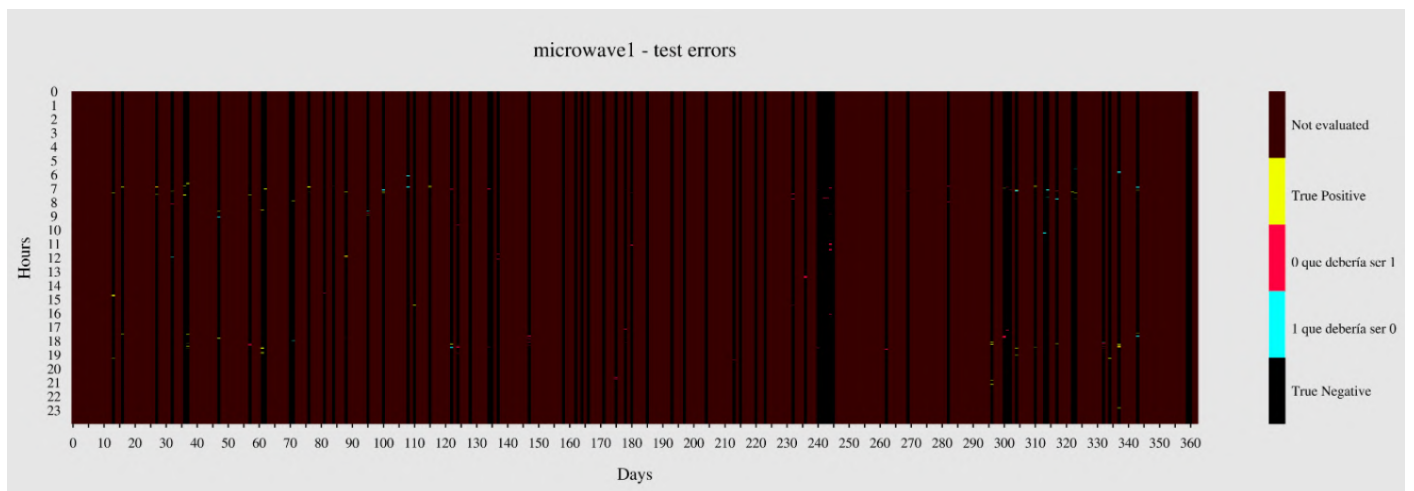
Anexo C – Figura 3 – Resultados CNN sobre otro hogar – *air1* – train



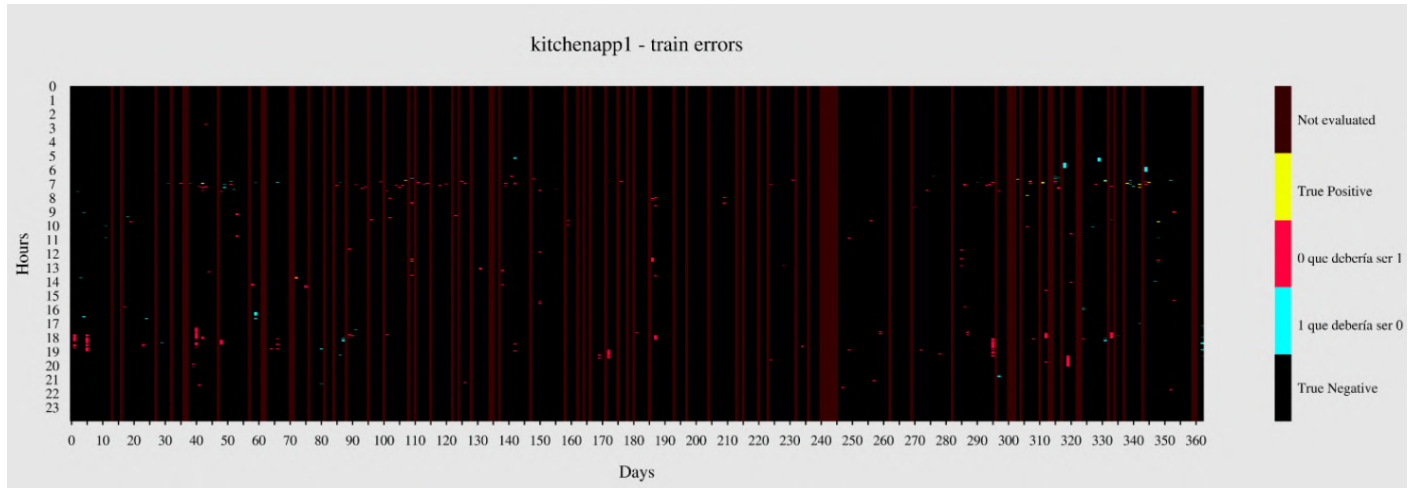
Anexo C – Figura 4 – Resultados CNN sobre otro hogar – *air1* – test



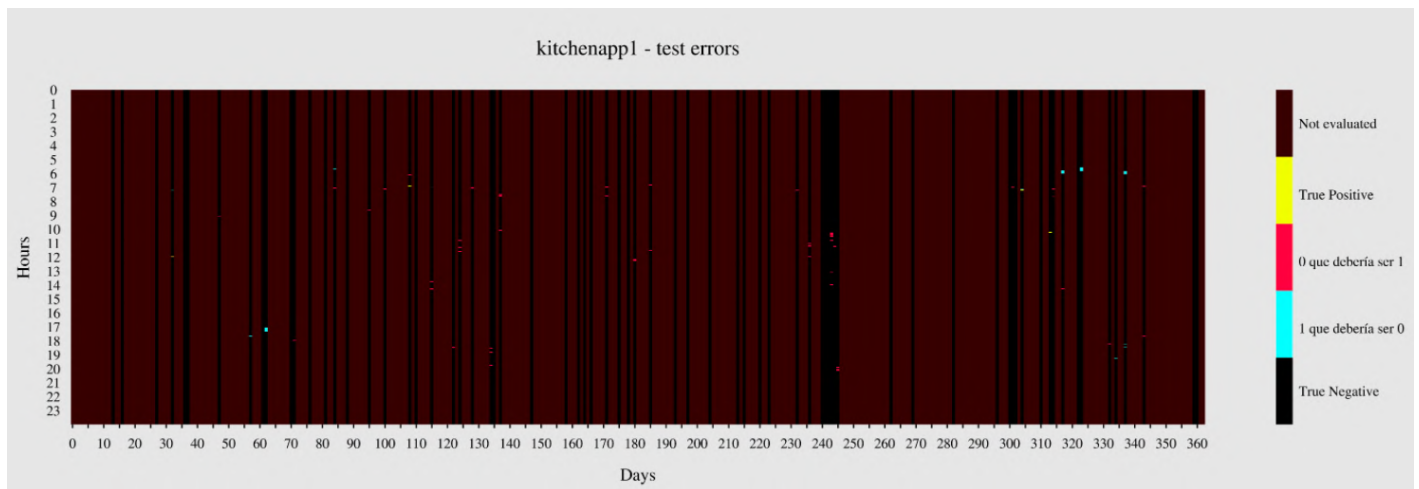
Anexo C – Figura 5 – Resultados CNN sobre otro hogar – `microwave1` – train



Anexo C – Figura 6 – Resultados CNN sobre otro hogar – `microwave1` – test



Anexo C – Figura 7 – Resultados CNN sobre otro hogar – *kitchenapp1* – train

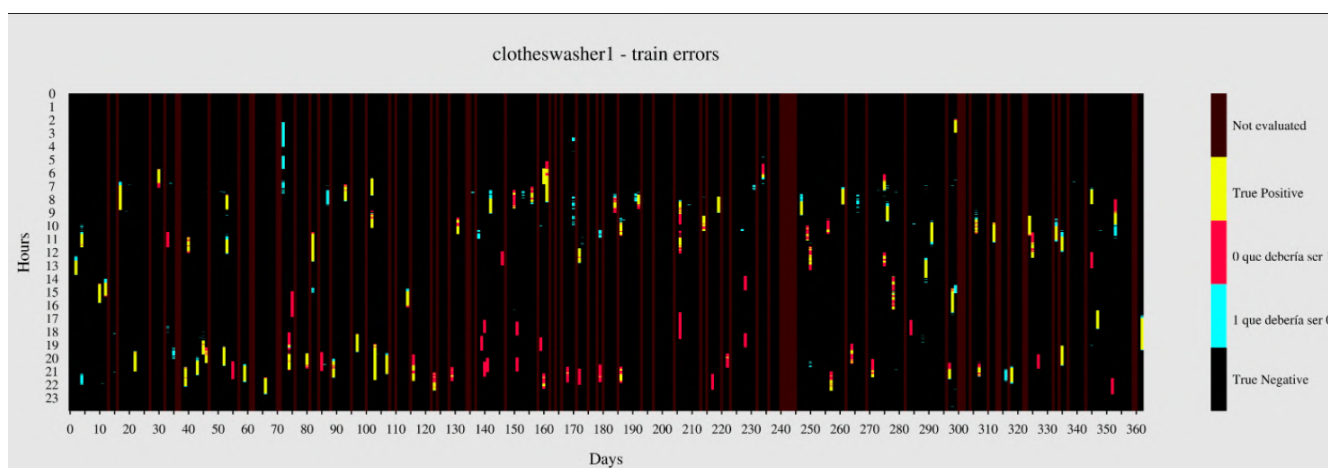


Anexo C – Figura 8 – Resultados CNN sobre otro hogar – *kitchenapp1* – test

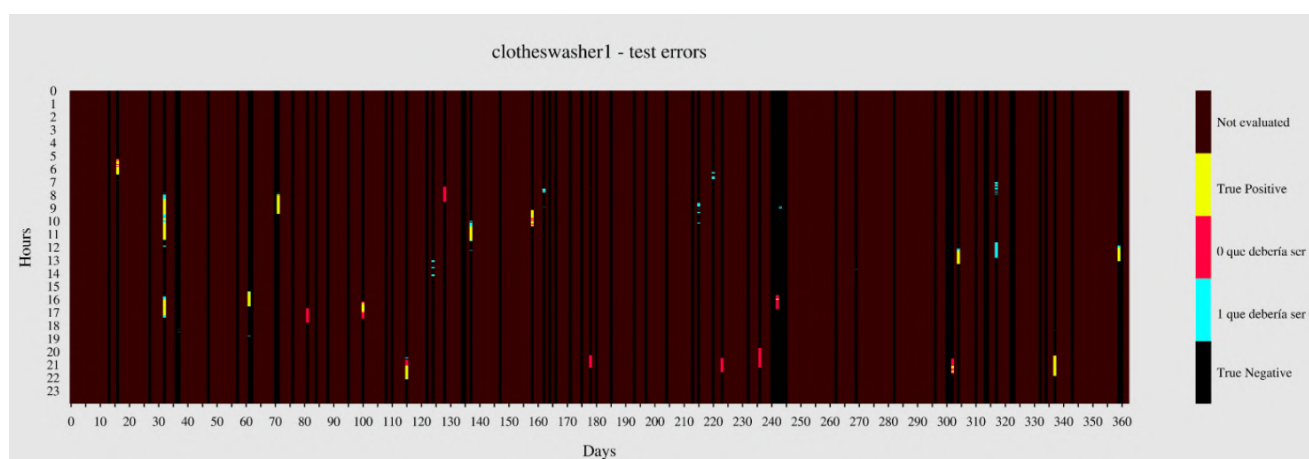
ANEXO D – RESULTADOS CNN SEGUNDA VIVIENDA

Y TRANSFER LEARNING

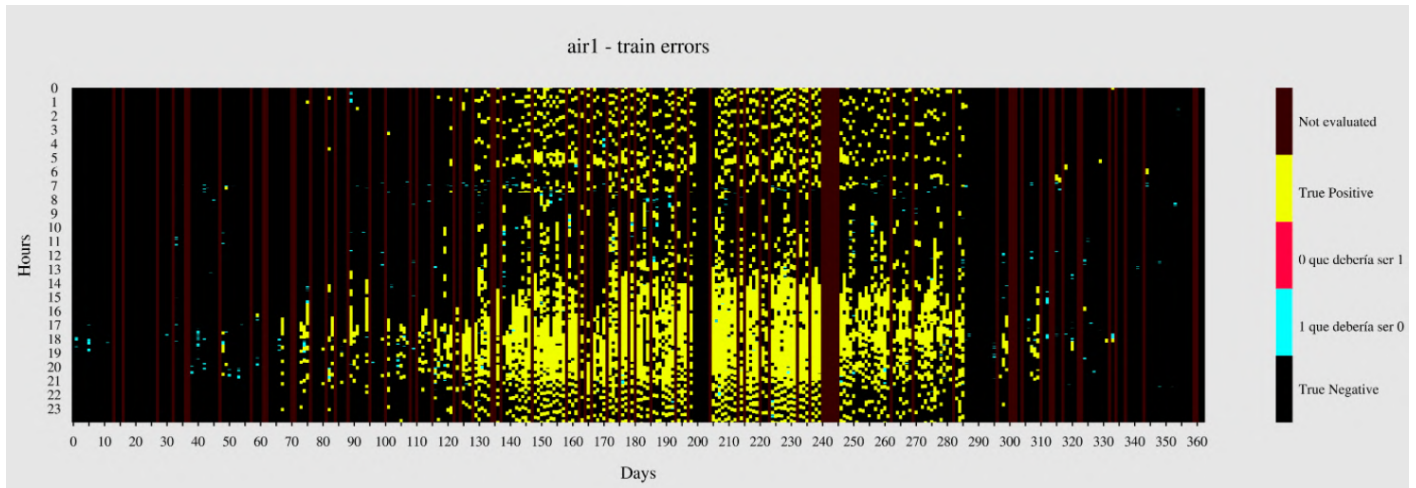
En este anexo se presentan los resultados equivalentes a la matriz de confusión de forma gráfica de los distintos electrodomésticos, evaluados con el mejor modelo CNN obtenido y sobre el que se ha realizado transfer learning de las dos últimas capas densas, La evaluación se realiza sobre los datos de una segunda vivienda.



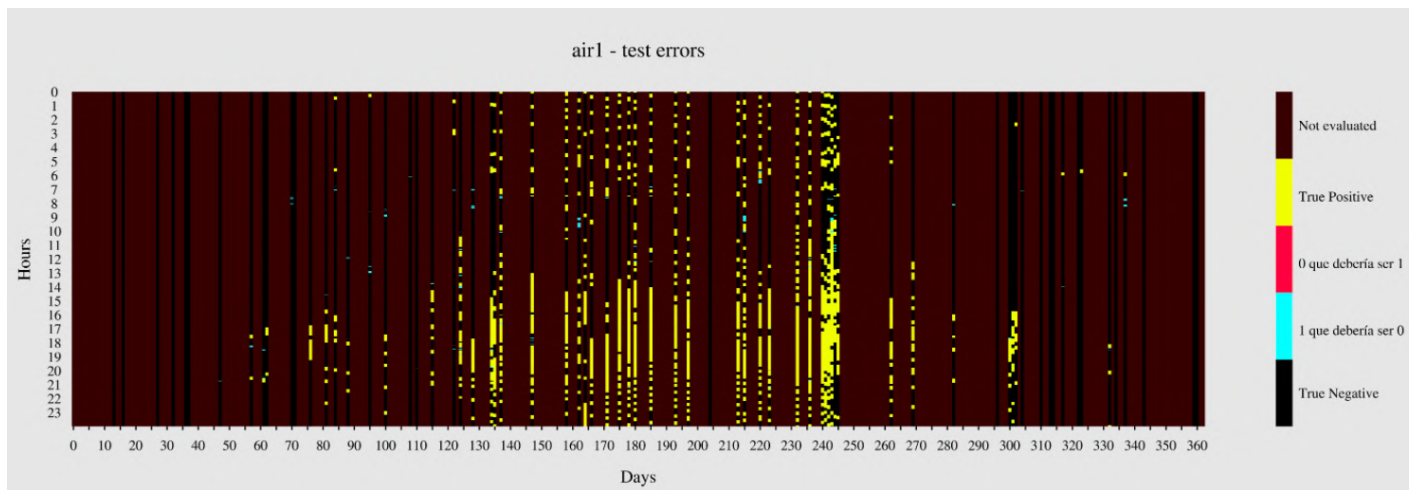
Anexo D – Figura 1 – Resultados CNN con transfer Learning sobre otro hogar – clotheswasher1 – train



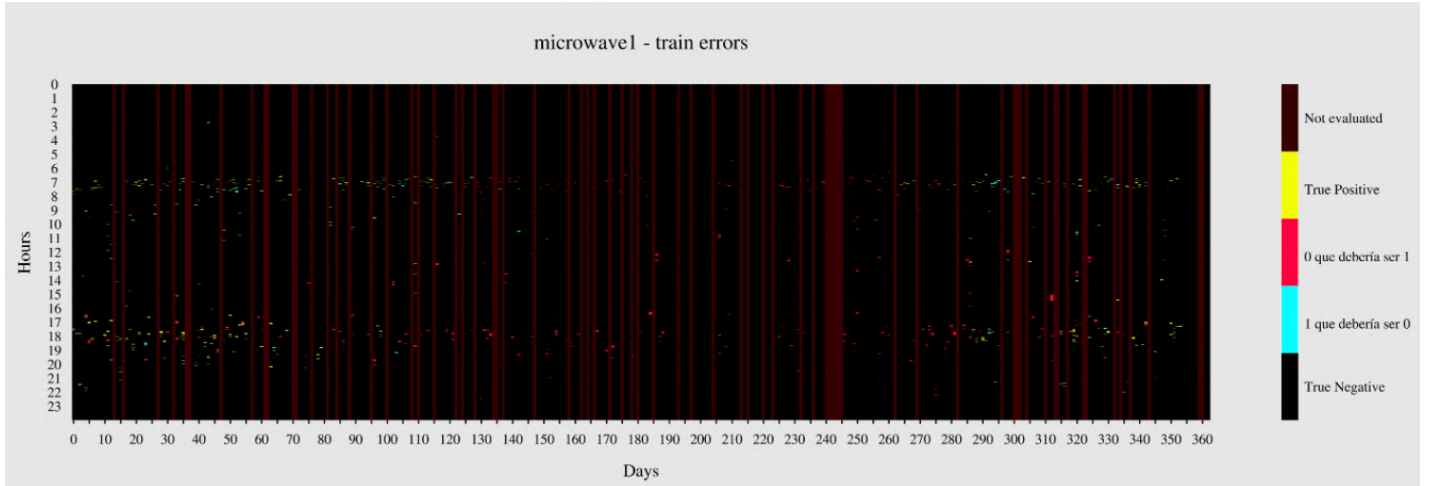
Anexo D – Figura 2 – Resultados CNN con transfer Learning sobre otro hogar – clotheswasher1 – test



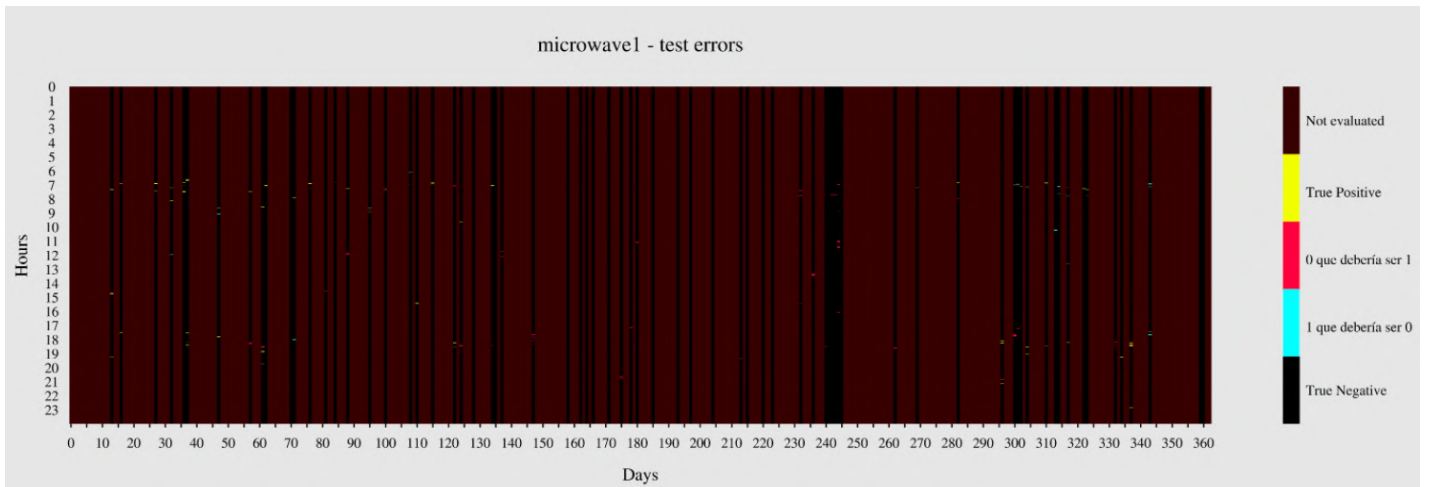
Anexo D – Figura 3 – Resultados CNN con transfer Learning sobre otro hogar – air1 – train



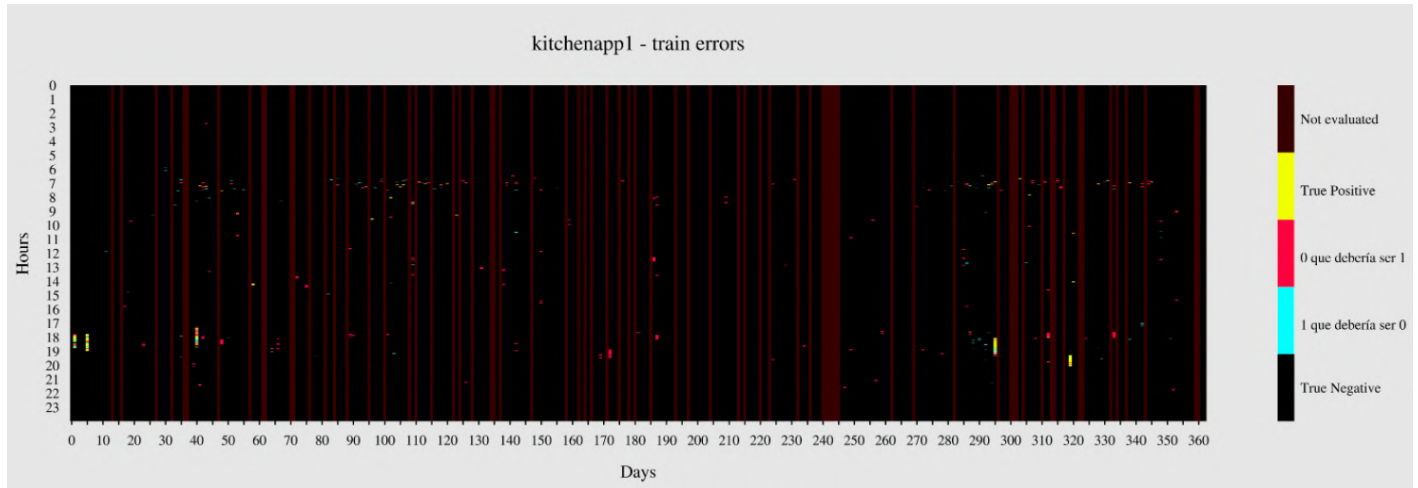
Anexo D – Figura 4 – Resultados CNN con transfer Learning sobre otro hogar – air1 – test



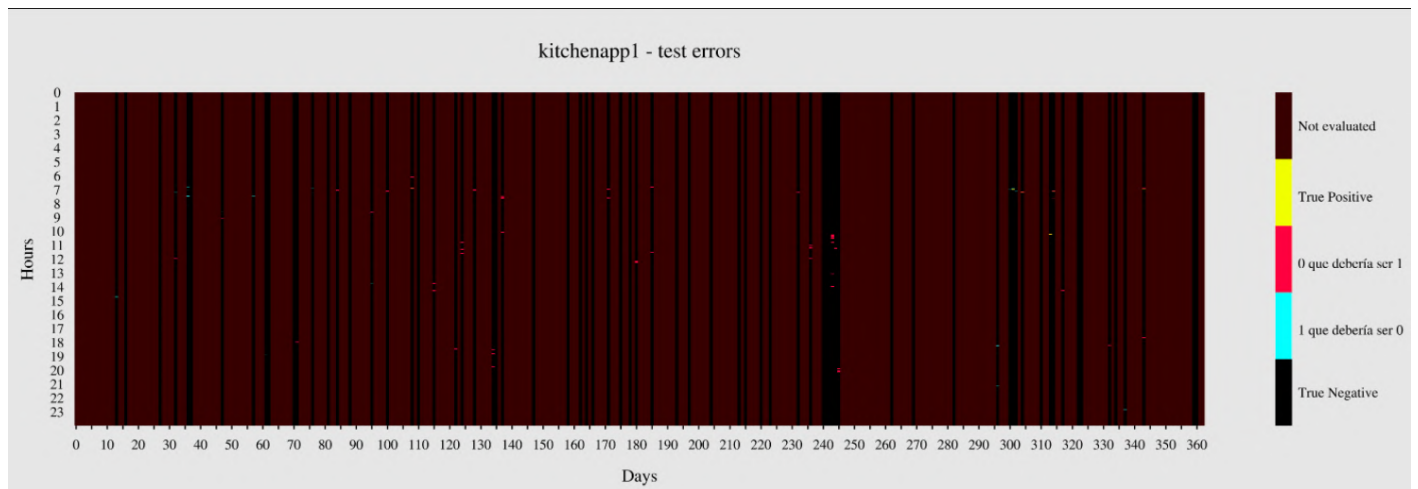
Anexo D – Figura 5 – Resultados CNN con transfer Learning sobre otro hogar – microwave1 – train



Anexo D – Figura 6 – Resultados CNN con transfer Learning sobre otro hogar – microwave1 – test



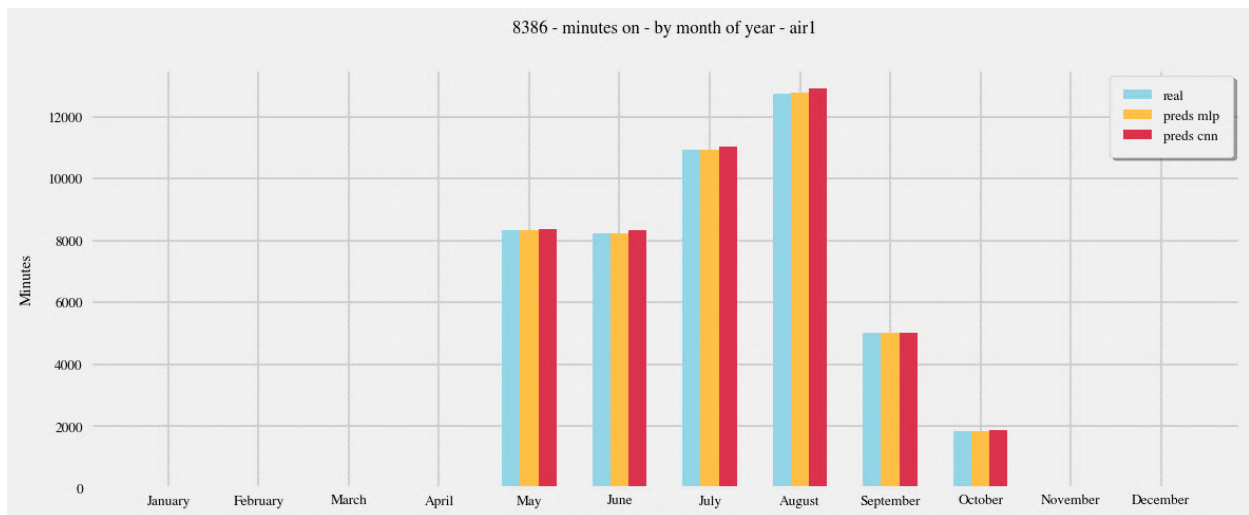
Anexo D – Figura 7 – Resultados CNN con transfer Learning sobre otro hogar – kitchenapp1 – train



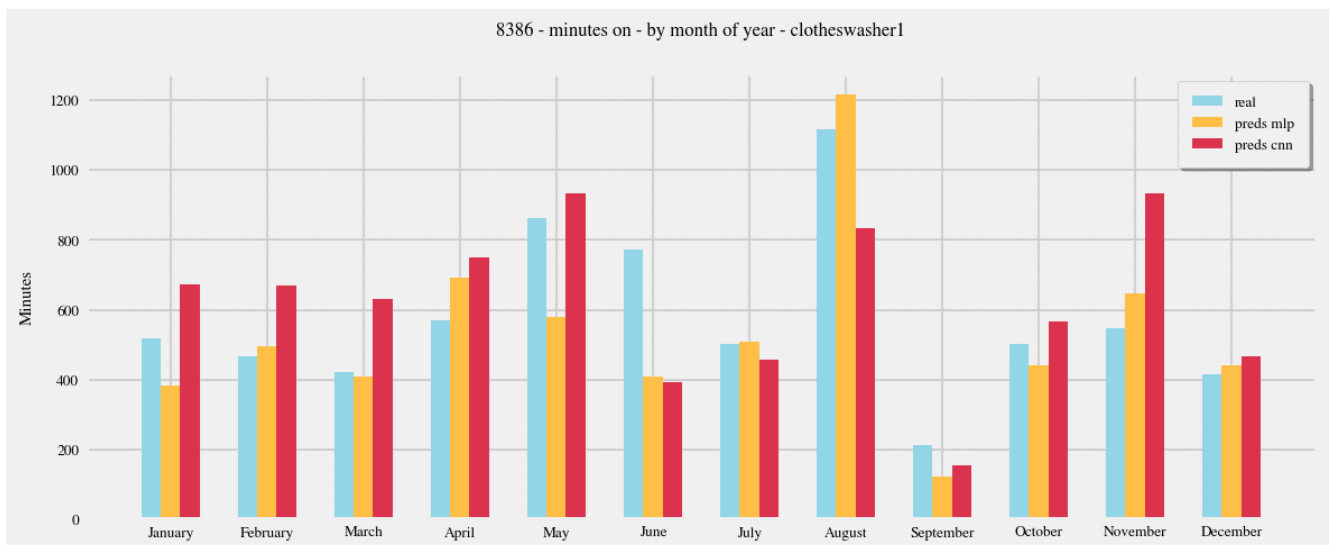
Anexo D – Figura 8 – Resultados CNN con transfer Learning sobre otro hogar – kitchenapp1 – test

ANEXO E – COMPARATIVA ENTRE MODELOS

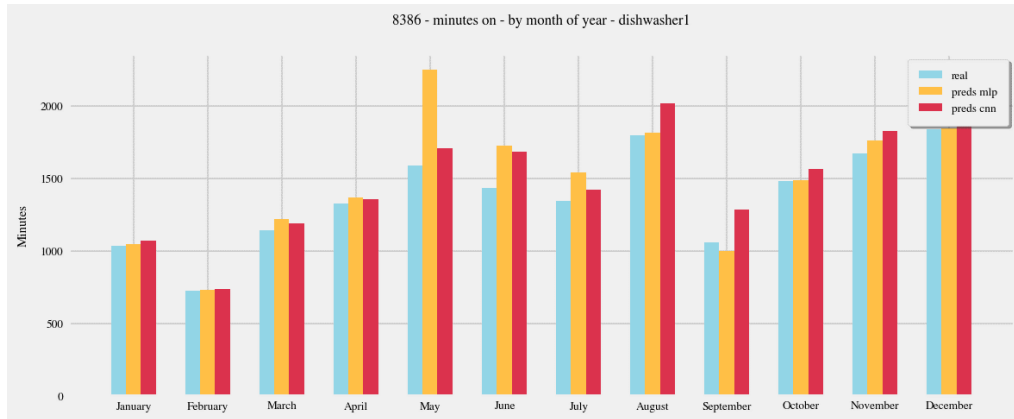
En este anexo se presentan las distintas gráficas que se explican en el apartado 4.3 para todos los electrodomésticos y todos los posibles casos de uso de agrupación (por hora del día, día de la semana y mes del año).



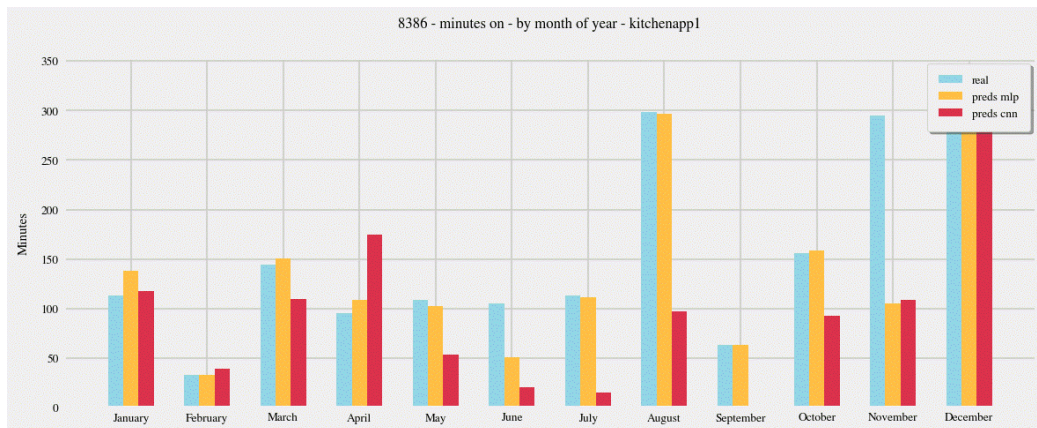
Anexo E – Figura 1 – Cantidad de minutos 'on' por mes del año – air1



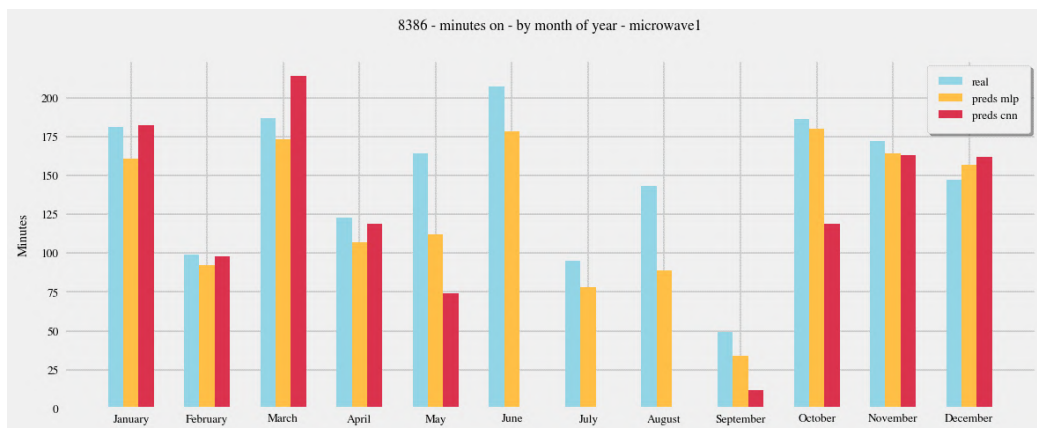
Anexo E – Figura 2 – Cantidad de minutos 'on' por mes del año – clotheswasher1



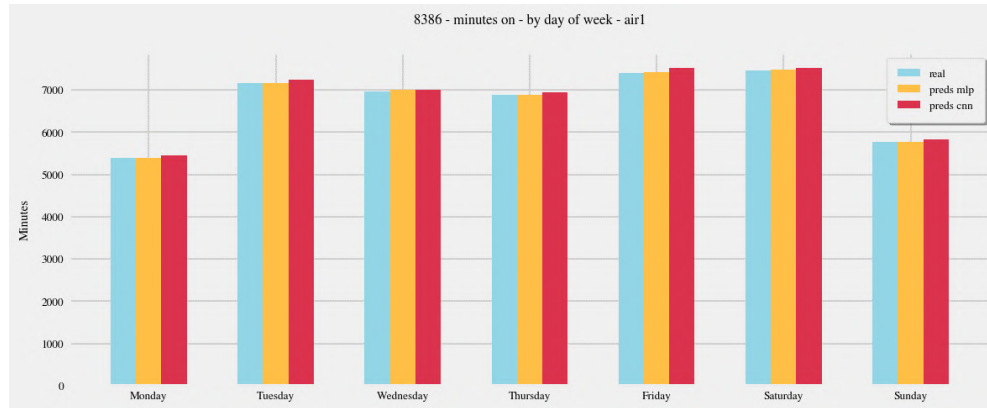
Anexo E – Figura 3 – Cantidad de minutos ‘on’ por mes del año – dishwasher1



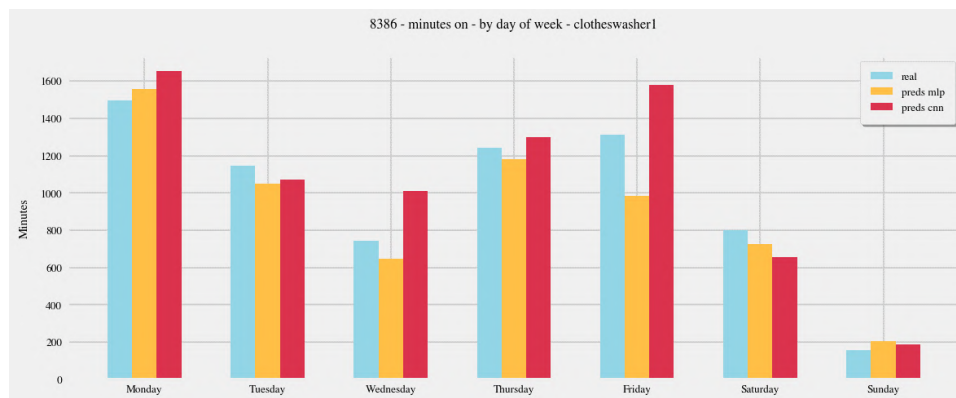
Anexo E – Figura 4 – Cantidad de minutos ‘on’ por mes del año – kitchenapp1



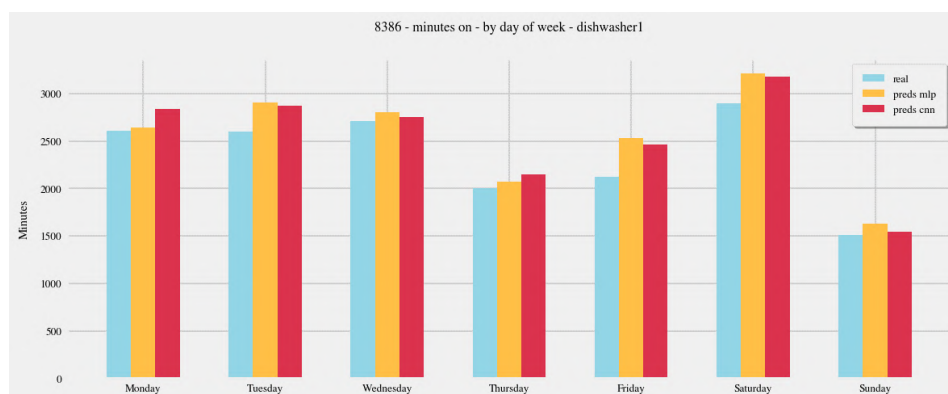
Anexo E – Figura 5 – Cantidad de minutos ‘on’ por mes del año – microwave1



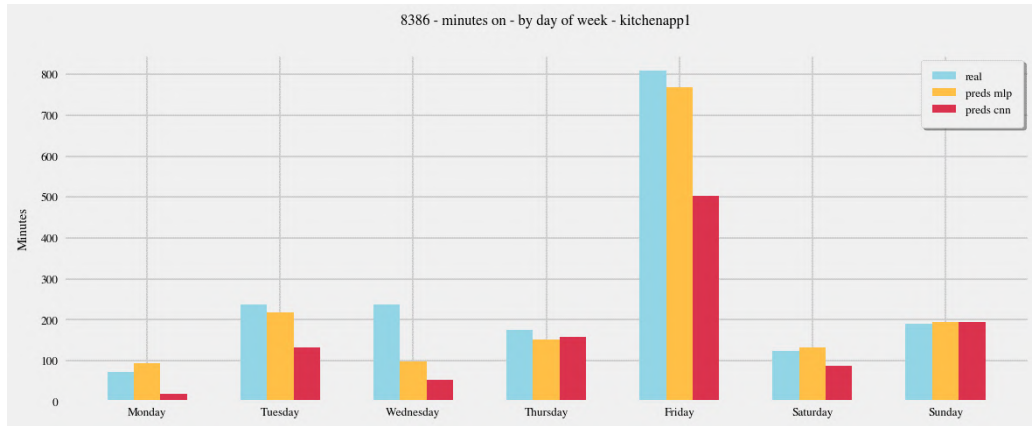
Anexo E – Figura 6 – Cantidad de minutos 'on' por día de la semana– *air1*



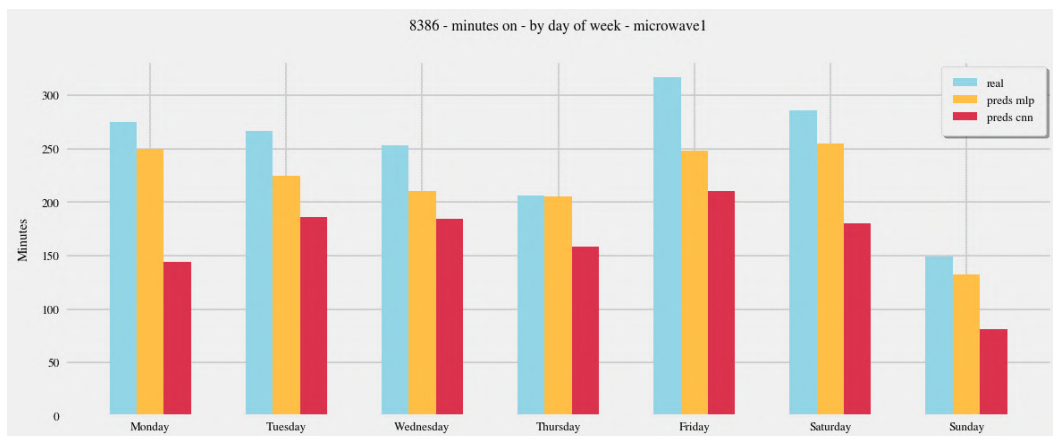
Anexo E – Figura 7 – Cantidad de minutos 'on' por día de la semana– *clotheswasher1*



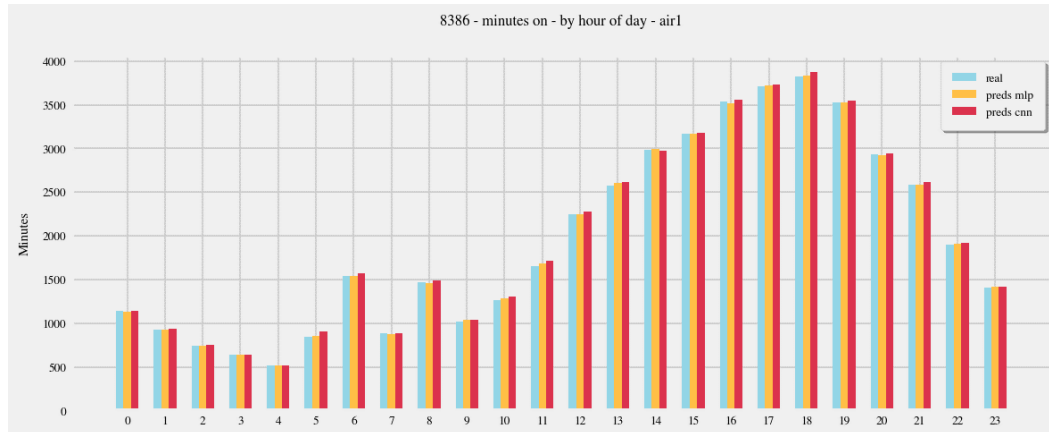
Anexo E – Figura 8 – Cantidad de minutos 'on' por día de la semana– *dishwasher1*



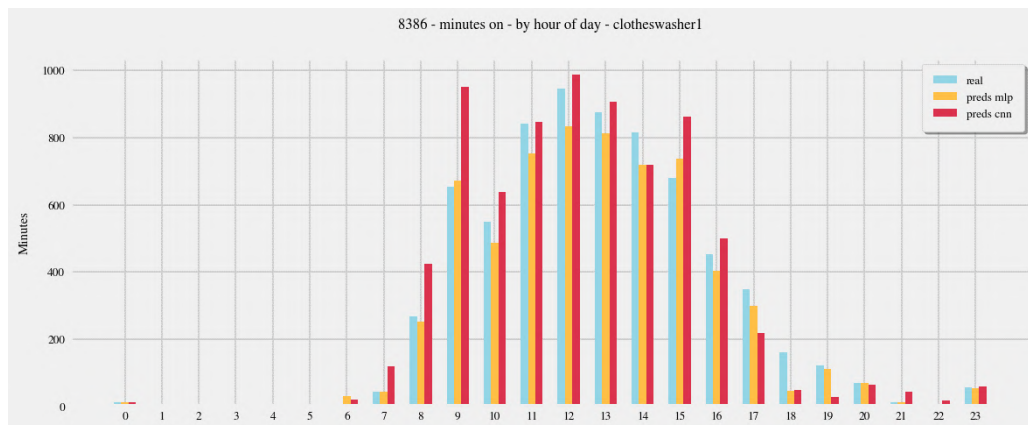
Anexo E – Figura 9 – Cantidad de minutos 'on' día de la semana – kitchenapp1



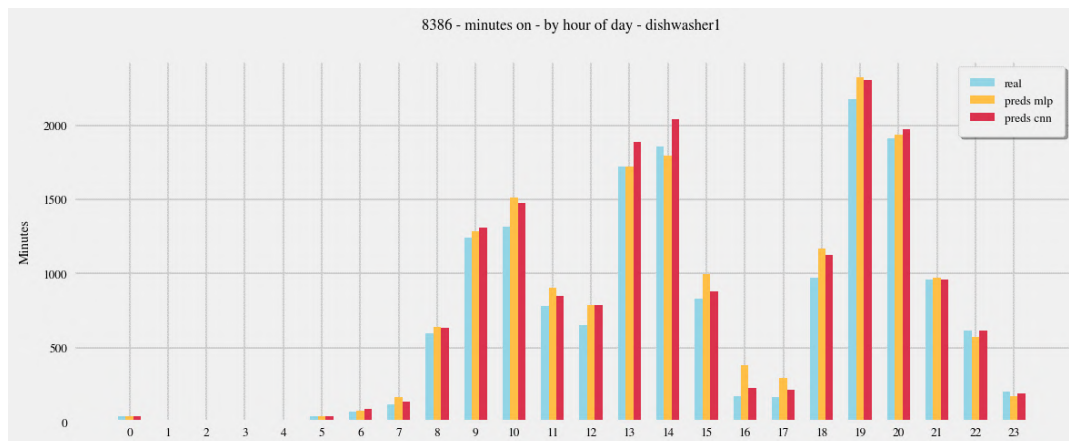
Anexo E – Figura 10 – Cantidad de minutos 'on' por día de la semana – microwave1



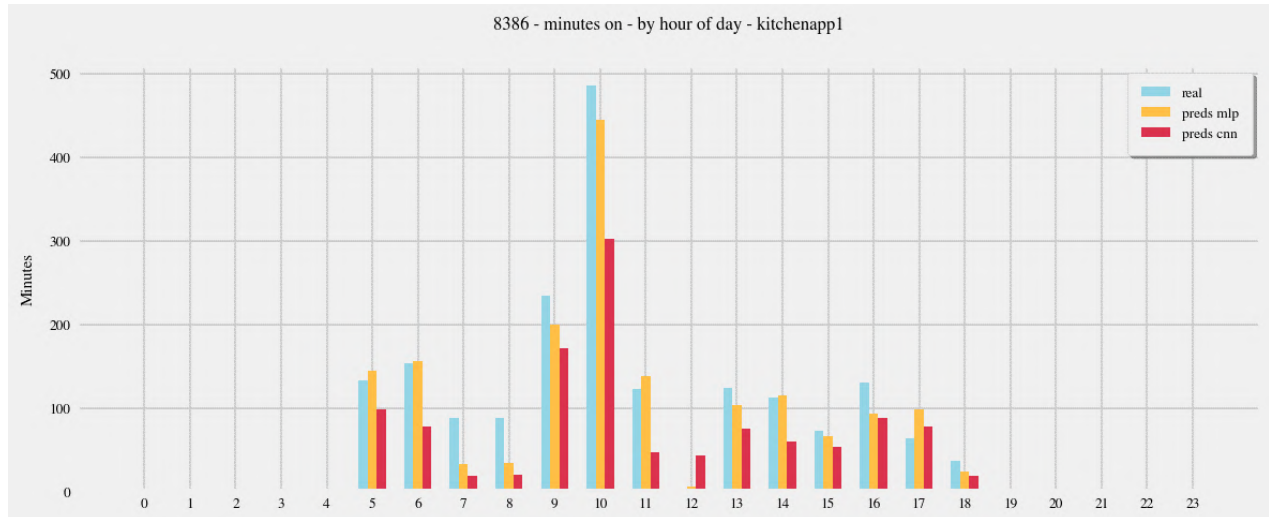
Anexo E – Figura 11 – Cantidad de minutos ‘on’ por hora del día – air1



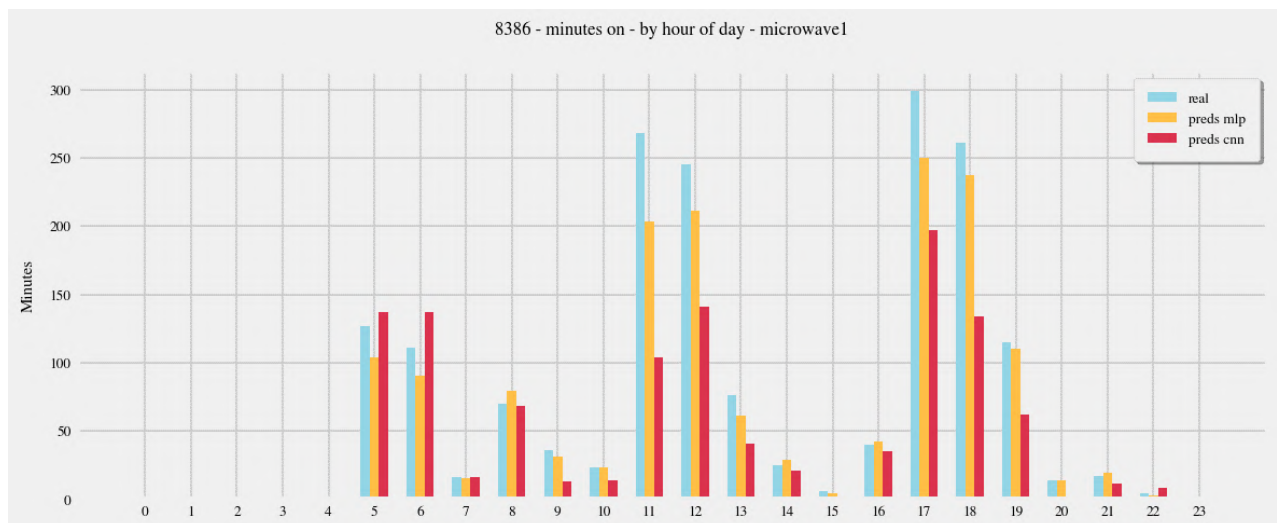
Anexo E – Figura 12 – Cantidad de minutos ‘on’ por hora del día – clotheswasher1



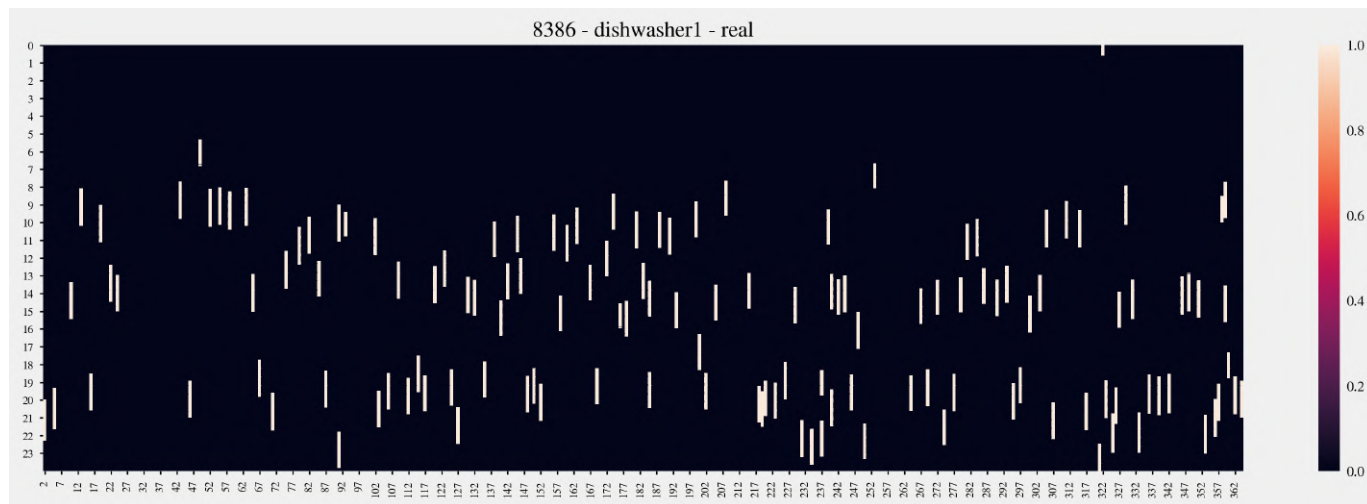
Anexo E – Figura 13 – Cantidad de minutos ‘on’ por hora del día – dishwasher1



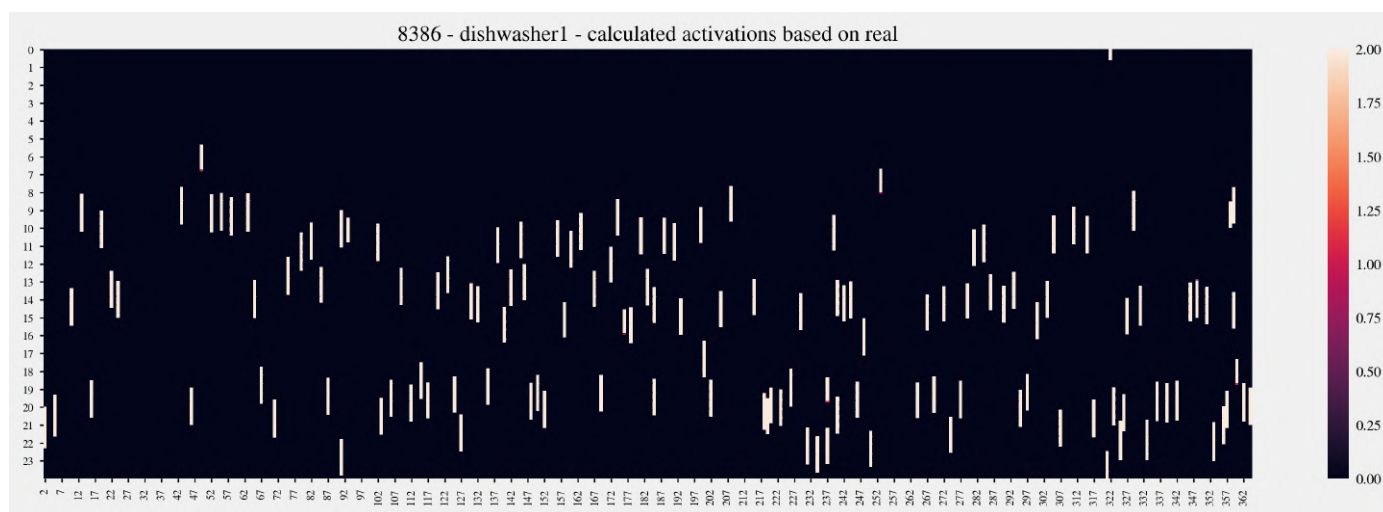
Anexo E – Figura 14 – Cantidad de minutos ‘on’ por hora del día – kitchenapp1



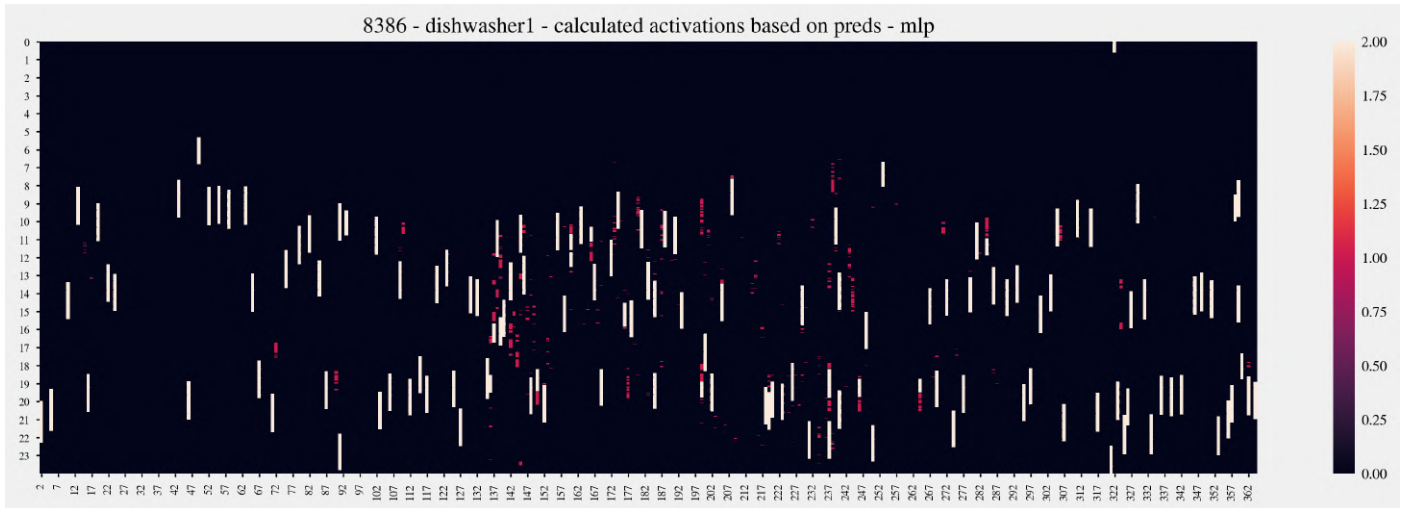
Anexo E – Figura 15 – Cantidad de minutos ‘on’ por hora del día – microwave1



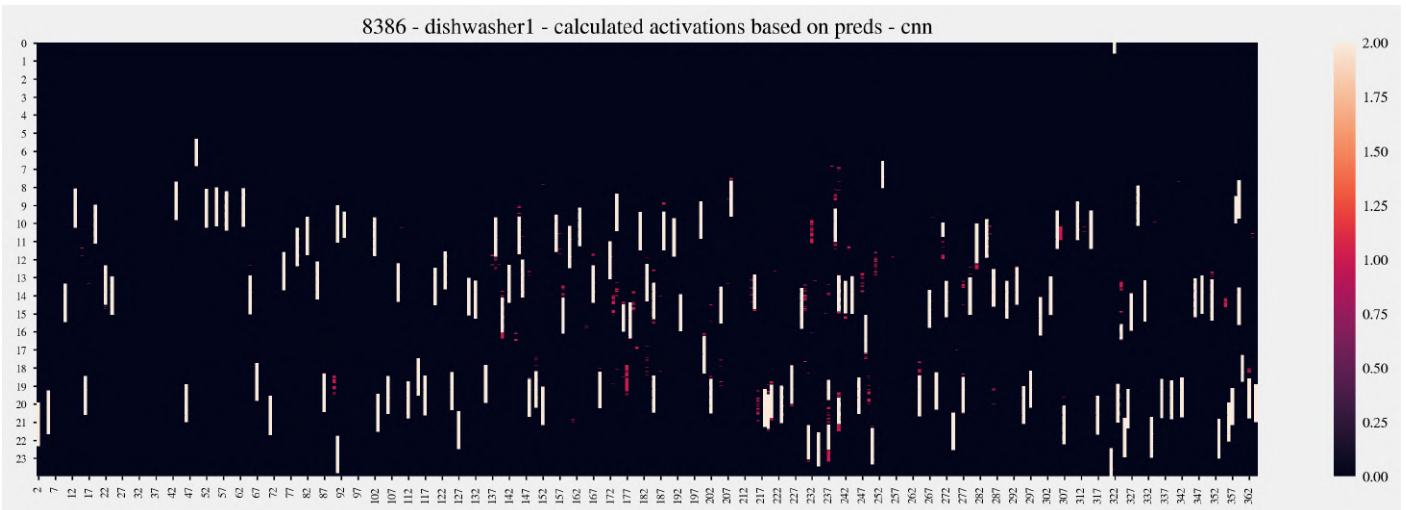
Anexo E – Figura 16 – Representación visual de la salida Y – dishwasher1



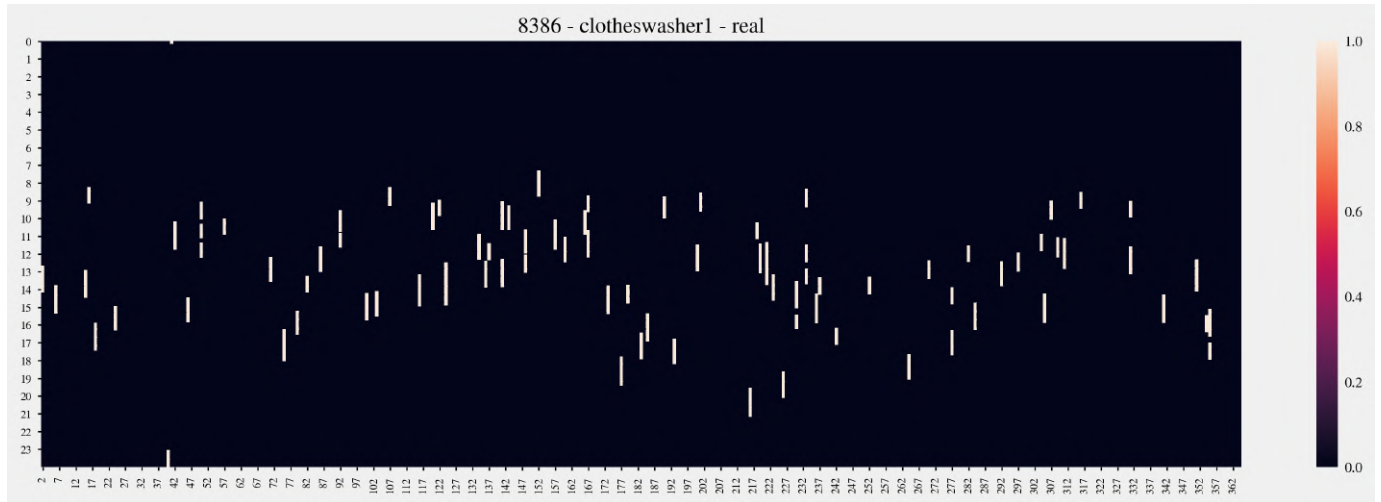
Anexo E – Figura 17 – Representación visual de detección de activaciones basada en datos reales – dishwasher1



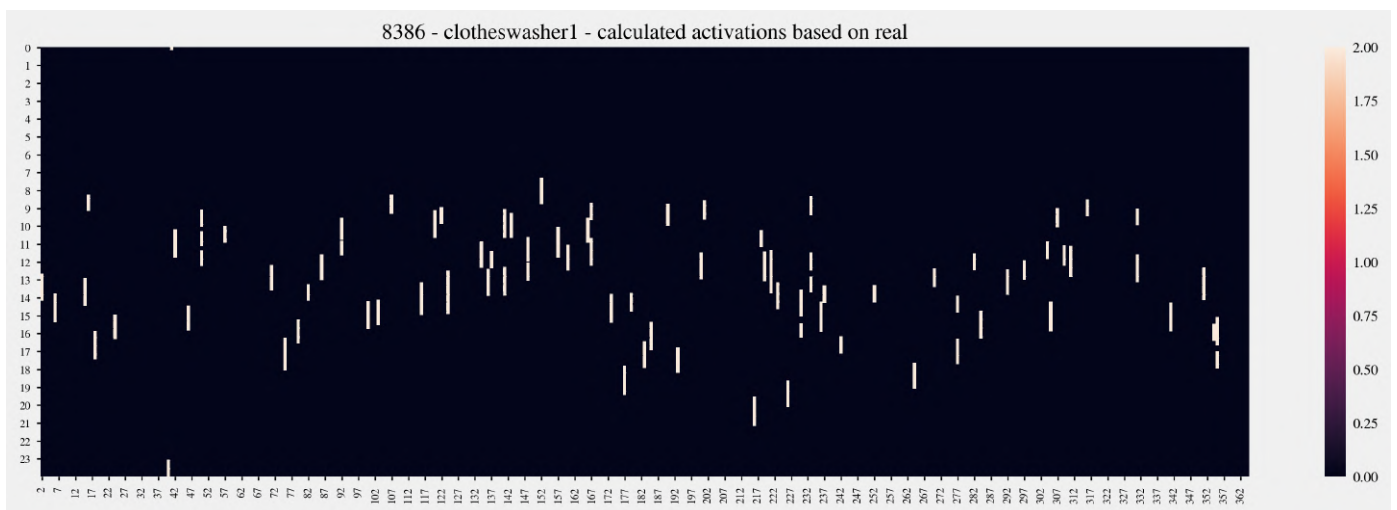
Anexo E – Figura 18 – Representación visual de detección de activaciones basada en predicciones – MLP – dishwasher1



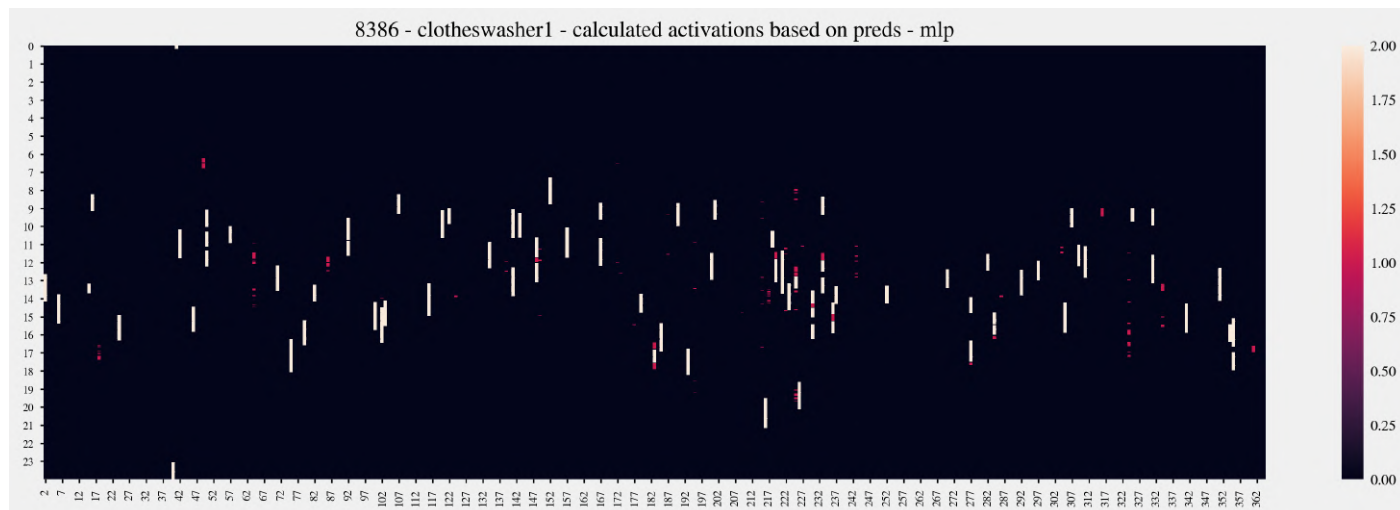
Anexo E – Figura 19 – Representación visual de detección de activaciones basada en predicciones – CNN – dishwasher1



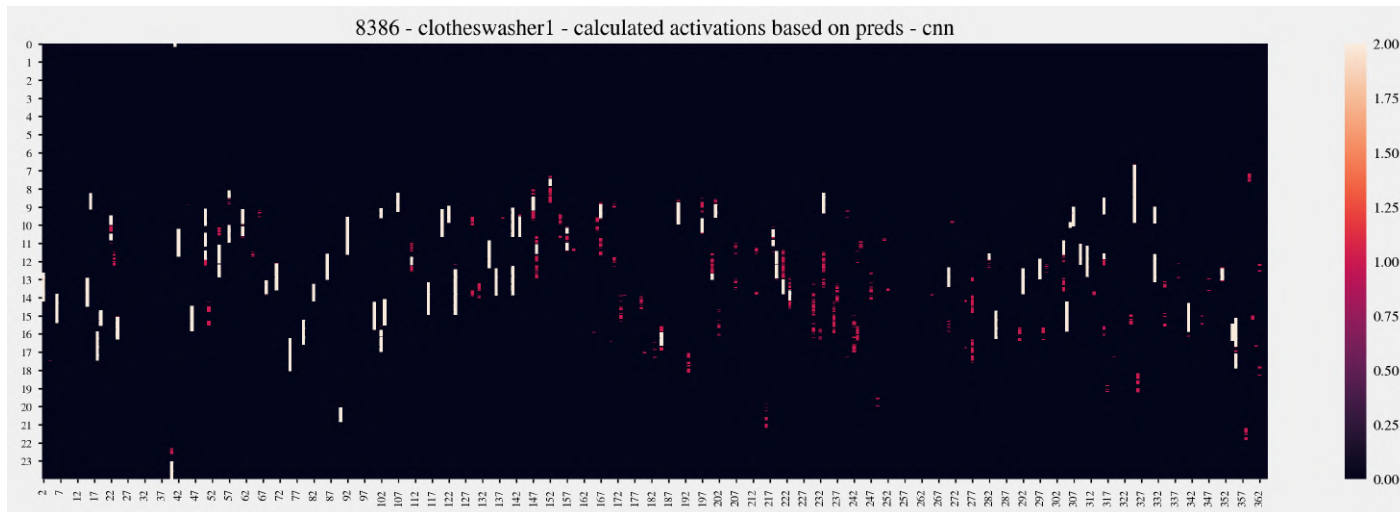
Anexo E – Figura 20 – Representación visual de la salida Y – clotheswasher1



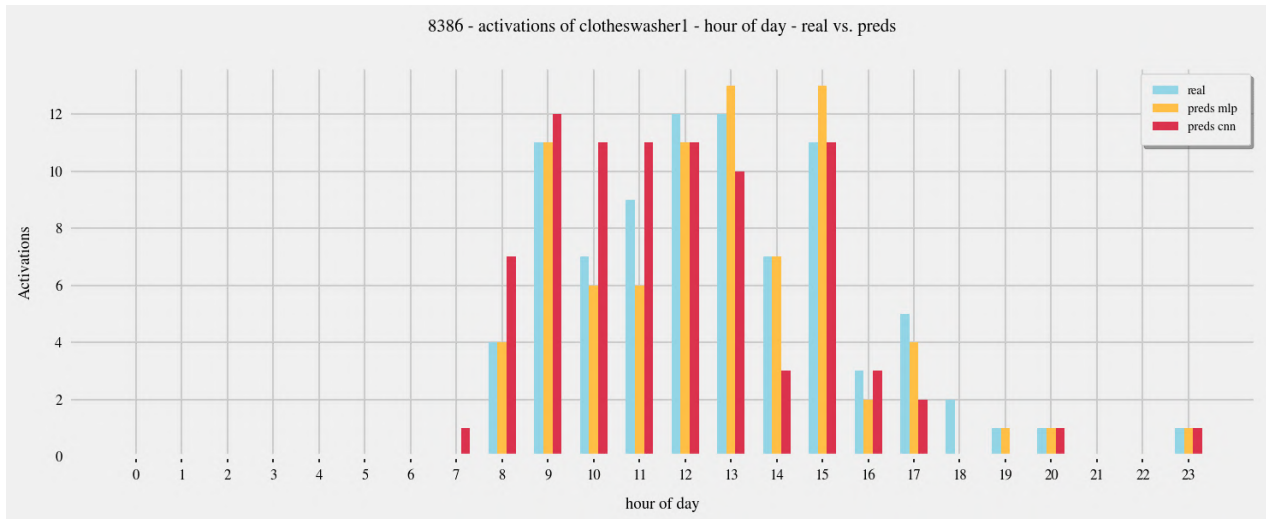
Anexo E – Figura 21 – Representación visual de detección de activaciones basada en datos reales – clotheswasher1



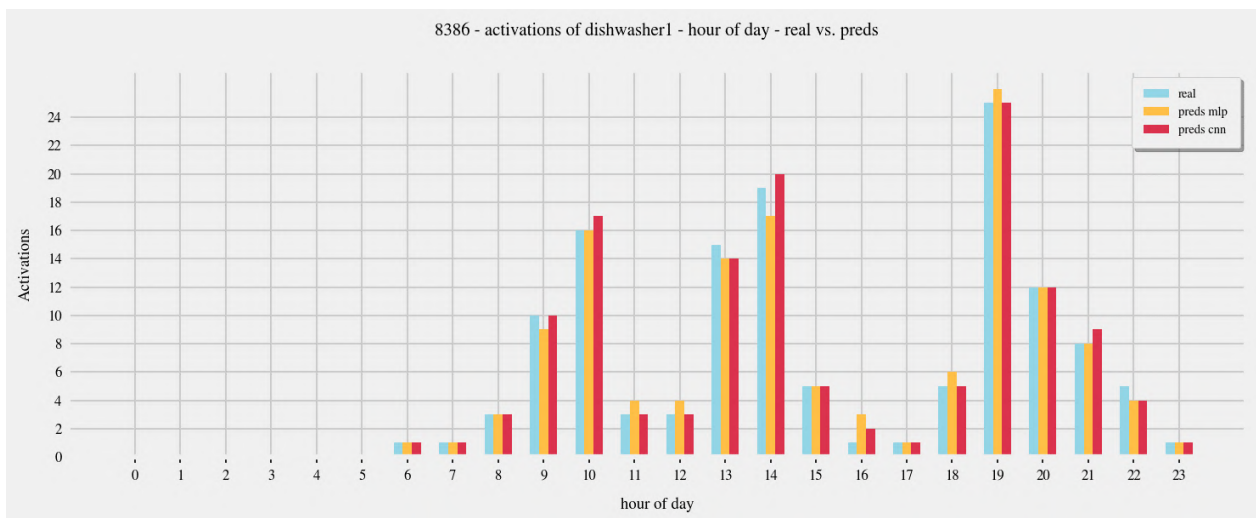
Anexo E – Figura 22 – Representación visual de detección de activaciones basada en predicciones – MLP – `c clotheswasher1`



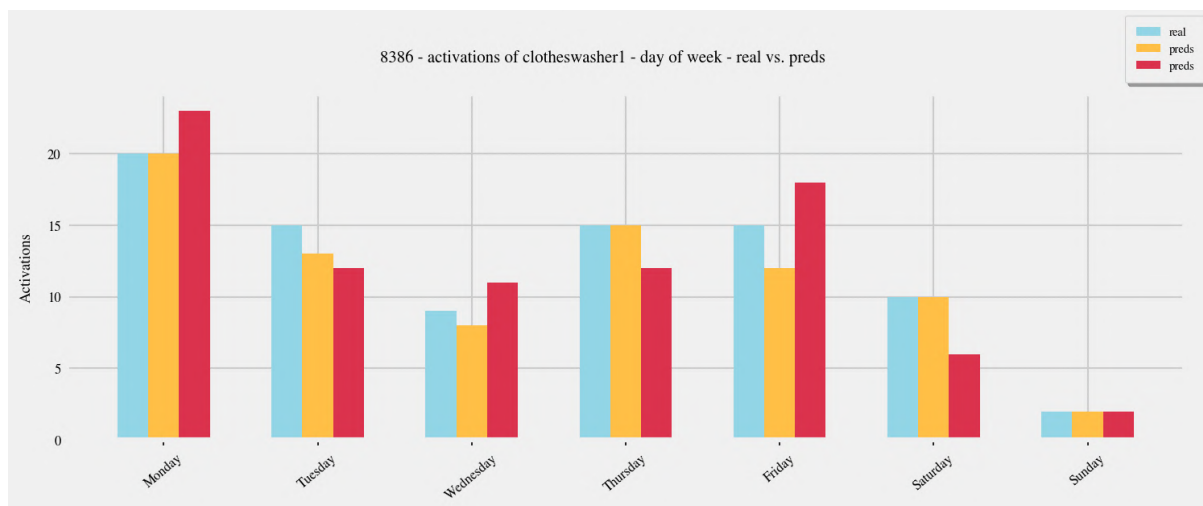
Anexo E – Figura 23 – Representación visual de detección de activaciones basada en predicciones – CNN – `c clotheswasher1`



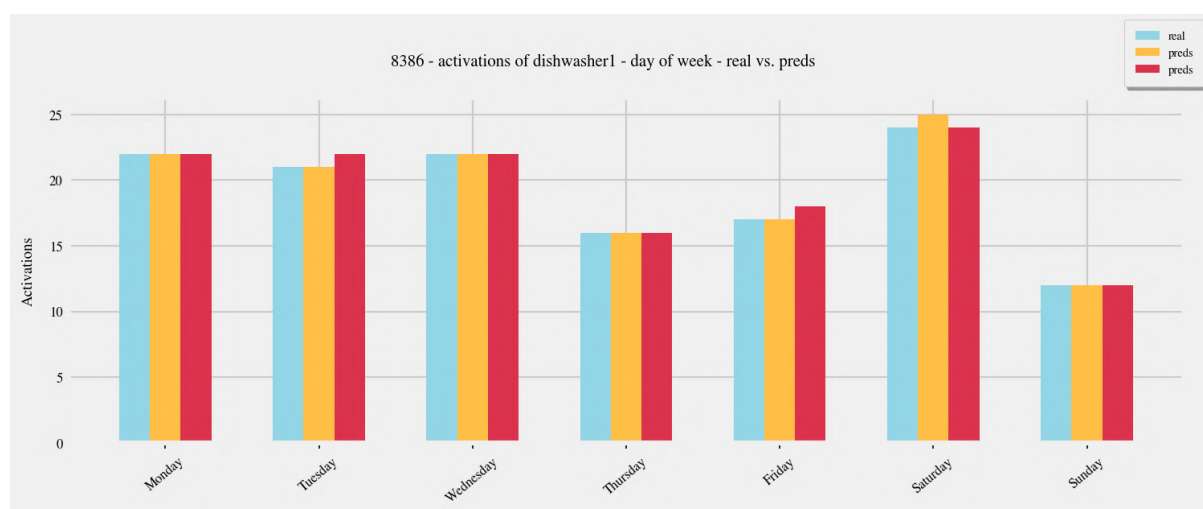
Anexo E – Figura 24 – Número de activaciones por hora del día – clotheswasher1



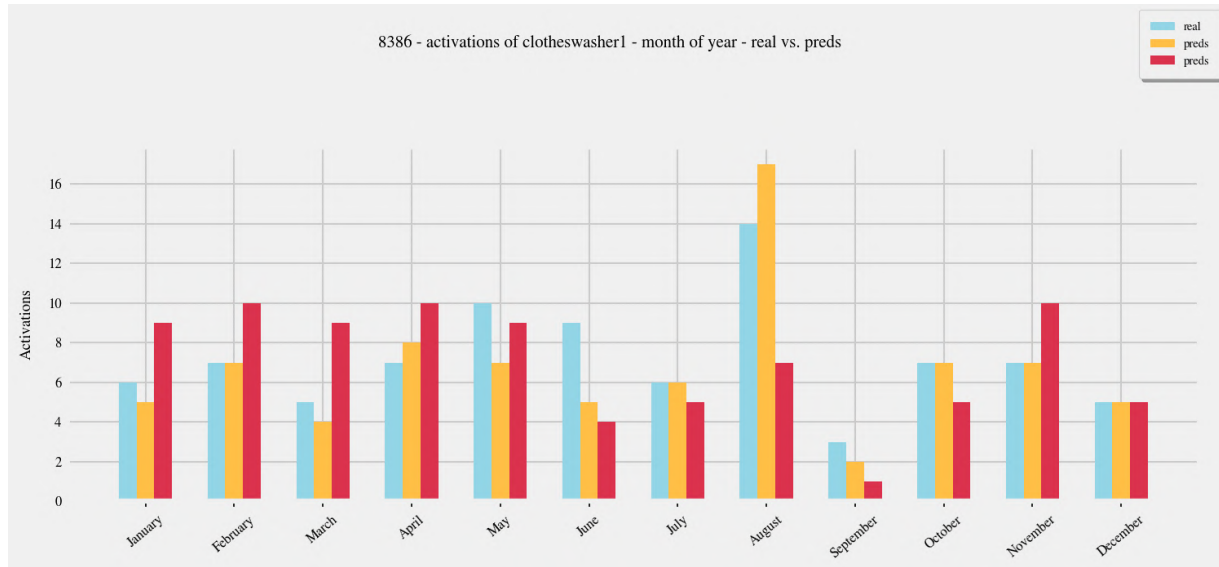
Anexo E – Figura 25 – Número de activaciones por mes del año – dishwasher1



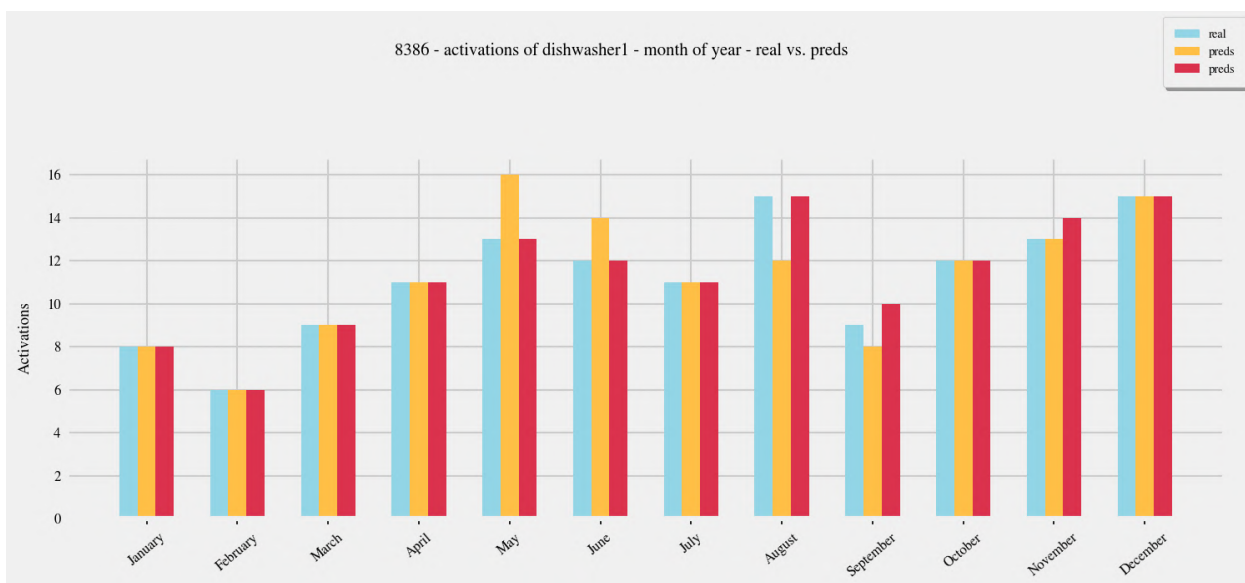
Anexo E – Figura 26 – Número de activaciones por día de la semana – clotheswasher1



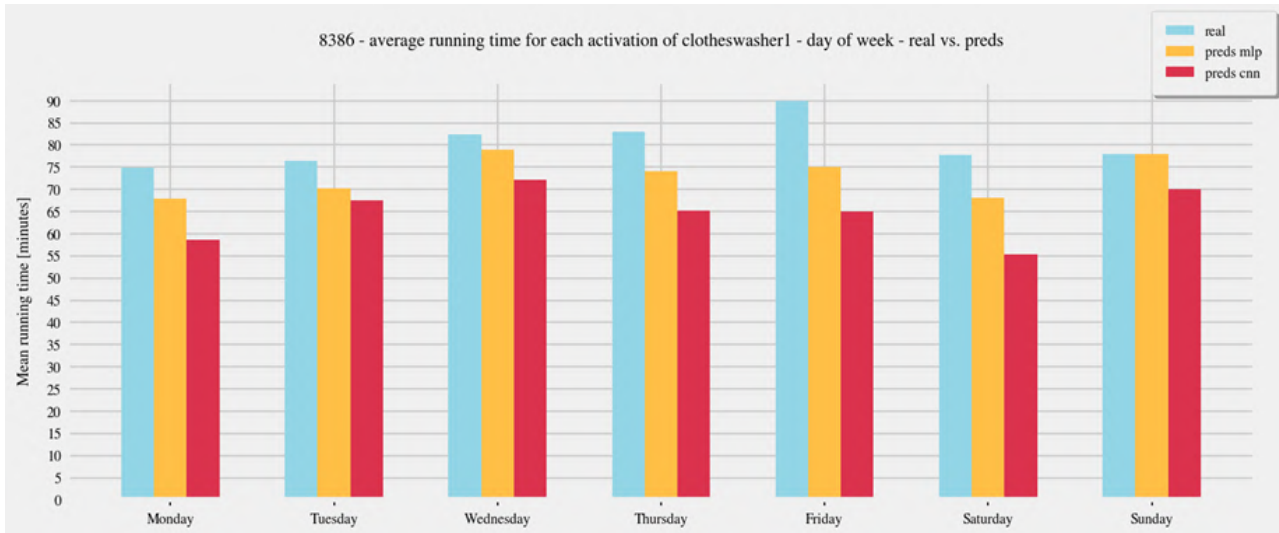
Anexo E – Figura 27 – Número de activaciones por día de la semana – dishwasher1



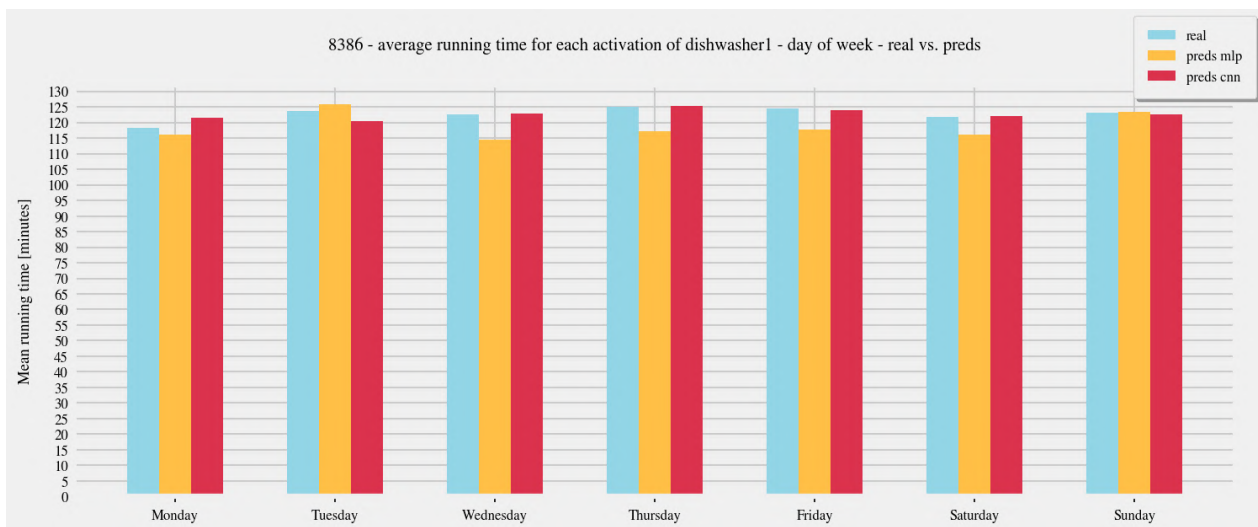
Anexo E – Figura 28 – Número de activaciones por día de la semana – clotheswasher



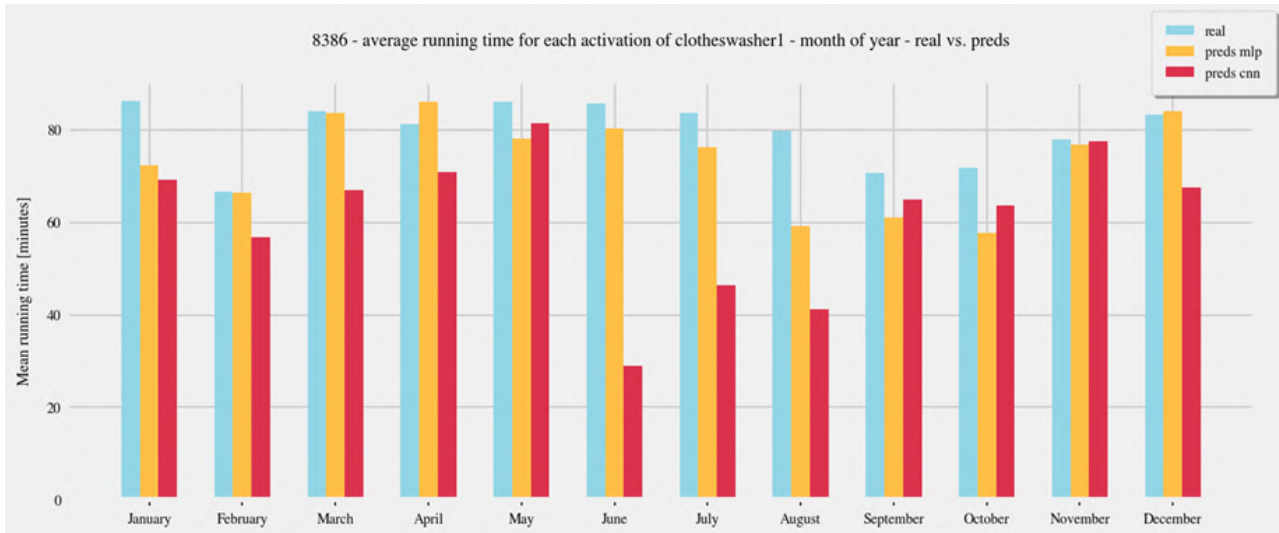
Anexo E - Figura 29 – Número de activaciones por día de la semana – dishwasher1



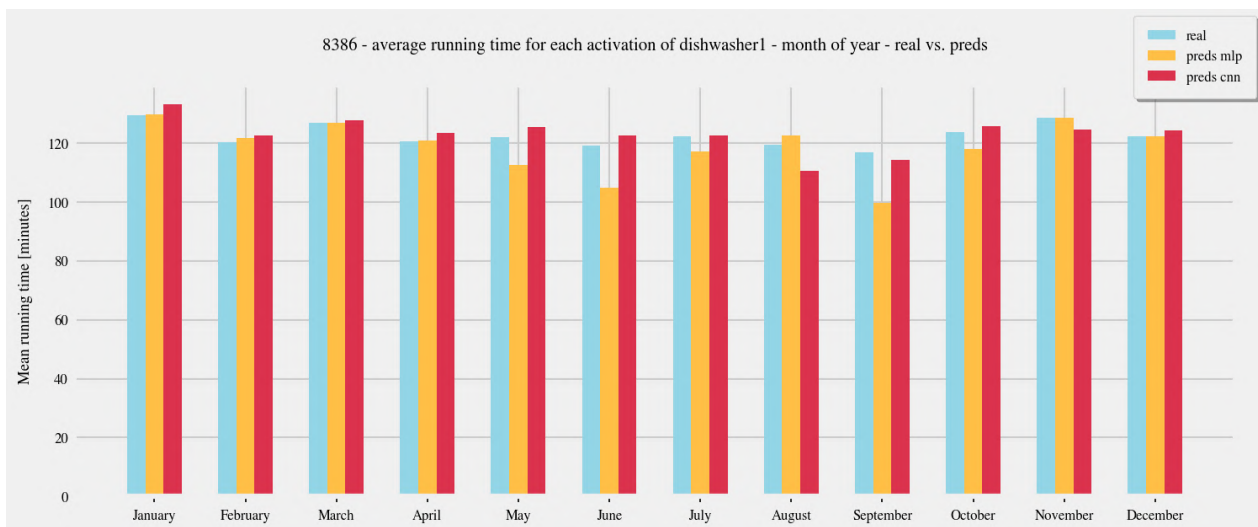
Anexo E – Figura 30 – Tiempo medio de activación por día de la semana – clotheswasher1



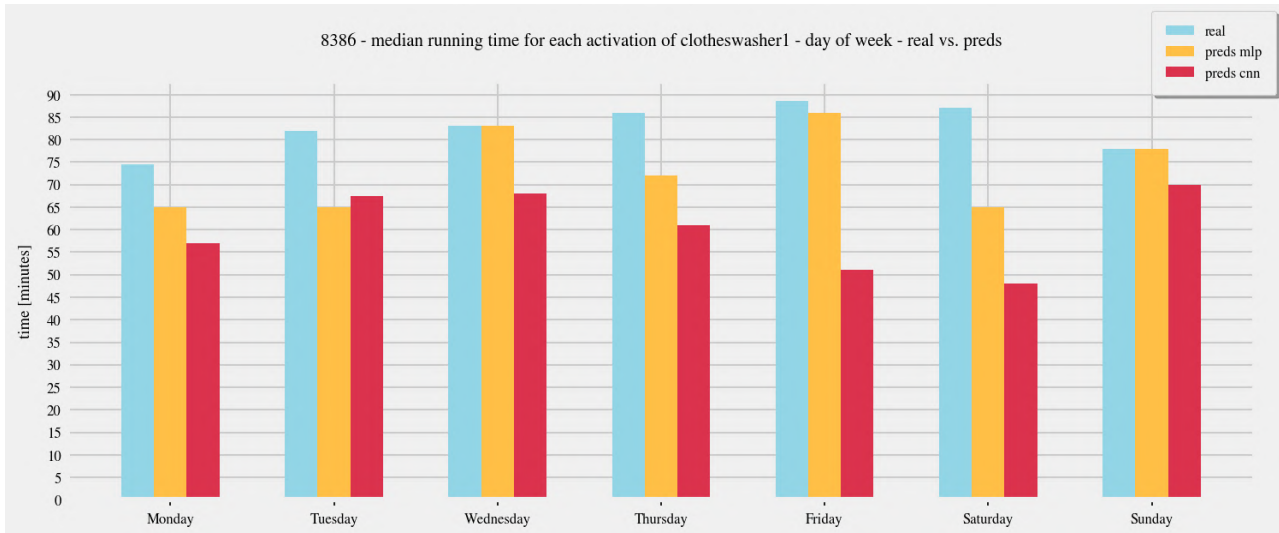
Anexo E – Figura 31 – Tiempo medio de activación por día de la semana – dishwasher1



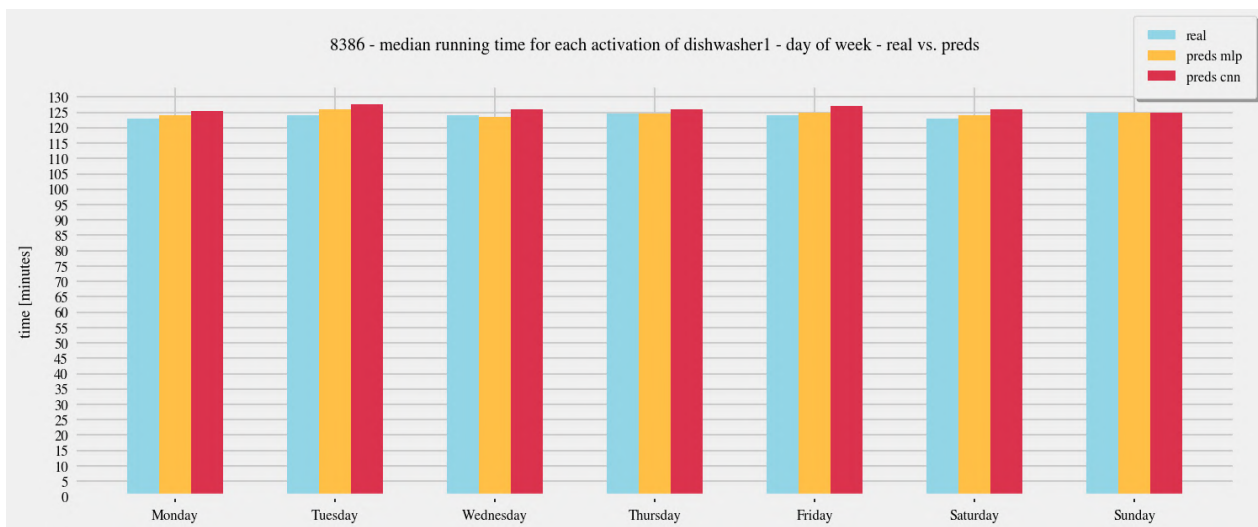
Anexo E – Figura 32 – Tiempo medio de activación por mes del año – clotheswasher1



Anexo E – Figura 33 – Tiempo medio de activación por mes del año – dishwasher1



Anexo E – Figura 34 – Tiempo mediano de activación por día de la semana – clotheswasher1

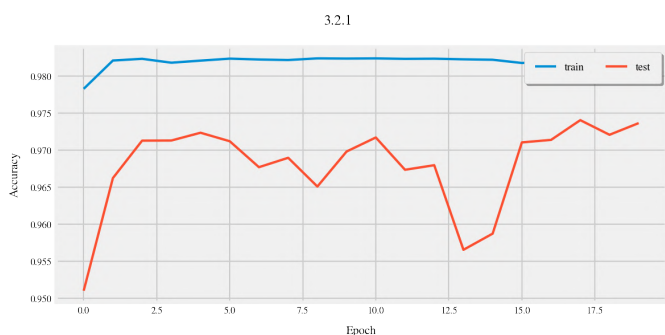


Anexo E – Figura 35 – Tiempo mediano de activación por día de la semana – dishwasher1

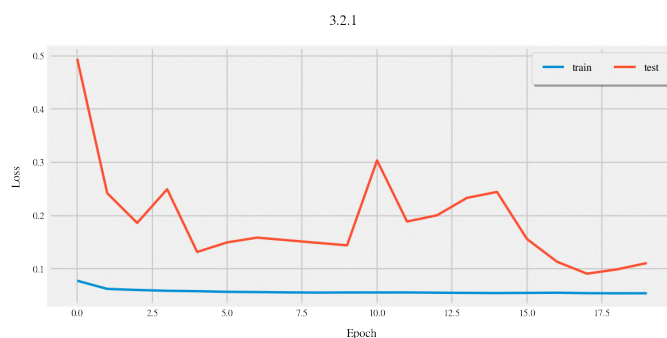
ANEXO F – ITERACIÓN DE MODELOS CNN

En este anexo se recogen los distintos modelos CNN explorados hasta dar con el mejor resultado. Los números identificativos no tienen otro valor que el de temporalidad, siendo el 3.2.1 el primer modelo y el 3.2.22 el último.

6.1 CNN - 3.2.1



Anexo F – Figura 1 – Accuracy por epoch – modelo 3.2.1



Anexo F – Figura 2 – Loss por epoch – modelo 3.2.1

```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

model = Sequential()
model.add(layers.Conv2D(8, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((4,4)))
model.add(layers.Dropout(0.5))
model.add(layers.Flatten())
model.add(Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
model.add(Dense(5, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()
executed in 370ms. finished 11:33:19 2021-01-24
Model: "sequential_10"
```

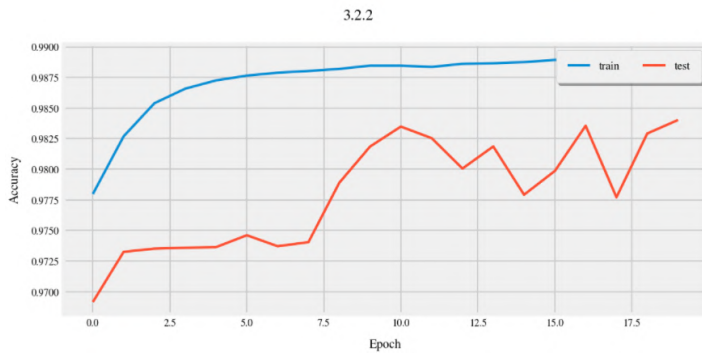
Layer (type)	Output Shape	Param #
conv2d_30 (Conv2D)	(None, 144, 13, 8)	80
max_pooling2d_22 (MaxPooling)	(None, 36, 3, 8)	0
dropout_10 (Dropout)	(None, 36, 3, 8)	0
flatten_10 (Flatten)	(None, 864)	0
dense_20 (Dense)	(None, 40)	34600
dense_21 (Dense)	(None, 5)	205
Total params: 34,885		
Trainable params: 34,885		
Non-trainable params: 0		

Anexo F – Figura 3 – Arquitectura – modelo 3.2.1

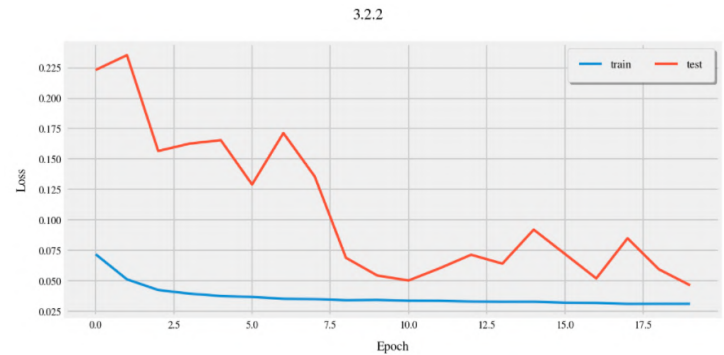
Train accuracy: 0.9821

val accuracy: 0.9737

6.2 CNN - 3.2.2



Anexo F – Figura 4 – Accuracy por epoch – modelo 3.2.2



Anexo F – Figura 5 – Loss por epoch – modelo 3.2.2

```
[26]: callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

model = Sequential()
model.add(layers.Conv2D(8, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((4,4)))
model.add(layers.Conv2D(8, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
model.add(layers.Dense(5, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()

executed in 231ms, finished 18:40:37 2021-01-24

Model: "sequential"

```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 144, 13, 8)	80
max_pooling2d (MaxPooling2D)	(None, 36, 3, 8)	0
conv2d_1 (Conv2D)	(None, 36, 3, 8)	584
max_pooling2d_1 (MaxPooling2D)	(None, 18, 1, 8)	0
flatten (Flatten)	(None, 144)	0
dense (Dense)	(None, 40)	5800
dense_1 (Dense)	(None, 5)	205

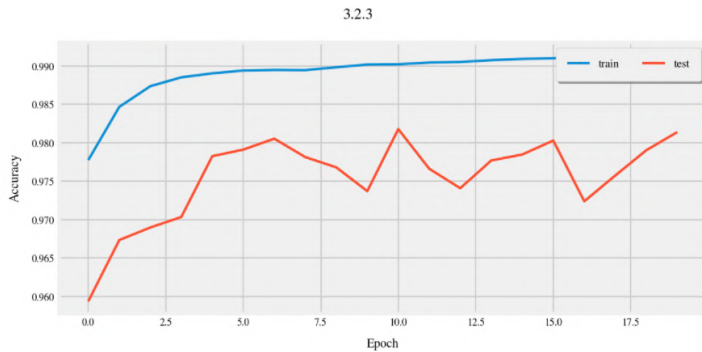
Total params: 6,669
 Trainable params: 6,669
 Non-trainable params: 0

Anexo F – Figura 6 – Arquitectura – modelo 3.2.2

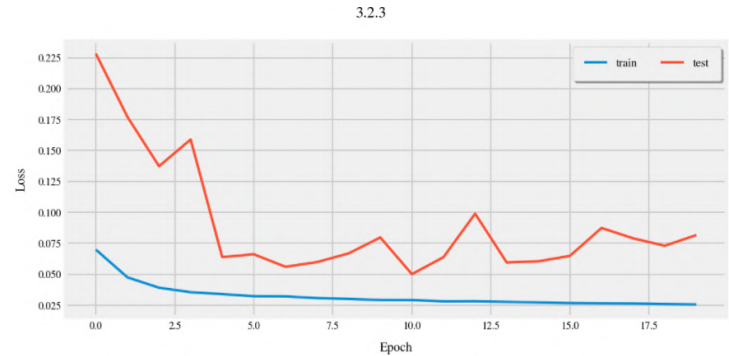
Train accuracy: 0.9892

Val accuracy: 0.9840

6.3 CNN - 3.2.3



Anexo F – Figura 7 – Accuracy por epoch – modelo 3.2.3



Anexo F – Figura 8 – Loss por epoch – modelo 3.2.3

```

callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

model = Sequential()
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((4,4)))
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(8, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.Flatten())
model.add(Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
model.add(Dense(5, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()

```

executed in 231ms, finished 22:27:47 2021-01-24

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 144, 13, 16)	160
max_pooling2d_2 (MaxPooling2D)	(None, 36, 3, 16)	0
conv2d_3 (Conv2D)	(None, 36, 3, 16)	2320
max_pooling2d_3 (MaxPooling2D)	(None, 18, 1, 16)	0
conv2d_4 (Conv2D)	(None, 18, 1, 8)	1160
flatten_1 (Flatten)	(None, 144)	0
dense_2 (Dense)	(None, 40)	5800
dense_3 (Dense)	(None, 5)	205

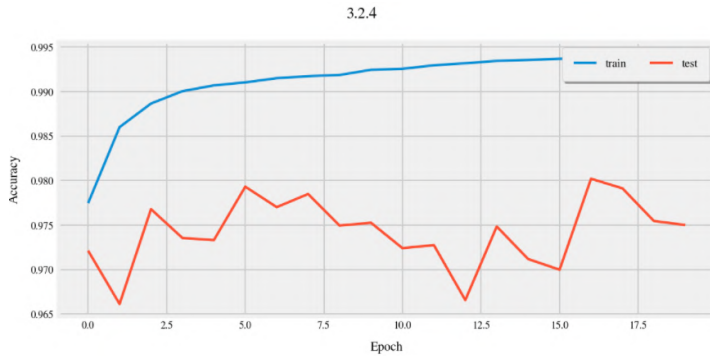
Total params: 9,645
 Trainable params: 9,645
 Non-trainable params: 0

Anexo F – Figura 9 – Arquitectura – modelo 3.2.3

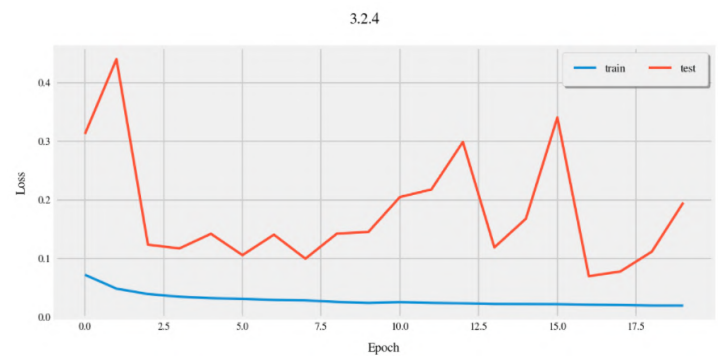
Train accuracy: 0.9915

Val accuracy: 0.9814

6.4 CNN - 3.2.4



Anexo F – Figura 10 – Accuracy por epoch – modelo 3.2.4



Anexo F – Figura 11 – Loss por epoch – modelo 3.2.4

```

: callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

model = Sequential()
model.add(layers.Conv2D(8, (5, 5), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((3,3)))
model.add(layers.Conv2D(64, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.Flatten())
model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
model.add(layers.Dense(5, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()

```

executed in 282ms, finished 23:35:38 2021-01-24

Model: "sequential_2"

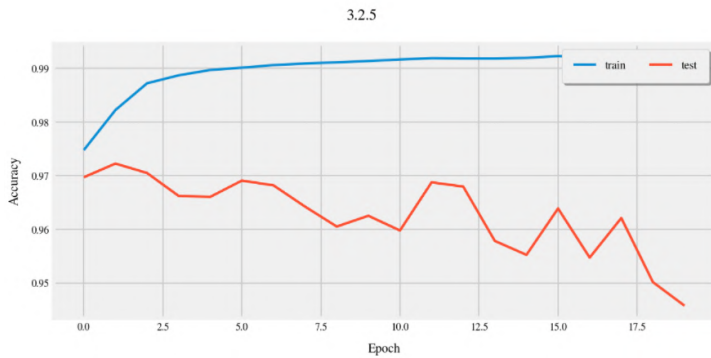
Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 144, 13, 8)	208
max_pooling2d_4 (MaxPooling2D)	(None, 72, 6, 8)	0
conv2d_6 (Conv2D)	(None, 72, 6, 16)	1168
max_pooling2d_5 (MaxPooling2D)	(None, 36, 3, 16)	0
conv2d_7 (Conv2D)	(None, 36, 3, 16)	2320
max_pooling2d_6 (MaxPooling2D)	(None, 12, 1, 16)	0
conv2d_8 (Conv2D)	(None, 12, 1, 64)	9280
flatten_2 (Flatten)	(None, 768)	0
dense_4 (Dense)	(None, 40)	30760
dense_5 (Dense)	(None, 5)	205
Total params:		43,941
Trainable params:		43,941
Non-trainable params:		0

Anexo F – Figura 12 – Arquitectura – modelo 3.2.4

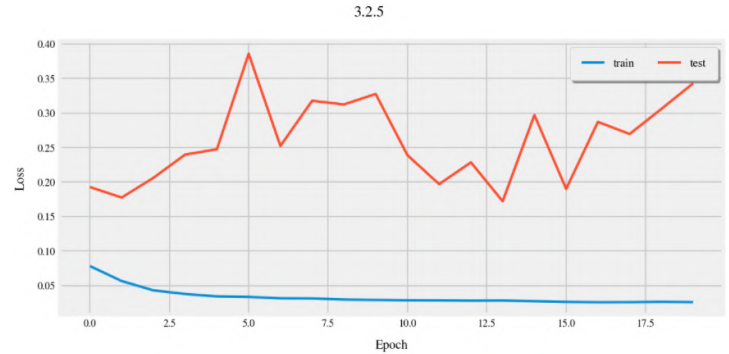
Train accuracy: 0.9942

Val accuracy: 0.9750

6.5 CNN - 3.2.5



Anexo F – Figura 13 – Accuracy por epoch – modelo 3.2.5



Anexo F – Figura 14 – Loss por epoch – modelo 3.2.5

```

]: callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

model = Sequential()
model.add(layers.Conv2D(8, (5, 5), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((3,3)))
model.add(layers.Conv2D(64, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
model.add(layers.Dense(5, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()
  
```

executed in 292ms. finished 01:18:14 2021-01-25

Model: "sequential_3"

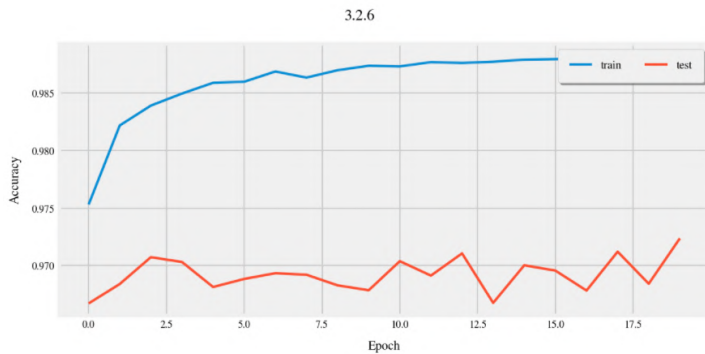
Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 144, 13, 8)	208
max_pooling2d_7 (MaxPooling2D)	(None, 72, 6, 8)	0
conv2d_10 (Conv2D)	(None, 72, 6, 16)	1168
max_pooling2d_8 (MaxPooling2D)	(None, 36, 3, 16)	0
conv2d_11 (Conv2D)	(None, 36, 3, 16)	2320
max_pooling2d_9 (MaxPooling2D)	(None, 12, 1, 16)	0
conv2d_12 (Conv2D)	(None, 12, 1, 64)	9280
flatten_3 (Flatten)	(None, 768)	0
dropout (Dropout)	(None, 768)	0
dense_6 (Dense)	(None, 40)	30760
dense_7 (Dense)	(None, 5)	205
Total params:		43,941
Trainable params:		43,941
Non-trainable params:		0

Anexo F – Figura 15 – Arquitectura – modelo 3.2.5

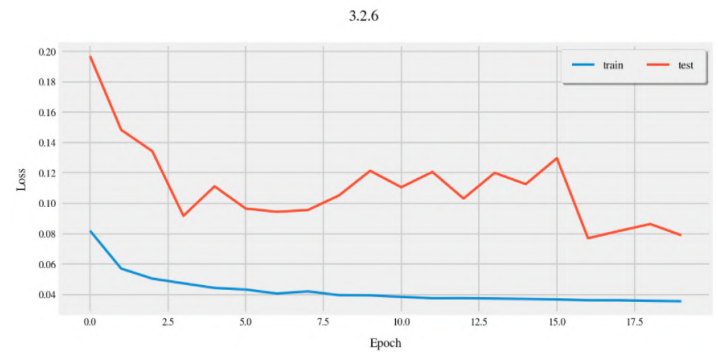
Train accuracy: 0.9923

val accuracy: 0.9457

6.6 CNN - 3.2.6



Anexo F – Figura 16 – Accuracy por epoch – modelo 3.2.6



Anexo F – Figura 17 – Loss por epoch – modelo 3.2.6

```

|: callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

model = Sequential()
model.add(layers.Conv2D(8, (5, 5), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Dropout(0.5))
model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.MaxPooling2D((3, 3)))
model.add(layers.Dropout(0.5))
model.add(layers.Conv2D(64, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
model.add(layers.Dense(5, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

model.summary()

```

executed in 307ms, finished 02:50:44 2021-01-25

Model: "sequential_4"

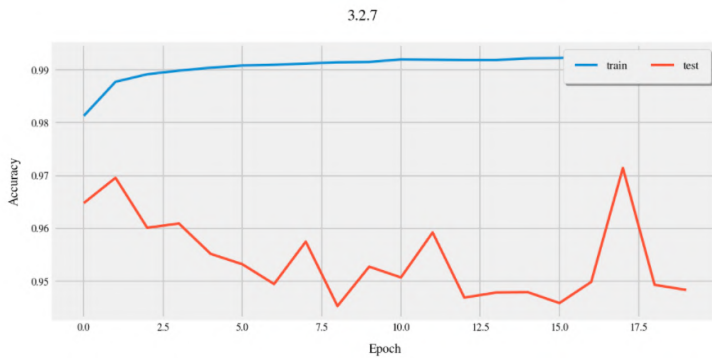
Layer (type)	Output Shape	Param #
=====		
conv2d_13 (Conv2D)	(None, 144, 13, 8)	208
max_pooling2d_10 (MaxPooling)	(None, 72, 6, 8)	0
conv2d_14 (Conv2D)	(None, 72, 6, 16)	1168
max_pooling2d_11 (MaxPooling)	(None, 36, 3, 16)	0
dropout_1 (Dropout)	(None, 36, 3, 16)	0
conv2d_15 (Conv2D)	(None, 36, 3, 16)	2320
max_pooling2d_12 (MaxPooling)	(None, 12, 1, 16)	0
dropout_2 (Dropout)	(None, 12, 1, 16)	0
conv2d_16 (Conv2D)	(None, 12, 1, 64)	9280
flatten_4 (Flatten)	(None, 768)	0
dropout_3 (Dropout)	(None, 768)	0
dense_8 (Dense)	(None, 40)	30760
dense_9 (Dense)	(None, 5)	205
=====		
Total params:	43,941	
Trainable params:	43,941	
Non-trainable params:	0	

Anexo F – Figura 18 – Arquitectura – modelo 3.2.6

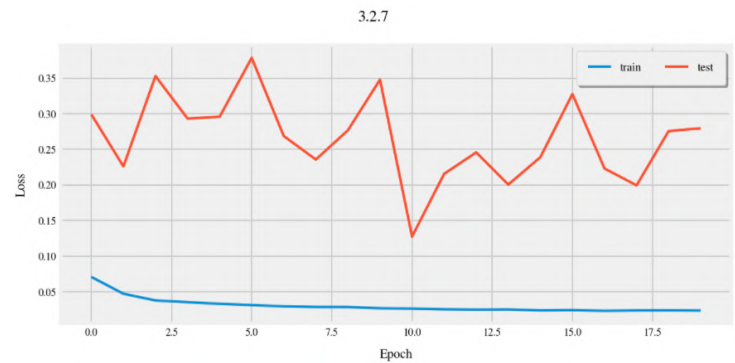
Train accuracy: 0.9882

Val accuracy: 0.9724

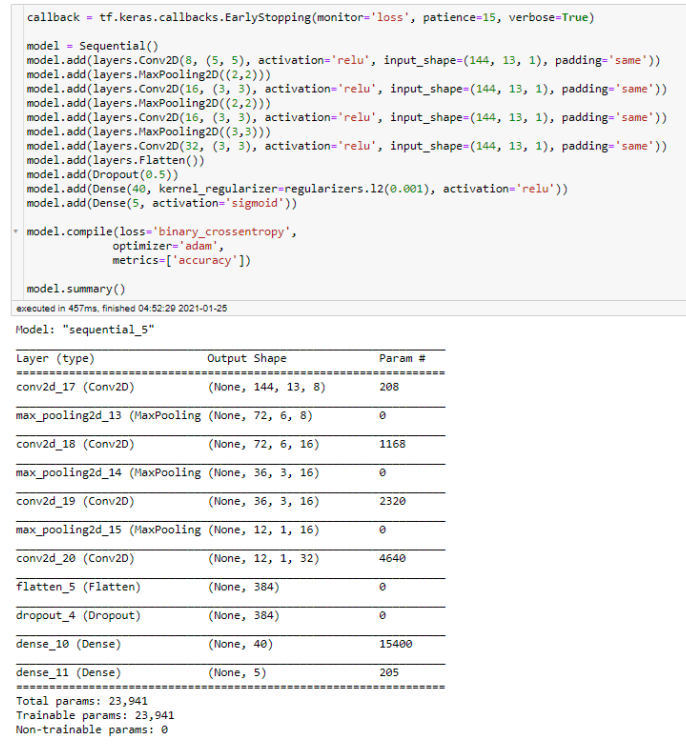
6.7 CNN - 3.2.7



Anexo F – Figura 19 – Accuracy por epoch – modelo 3.2.7



Anexo F – Figura 20 – Loss por epoch – modelo 3.2.7

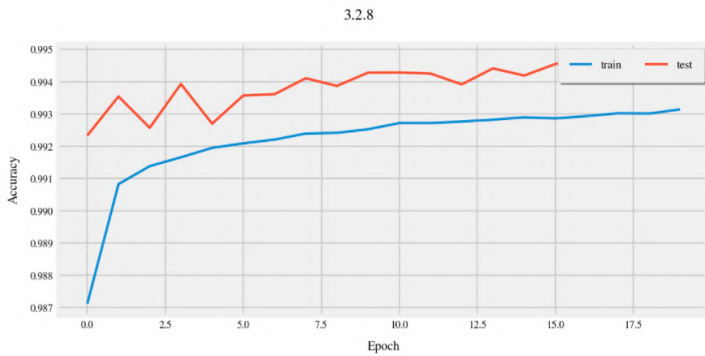


Anexo F – Figura 21 – Arquitectura – modelo 3.2.7

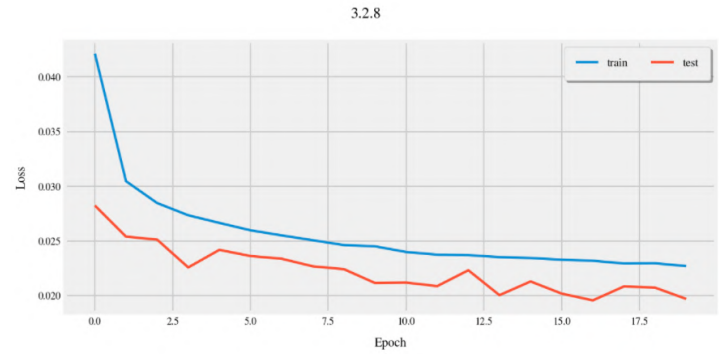
Train accuracy: 0.9924

Val accuracy: 0.9483

6.8 CNN - 3.2.8



Anexo F – Figura 22 – Accuracy por epoch – modelo 3.2.8



Anexo F – Figura 23 – Loss por epoch – modelo 3.2.8

```

with tf.device("/device:CPU:0"):
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

    model = Sequential()
    model.add(layers.Conv2D(16, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
    model.add(layers.MaxPooling2D((2,2)))
    model.add(layers.Dropout(0.5))
    model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2,2)))
    model.add(layers.Dropout(0.5))
    model.add(layers.Conv2D(32, (3, 3), activation='relu'))
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
    model.add(layers.Dense(5, activation='sigmoid'))

    sgd = SGD(lr=0.005, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer=sgd,
                  metrics=['accuracy'])

    model.summary()
  
```

executed in 386ms. finished 06:37:03 2021-01-25

Model: "sequential_6"

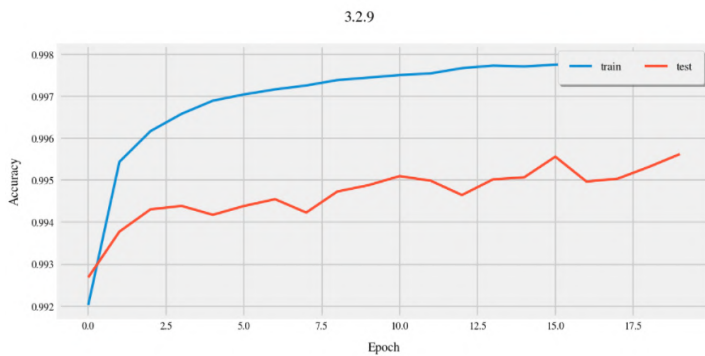
Layer (type)	Output Shape	Param #
conv2d_21 (Conv2D)	(None, 144, 13, 16)	160
max_pooling2d_16 (MaxPooling)	(None, 72, 6, 16)	0
dropout_5 (Dropout)	(None, 72, 6, 16)	0
conv2d_22 (Conv2D)	(None, 72, 6, 32)	4640
max_pooling2d_17 (MaxPooling)	(None, 36, 3, 32)	0
dropout_6 (Dropout)	(None, 36, 3, 32)	0
conv2d_23 (Conv2D)	(None, 34, 1, 32)	9248
flatten_6 (Flatten)	(None, 1088)	0
dropout_7 (Dropout)	(None, 1088)	0
dense_12 (Dense)	(None, 40)	43560
dense_13 (Dense)	(None, 5)	205
Total params:		57,813
Trainable params:		57,813
Non-trainable params:		0

Anexo F – Figura 24 – Arquitectura – modelo 3.2.8

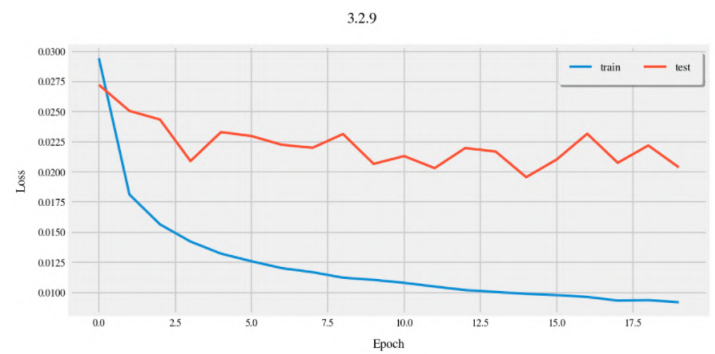
Train accuracy: 0.9931

Val accuracy: 0.9948

6.9 CNN - 3.2.9



Anexo F – Figura 25 – Accuracy por epoch – modelo 3.2.9



Anexo F – Figura 26 – Loss por epoch – modelo 3.2.9

```

with tf.device("/device:CPU:0"):
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

    model = Sequential()
    model.add(layers.Conv2D(8, (3, 3), activation='relu', input_shape=(144, 13, 1), padding='same'))
    model.add(layers.MaxPooling2D((2,2)))
    model.add(layers.Conv2D(32, (3, 3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2,2)))
    model.add(layers.Conv2D(32, (3, 3), activation='relu'))
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
    model.add(layers.Dense(5, activation='sigmoid'))

    sgd = SGD(lr=0.005, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

    model.summary()
  
```

executed in 310ms. finished 09:30:55 2021-01-25

Model: "sequential_7"

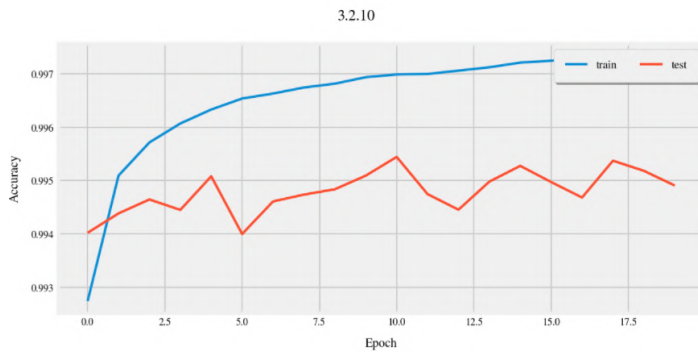
Layer (type)	Output Shape	Param #
conv2d_24 (Conv2D)	(None, 144, 13, 8)	80
max_pooling2d_18 (MaxPooling)	(None, 72, 6, 8)	0
conv2d_25 (Conv2D)	(None, 72, 6, 32)	2336
max_pooling2d_19 (MaxPooling)	(None, 36, 3, 32)	0
conv2d_26 (Conv2D)	(None, 34, 1, 32)	9248
flatten_7 (Flatten)	(None, 1088)	0
dropout_8 (Dropout)	(None, 1088)	0
dense_14 (Dense)	(None, 40)	43560
dense_15 (Dense)	(None, 5)	205
Total params:		55,429
Trainable params:		55,429
Non-trainable params:		0

Anexo F – Figura 27 – Arquitectura – modelo 3.2.9

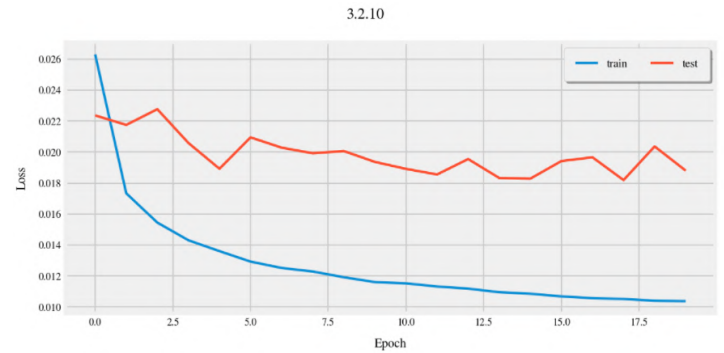
Train accuracy: 0.9979

Val accuracy: 0.9956

6.10 CNN-3.2.10



Anexo F – Figura 28 – Accuracy por epoch – modelo 3.2.10



Anexo F – Figura 29 – Loss por epoch – modelo 3.2.10

```

: with tf.device("/device:CPU:0"):
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

    model = Sequential()
    model.add(layers.Conv2D(8, (3, 3), strides=(2,2), activation='relu', input_shape=(144, 13, 1), padding='same'))
    model.add(layers.Conv2D(16, (3, 3), strides=(2,2), activation='relu', padding='same'))
    model.add(layers.Conv2D(32, (3, 3), strides=(2,2), activation='relu'))
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
    model.add(layers.Dense(5, activation='sigmoid'))

    sgd = SGD(lr=0.005, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

    model.summary()
  
```

executed in 1.14s, finished 11:31:34 2021-01-25

Model: "sequential_8"

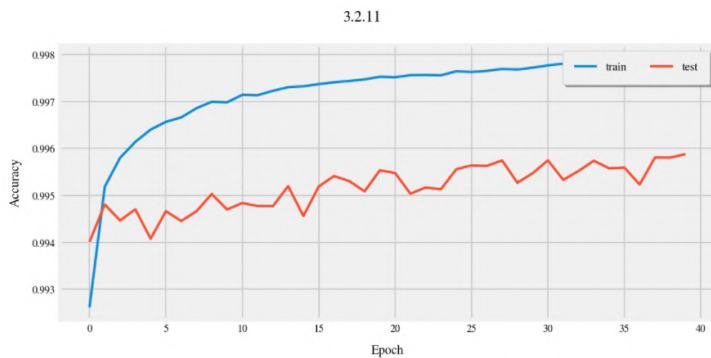
Layer (type)	Output Shape	Param #
conv2d_27 (Conv2D)	(None, 72, 7, 8)	80
conv2d_28 (Conv2D)	(None, 36, 4, 16)	1168
conv2d_29 (Conv2D)	(None, 17, 1, 32)	4640
flatten_8 (Flatten)	(None, 544)	0
dropout_9 (Dropout)	(None, 544)	0
dense_16 (Dense)	(None, 40)	21800
dense_17 (Dense)	(None, 5)	205
Total params: 27,893		
Trainable params: 27,893		
Non-trainable params: 0		

Anexo F – Figura 30 – Arquitectura – modelo 3.2.10

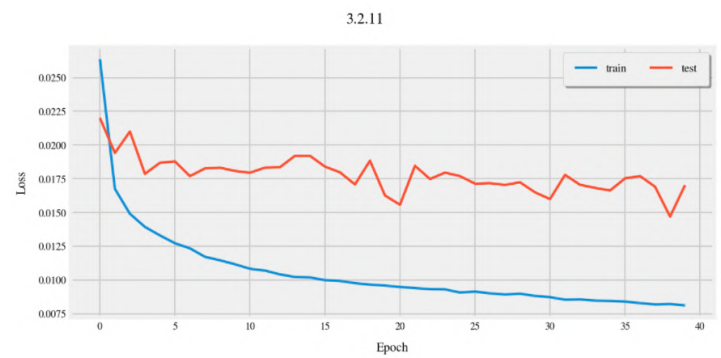
Train accuracy: 0.9973

Val accuracy: 0.9949

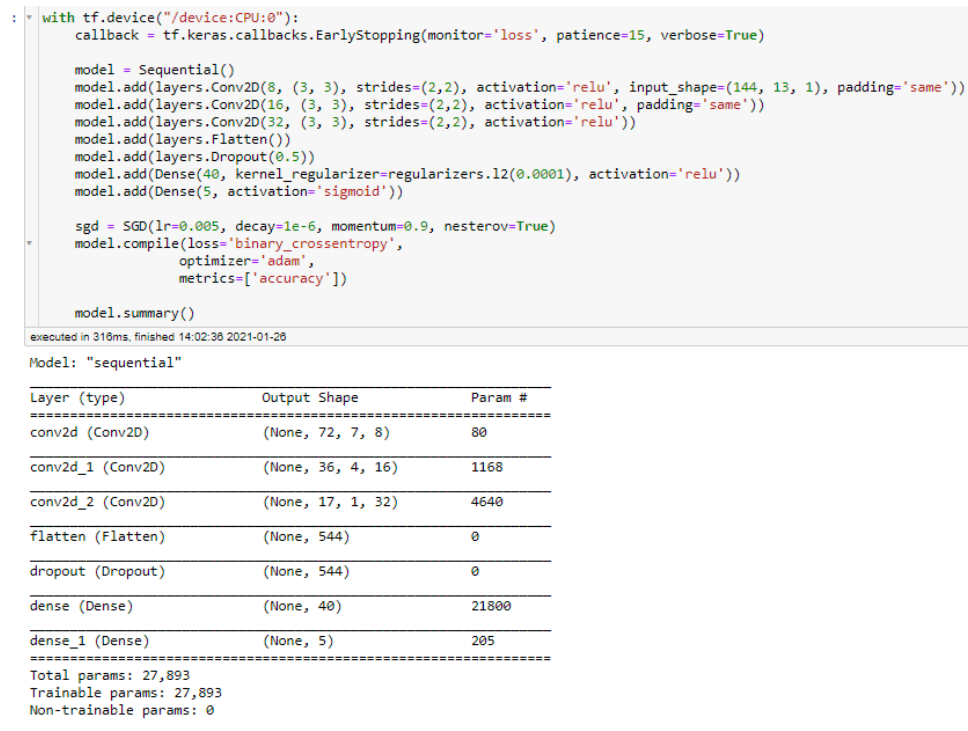
6.11 CNN-3.2.11



Anexo F – Figura 31 – Accuracy por epoch – modelo 3.2.11



Anexo F – Figura 32 – Loss por epoch – modelo 3.2.11

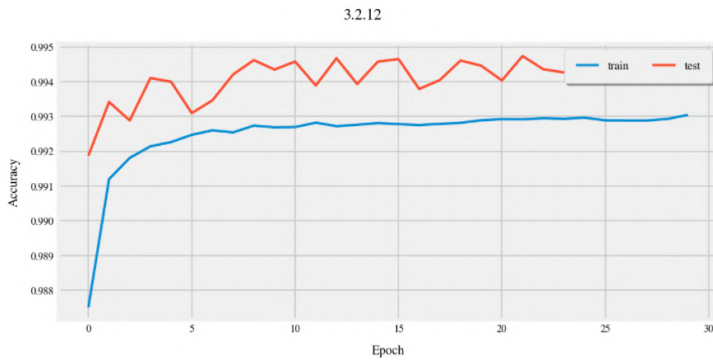


Anexo F – Figura 33 – Arquitectura – modelo 3.2.11

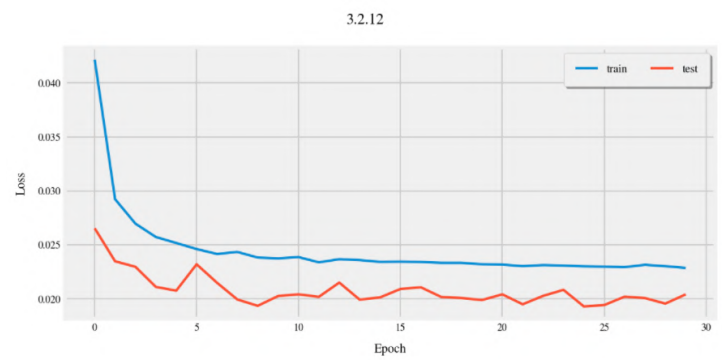
Train accuracy: 0.9979

Val accuracy: 0.9959

6.12 CNN-3.2.12



Anexo F – Figura 34 – Accuracy por epoch – modelo 3.2.12



Anexo F – Figura 35 – Loss por epoch – modelo 3.2.12

```

: with tf.device("/device:CPU:0"):
  callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

  model = Sequential()
  model.add(layers.Conv2D(8, (3, 3), strides=(2,2), activation='relu', input_shape=(144, 13, 1), padding='same'))
  model.add(layers.Dropout(0.5))
  model.add(layers.Conv2D(16, (3, 3), strides=(2,2), activation='relu', padding='same'))
  model.add(layers.Dropout(0.5))
  model.add(layers.Conv2D(32, (3, 3), strides=(2,2), activation='relu'))
  model.add(layers.Flatten())
  model.add(layers.Dropout(0.5))
  model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
  model.add(layers.Dense(5, activation='sigmoid'))

  sgd = SGD(lr=0.005, decay=1e-6, momentum=0.9, nesterov=True)
  model.compile(loss='binary_crossentropy',
                optimizer='adam',
                metrics=['accuracy'])

  model.summary()
  
```

executed in 274ms, finished 15:41:58 2021-01-20

Model: "sequential_1"

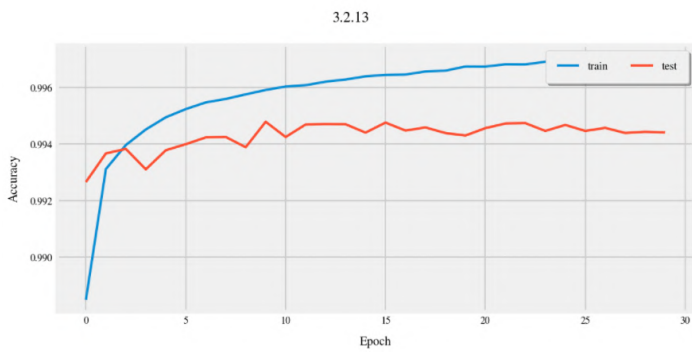
Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 72, 7, 8)	80
dropout_1 (Dropout)	(None, 72, 7, 8)	0
conv2d_4 (Conv2D)	(None, 36, 4, 16)	1168
dropout_2 (Dropout)	(None, 36, 4, 16)	0
conv2d_5 (Conv2D)	(None, 17, 1, 32)	4640
flatten_1 (Flatten)	(None, 544)	0
dropout_3 (Dropout)	(None, 544)	0
dense_2 (Dense)	(None, 40)	21800
dense_3 (Dense)	(None, 5)	205
Total params: 27,893		
Trainable params: 27,893		
Non-trainable params: 0		

Anexo F – Figura 36 – Arquitectura – modelo 3.2.12

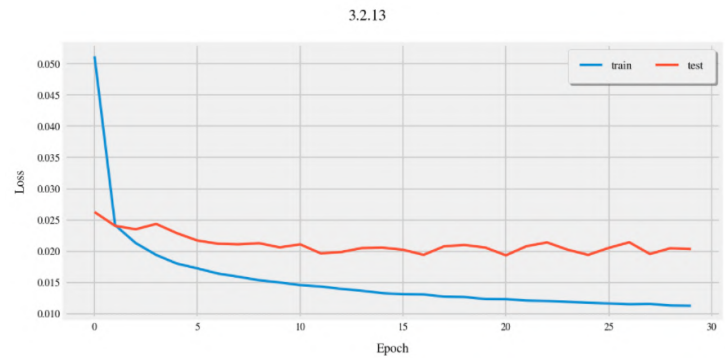
Train accuracy: 0.9930

Val accuracy: 0.9943

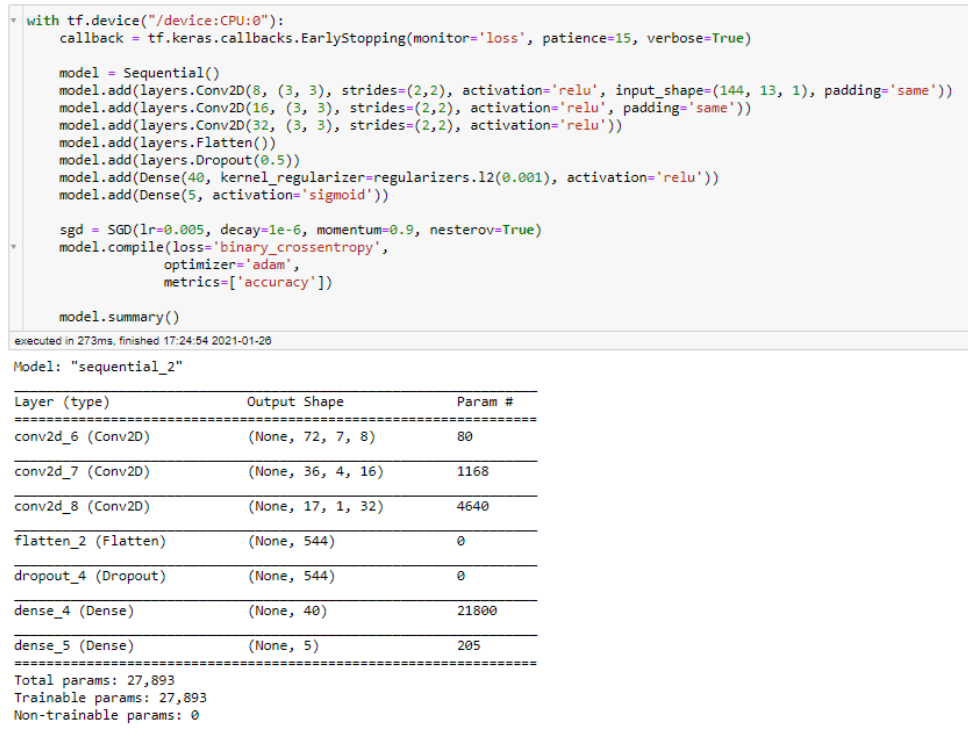
6.13 CNN-3.2.13



Anexo F – Figura 37 – Accuracy por epoch – modelo 3.2.13



Anexo F – Figura 38 – Loss por epoch – modelo 3.2.13

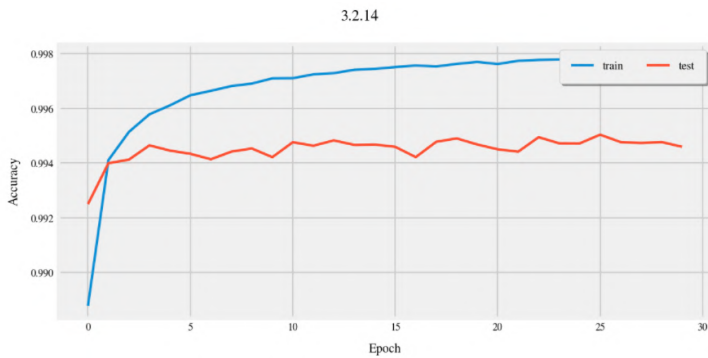


Anexo F – Figura 39 – Arquitectura – modelo 3.2.13

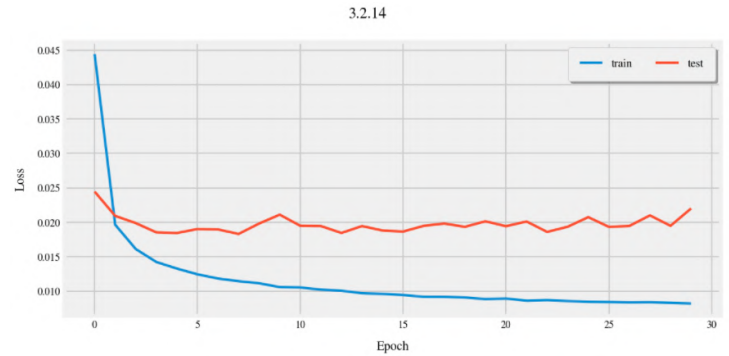
Train accuracy: 0.9971

Val accuracy: 0.9944

6.14 CNN - 3.2.14



Anexo F – Figura 40 – Accuracy por epoch – modelo 3.2.14



Anexo F – Figura 41 – Loss por epoch – modelo 3.2.14

```

with tf.device("/device:CPU:0"):
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

    model = Sequential()
    model.add(layers.Conv2D(8, (3, 3), strides=(2,2), activation='relu',
                           input_shape=(144, 13, 1), padding='same'))
    model.add(layers.Conv2D(16, (3, 3), strides=(2,2), activation='relu', padding='same'))
    model.add(layers.Conv2D(32, (3, 3), strides=(2,2), activation='relu', padding='same'))
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
    model.add(layers.Dense(5, activation='sigmoid'))

    sgd = SGD(lr=0.005, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

    model.summary()
  
```

executed in 273ms, finished 18:23:50 2021-01-20

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 72, 7, 8)	80
conv2d_10 (Conv2D)	(None, 36, 4, 16)	1168
conv2d_11 (Conv2D)	(None, 17, 1, 32)	4640
flatten_3 (Flatten)	(None, 544)	0
dropout_5 (Dropout)	(None, 544)	0
dense_6 (Dense)	(None, 40)	21800
dense_7 (Dense)	(None, 5)	205

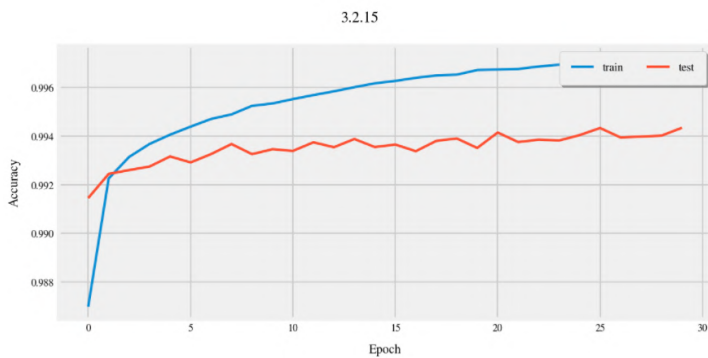
Total params: 27,893
 Trainable params: 27,893
 Non-trainable params: 0

Anexo F – Figura 42 – Arquitectura – modelo 3.2.14

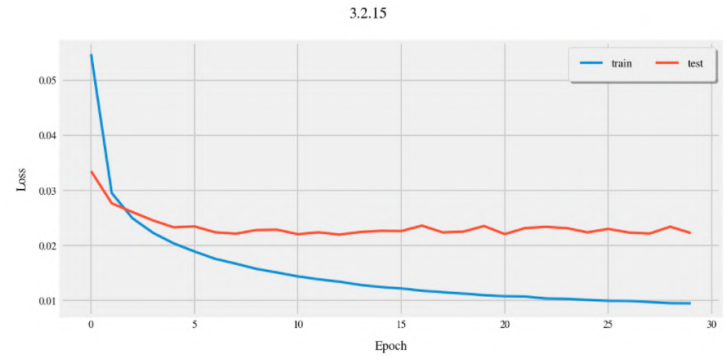
Train accuracy: 0.9979

Val accuracy: 0.9946

6.15 CNN-3.2.15



Anexo F – Figura 43 – Accuracy por epoch – modelo 3.2.15



Anexo F – Figura 44 – Loss por epoch – modelo 3.2.15

```
]: with tf.device("/device:CPU:0"):
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

    model = Sequential()
    model.add(layers.Conv2D(8, (3, 3), strides=(2,2), activation='relu', input_shape=(144, 13, 1), padding='same'))
    model.add(layers.Conv2D(16, (3, 3), strides=(2,2), activation='relu', padding='same'))
    model.add(layers.Conv2D(32, (3, 3), strides=(2,2), activation='relu'))
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
    model.add(layers.Dense(5, activation='sigmoid'))

    sgd = SGD(lr=0.005, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer=sgd,
                  metrics=['accuracy'])

    model.summary()

executed in 288ms, finished 19:22:16 2021-01-28
```

Model: "sequential_4"

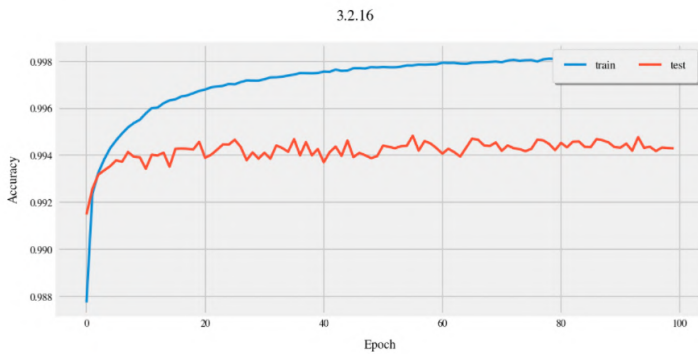
Layer (type)	Output Shape	Param #
conv2d_12 (Conv2D)	(None, 72, 7, 8)	80
conv2d_13 (Conv2D)	(None, 36, 4, 16)	1168
conv2d_14 (Conv2D)	(None, 17, 1, 32)	4640
flatten_4 (Flatten)	(None, 544)	0
dropout_6 (Dropout)	(None, 544)	0
dense_8 (Dense)	(None, 40)	21800
dense_9 (Dense)	(None, 5)	205
Total params: 27,893		
Trainable params: 27,893		
Non-trainable params: 0		

Anexo F – Figura 45 – Arquitectura – modelo 3.2.15

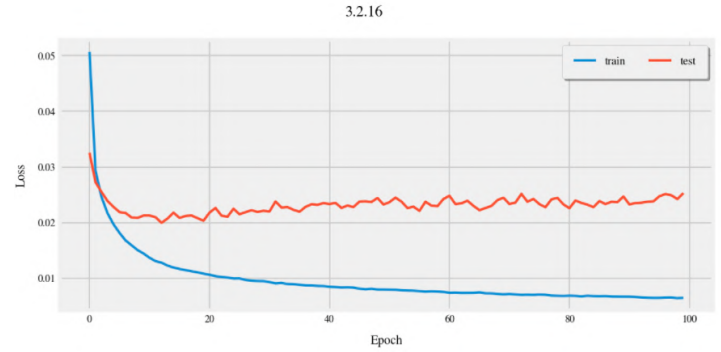
Train accuracy: 0.9972

Val accuracy: 0.9943

6.16 CNN-3.2.16



Anexo F – Figura 46 – Accuracy por epoch – modelo 3.2.16



Anexo F – Figura 47 – Loss por epoch – modelo 3.2.16

```

with tf.device("/device:CPU:0"):
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

    model = Sequential()
    model.add(layers.Conv2D(8, (3, 3), strides=(2,2), activation='relu', input_shape=(144, 13, 1), padding='same'))
    model.add(layers.Conv2D(16, (3, 3), strides=(2,2), activation='relu', padding='same'))
    model.add(layers.Conv2D(32, (3, 3), strides=(2,2), activation='relu'))
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
    model.add(layers.Dense(5, activation='sigmoid'))

    sgd = SGD(lr=0.005, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer=sgd,
                  metrics=['accuracy'])

    model.summary()
  
```

executed in 424ms, finished 21:14:57 2021-01-26

Model: "sequential_5"

Layer (type)	Output Shape	Param #
conv2d_15 (Conv2D)	(None, 72, 7, 8)	80
conv2d_16 (Conv2D)	(None, 36, 4, 16)	1168
conv2d_17 (Conv2D)	(None, 17, 1, 32)	4640
flatten_5 (Flatten)	(None, 544)	0
dropout_7 (Dropout)	(None, 544)	0
dense_10 (Dense)	(None, 40)	21800
dense_11 (Dense)	(None, 5)	205

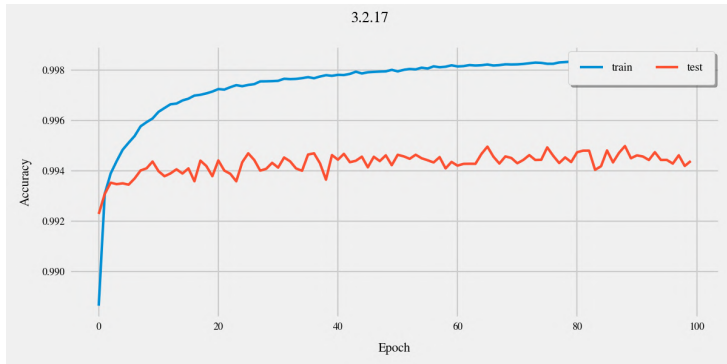
Total params: 27,893
 Trainable params: 27,893
 Non-trainable params: 0

Anexo F – Figura 48 – Arquitectura – modelo 3.2.16

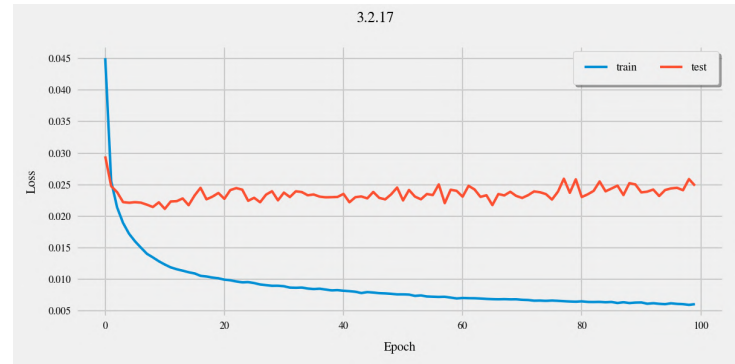
Train accuracy: 0.9982

Val accuracy: 0.9943

6.17 CNN-3.2.17



Anexo F – Figura 49 – Accuracy por epoch – modelo 3.2.17



Anexo F – Figura 50 – Loss por epoch – modelo 3.2.17

```

5]: with tf.device("/device:CPU:0"):
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

    model = Sequential()
    model.add(layers.Conv2D(8, (3, 3), strides=(2,2), activation='relu', input_shape=(144, 13, 1), padding='same'))
    model.add(layers.Conv2D(16, (3, 3), strides=(2,2), activation='relu', padding='same'))
    model.add(layers.Conv2D(32, (3, 3), strides=(2,2), activation='relu'))
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
    model.add(layers.Dense(5, activation='sigmoid'))

    sgd = SGD(lr=0.009, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer=sgd,
                  metrics=['accuracy'])

    model.summary()
  
```

executed in 271ms, finished 14:00:43 2021-01-27

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 72, 7, 8)	80
conv2d_1 (Conv2D)	(None, 36, 4, 16)	1168
conv2d_2 (Conv2D)	(None, 17, 1, 32)	4640
flatten (Flatten)	(None, 544)	0
dropout (Dropout)	(None, 544)	0
dense (Dense)	(None, 40)	21800
dense_1 (Dense)	(None, 5)	205

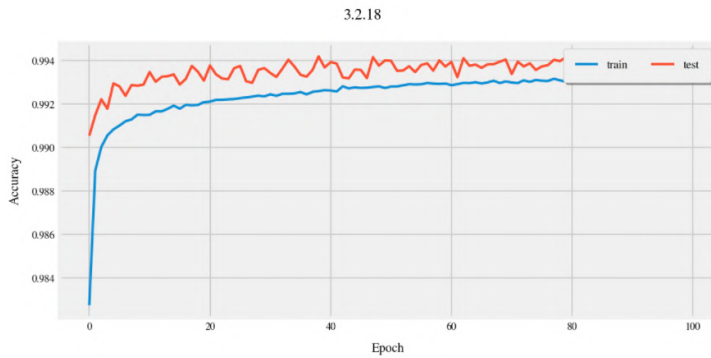
Total params: 27,893
 Trainable params: 27,893
 Non-trainable params: 0

Anexo F – Figura 51 – Arquitectura – modelo 3.2.17

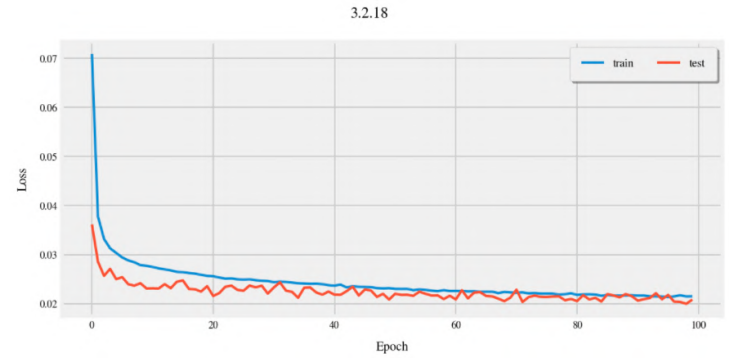
Train accuracy: 0.9823

val accuracy: 0.9943

6.18 CNN-3.2.18



Anexo F – Figura 52 – Accuracy por epoch – modelo 3.2.18



Anexo F – Figura 53 – Loss por epoch – modelo 3.2.18

```

]: with tf.device("/device:CPU:0"):
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

    model = Sequential()
    model.add(layers.Conv2D(8, (3, 3), strides=(2,2), activation='relu', input_shape=(144, 13, 1), padding='same'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Conv2D(16, (3, 3), strides=(2,2), activation='relu', padding='same'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Conv2D(32, (3, 3), strides=(2,2), activation='relu'))
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0005), activation='relu'))
    model.add(layers.Dense(5, activation='sigmoid'))

    sgd = SGD(lr=0.009, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer=sgd,
                  metrics=['accuracy'])

    model.summary()
  
```

executed in 278ms, finished 20:39:09 2021-01-27

Model: "sequential"

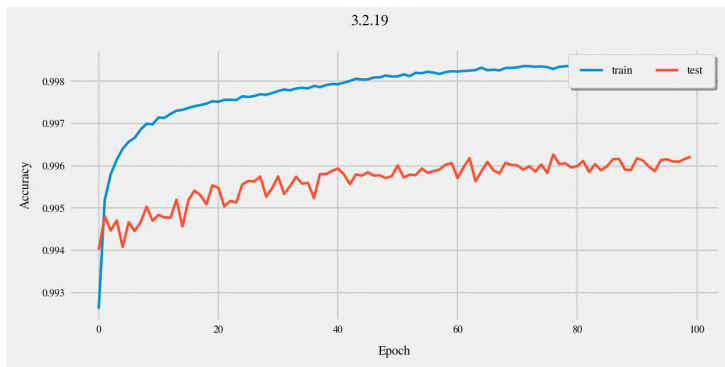
Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 72, 7, 8)	80
dropout (Dropout)	(None, 72, 7, 8)	0
conv2d_1 (Conv2D)	(None, 36, 4, 16)	1168
dropout_1 (Dropout)	(None, 36, 4, 16)	0
conv2d_2 (Conv2D)	(None, 17, 1, 32)	4640
flatten (Flatten)	(None, 544)	0
dropout_2 (Dropout)	(None, 544)	0
dense (Dense)	(None, 40)	21800
dense_1 (Dense)	(None, 5)	205
Total params: 27,893		
Trainable params: 27,893		
Non-trainable params: 0		

Anexo F – Figura 54 – Arquitectura – modelo 3.2.18

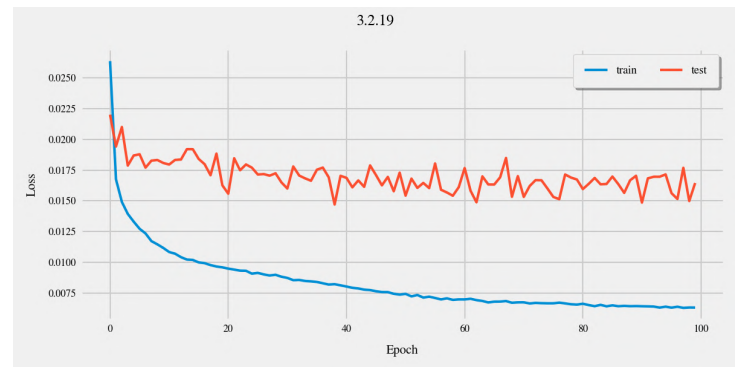
Train accuracy: 0.9932

Val accuracy: 0.9939

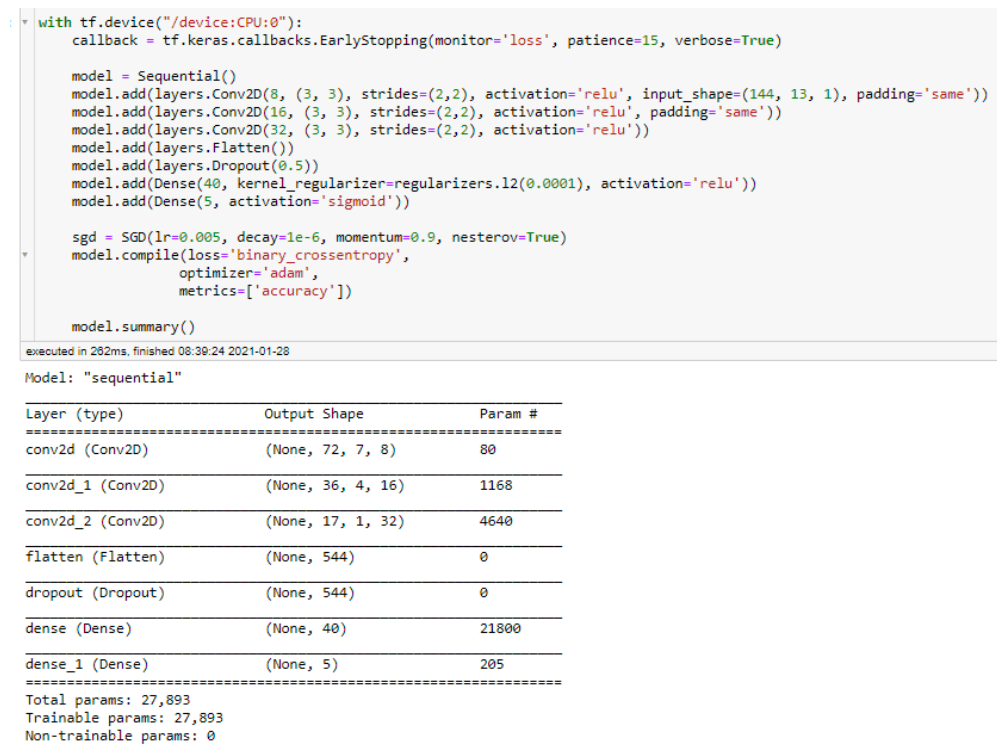
6.19 CNN-3.2.19



Anexo F – Figura 55 – Accuracy por epoch – modelo 3.2.19



Anexo F – Figura 56 – Loss por epoch – modelo 3.2.19

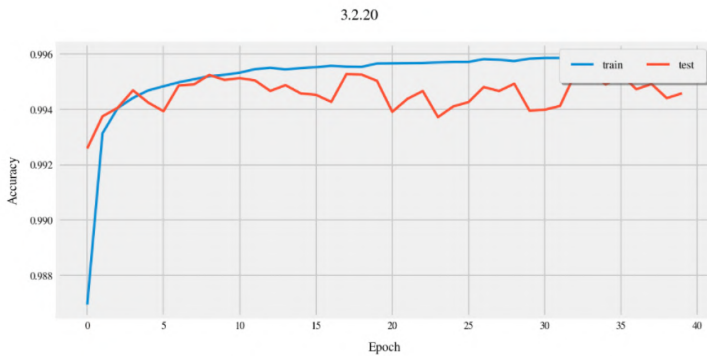


Anexo F – Figura 57 – Arquitectura – modelo 3.2.19

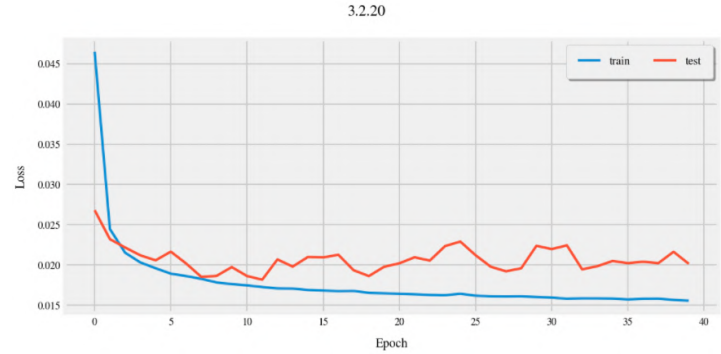
Train accuracy: 0.9987

val accuracy: 0.9949

6.20 CNN - 3.2.20



Anexo F – Figura 58 – Accuracy por epoch – modelo 3.2.20



Anexo F – Figura 59 – Loss por epoch – modelo 3.2.20

```

with tf.device("/device:CPU:0"):
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

    model = Sequential()
    model.add(layers.Conv2D(8, (3, 3), strides=(2,2), activation='relu', input_shape=(144, 13, 1), padding='same'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Conv2D(16, (3, 3), strides=(2,2), activation='relu', padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv2D(32, (3, 3), strides=(2,2), activation='relu'))
    model.add(layers.BatchNormalization())
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
    model.add(layers.BatchNormalization(
        momentum=0.95,
        epsilon=0.005,
        beta_initializer=RandomNormal(mean=0.0, stddev=0.05),
        gamma_initializer=Constant(value=0.9)
    ))
    model.add(layers.Dense(5, activation='sigmoid'))

    sgd = SGD(lr=0.005, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])

    model.summary()
  
```

executed in 367ms. finished 10:32:21 2021-01-28

Model: "sequential_1"

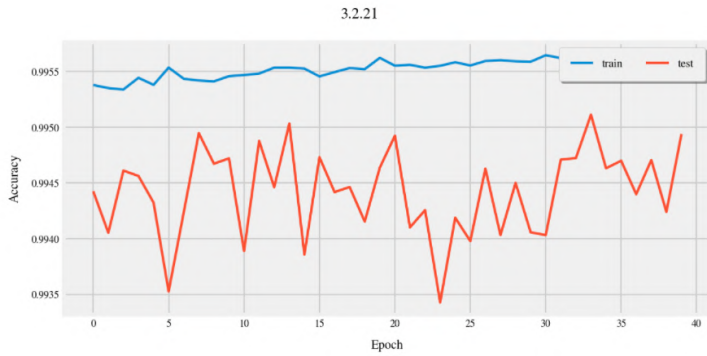
Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 72, 7, 8)	80
dropout_1 (Dropout)	(None, 72, 7, 8)	0
batch_normalization_1 (Batch Normalization)	(None, 72, 7, 8)	32
conv2d_4 (Conv2D)	(None, 36, 4, 16)	1168
batch_normalization_2 (Batch Normalization)	(None, 36, 4, 16)	64
conv2d_5 (Conv2D)	(None, 17, 1, 32)	4640
batch_normalization_3 (Batch Normalization)	(None, 17, 1, 32)	128
flatten_1 (Flatten)	(None, 544)	0
dropout_2 (Dropout)	(None, 544)	0
dense_2 (Dense)	(None, 40)	21800
batch_normalization_4 (Batch Normalization)	(None, 40)	160
dense_3 (Dense)	(None, 5)	205
Total params:	28,277	
Trainable params:	28,085	
Non-trainable params:	192	

Anexo F – Figura 60 – Arquitectura – modelo 3.2.20

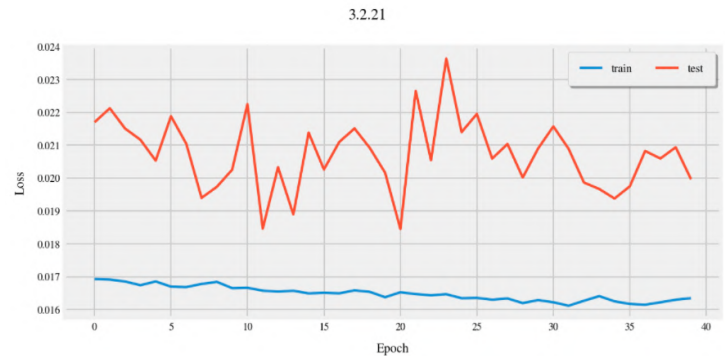
Train accuracy: 0.9959

Val accuracy: 0.9946

6.21 CNN - 3.2.21



Anexo F – Figura 61 – Accuracy por epoch – modelo 3.2.21



Anexo F – Figura 62 – Loss por epoch – modelo 3.2.21

```

with tf.device("/device:CPU:0"):
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3,
                                                min_delta=0.0001,
                                                verbose=True)

    model = Sequential()
    model.add(layers.Conv2D(8, (3, 3), strides=(2,2), activation='relu', input_shape=(144, 13, 1), padding='same'))
    model.add(layers.Dropout(0.5))
    model.add(BatchNormalization())
    model.add(layers.Conv2D(16, (3, 3), strides=(2,2), activation='relu', padding='same'))
    model.add(BatchNormalization())
    model.add(layers.Conv2D(32, (3, 3), strides=(2,2), activation='relu'))
    model.add(BatchNormalization())
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
    model.add(BatchNormalization(
        momentum=0.95,
        epsilon=0.005,
        beta_initializer=RandomNormal(mean=0.0, stddev=0.05),
        gamma_initializer=Constant(value=0.9)
    ))
    model.add(layers.Dense(5, activation='sigmoid'))

    sgd = SGD(lr=0.005, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer='sgd',
                  metrics=['accuracy'])

    model.summary()
  
```

Executed in 863ms. Finished 18:53:16 2021-01-28

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 72, 7, 8)	80
dropout_3 (Dropout)	(None, 72, 7, 8)	0
batch_normalization_4 (Batch Normalization)	(None, 72, 7, 8)	32
conv2d_7 (Conv2D)	(None, 36, 4, 16)	1168
batch_normalization_5 (Batch Normalization)	(None, 36, 4, 16)	64
conv2d_8 (Conv2D)	(None, 17, 1, 32)	4640
batch_normalization_6 (Batch Normalization)	(None, 17, 1, 32)	128
flatten_2 (Flatten)	(None, 544)	0
dropout_4 (Dropout)	(None, 544)	0
dense_4 (Dense)	(None, 40)	21800
batch_normalization_7 (Batch Normalization)	(None, 40)	160
dense_5 (Dense)	(None, 5)	205

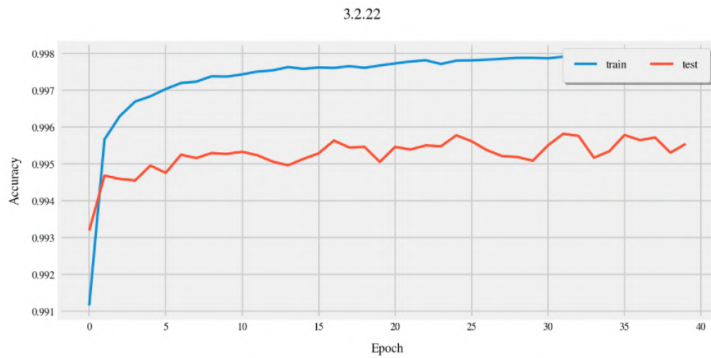
Total params: 28,277
 Trainable params: 28,085
 Non-trainable params: 192

Anexo F – Figura 63 – Arquitectura – modelo 3.2.21

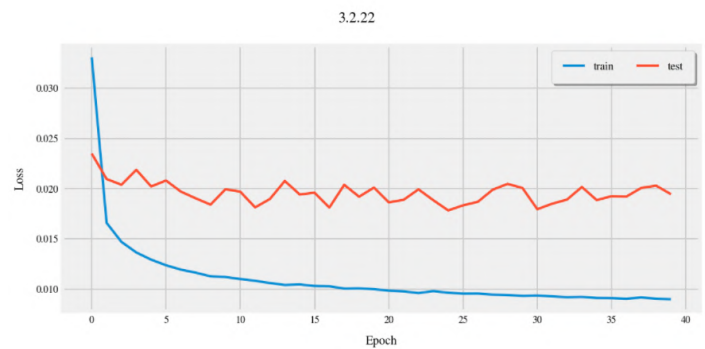
Train accuracy: 0.9956

Val accuracy: 0.9949

6.22 CNN - 3.2.22



Anexo F – Figura 64 – Accuracy por epoch – modelo 3.2.22



Anexo F – Figura 65 – Loss por epoch – modelo 3.2.22

```

with tf.device("/device:CPU:0"):
    callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=15, verbose=True)

    model = Sequential()
    model.add(layers.Conv2D(8, (3, 3), strides=(2,2), activation='relu', input_shape=(144, 13, 1), padding='same'))
    model.add(layers.Dropout(0.2))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv2D(16, (3, 3), strides=(2,2), activation='relu', padding='same'))
    model.add(layers.BatchNormalization())
    model.add(layers.Conv2D(32, (3, 3), strides=(2,2), activation='relu'))
    model.add(layers.BatchNormalization())
    model.add(layers.Flatten())
    model.add(layers.Dropout(0.2))
    model.add(layers.Dense(40, kernel_regularizer=regularizers.l2(0.0001), activation='relu'))
    model.add(layers.BatchNormalization())
    model.add(layers.Dense(5, activation='sigmoid'))

    sgd = SGD(lr=0.005, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='binary_crossentropy',
                  optimizer=sgd,
                  metrics=['accuracy'])

    model.summary()
  
```

Executed in 553ms, finished 12:08:07 2021-01-28
 Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 72, 7, 8)	80
dropout (Dropout)	(None, 72, 7, 8)	0
batch_normalization (Batch Normalization)	(None, 72, 7, 8)	32
conv2d_1 (Conv2D)	(None, 36, 4, 16)	1168
batch_normalization_1 (Batch Normalization)	(None, 36, 4, 16)	64
conv2d_2 (Conv2D)	(None, 17, 1, 32)	4640
batch_normalization_2 (Batch Normalization)	(None, 17, 1, 32)	128
flatten (Flatten)	(None, 544)	0
dropout_1 (Dropout)	(None, 544)	0
dense (Dense)	(None, 40)	21800
batch_normalization_3 (Batch Normalization)	(None, 40)	160
dense_1 (Dense)	(None, 5)	205

Total params: 28,277
 Trainable params: 28,085
 Non-trainable params: 192

Anexo F – Figura 66 – Arquitectura – modelo 3.2.22

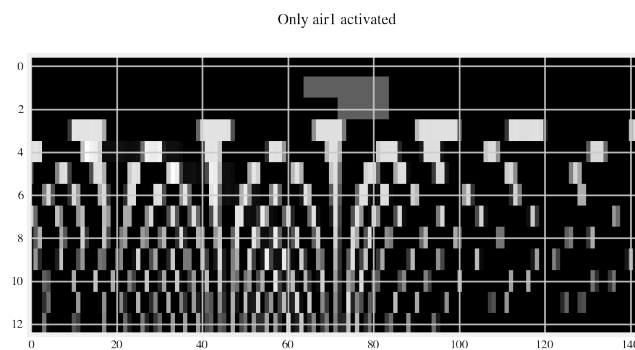
Train accuracy: 0.9980

Val accuracy: 0.9950

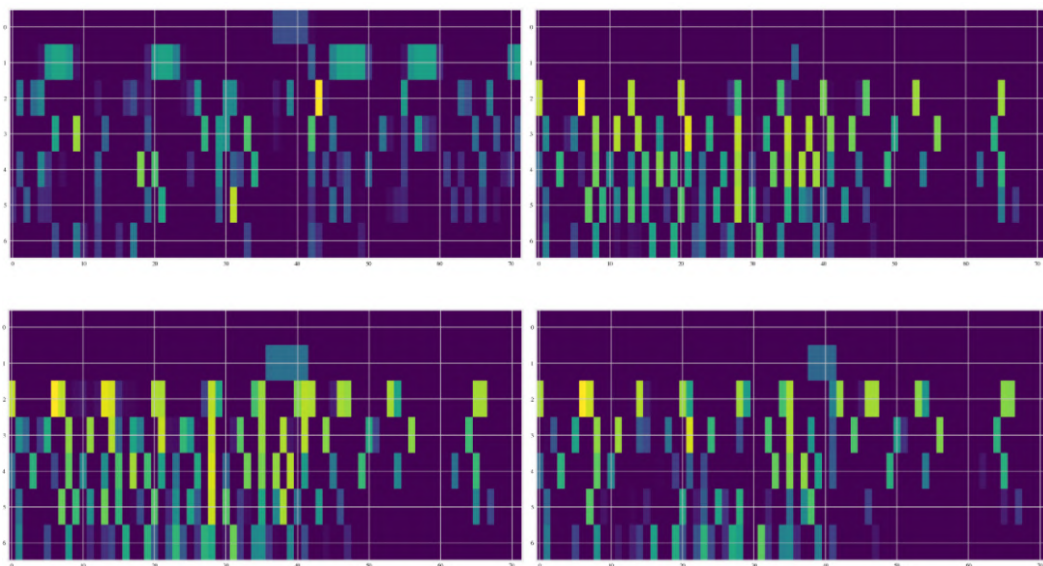
ANEXO G – FEATURE MAPS

Este anexo contiene los distintos feature maps para cada capa convolucional del mejor modelo CNN obtenido para cada uno de los distintos electrodomésticos seleccionados. Cada caso representa el estudio de un vizgrama que contenía la activación únicamente del electrodoméstico en cuestión.

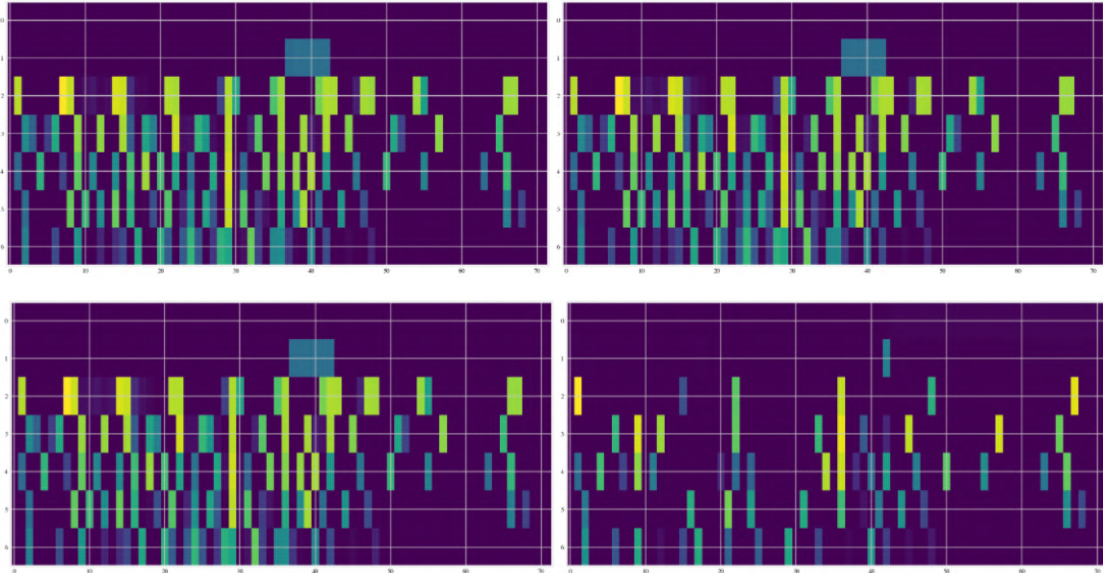
AIR1



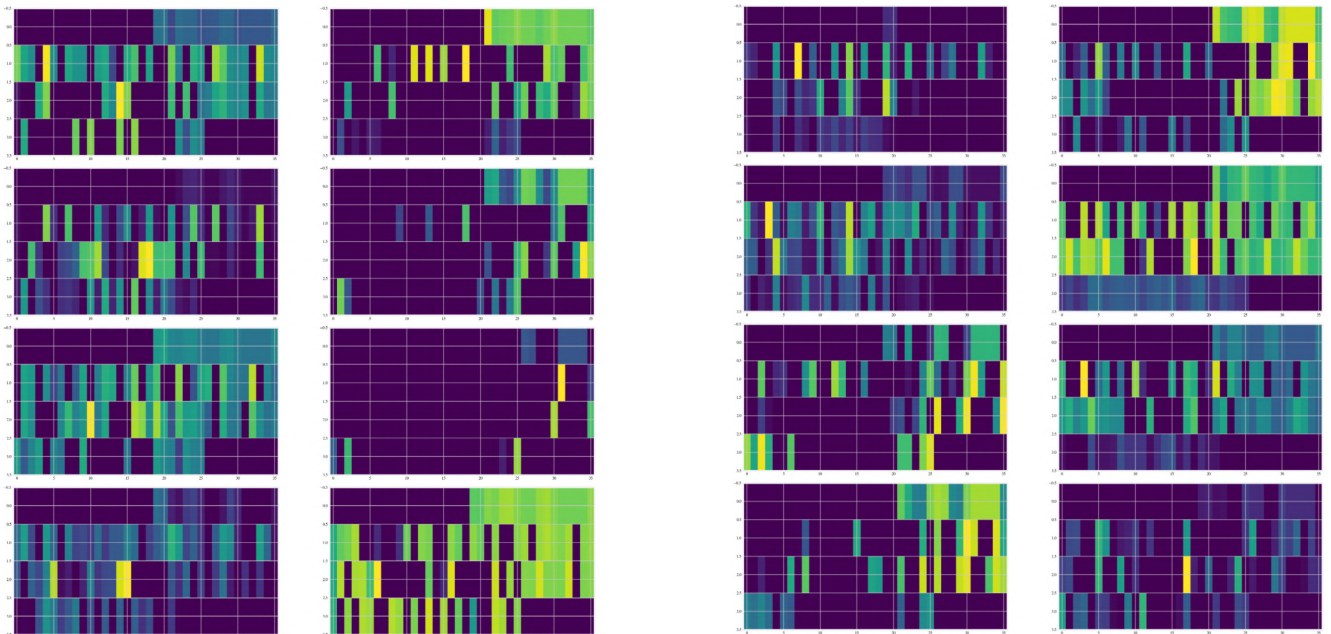
Anexo G – Figura 1 – Vizgrama de entrada – air1



Anexo G – Figura 2 – Feature maps de la primera capa convolucional (filtro 1 a 4) vizgrama con air1 como input

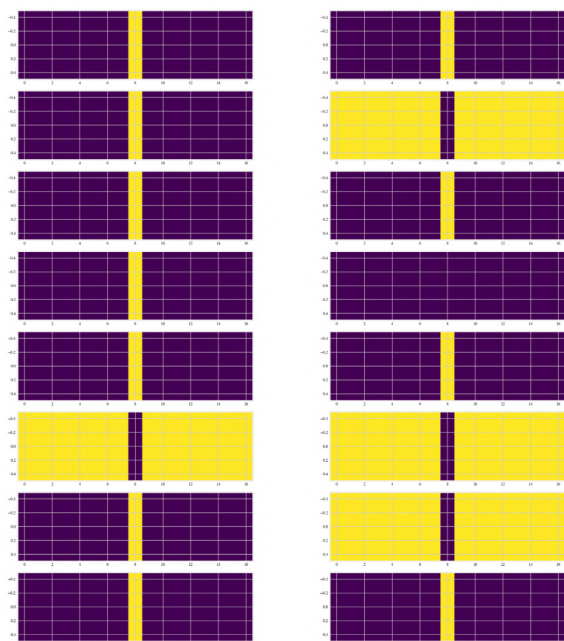


Anexo G – Figura 3 – Feature maps de la primera capa convolucional (filtro 5 a 8) vizgrama con `air1` como input

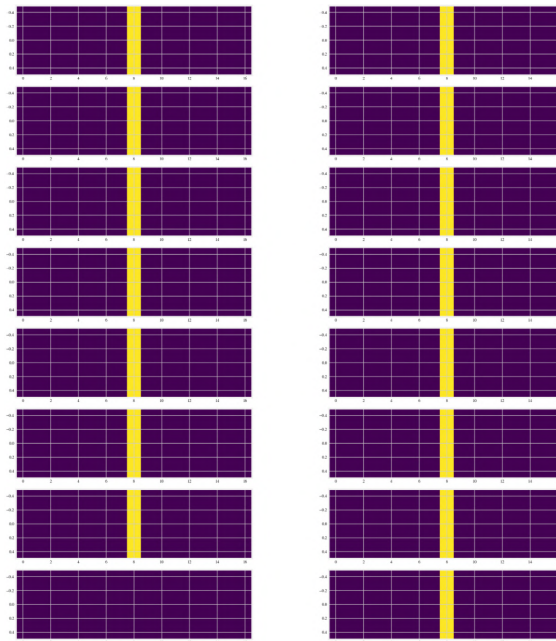


Anexo G – Figura 4 – Feature maps de la segunda capa convolucional (filtro 1 a 8) – vizgrama con `air1` como input

Anexo G – Figura 5 – Feature maps de la segunda capa convolucional (filtro 9 a 16) – vizgrama con `air1` como input



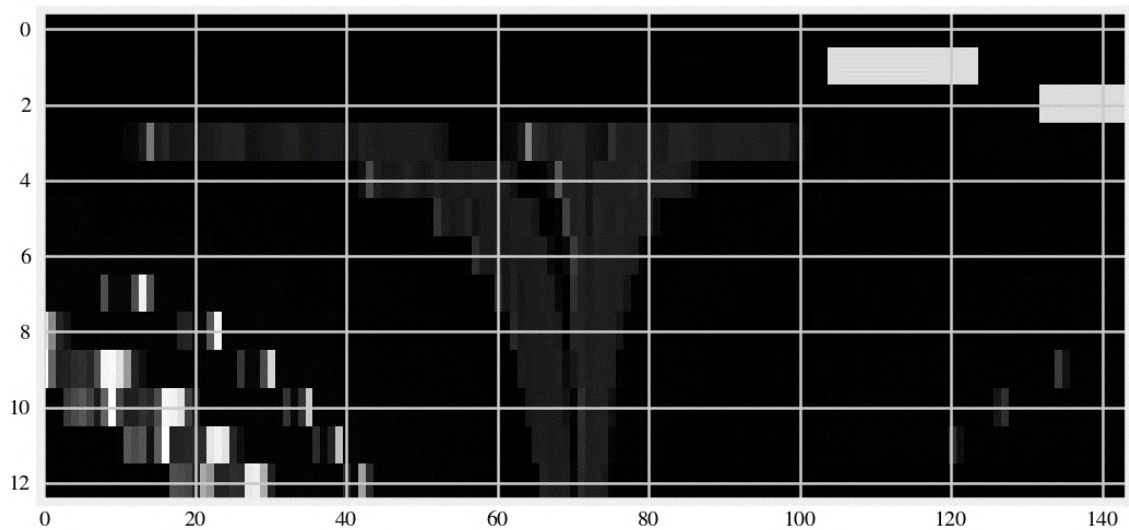
Anexo G – Figura 6 – Feature maps de la tercera capa convolucional (filtro 1 a 16) – vizgrama con `air1` como input



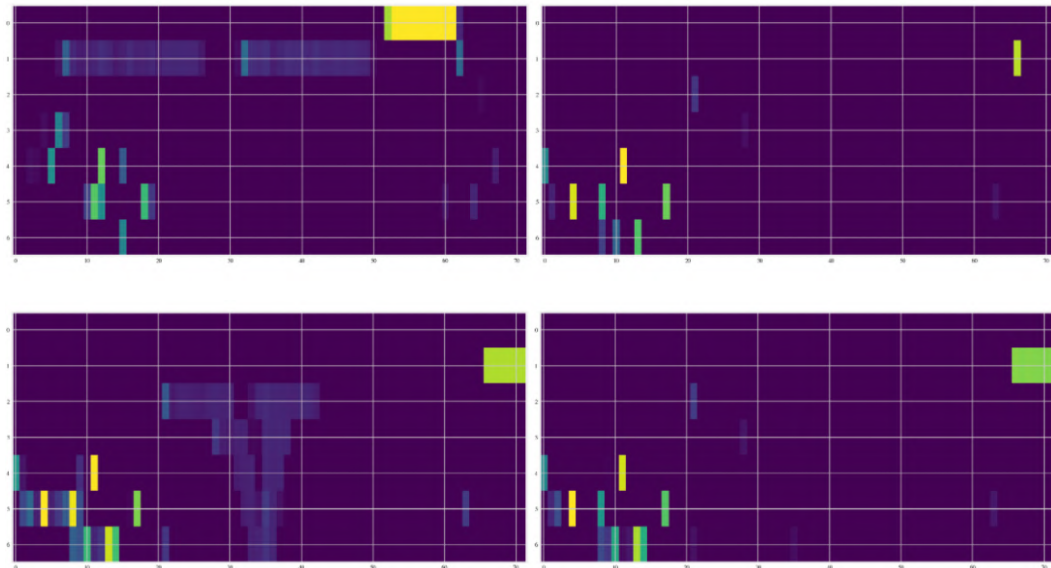
Anexo G – Figura 7 – Feature maps de la tercera capa convolucional (filtro 17 a 32) – vizgrama con `air1` como input

CLOTHESWASHER1

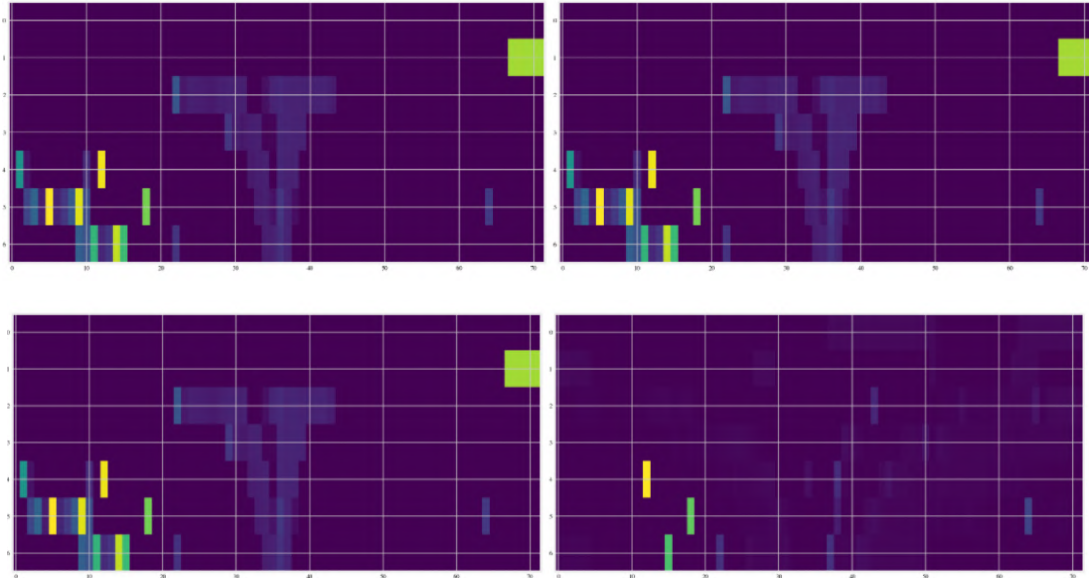
Only clotheswasher1 activated



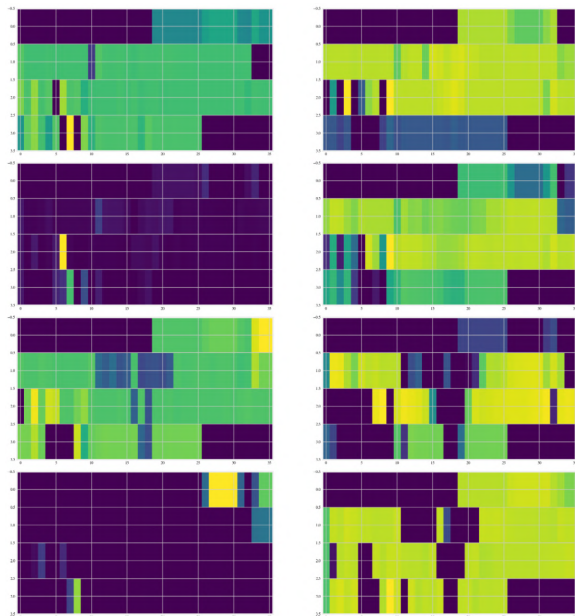
Anexo G – Figura 8 – Vizgrama de entrada



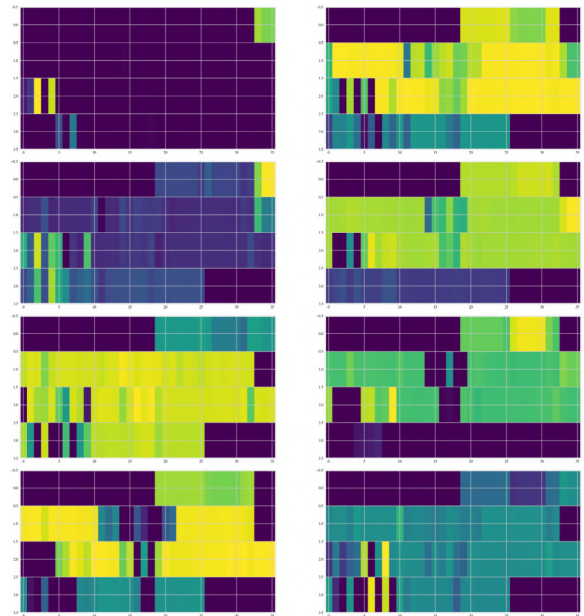
Anexo G – Figura 9 – Feature maps de la primera capa convolucional (ftrlo 1 a 4)
vizgrama con clotheswasher1 como input



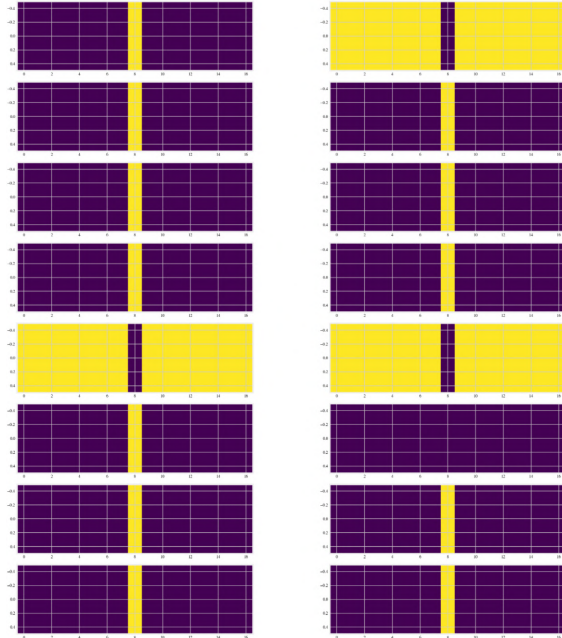
Anexo G – Figura 10 – Feature maps de la primera capa convolucional (filtro 5 a 8)
vizgrama con clotheswasher1 como input



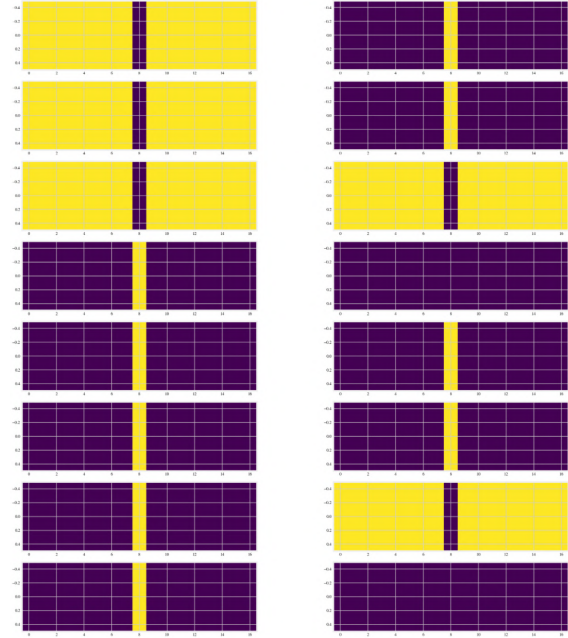
Anexo G – Figura 11 – Feature maps de la segunda capa convolucional (filtro 1 a 8) – vizgrama con clotheswasher1 como input



Anexo G – Figura 12 – Feature maps de la segunda capa convolucional (filtro 9 a 16) – vizgrama con clotheswasher1 como input



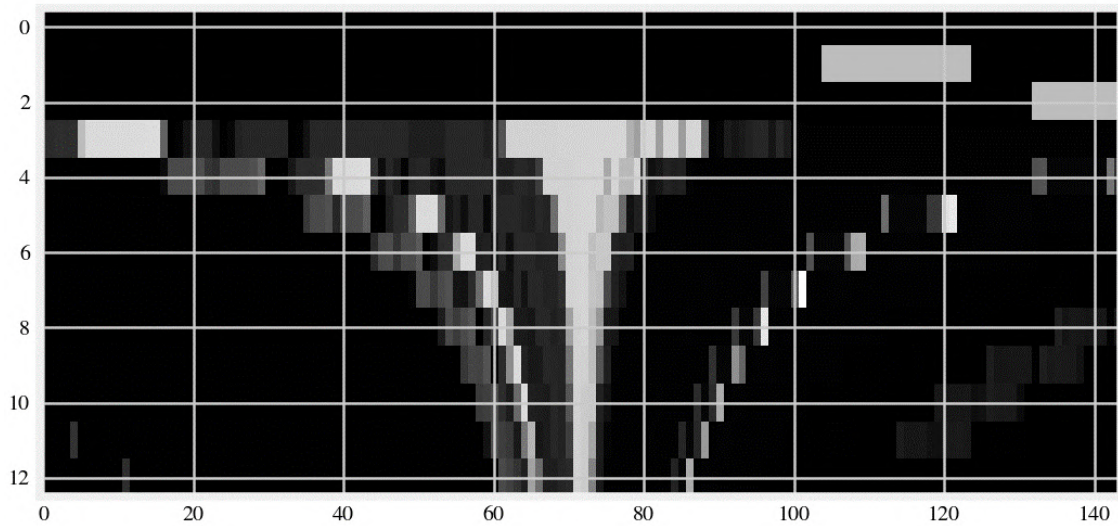
Anexo G – Figura 13 – Feature maps de la tercera capa convolucional (filtro 1 a 16) – vizgrama con clotheswasher1 como input



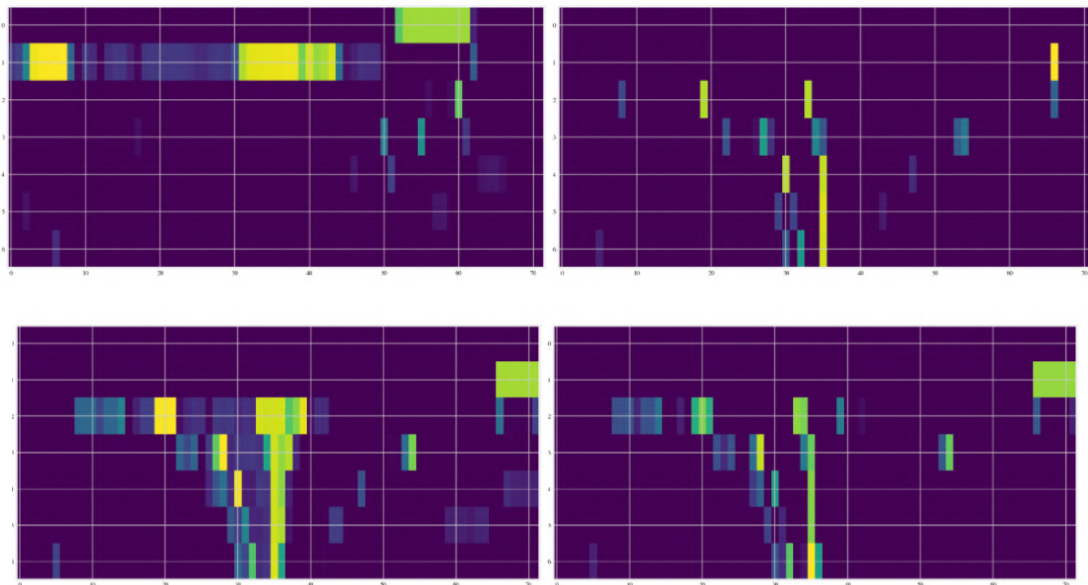
Anexo G – Figura 14 – Feature maps de la tercera capa convolucional (filtro 17 a 32) – vizgrama con clotheswasher1 como input

DISHWASHER1

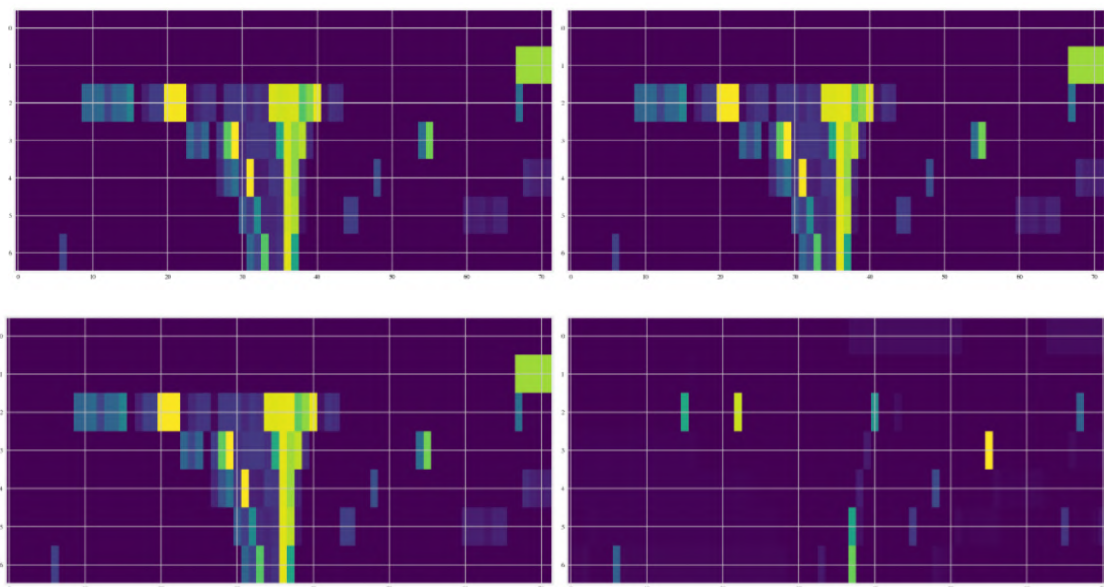
Only dishwasher1 activated



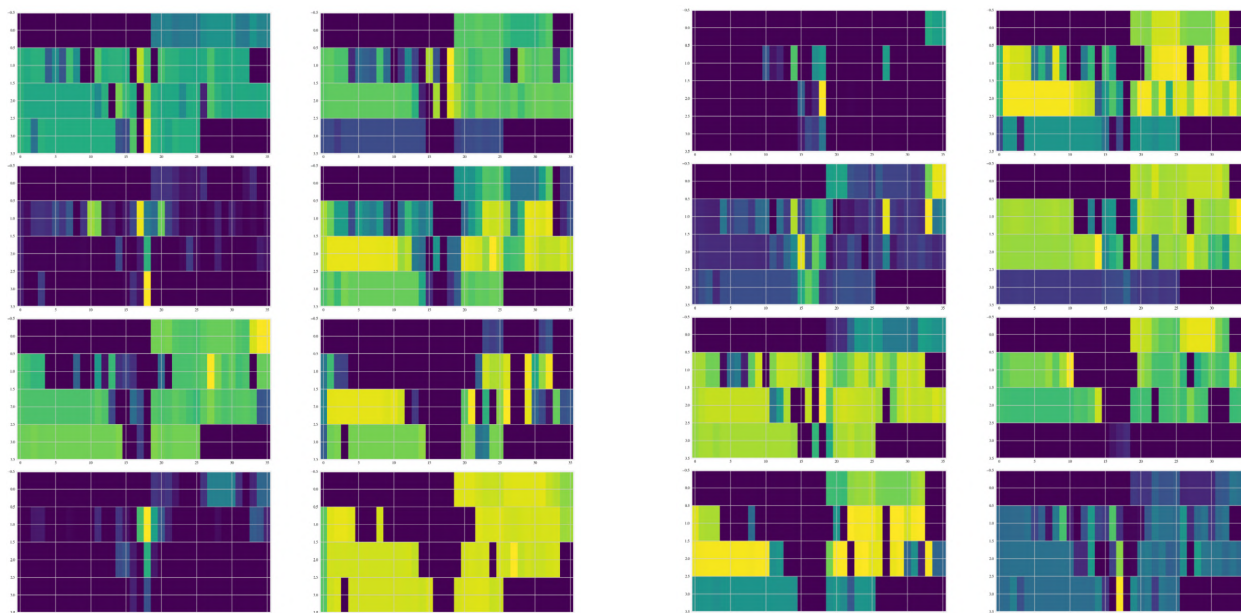
Anexo G – Figura 15 – Vizgrama de entrada



Anexo G – Figura 16 – Feature maps de la primera capa convolucional (filtro 1 a 4)
vizgrama con **dishwasher1** como input

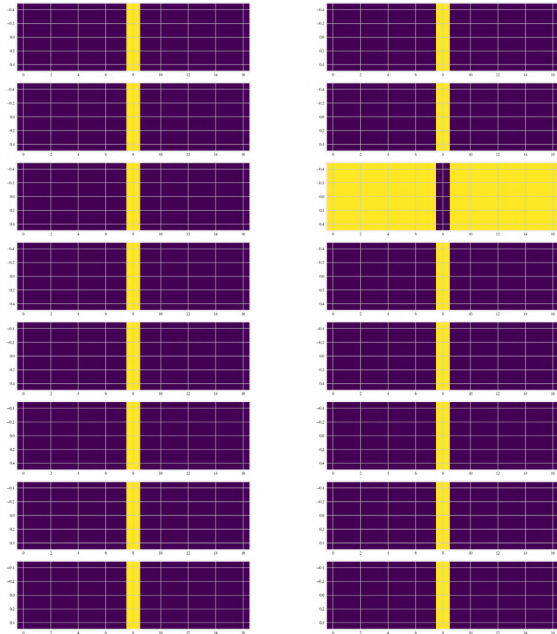


Anexo G – Figura 17 – Feature maps de la primera capa convolucional (filtro 5 a 8) vizgram con dishwasher1 como input

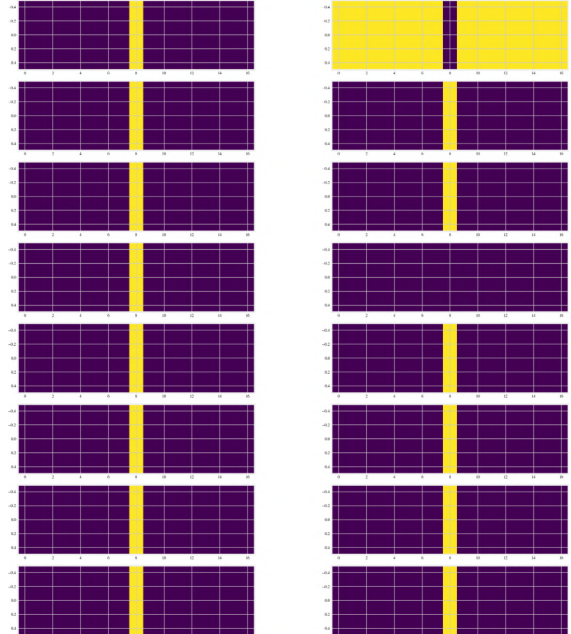


Anexo G – Figura 18 – Feature maps de la segunda capa convolucional (filtro 1 a 8) – vizgram con dishwasher1 como input

Anexo G – Figura 19 – Feature maps de la segunda capa convolucional (filtro 9 a 16) – vizgram con dishwasher1 como input



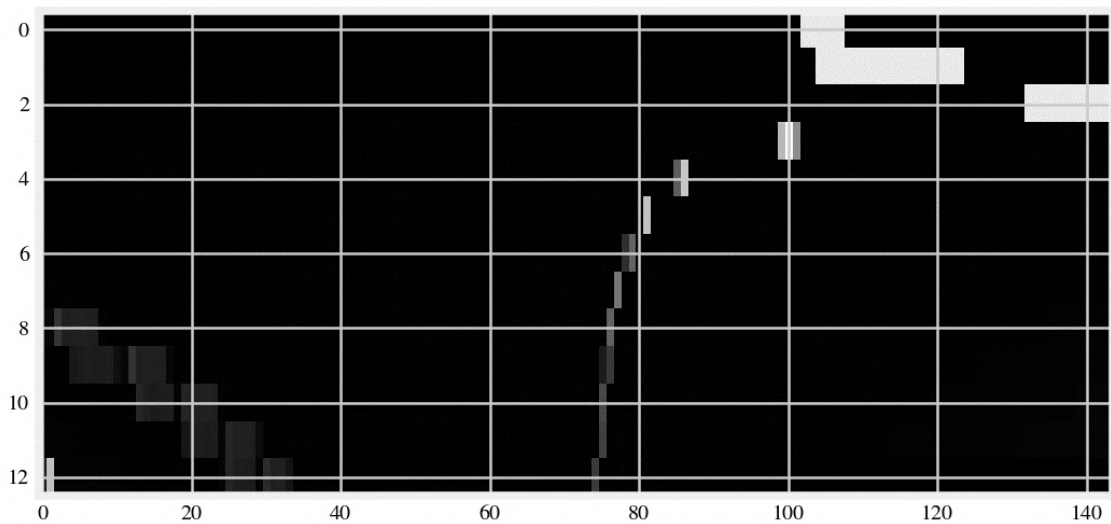
Anexo G – Figura 20 – Feature maps de la tercera capa convolucional (filtro 1 a 16) – vizgrama con dishwasher1 como input



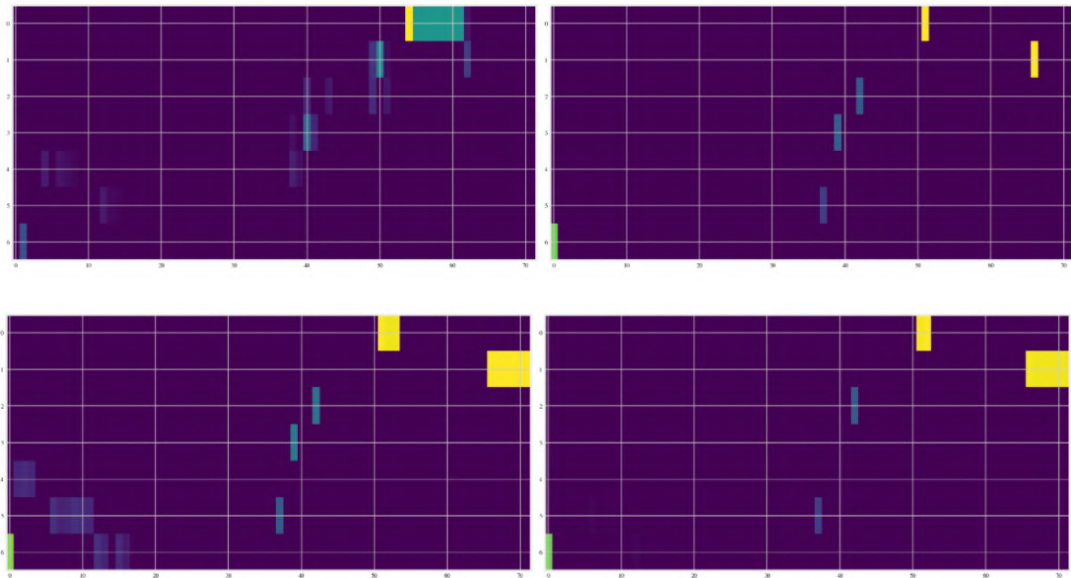
Anexo G - Figura 21 – Feature maps de la tercera capa convolucional (filtro 17 a 32) – vizgrama con dishwasher1 como input

KITCHENAPP1

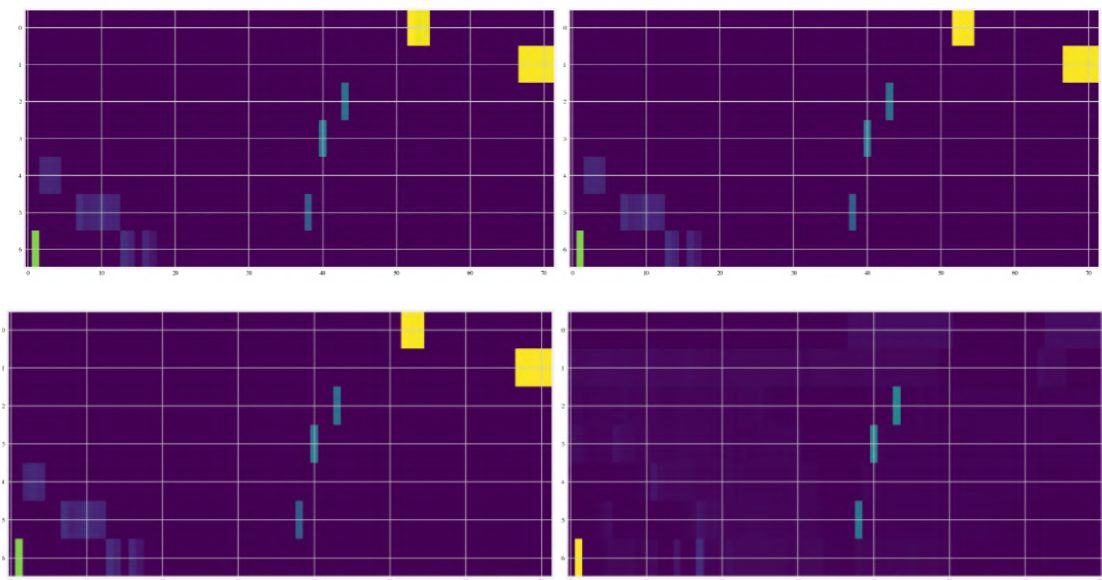
Only kitchenapp1 activated



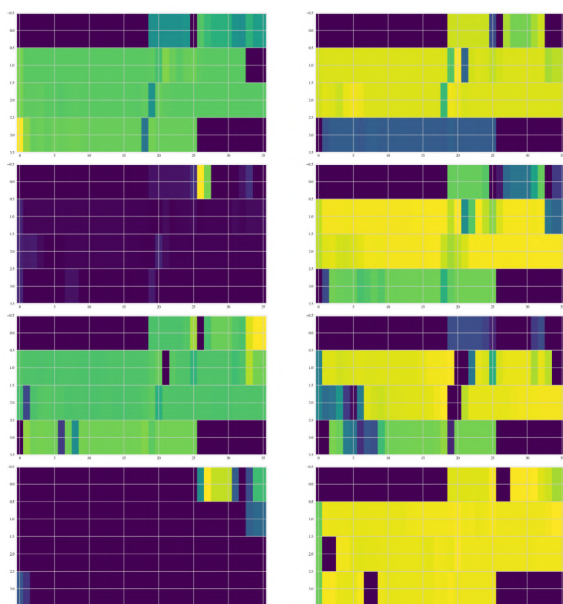
Anexo G – Figura 22 – Vizgrama de entrada



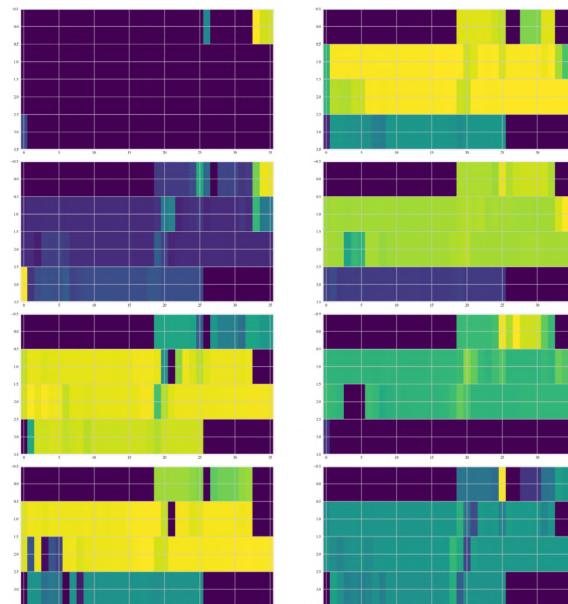
Anexo G – Figura 23 – Feature maps de la primera capa convolucional (filtro 1 a 4)
vizgrama con kitchenapp1 como input



Anexo G – Figura 24 – Feature maps de la primera capa convolucional (filtro 5 a 8) vizgrama con *kitchenapp1* como input



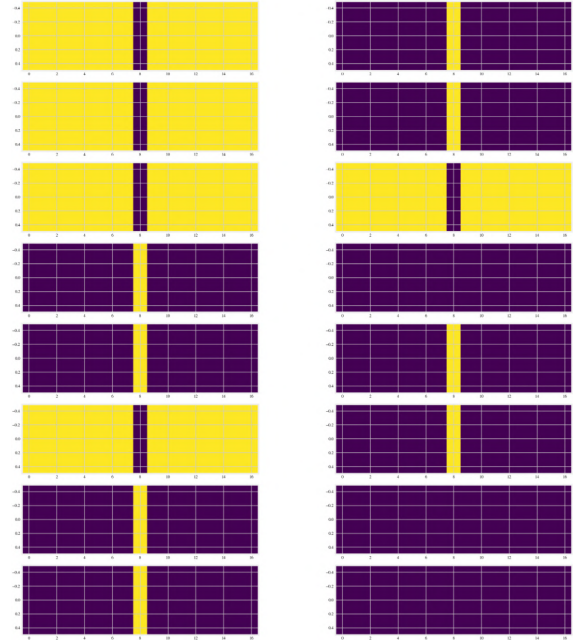
Anexo G – Figura 25 – Feature maps de la segunda capa convolucional (filtro 1 a 8) – vizgrama con *kitchenapp1* como input



Anexo G – Figura 26 – Feature maps de la segunda capa convolucional (filtro 9 a 16) – vizgrama con *kitchenapp1* como input



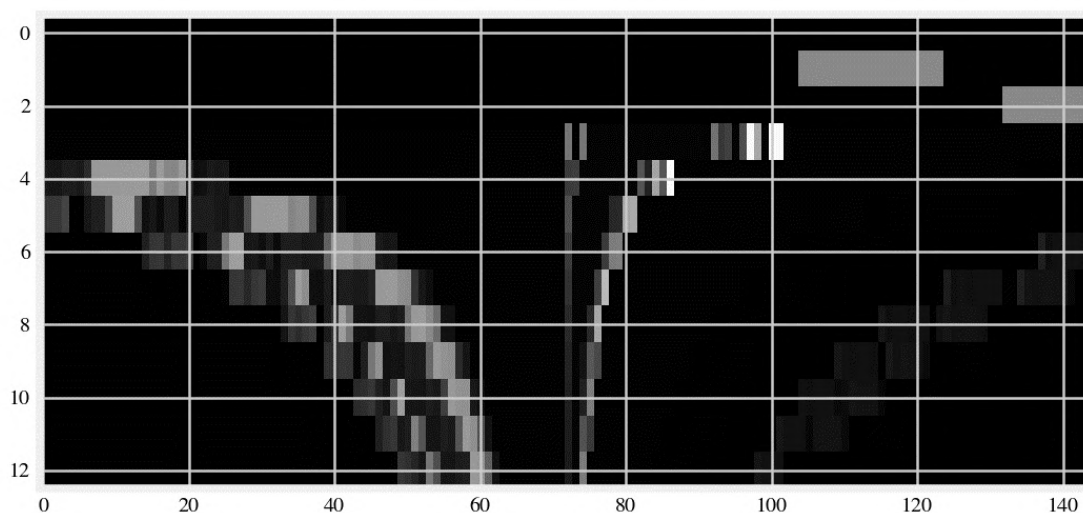
Anexo G – Figura 27 – Feature maps de la tercera capa convolucional (filtro 1 a 16) – vizgrama con kitchenapp1 como input



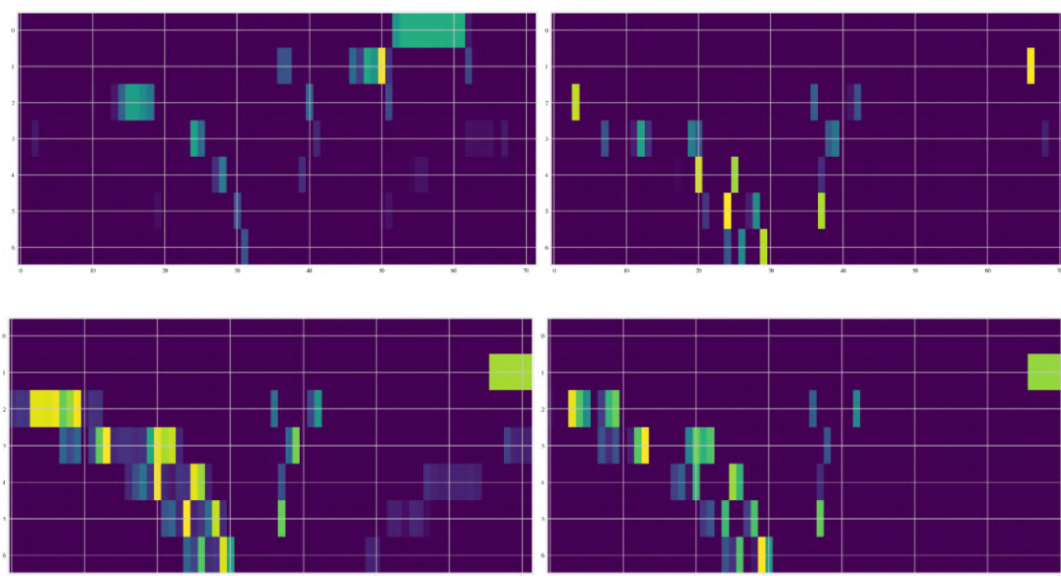
Anexo G – Figura 28 – Feature maps de la tercera capa convolucional (filtro 17 a 32) – vizgrama con kitchenapp1 como input

MICROWAVE1

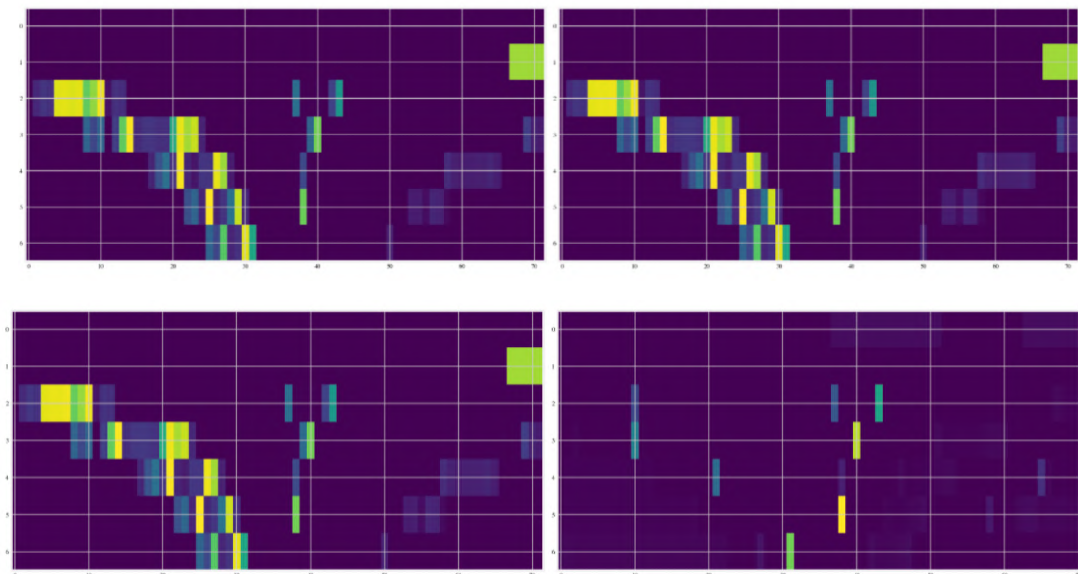
Only microwave1 activated



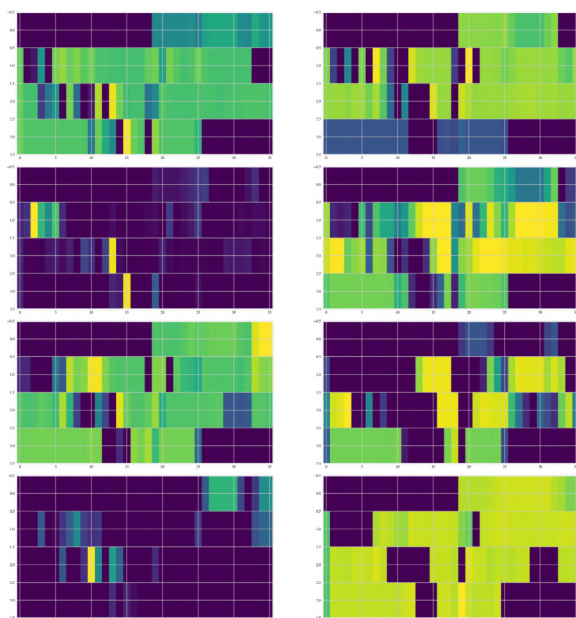
Anexo G – Figura 29 – Vizgrama de entrada



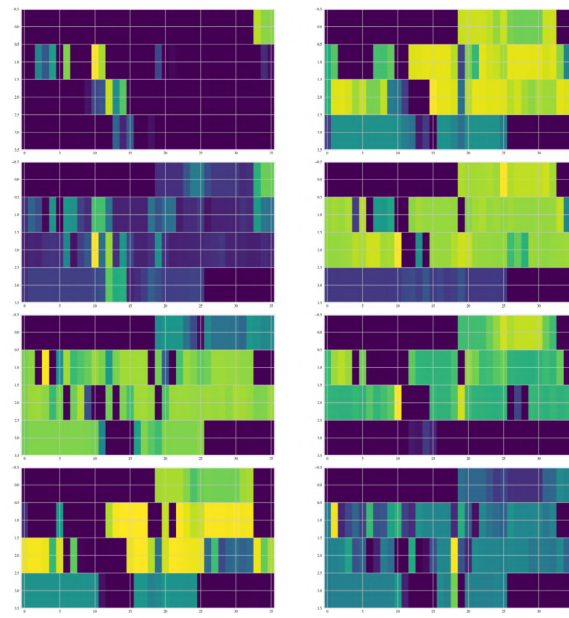
Anexo G – Figura 30– Feature maps de la primera capa convolucional (filtro 1 a 4)
vizgrama con **microwave1** como input



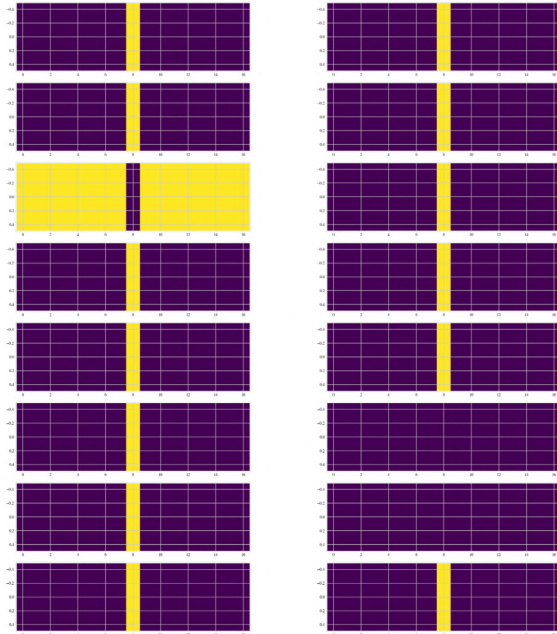
Anexo G – Figura 31 – Feature maps de la primera capa convolucional (filtro 5 a 8) vizgrama con microwave1 como input



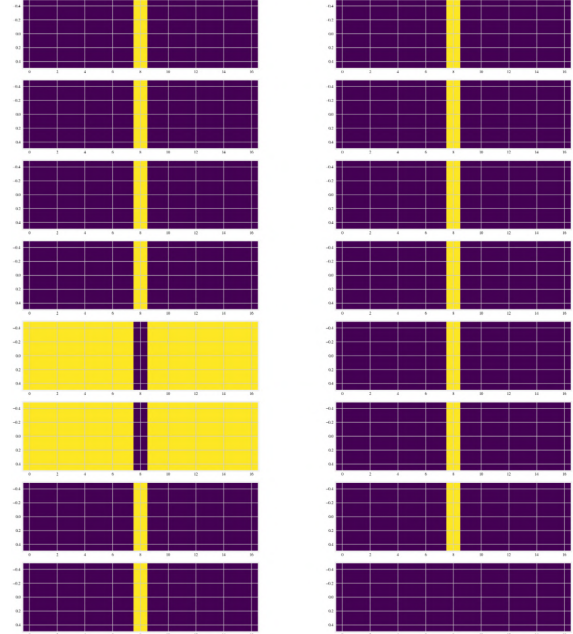
Anexo G – Figura 32 – Feature maps de la segunda capa convolucional (filtro 1 a 8) – vizgrama con microwave1 como input



Anexo G – Figura 33 – Feature maps de la segunda capa convolucional (filtro 9 a 16) – vizgrama con microwave1 como input



Anexo G – Figura 34 – Feature maps de la tercera capa convolucional (filtro 1 a 16) – vizgrama con microwave1 como input



Anexo G – Figura 35 – Feature maps de la tercera capa convolucional (filtro 17 a 32) – vizgrama con microwave1 como input