UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENT DE SISTEMES INFORMÀTICS I COMPUTACIÓ

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Sistemas Informáticos y Computación

Diseño, experimentación y puesta en producción de un sistema de detección de anomalías para la industria

Trabajo Fin de Máster

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

AUTOR/A: Fornés Gabaldón, Héctor

Tutor/a: Martínez Hinarejos, Carlos David

Cotutor/a externo: MARTINEZ MORET, LORENZO

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Diseño, experimentación y puesta en producción de un sistema de detección de anomalías para la industria

TRABAJO FIN DE MÁSTER

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor: Héctor Fornes Gabaldón
Tutor: Carlos David Martínez Hinarejos
Cotutor: Lorenzo Martínez Moret

Curso 2021-2022

Resumen

El objetivo de este trabajo es desarrollar un sistema de detección de anomalías para la industria, dentro del marco de mi puesto como trabajador en la empresa NUNSYS SA. Se cubrirán todas las fases del proyecto, desde la adquisición de los datos, pasando por su análisis y preprocesado, para seguir con la experimentación de diferentes técnicas de detección de anomalías y finalmente la puesta en producción del sistema en las instalaciones de un cliente real. Se pondrá especial énfasis en la evaluación y comparación de los diferentes enfoques utilizados durante el desarrollo del proyecto, los cuales se corresponden con los modelos neuronales que definen ahora mismo el estado del arte.

Palabras clave: Detección de anomalías, Industria 4.0, Mantenimiento predictivo

Resum

L'objectiu d'aquest treball és desenvolupar un sistema de detecció d'anomalies per a la indústria dins del marc del meu lloc com a treballador a l'empresa NUNSYS SA. Es cobriran totes les fases del projecte, des de l'adquisició de les dades, passant per la seva anàlisi i preprocessament, per seguir amb l'experimentació de diferents tècniques de detecció d'anomalies i finalment la posada en producció del sistema a les instal·lacions d'un client real. Es posarà especial èmfasi en l'avaluació i la comparació dels diferents enfocaments utilitzats durant el desenvolupament del projecte, els quals es corresponen amb els models neuronals que defineixen ara mateix l'estat de l'art.

Paraules clau: Detecció d'anomalies, Indústria 4.0, Manteniment predictiu

Abstract

The objective of this thesis is to develop an anomaly detection system for the industry, within the framework of my position as a worker in the company NUNSYS SA. All phases of the project will be covered, from the acquisition of the data, through its analysis and pre-processing, to continue with the experimentation of different anomaly detection techniques and finally the deployment of the system in the facilities of a real client. Special emphasis will be placed on the evaluation and comparison of the different approaches used during the development of the project, which correspond to the neural models that currently define the state of the art.

Key words: Anomaly detection, Industry 4.0, Predictive maintenance

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En los últimos años, la importancia de los datos en la industria ha ido creciendo hasta llegar a considerarse uno de sus principales activos, útiles tanto para asegurar la calidad y la seguridad de los sistemas como para mejorar y optimizar los procesos, e incluso para descubrir nuevas necesidades y tendencias en el mercado. Aquellas industrias que sepan sacarle partido a sus datos serán más eficientes, reducirán sus costes de producción y podrán anticiparse a sus competidores [1].

Teniendo en cuenta todas estas ventajas, no hay industria que no haya comenzado a sensorizar e interconectar los dispositivos de sus instalaciones, registrando más y más datos con la esperanza de extraer de ellos información de utilidad. Ahora bien, esta cantidad ingente de datos en crudo, proveniente de sensores y máquinas de diferentes tipos, conlleva una complejidad que impide que puedan ser tratados por humanos o por métodos basados en reglas de menor o mayor sofisticación [2]. Es por ello que durante los últimos años se ha dedicado un gran esfuerzo e inversión en la investigación de técnicas estadísticas y basadas en redes neuronales que consigan procesar y analizar estos datos de forma automática y utilizar la información extraída con diversos objetivos como, por ejemplo, el mantenimiento predictivo (que trata de predecir cuándo una máquina fallará para así anticiparse al problema y solventarlo antes de que sea demasiado tarde) o la detección de anomalías, que es la pieza central de este trabajo y que se define a continuación.

La detección de anomalías consiste en identificar observaciones que se desvían significativamente de la mayoría del resto de datos y que, por tanto, no corresponden a un comportamiento normal del objeto de la medición, dando pie a sospechar que ha habido algún tipo de alteración en el sistema. Por ejemplo, en el ámbito de la ciberseguridad, estas alteraciones podrían deberse a la intrusión a una red privada por parte de un ente ajeno, o una compra fraudulenta en un *ecommerce*. Sin embargo, este trabajo se centra en la detección de anomalías en la industria, y en este caso suelen venir provocadas por la propia naturaleza de las máquinas, las cuales se van deteriorando y dañando con el paso del tiempo debido a un uso intensivo y continuado de las mismas. Estos defectos ocasionan variaciones en las mediciones que, si son detectadas a tiempo, pueden prevenir efectos colaterales adversos, paradas del sistema y pérdidas millonarias [1].

Lamentablemente, la tarea no es tan sencilla como podría parecer a simple vista. En primer lugar, y como se ha comentado anteriormente, los sensores son de muy diversos tipos, y almacenan una cantidad de datos tan grande que hacen imposible temporal y económicamente que la tarea pueda ser llevada a cabo por expertos humanos. Además, y por los mismos motivos, los métodos basados en umbrales, reglas y alarmas no ofrecen una capacidad de modelado lo suficientemente potente, lanzando muchos falsos positivos y/o pasando por alto muchos verdaderos positivos, llegando a causar pérdidas aún

mayores que si dichos métodos no se utilizasen [2]. Por si esto fuera poco, debido a los requisitos cambiantes de la industria a lo largo del año e incluso del día, las máquinas operan bajo diferentes perfiles operacionales, alternando etapas de gran intensidad con otras de menos estrés; y aunque estas fases pueden darse con cierta estacionalidad, ésta no siempre se cumple.

Con el objetivo de combatir toda esta complejidad y realizar una detección de anomalías automática y eficaz, en este trabajo se analizan varias técnicas que han demostrado su valía en procesos industriales. En el resto de la memoria, se hace un repaso a los conceptos y técnicas con los que se ha experimentado, y se realiza un estudio más completo de la solución final implantada por parte de la empresa Nunsys SA [1] en la planta de producción de uno de sus clientes.

1.1 Motivación

La motivación del presente trabajo nace de un cliente de la empresa Nunsys SA a la cual pertenezco como trabajador. Este cliente tiene la necesidad de desarrollar un sistema automático de detección de anomalías para utilizarlo en las máquinas de sus plantas de producción.

1.2 Objetivos

El objetivo general del proyecto es satisfacer la necesidad de dicho cliente, y de forma más concreta se puede subdividir en varios subobjetivos.

El primero de ellos, es realizar un estudio de los diferentes algoritmos para la detección automática de anomalías en series de datos temporales, preferiblemente aquellos más novedosos que representan ahora mismo el estado del arte.

En segundo lugar, y aplicando los conocimientos adquiridos tras la fase de estudio, está el objetivo de implementar las técnicas más prometedoras, para después realizar una comparativa entre ellas y así poder concluir cuál ofrece un mejor rendimiento.

Por último, para que el proyecto sea satisfactorio, se debe implantar la solución en las instalaciones del cliente, y realizar los ajustes necesarios para que la solución final cumpla con los requisitos pactados.

1.3 Estructura de la memoria

La memoria está estructurada en capítulos, los cuales a su vez pueden estar divididos en secciones y subsecciones. A continuación, se explica brevemente cuál es el contenido de cada uno de los capítulos. Cabe destacar que esta estructura también queda reflejada en el índice de la memoria.

Se comienza con una recopilación de los conceptos fundamentales revisados durante la fase de estudio, y necesarios para entender los siguientes capítulos de implementación y experimentación. Entre los conceptos estudiados se encuentra la propia definición de *anomalía*, así como el fundamento teórico de las topologías de redes neuronales utilizadas.

[1] https://www.nunsys.com

Más adelante, se realiza una descripción del conjunto de datos proporcionado por el cliente, con el objetivo de mostrar cuál es la estructura de la información recogida por los sensores instalados en las máquinas de sus plantas de producción.

Seguidamente, se detallan los aspectos más relacionados con la implementación de las arquitecturas seleccionadas, para después realizar una comparativa de las mismas mediante una serie de experimentos.

Por último, se hace una conclusión de las lecciones aprendidas durante el proyecto, así como una recopilación de posibles extensiones del mismo que pueden acometerse en el futuro.

CAPÍTULO 2
Conceptos fundamentales

En este capítulo, se repasan los conceptos que se han estudiado y las técnicas con las que se ha experimentado durante el proceso de construcción de un sistema de detección de anomalías. En cuanto a los conceptos, se incluyen la propia definición de anomalía y los desafíos que conlleva su detección en el mundo de la industria. En cuanto a las técnicas, se explican diferentes arquitecturas de redes neuronales profundas que se consideraron prometedoras, ya que gracias a su capacidad de realizar tareas de aprendizaje *Seq2Seq*, hoy en día representan el estado del arte.

Debido a los buenos resultados obtenidos con estas arquitecturas más potentes, no se ha incluido una explicación tan exhaustiva de otras técnicas más clásicas que se han aplicado satisfactoriamente en otros dominios, pero que presentan ciertas desventajas relacionadas con la especificación de parámetros, la interpretabilidad, la generalización o el coste computacional. Ejemplos de estas técnicas son el *clustering* [3], los vecinos más cercanos [4], los sistemas expertos [5] y la reducción de dimensionalidad [6].

Tal y como se ha mencionado anteriormente, la detección y predicción de anomalías a partir de series de datos temporales y con múltiples variables son tareas críticas en los procesos industriales actuales. Pero no lo son únicamente en este sector, sino que lo son en otros tan variados como por ejemplo el financiero, el de seguros, o el aeroespacial [7]. Es por ello que las referencias sobre detección de anomalías son abundantes en la literatura sobre ciencia de datos. En cada uno de siguientes apartados se aprovecha para mencionar otros trabajos que se han encontrado interesantes en relación a los respectivos conceptos y técnicas.

2.1 Definición de anomalía

Teniendo en cuenta los muchos intentos de definir la naturaleza de los datos anómalos en la literatura científica, una definición general en el contexto de la industria 4.0 podría ser que los datos anómalos son las consecuencias medibles de un cambio de estado inesperado de un sistema, que se sale fuera de su norma local o global [8]. Esta definición puede desestructurarse en una serie de importantes observaciones que reflejan la naturaleza de este tipo de datos. En primer lugar, la mayoría de los datos capturados por los sensores de una máquina pueden ser considerados "normales", ya que representan las características de operativa habituales de dicha máquina. En segundo lugar, el concepto de operativa normal de un sistema puede variar a lo largo del tiempo por diversos motivos. Y en tercer y último lugar, los datos generados por estos sensores únicamente representan una parte de los procesos que gobiernan la máquina siendo monitoreada.

2.1.1. Tipología

En cuanto a los tipos de anomalías, también cabe destacar varios de ellos. Por un lado, están las anomalías puntuales (figura 2.1), que corresponden a la definición dada por Hawkins de que "un dato anómalo es una observación que se desvía tanto del resto de observaciones como para crear la sospecha de que fue creada por un mecanismo generador diferente" [9]. En el ámbito de las series de datos temporales, este tipo de anomalías se caracteriza por volver al estado normal previo tras un corto periodo de tiempo, y pueden corresponder a simplemente ruido causado, por ejemplo, por un desajuste en el sensor, o realmente a un evento que, aunque corto en el tiempo, pueda ser interesante estudiar más detenidamente. Por otro lado, están las anomalías contextuales (figura 2.2), las cuales son observaciones o secuencias que, si se miran de forma aislada, parecen dentro de los valores esperados para esa señal, pero que si se miran teniendo en cuenta el contexto de las observaciones vecinas, es decir, las de los instantes anteriores y posteriores en el contexto de las series de datos temporales, sí que se desvían de los patrones o la norma esperados. Por último, también existen las anomalías colectivas (figura 2.3), que corresponden a una colección de observaciones que, si se miran de forma individual pueden considerarse como no anómalas, pero que si se hacen de forma grupal sí que levantan sospechas.

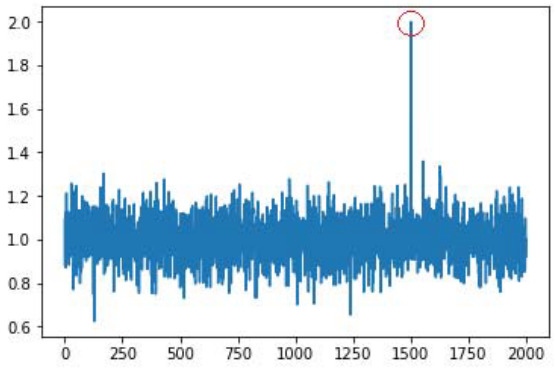

Figura 2.1: Anomalía puntual (rodeada en rojo) en un ruido aleatorio Gaussiano. Extraída de [8].

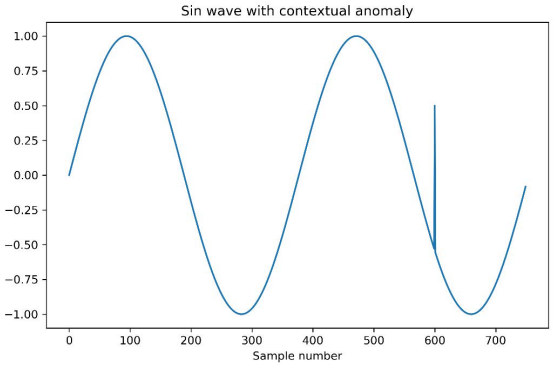

Figura 2.2: Anomalía contextual en la que el valor anómalo en el instante 600 es el mismo que en otros instantes, pero dentro de su contexto es anómalo. Extraída de [8].

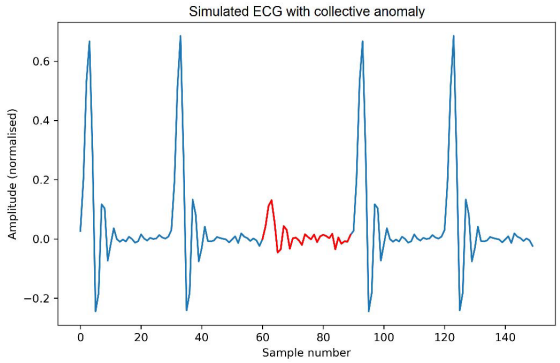

Figura 2.3: Anomalía colectiva (marcada en rojo) en una serie de datos temporales simulada. Extraída de [8].

2.1.2. Detección de anomalías

Una vez definido el concepto de anomalía, se puede pasar a estudiar el por qué su detección automática es tan importante en el ámbito de la industria. Como se ha comentado en la introducción, el principal motivo es que la cantidad de datos recogidos por los sensores, cada vez más abundantes en las plantas de producción, es tan grande que su tratamiento por parte de humanos o sistemas basados en reglas simples no es factible. Dentro de este contexto, la detección de anomalías es el proceso por el cual, a partir del histórico de datos recogidos por los sensores de un sistema, se modela mediante técnicas de *machine learning* el funcionamiento normal de dicho sistema, para así poder identificar de forma automática cuándo los nuevos datos recogidos no se corresponden a los patrones que podrían considerarse dentro de lo normal. Sin embargo, aunque esta tarea puede parecer sencilla a simple vista, existen una serie de elementos que la convierten en una tarea realmente desafiante, los cuales se analizan a continuación.

2.1.3. Primer desafío

El primero de estos desafíos viene de la mano de la propia naturaleza de las series de datos temporales recogidos por sensores, que presentan una serie de características que pueden afectar a los algoritmos de detección de anomalías. Estos sensores son capaces de incluir en sus mediciones información contextual sobre el ambiente que les rodea, los cuales pueden verse afectados por factores externos. Por ejemplo, la temperatura del interior de unas instalaciones se ve influida por la temperatura y condiciones atmosféricas en general del exterior, por lo que únicamente tener en cuenta la temperatura del interior sin prestar atención a estos factores externos puede provocar la interpretación errónea de los datos. Por este motivo, se considera que la información contextual puede enriquecer la capacidad de un algoritmo de detección de anomalías a la hora de identificar correctamente las observaciones que no se conforman al comportamiento normal, pero esto incrementa la complejidad del proceso y es importante seleccionar bien qué información contextual es realmente necesaria a la hora de diseñar la estructura de dichos sensores.

Otra característica de este tipo de datos es su dimensionalidad, que suele ser multivariable. Esto implica que a la hora de interpretar la observación de un sensor en concreto no es solo importante su propio histórico, sino que también lo es su relación con el resto de mediciones tomadas en el mismo instante por otros sensores; por tanto, puede ser necesario tenerlas en cuenta de forma combinada. También hay que atender al ruido, presente de forma inherente en los sistemas del mundo real, y que representa fluctuaciones en los valores obtenidos que no son significativas a la hora de evaluar la estructura de

los datos en su conjunto. Pueden ser ocasionadas por variaciones en la sensibilidad del sensor, o por errores en el proceso de transmisión de los datos. Sin embargo, en ocasiones un cambio en la distribución del propio ruido sí puede representar un evento significativo, por lo que podría ser importante entender la naturaleza y la causa de dicho ruido. Esto hace que pueda no ser conveniente aplicar técnicas de reducción del ruido como sí se hace en muchos otros campos.

Como última característica de las series de datos temporales recogidas por los sensores habituales en la industria 4.0, se encuentra su estacionalidad, y es que los procesos industriales se ven sometidos a cambios cíclicos repetidos en el tiempo, como por ejemplo podría ser una reducción de la potencia en el turno de noche. Por este motivo, resulta de suma importancia que los métodos de detección de anomalías sean capaces de adaptarse a dichos cambios en la estructura de datos en despliegues pensados para el largo plazo, pues una observación que se consideró como anómala en el pasado puede ser vista como un dato normal en un momento posterior, atendiendo a un cambio en el estado de funcionamiento del sistema.

2.1.4. Segundo desafío

Como segundo desafío que los algoritmos de detección de anomalías deben afrontar, está el hecho de que la mayor parte de las veces se cuenta con mucha información histórica sobre el estado operativo normal de una máquina, pero con muy pocos datos, o incluso ninguno, que representen anomalías. Es por ello que los métodos más tradicionales de entrenamiento supervisado son de difícil aplicación sobre estos conjuntos de datos. Para combatir este desequilibrio, pueden utilizarse técnicas de *resampling* [10], reduciendo la cantidad de datos normales, introduciendo copias de anomalías conocidas, o creando datos anómalos de forma sintética, aunque si no se usan de forma correcta pueden conducir a problemas de *under* u *overfitting*. Otra técnica que puede utilizarse para reducir el impacto de este problema es el *transfer learning* [11], pues el conocimiento adquirido a partir de los datos de una máquina en concreto puede ser usado en máquinas del mismo tipo, o incluso adaptado a sistemas y entornos semejantes. En una sección posterior se profundizará en este tema.

Sin embargo, para la mayoría de los casos, son los métodos no supervisados o semisupervisados los que mejor combaten esta falta de conocimiento *a priori*, ya que únicamente necesitan datos de operativa normales para ser entrenados, siendo capaces de detectar cuándo un nuevo dato se sale de estos patrones normales, marcando dicha observación como anómala [8]. Otra ventaja implícita de este enfoque frente a los métodos supervisados es que permite identificar anomalías de distintos tipos a las que se habían podido etiquetar en el pasado, incluso de tipos que nunca habían sido vistos con anterioridad.

2.1.5. Tercer desafío

El tercero de los desafíos tiene que ver con las restricciones de tiempo y recursos que suele implicar el uso de sensores a gran escala. Cuando el procesamiento de los datos se hace de forma centralizada en un dispositivo en red con mayor capacidad, existe una latencia debido a la necesidad de transmitir dichos datos por la red, suponiendo esto un reto para los sistemas que requieren un tiempo de reacción minúsculo para ser de utilidad [12]. En estos casos, puede optarse por realizar el procesamiento en los propios dispositivos de recolección de datos, pero entonces aparecen restricciones de recursos, ya que estos dispositivos cuentan con una capacidad de procesamiento muy reducida. Otro problema derivado del uso de enormes cantidades de sensores es que la magnitud de datos que recogen es aún mayor, y pretender almacenar todo el histórico de dichos datos

puede ser inviable. Para tratar de solucionar esto, pueden mantenerse únicamente los datos más recientes, pero es necesario entonces que los modelos sean capaces de recordar de algún modo las tendencias y patrones del pasado más lejano para que las prestaciones del sistema sean las adecuadas.

2.1.6. Métodos de obtención de información

Para concluir esta sección, se describen a continuación los métodos de obtención de información más utilizados por los sistemas de detección de anomalías [13, 14]. El primero de ellos consiste en una puntuación de anomalía, es decir, un valor que representa el grado con el que una observación se desvía del valor esperado proporcionado por el modelo de detección. A la hora de analizar estas puntuaciones, la persona encargada de dicha tarea puede establecer un umbral para diferenciar entre anomalía y no anomalía, o considerar únicamente las N predicciones con las puntuaciones más altas. El otro tipo de obtención de información es aquel en el que el propio sistema es el encargado de etiquetar un dato como anómalo o no, normalmente en base a un umbral que puede ser parametrizado. Este tipo de obtención de información es el más utilizado en sistemas que requieren notificación inmediata, al no requerir una fase de análisis posterior, y también permiten la clasificación de diferentes niveles de anomalías, con mayor o menor gravedad, y que pueden desencadenar el lanzamiento de alertas más o menos críticas.

2.2 Generalidades de redes neuronales y su entrenamiento

En esta sección, se repasan los conceptos fundamentales que componen un red neuronal, así como los principales parámetros con los que se puede experimentar durante su entrenamiento para conseguir el mejor rendimiento posible.

En primer lugar, para comprender de forma intuitiva cómo se estructura una red neuronal, se puede pensar en ellas como una secuencia de capas, conteniendo cada una de esas capas una o más neuronas. La capa de entrada es la que recoge los datos usados para el entranamiento, y la de salida es la que se encarga de entregar los resultados. Las capas intermedias se conocen como ocultas, y es en ellas donde tiene lugar el ajuste de pesos durante el entrenamiento que consigue que la red vaya aprendiendo a cumplir con su objetivo. En la figura 2.4 se puede ver una representación visual de esta estructura de capas.

Cabe destacar que este aprendizaje se realiza poco a poco, en un proceso iterativo que va recorriendo los datos de entrenamiento una y otra vez. Cada una de estas pasadas a través de los datos de entrenamiento se conoce como *epoch*. Cuantos más *epochs* se ejecuten, más tiempo le estamos dando a la red para aprender. Sin embargo, esto puede ser contraproducente si se hace *overfitting*, es decir, que la red haya aprendido a la perfección los datos de entrenamiento, pero no sea capaz de generalizar sobre datos que no ha visto nunca como los de validación y *testing*. El efecto contrario es el *underfitting*, que es un aprendizaje pobre durante el entrenamiento que se traduce en un mal rendimiento.

A continuación se introduce un parámetro con el que se ha experimentado en el capítulo 5, y que junto al número de *epochs*, puede ayudar a evitar las situaciones de *overfitting* y *underfitting* descritas anteriormente. Se trata del *learning rate*, el cual determina la velocidad de aprendizaje, indicándole al algoritmo optimizador cuánto debe avanzar en cada actualización. Si se usa un número demasiado grande, la búsqueda del mínimo glo-

[1] https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a

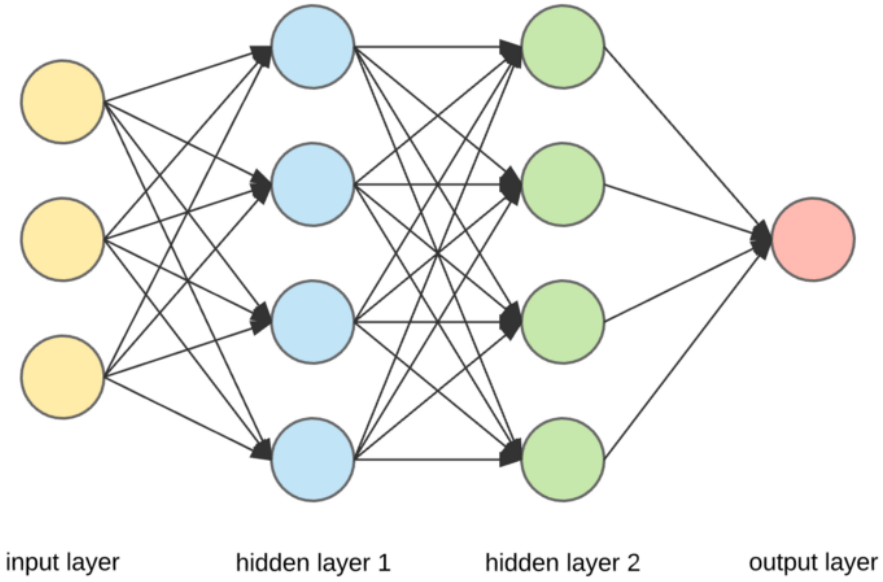

Figura 2.4: Arquitectura de una red neuronal. [1]

bal va dando muchos bandazos y puede que nunca se llegue a encontrar. Al contrario, si el *learning rate* es demasiado pequeño, puede que la búsqueda se quede atrapada en un mínimo local, o que el número de *epochs* sea insuficiente para alcanzar el mínimo global.

El objetivo de esta sección es únicamente introducir los parámetros que se han utilizado en la fase de experimentación 5. Existen muchos más componentes y parámetros que describen una red neuronal y su proceso de entrenamiento que no se han tratado aquí. El libro de C. Bishop [26] es un clásico que explica todos estos conceptos de forma excelente.

2.3 Redes *Long short term memory* (LSTM)

LSTM es una tipología de red neuronal recurrente (RNN) que permite retener dependencias de largo alcance entre secuencias de una serie de datos temporales [15]. Su estructura consiste en una concatenación de módulos, y cada uno de estos módulos incluye a su vez tres puertas de control: la puerta de olvido, la puerta de entrada y la puerta de salida. Cada una de estas puertas está compuesta por una capa sigmoide y una operación de multiplicación. La salida de las capas sigmoides está en el intervalo [0, 1], y este número representa el porcentaje de información que deben dejar pasar. Al tratarse de una tipología RNN, la red LSTM se nutre de una secuencia de vectores M-dimensionales de entrada que representan una serie temporal, correspondiendo cada vector a un instante en el tiempo. Normalmente, se considera el escenario en el que múltiples series de datos son obtenidas aplicando una ventana a una serie de datos temporales más grande.

A continuación, se explica de forma más específica el comportamiento de un módulo LSTM, el cual queda reflejado en la figura 2.5. Asumiendo que x_t representa el vector de entrada en el instante t, en primer lugar, la red LSTM decide qué parte de la información del estado anterior h_{t-1} debe ser olvidada, multiplicando dicha salida h_{t-1} por la salida de la puerta de olvido, que es un número en el intervalo [0, 1]. Este cómputo representa el valor f_t en la figura 2.5, y responde a la siguiente fórmula:

$$f_t = \sigma_1 \left(W_f \left[h_{t-1}, x_t \right] + b_f \right),$$

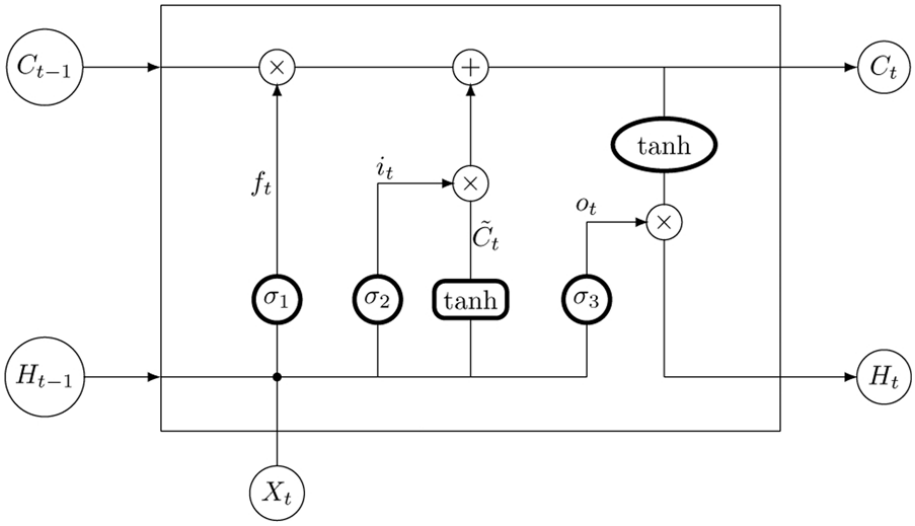

Figura 2.5: Un módulo de una red LSTM. Extraído de [7]

.

donde W_f y b_f son la matriz de pesos y el *bias* (término independiente o umbral) de la puerta de olvido. Seguidamente, y antes de almacenar el resultado en el estado de la celda, x_t es procesada por la puerta de entrada, la cual determina el valor i_t junto con un vector de valores candidatos $\widetilde{C}_t$ generado por una capa tangente hiperbólica (tanh):

$$i_t = \sigma_2 \left(W_i \left[h_{t-1}, x_t \right] + b_i \right),$$

$$\widetilde{C}_t = \tanh \left(W_c \left[h_{t-1}, x_t \right] + b_c \right),$$

donde (W_i, b_i) y (W_c, b_c) son las matrices de pesos y los *biases* de las puertas de entrada y la memoria del estado de la celda, respectivamente. Ambos resultados son utilizados, junto con el resultado de la puerta de olvido y el estado de la celda en el instante anterior C_{t-1}, para generar el estado de la celda en el instante actual C_t, aplicando la siguiente fórmula:

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}_t.$$

Por último, la puerta de salida queda definida por:

$$o_t = \sigma_3 \left(W_o \left[h_{t-1}, x_t \right] + b_o \right),$$

donde W_o y b_o son la matriz de pesos y el *bias* de dicha puerta de salida, la cual finalmente determina qué parte del estado de la celda debe ser devuelto por el módulo LSTM. Dicha salida corresponde a h_t en la figura 2.5, y su fórmula es:

$$h_t = o_t \cdot \tanh \left(C_t \right).$$

Cabe destacar que existen múltiples variantes de la tipología LSTM, y la que aquí se ha revisado es la correspondiente a la descrita en [15].

2.4 Redes *Autoencoder*

La tipología de red neuronal *Autoencoder* tiene como objetivo aprender, de forma no supervisada, el mejor esquema *encoding-decoding* que, a partir de unos datos, y pasando por un espacio de representación más reducido, sea capaz de reconstruir los mismos datos de nuevo [7]. Más concretamente, consiste en una capa de entrada, una capa de salida, una red *encoder*, una red *decoder* y un espacio latente. En cuanto a su funcionamiento, cuando el *Autoencoder* es alimentado con unos datos, el *encoder* comprime dichos datos en el espacio latente, y seguidamente el *Decoder* descomprime dicha representación comprimida y la envía a la puerta de salida. Dichos datos de salida se comparan con los datos proporcionados en la entrada y el error es propagado hacia atrás a través de la arquitectura para actualizar los pesos de la red. La estructura aquí descrita puede verse en la figura 2.6.

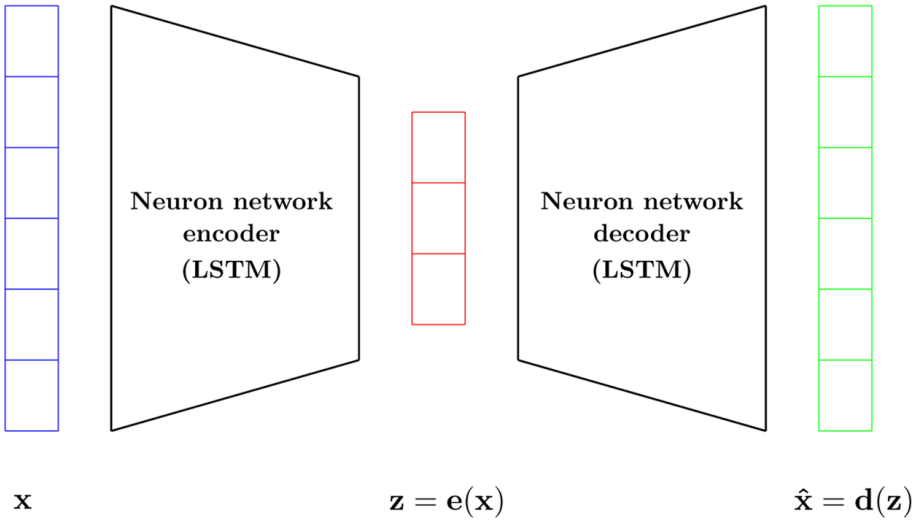

Figura 2.6: Red LSTM *Autoencoder*. Extraída de [7].

Aunque se ha comentado anteriormente, merece la pena destacar que el *Autoencoder* no se limita a copiar los datos de entrada a la salida, sino que, gracias a utilizar un espacio de representación con una dimensión menor que la de los propios datos, la red es forzada a aprender las características más importantes y quedarse únicamente con ellas. En otras palabras, el *Autoencoder* es capaz de reducir la dimensionalidad de los datos a la vez que mantiene la mayor parte de la información estructural de los mismos.

2.5 *Generative Adversarial Networks* (GAN)

Las *Generative Adversarial Networks* (GAN) se encuentran dentro de la categoría los algoritmos de *machine learning* no supervisados. En su concepción inicial [16], su principal objetivo es generar imágenes realistas. Para conseguirlo, se utilizan dos redes, una conocida como generador y otra como discriminador, las cuales compiten entre ellas durante la fase de entrenamiento, de manera que la primera trata de generar una imagen, mientras la segunda se encarga de decidir si esa imagen generada es real o falsa. El generador suele consistir en un *Autoencoder*, cuyo objetivo, como se ha mencionado anteriormente, es modelar los datos de entrada en un espacio latente más reducido que consiga capturar la distribución de los datos reales, para luego ser capaz de reconstruir la entrada original a partir de dicha representación intermedia. En cuanto al discriminador, normalmente

tiene una arquitectura de clasificación típica, es decir, lee una imagen de entrada y debe determinar si es o no es real.

Sin embargo, aunque las GAN han demostrado un buen rendimiento en tareas relacionadas con la visión por computador, incluida la detección de anomalías en imágenes, algunos investigadores empezaron a usarlas sobre series de datos temporales. Más concretamente, y en lo que se refiere a este trabajo, es de interés su aplicación para detección de anomalías basada en la reconstrucción de series temporales [17]. La idea detrás de este enfoque es aprender un modelo que pueda codificar un segmento de los datos temporales, y luego decodificarlo o, lo que es lo mismo, reconstruirlo, para obtener de nuevo la entrada original. De esta manera, un modelo efectivo no debería ser capaz de reconstruir segmentos con presencia de anomalías tan fielmente como lo haría con un segmento normal, ya que las anomalías provocan pérdida de información durante la fase de codificación.

CAPÍTULO 3
Descripción de los datos

Para poder realizar las pruebas e implementar un modelo de detección de anomalías adaptado a los requisitos específicos del cliente, éste mismo proporcionó un *dataset* en formato CSV con los datos recogidos por varios sensores instalados en una de las máquinas de sus instalaciones. En concreto, el fichero contiene información sobre la vibración de la máquina en los ejes X, Y y Z, la temperatura en tres puntos diferentes y la corriente eléctrica, contando en total con siete canales de información. En cuanto al tamaño de dicho *dataset*, recoge información desde el 22 de enero de 2020 hasta el 5 de febrero de 2020, con mediciones registradas cada segundo. Una vez recibidos estos datos se procedió a realizar un análisis exploratorio de los mismos, cuyos pasos y conclusiones se describen a continuación.

Utilizando las librerías *numpy* y *pandas* del lenguaje de programación *Python*, se cargaron los datos del fichero CSV en un *DataFrame*. La descripción estadística básica del *dataset* puede verse en la tabla 3.1.

	mean	std	min	25 %	50 %	75 %	max
VIB_X	1.96	0.40	$5.15 \cdot 10^{-1}$	1.68	1.95	2.24	3.78
VIB_Y	$1.01e+01$	5.09	$9.18 \cdot 10^{-2}$	$1.03 \cdot 10^{1}$	$1.22 \cdot 10^{1}$	$1.34 \cdot 10^{1}$	$1.98e+01$
VIB_Z	9.98	5.07	$9.34 \cdot 10^{-2}$	10.00	$1.21 \cdot 10^{1}$	$1.33 \cdot 10^{1}$	$1.89 \cdot 10^{1}$
TEMP_1	$1.78 \cdot 10^{1}$	0.55	$1.56 \cdot 10^{1}$	$1.75 \cdot 10^{1}$	$1.79 \cdot 10^{1}$	$1.82 \cdot 10^{1}$	$1.98 \cdot 10^{1}$
TEMP_2	$2.53 \cdot 10^{1}$	9.97	$1.60 \cdot 10^{1}$	$1.79 \cdot 10^{1}$	$1.88 \cdot 10^{1}$	$2.92 \cdot 10^{1}$	$5.46 \cdot 10^{1}$
TEMP_3	$6.08 \cdot 10^{1}$	4.00	6.00	$5.83 \cdot 10^{1}$	$6.08 \cdot 10^{1}$	$6.33 \cdot 10^{1}$	$1.00 \cdot 10^{2}$
CORR_4	5.49	0.34	$4.27 \cdot 10^{1}$	5.30	5.49	5.68	$1.00 \cdot 10^{1}$

Tabla 3.1: Descripción estadística de los datos: *mean* es la media, *std* la desviación típica, *min* hace referencia al valor mínimo, *max* al valor máximo, y *25 %*, *50 %* y *75 %* a los cuartiles 1, 2 y 3 respectivamente.

Como curiosidad, las mediciones de temperatura están en una escala de 0 a 100 y las de corriente en una escala de 0 a 10. Se desconocen las correspondencias entre dichas escalas y los valores de temperatura y corriente en sus unidades de medida originales, pero como a la hora de entrenar los modelos todos los canales se han escalado entre -1 y 1, esto no resulta preocupante.

El siguiente paso fue analizar el *dataset* para detectar la posible presencia de valores nulos. La librería *pandas* tiene una función *isnull* que puede utilizarse justamente con esta finalidad. Resulta que únicamente se encontró un valor nulo, correspondiente a la última medición del sensor de corriente. Se procedió a sustituir dicho valor nulo por el valor

del *timestamp* anterior, ya que al no existir mediciones posteriores no se podía realizar la media entre el anterior y el siguiente.

A modo de ejemplo, a continuación se muestra la evolución de las mediciones de algunos de los sensores durante un minuto del día 22 de enero de 2020. En la figura 3.1 se incluyen los sensores de vibración, y en la figura 3.2, los de temperatura.

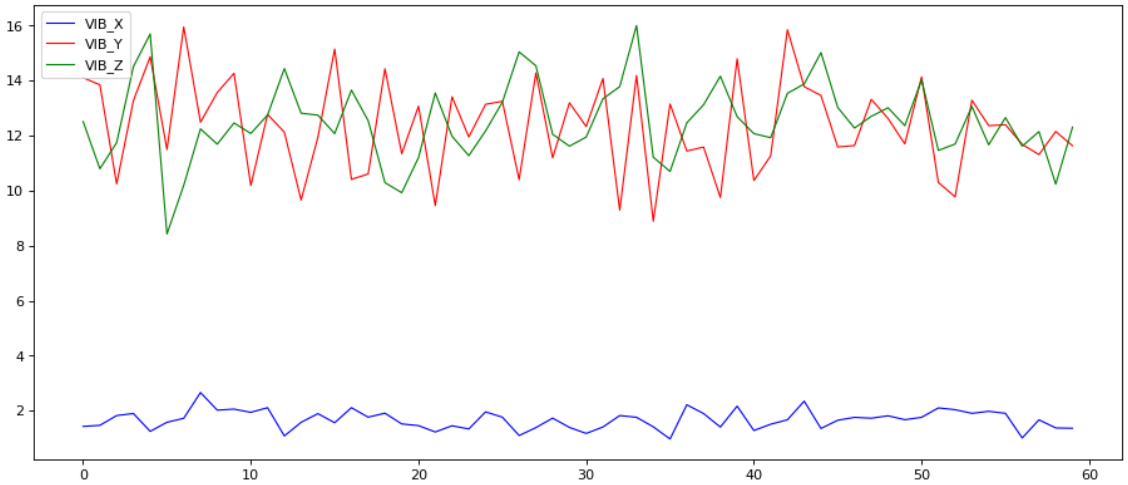

Figura 3.1: Mediciones de vibración durante un minuto del día 22 de enero de 2020.

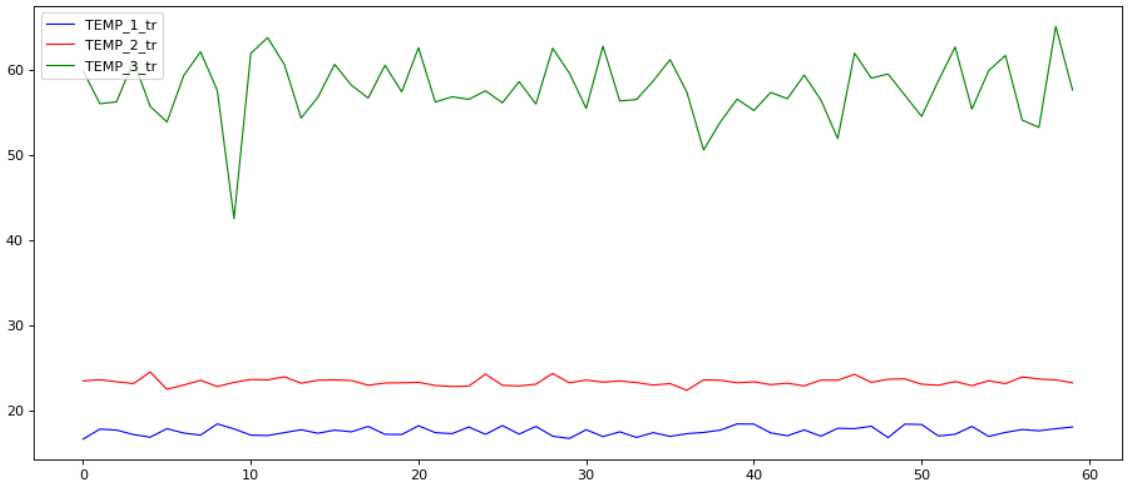

Figura 3.2: Mediciones de temperatura durante un minuto del día 22 de enero de 2020.

Por último, se realizó un análisis de la correlación entre las diferentes variables, utilizando el coeficiente de correlación de Pearson. Dicho coeficiente es una medida de dependencia lineal entre dos variables aleatorias cuantitativas, en el que un valor de 1 representa una relación positiva perfecta, un valor de -1 indica una relación negativa perfecta, y un valor de 0, la ausencia de relación entre ambas variables. Su fórmula es la siguiente:

$$\rho = \frac{\text{cov}(X, Y)}{\sigma_x \sigma_y}$$

donde X e Y hacen referencia a las variables, cov es la covarianza y σ es la desviación típica.

La figura 3.3 muestra el resultado de calcular dicha correlación entre las 7 variables del *dataset*. A continuación se exponen las conclusiones extraídas tras analizar dichas correlaciones.

Figura 3.3: Correlación entre las variables utilizando el coeficiente de correlación de Pearson.

En la figura 3.3 se puede apreciar cómo las vibraciones en los ejes Y y Z están altamente correlacionadas, pero no ocurre lo mismo con la vibración en el eje X. Esto puede deberse a que la máquina en la que se ha instalado el sensor de vibración, por su modo de funcionamiento, presenta una vibración residual en el eje X en comparación con los otros dos ejes. Este hecho también queda reflejado en la figura 3.1. A la hora de detectar anomalías, registrar valores altos en el eje X debería llamarle bastante la atención al modelo desarrollado.

También en referencia a la figura 3.3, llama la atención cómo las temperaturas 1 y 2 presentan cierta correlación negativa. Una de las posibles explicaciones a este fenómeno sería que uno de los dos sensores esté colocado cerca de algún tipo de sistema de refrigeración. En cuanto a la corriente, no parece estar relacionada con ninguna otra variable.

CAPÍTULO 4
Detalles de implementación

En este capítulo se explican los principales componentes que se han implementado para construir un sistema de detección de anomalías no supervisado. Más concretamente, se ha optado por utilizar las tipologías *LSTM Autoencoder* y *GAN*, cuyos fundamentos teóricos ya se han explicado anteriormente. En primer lugar, se abordan sus detalles más a nivel de implementación. Seguidamente, se explica un método utilizado para seleccionar los umbrales de decisión de forma dinámica y no supervisada, y poder así determinar de forma automática qué errores de predicción representan o no una anomalía. Por último, se describen diferentes estrategias implementadas para mitigar el número de falsos positivos, y de esta manera mejorar la confianza del cliente a la hora de utilizar esta solución en sus sistemas de producción.

Toda la implementación se ha realizado utilizando el *API* de *Keras* [1], un *framework* de *machine learning* construido sobre *Tensorflow* [2].

Más adelante, en el capítulo 5, se comentarán los experimentos que se han realizado para evaluar cuál de estas tipologías muestra un mejor rendimiento a la hora de detectar anomalías, y de esta forma poder decidir qué arquitectura debía implantarse en las instalaciones del cliente.

4.1 Reconstrucción de secuencias temporales

Como se ha explicado anteriormente, los enfoques de detección de anomalías basados en la reconstrucción aprenden un modelo que captura la estructura latente, es decir, la representación en un espacio dimensional menor, de una serie de datos temporales para luego tratar de reconstruir la misma serie de forma sintética. Bajo este funcionamiento, se asume que si un modelo recibe como entrada una secuencia de datos normales éste podrá reconstruirla de manera fiel, mientras que presentará problemas para hacer lo mismo si la entrada contiene anomalías. Sin embargo, si se quiere una definición más formal del problema, éste consistiría en el aprendizaje de dos funciones de correspondencia entre dos dominios X y Z, llamadas $\mathcal{E} : X \to Z$ and $\mathcal{G} : Z \to X$. X corresponde al dominio de los datos de entrada y contiene los datos de entrenamiento, mientras que Z representa el dominio latente. Más concretamente, y usando la nomenclatura x_i para hacer referencia a la secuencia de longitud t comenzando desde el punto en el tiempo i, el problema consistiría en obtener $\mathcal{E}$ y $\mathcal{G}$ de manera que $x_i \to \mathcal{E}(x_i) \to \mathcal{G}(\mathcal{E}(x_i)) \approx \hat{x}_i$.

Cabe destacar que se ha creado un modelo diferente para cada canal, es decir, que existen siete modelos, cada uno de ellos especializado en realizar predicciones para cada

[1] https://keras.io
[2] https://www.tensorflow.org

canal específico. Sin embargo, para el entrenamiento sí que se utilizan los siete canales. Se trata por tanto de modelos multivariable que producen una salida univariable. Esto se ha hecho así porque las redes LSTM muestran problemas a la hora de predecir salidas multivariables. Además, esto tiene la ventaja de que se puede estudiar la trazabilidad del rendimiento de cada uno de los canales por separado, permitiendo una vista más granulada de las anomalías detectadas y de sus posibles causas.

4.2 Implementación *LSTM Autoencoder*

La estructura de capas que se ha seleccionado para implementar la red LSTM *Autoencoder* se compone de 3 capas LSTM bidireccionales en la parte del *Encoder* con una reducción en el número de neuronas hasta llegar al espacio latente. Más concretamente, la primera capa tiene 64 nueronas, la segunda 32 y la tercera 16. Seguidamente, el *Decoder* realiza el proceso contrario, también con 3 capas LSTM bidireccionales pero esta vez aumentando el número de neuronas hasta alcanzar la capa de salida. En este caso, la primera capa tiene 16 neuronas, la segunda 32 y la tercera 64. Como optimizador se ha utilizado *Adam*.

4.3 Implementación *GAN*

En cuanto a la parte generadora de la arquitectura GAN, la función generadora $\mathcal{E}$ corresponde con el *Encoder*, mientras que la función generadora $\mathcal{G}$ hace el papel de *Decoder*. En lo que respecta a la parte *adversarial*, existen dos funciones críticas C_x y C_z, también conocidas como discriminadoras. El objetivo de C_x es distinguir entre secuencias reales extraídas de X y secuencias sintéticas generadas con $\mathcal{G}(z)$, mientras que C_z mide la eficiencia de la codificación en el espacio latente. Básicamente, $\mathcal{G}$ está intentando engañar a C_x generando secuencias que parezcan reales. Sin embargo, entrenar una red con estas características presenta una serie de desafíos, los cuales se consiguen solventar utilizando una función de pérdida multiobjetivo consistente en la combinación de dos pérdidas.

4.3.1. Función de pérdida multiobjetivo

$\mathcal{G}$ tiende a aprender únicamente una pequeña fracción de la variabilidad de los datos, por lo que es incapaz de reconstruir de forma completa secuencias que respondan a la misma distribución que la de los datos de entrada. La causa de esto es que $\mathcal{G}$ prefiere seguir produciendo secuencias que ya han conseguido engañar correctamente a los críticos antes que buscar nuevos tipos de variaciones, o modos, que también conseguirían engañarlos. Esta limitación se conoce como problema de colapso de modo, y puede solucionarse aplicando la pérdida Wassertein [18] para entrenar la parte crítica de la arquitectura. Formalmente, siendo $\mathbb{P}_X$ la distribución sobre X, para la función de correspondencia $\mathcal{G} : Z \to X$ y su función crítica C_x, se tiene el siguiente objetivo:

$$\min_{\mathcal{G}} \max_{\mathcal{C}_x \in \mathbf{C_x}} V_X\left(\mathcal{C}_x, \mathcal{G}\right)$$

con

$$V_X\left(\mathcal{C}_x, \mathcal{G}\right) = \mathbb{E}_{x \sim \mathbb{P}_X}\left[\mathcal{C}_x(x)\right] - \mathbb{E}_{z \sim \mathbb{P}_Z}\left[\mathcal{C}_x(\mathcal{G}(z))\right]$$

donde $\mathcal{C}_x \in \mathbf{C_x}$.

$\mathbf{C_x}$ denota el conjunto de funciones continuas 1-Lipschitz [19], las cuales restringen el límite superior de la función, consiguiendo así suavizarla. De esta manera, los pesos no cambian de forma dramática al actualizarlos con métodos de descenso por gradiente. Esto reduce el riesgo de explosión de gradiente, y hace que el entrenamiento del modelo sea más estable.

Siguiendo un enfoque similar, también se aplica una pérdida Wassertein para la función de correspondencia $\mathcal{E} : X \to Z$ y su función crítica C_z, cuyo objetivo se expresa de la siguiente manera:

$$\min_{\mathcal{E}} \max_{\mathcal{C}_z \in \mathbf{C_z}} V_Z\left(\mathcal{C}_z, \mathcal{E}\right)$$

Conviene recordar que el objetivo de la función crítica C_z es diferenciar entre muestras aleatorias del espacio latente $z \sim \mathbb{P}_Z$ y muestras codificadas $\mathcal{E}(x)$ utilizando $x \sim \mathbb{P}_X$.

Sin embargo, el uso de funciones de pérdida adversariales como la Wassertein de forma individual no puede garantizar que la reconstrucción sea satisfactoria, y es necesario reducir el espacio de búsqueda mediante el uso de funciones de pérdida de consistencia de ciclo. Más concretamente, se entrena la red introduciendo como objetivo el minimizar la normalización L2 de la diferencia entre los datos originales y los reconstruidos, consiguiendo así reducir las contradicciones entre $\mathcal{E}$ y $\mathcal{G}$:

$$V_{L2}(\mathcal{E}, \mathcal{G}) = \mathbb{E}_{x \sim \mathbb{P}_X}\left[\|x - \mathcal{G}(\mathcal{E}(x))\|_2\right]$$

La combinación de ambos objetivos conduce al problema MinMax final:

$$\min_{\{\mathcal{E},\mathcal{G}\}} \max_{\{\mathcal{C}_x \in \mathbf{C_x}, \mathcal{C}_z \in \mathbf{C_z}\}} V_X\left(\mathcal{C}_x, \mathcal{G}\right) + V_Z\left(\mathcal{C}_z, \mathcal{E}\right) + V_{L2}(\mathcal{E}, \mathcal{G}).$$

Aquí se ve reflejado que, además de tratar de conseguir generar secuencias lo más similares posible a las originales, existe una función crítica C_x entrenada específicamente para distinguir entre datos reales y falsos. En teoría, la salida obtenida a partir de dicha función crítica debería suponer una ventaja con respecto al uso de *Autoencoders* LSTM más simples, pues su uso a la hora de calcular las puntuaciones de anomalía otorga más información útil al sistema, como se explicará en la subsección 4.3.2.

4.3.2. Obtención de puntuaciones de anomalía

Antes de estudiar cómo se asignan puntuaciones a las anomalías encontradas, conviene repasar el proceso completo del sistema mediante el cual éstas son obtenidas. En primer lugar, se parte de la base de que los datos de entrenamiento son de la forma $\mathbf{X} = \left(x^1, x^2, \cdots, x^T\right)$, donde x^i perteneciente a $\mathbb{R}^{M \times 1}$ indica M tipos de medidas en el momento i, y también que dichos datos corresponden a observaciones de funcionamiento sin anomalías. Por simplicidad, en esta explicación se va a usar $M = 1$, lo que significa que se considera que $\mathbf{X}$ es una secuencia de datos temporales univariable, y que x^i es un escalar.

Tras estas asunciones, el primer paso es aplicar una ventana de tamaño t que se va desplazando a lo largo de los datos de entrenamiento con un avance s para dividir la secuencia original en N subsecuencias, obteniendo así la primera de las dos entradas de la arquitectura $X = \left\{\left(x_i^{1\cdots t}\right)\right\}_{i=1}^{N}$, donde $N = \frac{T-t}{s}$. La segunda de las entradas es un vector $Z = \left\{\left(z_i^{1\cdots k}\right)\right\}_{i=1}^{N}$ obtenido de un espacio aleatorio, el cual obedece a una distribución normal, y donde k denota la dimensión del espacio latente.

Ahora sí, entrenando el modelo GAN utilizando la función pérdida descrita en la subsección 4.3.1, se pueden calcular las puntuaciones de anomalía en cada punto del tiempo, utilizando para ello los errores de reconstrucción y la salida de la red crítica C_x.

Utilización de los errores de reconstrucción

Como se ha visto anteriormente, dada una secuencia x_i de longitud t, el modelo implementado en el cliente genera otra secuencia reconstruida de la misma longitud y que responde a las transformaciones $x_i \rightarrow \mathcal{E}(x_i) \rightarrow \mathcal{G}(\mathcal{E}(x_i)) \approx \hat{x}_i$. Para calcular los errores de reconstrucciones en cada uno de los puntos del tiempo se pueden utilizar tres tipos de funciones diferentes.

El primero es la diferencia punto a punto, la cual, de manera intuitiva, simplemente calcula la diferencia entre el valor real y el reconstruido en cada punto del tiempo:

$$s_t = \left| x^t - \hat{x}^t \right|.$$

La segunda forma de calcular dicha diferencia, conocida como diferencia de área, es aplicando ventanas de determinado tamaño para medir la similitud entre regiones locales, consiguiendo de esta manera identificar de forma correcta regiones en las que existen pequeñas diferencias durante un largo periodo de tiempo. En este caso, se utiliza la diferencia media entre las áreas alojadas bajo dos curvas de longitud l, como indica la siguiente fórmula:

$$s_t = \frac{1}{2l} \left| \int_{t-l}^{t+l} x^t - \hat{x}^t dx \right|.$$

El último y más sofisticado de los métodos, se llama *Dynamic time warping* (DTW), y trata de calcular la correspondencia óptima entre dos secuencias temporales utilizando el procedimiento definido a continuación. Dadas dos secuencias $X = (x_{t-1}, x_{t-l+1}, \ldots, x_{t+l})$ y $\hat{X} = (\hat{x}_{t-1}, \hat{x}_{t-l+1}, \ldots, \hat{x}_{t+l})$, y una matriz $W \in \mathbb{R}^{2l \times 2l}$ de manera que su elemento en la posición (i, j) corresponde a la distancia entre x_i y $\hat{x}_j$, denominada w_k, se quiere encontrar el camino $W^* = (w_1, w_2, \ldots, w_K)$ que define la distancia mínima entre las dos curvas, asumiendo condiciones al inicio y al final, así como restricciones de continuidad y monotoneidad. Formalmente, la distancia DTW quedaría definida por

$$s_t = W^* = \text{DTW}(X, \hat{X}) = \min_W \left[\frac{1}{K} \sqrt{\sum_{k=1}^{K} w_k} \right].$$

De igual forma que la diferencia de área, DTW es capaz de identificar regiones con pequeñas diferencias durante largos periodos de tiempo, y además problemas de desplazamientos en el tiempo.

Utilización la salida de la red crítica C_x

Durante el proceso de entrenamiento, el objetivo de la red crítica C_x es diferenciar entre secuencias reales y sintéticas, y como para conseguirlo se utiliza la distancia Wassertein, significa que valores altos indican mayor confianza en que la secuencia es real, así como valores más pequeños indican lo contrario. Por tanto, una vez la red crítica está entrenada, su salida puede usarse directamente para medir cómo de anómala es una secuencia de datos temporales.

Combinación de ambas puntuaciones

Para normalizar los errores de reconstrucción $RE(x)$ y las salidas de la red crítica C_x, se ha optado por calcular en primer lugar sus medias y desviaciones típicas, y así poder calcular sus respectivos *z-scores* $Z_{RE}(x)$ y $Z_{\mathcal{C}_x}(x)$. De esta manera, un mayor *z-score* significa una puntuación de anomalía también mayor. Seguidamente, para combinar ambas puntuaciones, se han probado dos aproximaciones. La primera de ellas es mediante una combinación convexa:

$$\boldsymbol{a}(x) = \alpha Z_{RE}(x) + (1-\alpha) Z_{\mathcal{C}_x}(x),$$

donde α controla la importancia relativa de ambos términos, y es 0.5 por defecto.

En la segunda de las aproximaciones se opta por multiplicar las puntuaciones para tratar de dar énfasis a los valores altos:

$$\boldsymbol{a}(x) = \alpha Z_{RE}(x) \cdot Z_{\mathcal{C}_x}(x),$$

donde $\alpha = 1$ por defecto. Ambas aproximaciones han dado buenos resultados en los experimentos realizados.

4.4 Umbrales de error dinámicos

Debido a que no se poseen datos etiquetados, el establecimiento del umbral debe hacerse de forma no supervisada. Para dotar de flexibilidad al sistema, esto se hace de forma separada para cada secuencia de errores, por lo que estos umbrales pueden considerarse dinámicos. Más concretamente, el umbral ϵ es escogido de entre el conjunto

$$\boldsymbol{\epsilon} = \mu(\mathbf{e}_s) + \mathbf{z}\sigma(\mathbf{e}_s),$$

donde ϵ es determinado por:

$$\epsilon = \operatorname{argmax}(\boldsymbol{\epsilon}) = \frac{\Delta\mu(\mathbf{e}_s)/\mu(\mathbf{e}_s)) + (\Delta\sigma(\mathbf{e}_s)/\sigma(\mathbf{e}_s)}{|\mathbf{e}_a| + |\mathbf{E}_{seq}|^2}$$

tal que:

$$\begin{aligned}
&\Delta\mu(\mathbf{e}_s) = \mu(\mathbf{e}_s) - \mu(\{e_s \in \mathbf{e}_s \mid e_s < \epsilon\}) \\
&\Delta\sigma(\mathbf{e}_s) = \sigma(\mathbf{e}_s) - \sigma(\{e_s \in \mathbf{e}_s \mid e_s < \epsilon\}) \\
&\mathbf{e}_a = \{e_s \in \mathbf{e}_s \mid e_s > \epsilon\} \\
&\mathbf{E}_{seq} = \text{ secuencias continuas de } e_a \in \mathbf{e}_a
\end{aligned}$$

donde μ es la media y σ la varianza.

Los valores para ϵ se determinan utilizando $z \in \mathbf{z}$, donde $\mathbf{z}$ es un conjunto ordenado de valores enteros entre [2, 10]. Este rango ha sido establecido por ofrecer buenos resultados en la fase de experimentación. Una vez que $\operatorname{argmax}(\boldsymbol{\epsilon})$ ha sido determinado, se le asigna una puntuación de anomalía a cada secuencia de errores suavizados, la cual indica la severidad de la anomalía:

$$s^{(i)} = \frac{\max\left(\mathbf{e}_{\text{seq}}^{(i)}\right) - \operatorname{argmax}(\boldsymbol{\epsilon})}{\mu(\mathbf{e}_s) + \sigma(\mathbf{e}_s)}.$$

4.5 Mitigación de falsos positivos

La precisión de los enfoques de detección de anomalías basados en la reconstrucción de series temporales depende en gran medida de la cantidad de datos históricos utilizados. Cuantos más datos se dispongan para el entrenamiento, más confianza se tendrá sobre si los umbrales establecidos, y por tanto los juicios emitidos en base a ellos, son correctos. Sin embargo, y debido al coste que suele conllevar procesar dichos datos a gran escala y en tiempo real, muchas veces no se tienen suficientes datos históricos, lo cual provoca que haya un gran número de falsos positivos, pues lo que se ha considerado anómalo únicamente lo es porque hubo una falta de información y contexto durante el entrenamiento. Este gran número de falsos positivos suele suponer que el revisor humano encargado de evaluar los potenciales eventos anómalos se vea sobrepasado en su tarea.

Para mitigar los falsos positivos se ha utilizado un procedimiento de poda por el cual, del conjunto total de anomalías detectadas e_{seq}, únicamente se mantienen las N primeras ordenadas de forma decreciente. Este nuevo conjunto reducido se conoce como $\mathbf{e}_{\text{max}}$. Además, se establece un porcentaje máximo de decrecimiento p, de manera que si el porcentaje de decrecimiento entre dos anomalías consecutivas, $i, i+1$, sobrepasa dicho límite, y también lo hace para todos los pares subsecuentes, todas esas anomalías a partir de i en adelante pasan a considerarse normales. Esta poda ayuda a asegurar que las secuencias anómalas no son resultado del ruido de la señal y, junto con el uso de umbrales, suponen un procedimiento mucho más eficiente que la enorme cantidad de comparaciones valor a valor que habría que hacer si se utilizara otro procedimiento.

De forma adicional, y una vez ya se ha obtenido un histórico de anomalías considerable y dichas anomalías han sido evaluadas como verdaderos positivos, podría establecerse un umbral mínimo $s_{\text{mín}}$ para cada uno de los canales de entrada. De esta manera, aunque el umbral dinámico calculado de forma automática mediante el procedimiento explicado en la sección anterior considerara cierto evento como anómalo, éste pasaría a considerarse normal si su puntuación fuera menor que $s_{\text{mín}}$. Este mecanismo supone una forma sencilla de incorporar al sistema conocimiento experto aprendido de la propia experiencia.

CAPÍTULO 5
Experimentación y evaluación

Para experimentar con las dos tipologías implementadas y con sus parámetros explicados en el capítulo 4, y de esa forma poder determinar cuál es la configuración que mejor resultados ofrece, es necesario contar con un conjunto de pruebas con anomalías. El problema es que los datos proporcionados por el cliente corresponden al funcionamiento normal de la máquina, y por tanto no presentan ninguna observación anómala. Podría haberse optado por la introducción de anomalías de forma sintética, pero sin un profundo conocimiento del funcionamiento interno de la máquina, así como de sus procesos de deterioro provocados por el desgaste natural de sus piezas, existe una alta probabilidad de que las anomalías artificiales disten mucho de las que la máquina producirá en un futuro. Esto podría provocar que la configuración seleccionada, aunque sea la que mejor responde con las anomalías sintéticas, ofreciera un mal rendimiento una vez implantada en producción. Por este motivo, se ha preferido utilizar en esta fase un *dataset* habitualmente utilizado como *benchmark* en el ámbito de la detección de anomalías.

Este *dataset* se conoce como SMAP porque contiene datos recogidos por el satélite *Soil Moisture Active Passive*. Se trata de un *dataset* público [1]. Por razones de seguridad, los nombres de los sensores han sido anonimizados, y únicamente contienen la primera letra. Por ejemplo, el canal *P-1* corresponde a mediciones de potencia, o el canal *R-1* a radiación. De todos los canales presentes en el *dataset*, únicamente se han utilizado 7, en un intento de asimilarse lo máximo posible al *dataset* proporcionado por el cliente, que también consta de 7 canales. Para los sensores seleccionados, SMAP contiene muestras recogidas cada 21600 milisegundos desde el año 2008 hasta el año 2015. Durante ese rango de fechas, el satélite sufrió tres anomalías. La primera de ellas ocurrió entre marzo y abril de 2012, la segunda entre febrero y abril de 2013, y la tercera entre octubre de 2013 y enero de 2014. Como todas estas anomalías se encuentran en la segunda mitad del *dataset*, se ha decidido utilizar la primera mitad para entrenamiento y la segunda para *testing*. En la figura 5.1 puede verse la evolución en el tiempo de las mediciones del sensor *P-1* que se han utilizado para entrenamiento, y en la figura 5.2 la parte destinada a *testing*, con las anomalías representadas en color rojo. El entrenamiento se ha realizado utilizando los 7 canales, pero prediciendo únicamente para el canal *P-1*, por los motivos que se han explicado en la sección 4.1.

A continuación se detallan los parámetros con los que se ha querido experimentar, y sus respectivos rangos de valores. En primer lugar, y para tratar de encontrar el mejor equilibrio entre *underfitting* y *overfitting*, se ha querido probar con diferentes valores del *learning rate* y de *epochs*. Para el *learning rate* se han probado los valores 0.001, 0.0005 para la arquitectura GAN y 0.0005 y 0.0001 para la LSTM *Autoencoder*. Para los *epochs* se han entrenado modelos con 50, 100 y 200 iteraciones. En cuanto a los parámetros para

[1] https://s3-us-west-2.amazonaws.com/telemanom/data.zip

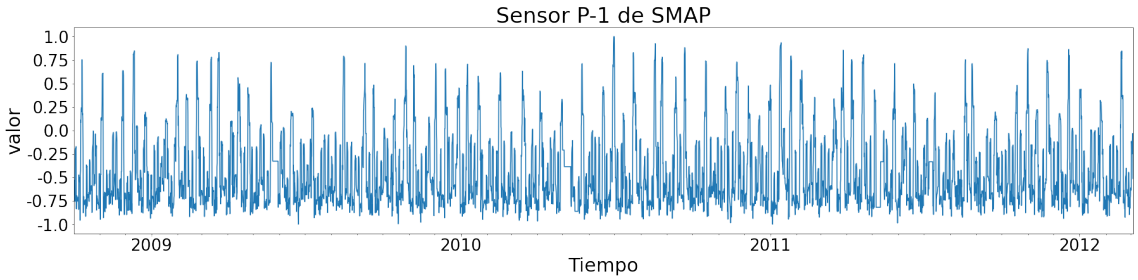

Figura 5.1: Mediciones del sensor P-1 de la parte de *training*.

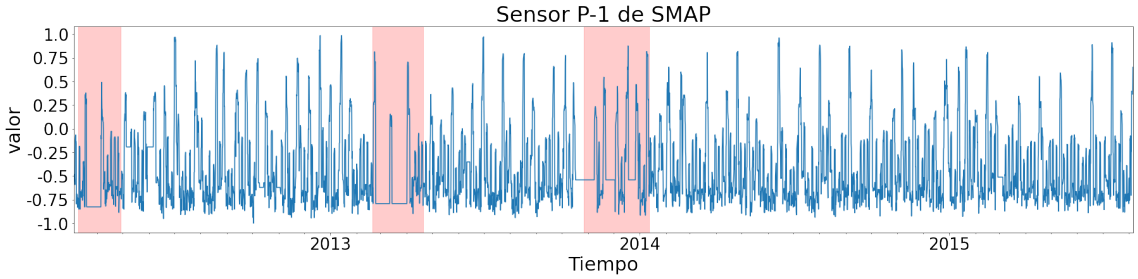

Figura 5.2: Mediciones del sensor P-1 de la parte de *testing* con anomalías en rojo.

la obtención de puntuaciones de anomalía, se han probado los tres métodos de cálculo de diferencias entre los valores reales y los reconstruidos, que son la diferencia punto a punto, la diferencia de área y la diferencia DTW. Además, y esto aplica únicamente a la tipología GAN, se ha experimentado con la combinación de puntuaciones mediante suma y mediante multiplicación. Por último, y en relación al proceso de determinar qué segmentos deben ser anotados como anómalos y cuáles no en base a sus puntuaciones, se ha probado el algoritmo con diferentes tamaños de ventana y de avance de ventana en cada paso. Más concretamente, se han utilizado los valores 0.2, 0.33, 0.5 y 0.8 para el tamaño de ventana, y los valores 0.1, 0.2, 0.33 y 0.5 para el avance de ventana en cada paso.

En lo que respecta a la evaluación, se ha decidido utilizar el valor F1 para realizar un balance entre la precisión y la sensibilidad. Esto hace más sencillo el poder comparar de un solo vistazo el rendimiento combinado de ambas métricas entre los diferentes experimentos lanzados. Más concretamente, el valor F1 se calcula haciendo la media armónica entre la precisión y la sensibilidad. A continuación se incluyen las fórmulas para calcular los tres valores:

$$Precision = \frac{TP}{TP + FP}$$

$$Sensibilidad = \frac{TP}{TP + FN}$$

$$F1 = \frac{2 \cdot Precision \cdot Sensibilidad}{Precision + Sensibilidad}$$

donde TP hace referencia a los verdaderos positivos, FP a los falsos positivos y FN a los falsos negativos. Como se verá más adelante, y para que sirva a modo de ejemplo, el mejor experimento ha obtenido un F1 de 0.86, que atendiendo a la fórmulas anteriores se calcularía así:

$$TP = 3$$

$$FP = 1$$

$$FN = 0$$

$$Precision = \frac{3}{3+1} = 0.75$$

$$Sensibilidad = \frac{3}{3+0} = 1$$

$$F1 = \frac{2 \cdot 0.75 \cdot 1}{0.75+1} = \frac{1.5}{1.75} = 0.8571$$

Pasando a mostrar los resultados de los diferentes experimentos realizados, es necesario primero aclarar las abreviaturas utilizadas en las tablas. *mult* y *sum* hacen referencia a los métodos de combinación de puntuaciones. WS representa el parámetro de ancho de ventana (*window size*) y WSS, el de avance de ventana en cada paso (*window step size*).

Se comienza con los resultados obtenidos utilizando la tipología GAN, entrenada con un *lr* de 0.0005 y durante 50 *epochs*. La tabla 5.1 recoge las puntuaciones F1 obtenidas utilizando la diferencia punto a punto.

	mult				sum			
WSS/WS	0.20	0.33	0.50	0.80	0.20	0.33	0.50	0.8
0.10	**0.50**	0.25	0.29	0.33	0.44	0.22	0.29	0.00
0.20	0.44	0.25	0.29	0.33	0.40	0.25	0.33	0.00
0.33	0.20	0.25	0.33	0.33	0.20	0.29	0.33	0.00
0.50	0.22	0.25	0.33	0.33	0.25	0.25	0.33	0.00

Tabla 5.1: Puntuaciones F1 obtenidas con GAN y diferencia punto a punto.

Mientras tanto, la tabla 5.2 muestra los valores obtenidos con la diferencia de área.

	mult				sum			
WSS/WS	0.20	0.33	0.50	0.80	0.20	0.33	0.50	0.80
0.10	0.50	0.57	0.40	0.50	0.44	0.44	0.33	0.33
0.20	0.50	0.57	0.40	0.50	0.40	0.44	0.33	0.40
0.33	0.50	**0.67**	0.50	0.50	0.40	0.50	0.40	0.40
0.50	0.50	0.57	0.50	0.00	0.44	0.29	0.40	0.00

Tabla 5.2: Puntuaciones F1 obtenidas con GAN y diferencia de área.

Puede observarse cómo los resultados que ofrece la diferencia de área son mejores en general que los ofrecidos por la diferencia punto a punto. También, y esta vez comparando los métodos de combinación de puntuaciones, parece que la multiplicación se

comporta algo mejor que la suma. Por este motivo, los resultados que se muestran a partir de este punto son únicamente para el método *mult*.

Para terminar con la comparativa entre los diferentes métodos de cálculo de diferencias, la tabla 5.3 contiene las puntuaciones F1 obtenidas con la diferencia DTW.

	mult			
WSS/WS	0.20	0.33	0.50	0.80
0.10	0.22	0.29	0.29	0.00
0.20	0.22	0.29	0.29	0.00
0.33	0.22	0.29	**0.33**	0.00
0.50	0.25	**0.33**	**0.33**	0.00

Tabla 5.3: Puntuaciones F1 obtenidas con GAN y diferencia DTW.

Analizando los valores de las tablas anteriores, puede concluirse que la mejor combinación de parámetros hasta ahora corresponde a la diferencia de área, combinación por multiplicación, WS de 0.33 y WSS de 0.33. La tabla 5.4 recoge los resultados obtenidos con esta configuración, pero variando los parámetros de entranamiento *lr* y *epochs*.

epochs/lr	0.0005	0.0001
50	0.67	0.50
100	**0.86**	0.57
200	0.25	-

Tabla 5.4: Puntuaciones F1 obtenidas con GAN y diferentes configuraciones de *lr* y *epochs*.

Puede observarse cómo utilizando 100 *epochs* y un *lr* de 0.0005 es como se obtiene la mejor puntuación F1. Más concretamente, ese valor se extrae a partir de una precisión del 75 % y una sensibilidad del 100 %. Esto quiere decir que el modelo ha sido capaz de detectar las tres anomalías, y únicamente ha tenido un falso positivo, como puede apreciarse en la figura 5.3.

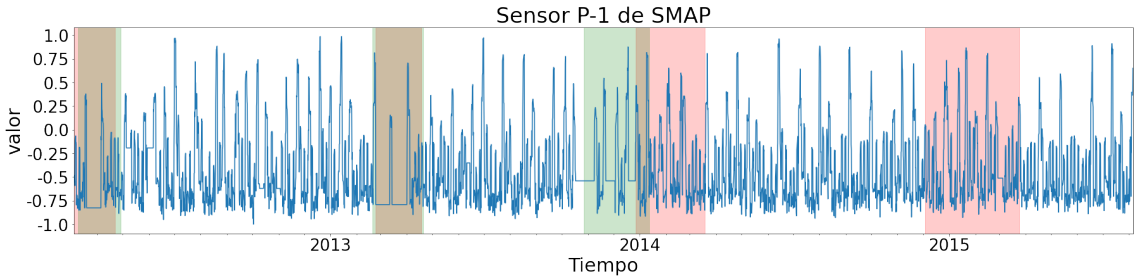

Figura 5.3: Anomalías detectadas (en rojo) por el modelo que obtuvo un F1 de 0.86. Las anomalías reales aparecen en verde.

La reconstrucción de la señal obtenida por el mejor modelo, dibujada junto con la señal original, puede verse en la figura 5.4. En la parte inferior de dicha figura se incluye el error calculado con el método de diferencia de área en cada instante del tiempo. A modo de comparativa, la figura 5.5 muestra una gráfica del mismo tipo, pero en este caso con un modelo menos potente que obtuvo un F1 de 0.33.

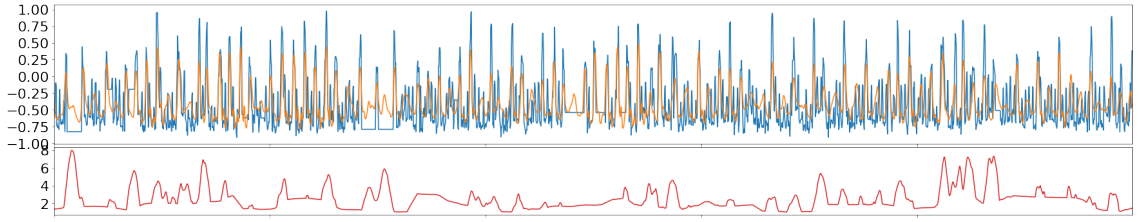

Figura 5.4: Señal original (en azul) junto con señal reconstruida por el mejor modelo (en naranja). En la parte inferior, error calculado en cada instante (en rojo).

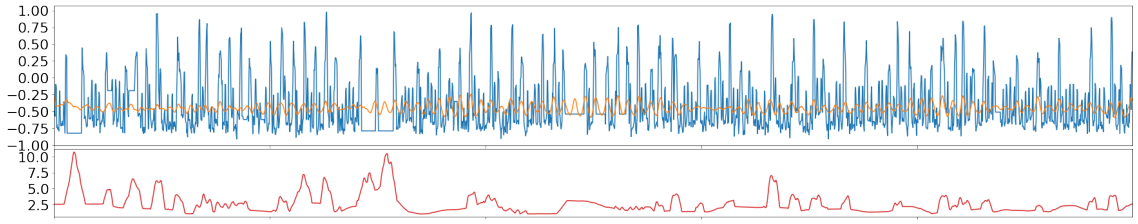

Figura 5.5: Señal original (en azul) junto con señal reconstruida modelo menos potente (en naranja). En la parte inferior, error calculado en cada instante (en rojo).

Es turno ahora de analizar el comportamiento de la otra tipología de red neuronal implementada, la LSTM *Autoencoder*. En este caso, se ha probado con los valores de *lr* 0.001 y 0.0005, pues el valor 0.0001 utilizado en la GAN se quedaba muy pequeño y el *Autoencoder* no lograba una buena reconstrucción de los datos. En cuanto a WS y WSS, se han utilizado los valores 0.2 y 0.5, obteniendo los resultados que se reflejan en la tabla 5.5.

	learning rate - epochs			
	0.0005 - 50	0.001 - 50	0.0005 - 100	0.001 - 100
WSS/WS				
0.20/0.50	**0.57**	0.22	0.22	0.33
0.50/0.20	0.50	0.25	0.20	0.44

Tabla 5.5: Puntuaciones F1 obtenidas con LSTM *Autoencoder* y diferentes configuraciones.

Atendiendo a los resultados de los experimentos, puede concluirse que la tipología LSTM *Autoencoder* ofrece un peor rendimiento a la hora de detectar anomalías que la tipología GAN, pues la mejor puntuación F1 conseguida por el LSTM *Autoencoder* es 0.57, quedando lejos del 0.86 alcanzado con GAN. Esta diferencia podría deberse a que la tipología GAN cuenta con más información a la hora de calcular las puntuaciones de anomalía, ya que aprovecha la salida de la función crítica. Es por ello que la tipología seleccionada para implantar en las instalaciones del cliente ha sido la GAN.

CAPÍTULO 6
Conclusiones y trabajos futuros

6.1 Conclusiones

En este trabajo se ha hecho un repaso a los principales conceptos referentes a la detección automática de anomalías mediante las técnicas de *machine learing* que representan el estado del arte en la actualidad. Se han desarrollado dos de esas técnicas, y se han descrito los detalles de de la implementación de cada una de ellas. También se ha realizado una comparativa entre ambas arquitecturas, tras una fase de experimentación y evaluación utilizando un *dataset* con anomalías reales y conocidas. El resultado de dicha comparativa ha permitido seleccionar cuál de las dos tipologías ofrece mejores resultados. La arquitectura seleccionada ha sido la que hace uso de *Generative Adversarial Networks*, frente a los LSTM *Autoencoders*.

Todo este trabajo se ha realizado dentro del marco de un proyecto para uno de los clientes de la empresa Nunsys SA. Cabe por tanto recalcar que la motivación del proyecto es profesional, buscando la consecución de los objetivos planteados por el cliente dentro de los plazos establecidos, y no una motivación de investigación. Teniendo esto en mente, también se quiere destacar que el problema de detección de anomalías sufre de una limitación por naturaleza, y es que normalmente se cuenta con muy pocos datos anómalos, pues la mayoría del tiempo la máquina funciona con normalidad. Este ha sido el caso de los datos proporcionados por el cliente, que no presentan ningún dato anómalo. Es por ello que la experimentación se ha realizado con otro *dataset* que, aunque cuenta con únicamente 3 fases anómalas, está corroborado que las mismas son reales. Esta decisión se tomó de forma conjunta con el cliente, tras planteársele también la opción de simular anomalías en sus datos, que quedó descartada debido al riesgo de que no se parecieran a las posibles anomalías que pudieran surgir en un futuro.

En el momento de la redacción de esta memoria, al cliente se le ha entregado un API accesible desde sus instalaciones para poder realizar pruebas con los datos recogidos por los sensores instalados en sus máquinas. Tras estas pruebas, se llevará a cabo un proceso iterativo de ajustes en el modelo, y posteriormente un despliegue más integrado con el modelo final.

Además de esta última fase que está en proceso, existen otras áreas de extensión del proyecto que se definen el la sección 6.2.

6.2 Trabajos futuros

Tras los conocimientos adquiridos durante la realización de este proyecto, se considera que la solución desarrollada podría beneficiarse enormemente de ciertas líneas de

investigación y de trabajo. De hecho, algunas de ellas han sido propuestas por el propio cliente, lo que muestra una vez más el alto interés dentro de la industria en contar con sistemas inteligentes que consigan reducir costes y elevar la productividad. A continuación, se realiza una breve descripción de las tres áreas de trabajo más interesantes de entre las posibles actuaciones que se han barajado.

6.2.1. Mecanismos de atención

En este trabajo, las arquitecturas con las que se ha experimentado son *Seq2Seq*, es decir, convierten una secuencia de entrada en otra secuencia de salida. Para modelar las relaciones temporales, y también para combatir los problemas de desvanecimiento de gradiente, se han utilizado las redes LSTM. Sin embargo, estas redes presentan ciertas limitaciones conforme la secuencia que deben procesar aumenta de tamaño, pues se les hace más difícil capturar dichas relaciones cuando están muy separadas en el tiempo. Dado el éxito de los mecanismos de atención en dominios relacionados con el procesamiento de lenguaje natural [20], se considera que aplicar dichos mecanismos, ya sea como sustitución de las LSTM o como complemento, podría ser un área de investigación provechosa.

Los mecanismos de atención fueron propuestos con el objetivo de darle la capacidad al *Decoder* de fijarse, de forma selectiva, en los estados ocultos más relevantes, ignorando a su vez los de menor relevancia. De forma general, su modo de funcionamiento es el siguiente. En cada instante de tiempo t, durante la fase de decodificación, el modelo de atención calcula un vector de contexto $\mathbf{c}_t$ como resultado de una suma ponderada de todos los estados ocultos producidos por el *Encoder*. Más concretamente, los pesos de dicha suma se calculan en base a una función que mide la similitud entre el estado oculto actual del *Decoder* $\mathbf{h}_t^{\mathrm{d}}$ y todos los estados ocultos del *Encoder* $\mathbf{h}^{\mathrm{e}} = \left(\mathbf{h}_1^{\mathrm{e}}, \mathbf{h}_2^{\mathrm{e}}, \ldots, \mathbf{h}_{T_x}^{\mathrm{e}}\right)$. Seguidamente, estos valores son normalizados usando la función *softmax*, de manera que sumen 1 a lo largo de la segunda dimensión.

Gracias a este vector de contexto generado por los mecanismos de atención, el *Decoder* obtiene una ayuda adicional para saber qué parte de la información codificada es relevante y cuál no, por lo que se piensa que su utilización dentro del sistema actualmente implantado en el cliente puede representar una línea de trabajo futuro interesante

6.2.2. *Transfer learning*

Contar con una gran cantidad de datos de monitorización del estado operacional de un conjunto de máquinas puede no ser suficiente para realizar detección de anomalías, pues incluso aunque las máquinas sean del mismo modelo y estén instaladas en entornos similares, las distribuciones de los datos recogidos por cada una de ellas van a contener discrepancias, lo cual dificulta enormemente la reutilización del conocimiento adquirido en una máquina para aplicarlo en otra. Para solventar este problema y así poder contar con más datos de entrenamiento para generar modelos más potentes, es necesario estrechar las discrepancias entre los datos adquiridos por las diferentes máquinas, para lo cual se han analizado diferentes técnicas de *transfer learning* [21] que se describen brevemente a continuación. Si bien estas técnicas no se han aplicado a la solución actual, se considera que el *transfer learning* supondrá una enorme mejora del rendimiento, por lo que es una de las funcionalidades prioritarias planificadas para futuros desarrollos.

Deep Domain Confusion (DDC)

La arquitectura DDC, entrenada sobre un dominio de origen y otro de destino de forma conjunta, aprende de forma automática una representación entre ambos dominios capaz de minimizar la distancia entre sus distribuciones [22]. Esto lo consigue utilizando una capa adaptativa y una función de pérdida entre ambos dominios basada en la *Maximum Mean Discrepancy* (MMD). Una vez aprendida dicha representación, un modelo entrenado sobre datos etiquetados del dominio origen puede ser aplicado directamente sobre el dominio destino con una pérdida mínima.

Deep Adaptation Networks (DAN)

La principal diferencia entre las arquitecturas DDC y DAN reside en que esta última utiliza una variante con múltiples *kernels* de MMD, conocida como MK-MMD, y que posee una mayor capacidad de caracterización [23].

Esta arquitectura se basa en el hecho de que las características aprendidas por una red profunda van transitando de más generales a más específicas conforme éstas avanzan por la red, por lo que su transferibilidad es mayor en las capas superiores que en las inferiores. Por este motivo, las primeras capas convolucionales están congeladas, mientras que el aprendizaje de las siguientes capas *fully-connected* se hace mediante *fine-tuning* para ajustarse al dominio destino, utilizando la capacidad de adaptación que otorga MK-MMD.

Correlation Alignment for Deep Domain Adaptation (Deep CORAL)

CORAL es un método no supervisado de adaptación de dominios que alinea las estadísticas de segundo orden de las distribuciones origen y destino mediante una transformación lineal [24]. Se trata de una técnica sencilla y eficaz, pero que cuenta con algunas limitaciones como el propio hecho de depender en una transformación lineal, y que requiere de la aplicación de varias fases, lo que le impide poder considerarse un método *end-to-end*.

Con el objetivo de superar dichas limitaciones y poder aplicarse a dominios más complejos de forma *end-to-end*, Deep CORAL surge como una extensión de CORAL capaz de aprender una transformación no lineal que minimiza la diferencia entre las correlaciones origen y destino [25]. Esto se consigue mediante la incorporación de la función de pérdida CORAL dentro de una red neuronal profunda.

6.2.3. Mantenimiento predictivo

En base a las conversaciones mantenidas con el cliente, también están interesados en aplicar mantenimiento predictivo a las máquinas de sus instalaciones, como una ampliación natural al sistema dedetección de anomalías que ya tienen implantado en producción. El mantenimiento predictivo, a diferencia de la detección de anomalías, no se limita a identificar patrones fuera de lo común en los datos, sino que aprovecha el conocimiento adquirido a partir de dichos patrones para tratar de predecir cuándo la máquina fallará. Más allá de aliviar la sobrecarga del personal responsable del mantenimiento de los equipos de las plantas de producción, una solución automática también permitiría evitar recambios innecesarios, reducir el tiempo de inactividad de las máquinas o incluso de cadenas completas, ayudar a encontrar las causas de los fallos y, en definitiva, a ahorrar costes y mejorar la productividad de forma general.

Como se ha podido comprobar, el mantenimiento predictivo puede ofrecer grandes beneficios, pero es una tarea más compleja que la detección de anomalías. Es por ello que habitualmente necesita de un enfoque supervisado para obtener buenos resultados. El cliente es consciente de que tendrá que recolectar información adicional, la cual puede ser dividida en tres categorías. En primer lugar, un histórico de fallos, pues no se puede entrenar un modelo de mantenimiento predictivo únicamente con datos del funcionamiento normal de la máquina. En segundo lugar, un registro sobre las reparaciones y recambios que se han ejecutado sobre los dispositivos. Y, por último, datos sobre el funcionamiento habitual de la máquina como los que se han utilizado para la detección de anomalías, pero en este caso recogidos durante un periodo de tiempo lo suficientemente largo como para reflejar el deterioro y desgaste naturales sufridos por los diferentes componentes.

Bibliografía

[1] J. Helsen. SCADA analysis for condition monitoring. *European Wind Energy Conference EWEA*, 2015.

[2] Ponemon Institute LLC. The Cost of Malware Containment. 2015.

[3] Y. Gao, T. Yang, M. Xu and N. Xing. An Unsupervised Anomaly Detection Approach for Spacecraft Based on Normal Behavior Clustering. *Fifth International Conference on Intelligent Computation Technology and Automation*, 2012, pp. 478-481.

[4] S.D. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. *Ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 29-38.

[5] T. Yairi, N. Takeishi, T. Oda, Y. Nakajima, N. Nishimura and N. Takata. A Data-Driven Health Monitoring Method for Satellite Housekeeping Data Based on Probabilistic Clustering and Dimensionality Reduction. *IEEE Transactions on Aerospace and Electronic Systems vol. 53(3)*, 2017, pp. 1384-1401.

[6] R. Fujimaki, T. Yairi and K. Machida. An approach to spacecraft anomaly detection problem using kernel feature space. *Eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 2005, pp. 401-410.

[7] H.D. Nguyen, K.P. Tran, S. Thomassey and M. Hamad. Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management. *International Journal of Information Management vol. 57(102292)*, 2021.

[8] A. A. Cook, G. Mısırlı and Z. Fan. Anomaly Detection for IoT Time-Series Data: A Survey. *IEEE Internet of Things Journal vol. 7(7)*, 2020, pp. 6481-6494.

[9] D.M. Hawkins. *Identification of Outliers*. Chapman & Hall, London, vol. 11, 1980.

[10] J. Gottschlich. Paranom: A Parallel Anomaly Dataset Generator. *arXiv:1801.03164*, 2018.

[11] K. Amarasinghe, K. Kenney and M. Manic. Toward Explainable Deep Neural Network Based Anomaly Detection. *11th International Conference on Human System Interaction (HSI)*, 2018, pp. 311-317.

[12] M.A. Razzaque, M. Milojevic-Jevric, A. Palade and S. Clarke. Middleware for Internet of Things: A Survey. *IEEE Internet of Things Journal vol. 3(1)*, 2016, pp. 70-95.

[13] V. Chandola, A. Banerjee and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys vol. 41(15)*, 2016, pp. 1-58.

[14] M. Goldstein and S. Uchida. A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data. *PLoS ONE 11(4)*, 2016, e0152173.

[15] K.P. Tran, H.D. Nguyen and S. Thomassey. Anomaly detection using Long Short Term Memory Networks and its applications in Supply Chain Management. *IFAC-PapersOnLine vol. 52(13)*, 2019, pp. 2408-2412.

[16] I.J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville and Y. Bengio. Generative Adversarial Networks. *Communications of the ACM vol. 63(11)*, 2020.

[17] S. Akcay, A. Atapour-Abarghouei and T.P. Breckon. GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training. *arXiv:1805.06725*, 2018.

[18] M. Arjovsky and L. Bottou. Towards Principled Methods for Training Generative Adversarial Networks. *5th International Conference on Learning Representations*, 2017.

[19] M. Arjovsky, S. Chintala and L. Bottou. Wasserstein GAN. *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 214-223.

[20] D. Bahdanau, K. Cho and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *3rd International Conference on Learning Representations*, 2015.

[21] Y Zhu, C. Zhu, J. Tan, Y. Tan and L. Rao. Anomaly detection and condition monitoring of wind turbine gearbox based on LSTM-FS and transfer learning. *Renewable Energy vol. 189*, 2022, pp. 90-103.

[22] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko and T.J.A. Darrell. Deep Domain Confusion: Maximizing for Domain Invariance. *CoRR:1412.3474*, 2014.

[23] M. Long, Y. Cao, J. Wang and M.I. Jordan. Learning Transferable Features with Deep Adaptation Networks. *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, 2015, pp. 97-105.

[24] B. Sun, J. Feng and K. Saenko. Return of Frustratingly Easy Domain Adaptation. *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 2058–2065.

[25] B. Sun and K. Saenko. Deep CORAL: Correlation Alignment for Deep Domain Adaptation. *Computer Vision – ECCV 2016 Workshops*, 2016, 443-450.

[26] C. Bishop. *Neural networks for pattern recognition*. Oxford University Press, USA, 1995.