UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Ingeniería de Sistemas y Automática

Evaluación de técnicas para la detección de anomalías de funcionamiento mecánico mediante el procesamiento de la información suministrada por sensores de aceleración.

Trabajo Fin de Máster

Máster Universitario en Automática e Informática Industrial

AUTOR/A: Martínez Díaz, María Ascensión

Tutor/a: Simó Ten, José Enrique

Cotutor/a: Navarro Herrero, José Luís

CURSO ACADÉMICO: 2023/2024

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Universitat Politècnica de València

Evaluación de técnicas para la detección de anomalías de funcionamiento mecánico mediante el procesamiento de la información suministrada por sensores de aceleración

Trabajo Fin de Máster

Máster Universitario en Automática e Informática Industrial

Autor: María Ascensión Martínez Díaz

Tutor: José Enrique Simó Ten

Curso 2023-2024

Resum

En aquest treball es proposa l'avaluació de diferents tècniques de processament d'informació, obtinguda mitjançant instrumentació, amb l'objectiu de la detecció d'anomalies de funcionament mecànic.

Com a punt de partida s'utilitzaran dades obtingudes mitjançant el mostreig d'un acceleròmetre muntat en un pòrtic XYZ, el moviment del qual està controlat amb motors pas a pas. Als datos obtinguts per la instrumentació s'aplicaran tècniques de processament digital, Big Data i xarxes neuronals per a determinar la possibilitat de predicció en l'aparició de funcionaments anòmals en la màquina.

Per a l'avaluació experimental de les tècniques de processament d'informació s'injectaran diferents tipus de fallades (soroll d'alta i baixa freqüència i capbussades) sota diferents escenaris de funcionament.

Els resultats obtinguts es compararan quantitativa i qualitativament, avaluant l'eficàcia de cada enfocament. Les conclusions derivades d'aquest treball contribuiran a l'avanç en la detecció primerenca d'anomalies en màquines de tall amb làser, proporcionant aplicacions pràctiques per al manteniment predictiu i la millora de l'eficiència operativa.

Paraules clau: detecció d'anomalies, màquines de tall amb làser, Big Data, xarxes neuronals, manteniment predictiu

---

Resumen

En este trabajo se propone la evaluación de diferentes técnicas de procesamiento de información, obtenida mediante instrumentación, con el objetivo de la detección de anomalías de funcionamiento mecánico.

Como punto de partida se usarán datos obtenidos mediante el muestro de un acelerómetro montado en un pórtico XYZ cuyo movimiento está controlado con motores paso a paso. A los datos obtenidos por la instrumentación se aplicarán técnicas de procesamiento digital, Big Data y redes neuronales para determinar la posibilidad de predicción en la aparición de funcionamientos anómalos en la máquina.

Para la evauación experimental de las técnicas de procesamiento de información se inyectarán diferentes tipos de fallos (ruido de alta y baja frecuencia y cabeceos) bajo diferentes escenarios de funcionamiento.

Los resultados obtenidos se comparan cuantitativa y cualitativamente, evaluando la eficacia de cada enfoque. Las conclusiones derivadas de este trabajo contribuirán al avance en la detección temprana de anomalías en máquinas de corte con láser, brindando aplicaciones prácticas para el mantenimiento predictivo y la mejora de la eficiencia operativa.

Palabras clave: detección de anomalías, máquinas de corte con láser, Big Data, redes neuronales, mantenimiento predictivo

---

Abstract

In this work, the evaluation of different information processing techniques obtained through instrumentation is proposed, aiming at the detection of mechanical malfunction anomalies.

As a starting point, data obtained by sampling an accelerometer mounted on an XYZ gantry, whose motion is controlled by stepper motors, will be used. Digital processing,

Big Data, and neural network techniques will be applied to the instrumentation data to determine the possibility of predicting abnormal machine behaviors.

For the experimental evaluation of information processing techniques, various types of faults (high and low-frequency noise, and tilting) will be injected under different operating scenarios.

The obtained results will be quantitatively and qualitatively compared, assessing the effectiveness of each approach. The conclusions derived from this work will contribute to the advancement in early detection of anomalies in laser cutting machines, providing practical applications for predictive maintenance and improving operational efficiency.

Key words: malfunction anomalies, laser cutting machines, Big Data, neural networks, predictive maintenance

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

La Cuarta Revolución Industrial, conocida como Industria 4.0 [1], está en pleno apogeo y tiene un gran impacto en empresas industriales de diferentes sectores, como la automatización, el sector de la automoción, la química, la construcción, los servicios al consumidor, la energía, las finanzas, salud, tecnologías de la información y telecomunicaciones. La cantidad de datos industriales generados por controladores de máquinas, sensores, sistemas de fabricación, etc. está creciendo exponencialmente. El Big Data [2] consiste en recopilar y analizar estos grandes volúmenes de datos, en principio sin ninguna relación alguna, para convertirlos en información procesable y útil. Así, mediante el empleo de sistemas inteligentes permite transformar esta enorme cantidad de datos en conocimiento, representado por modelos matemáticos y estadísticos. El aprendizaje automático, conocido como Machine Learning, es una parte de la inteligencia artificial que permite construir esos modelos. El aprendizaje automático comprende varios métodos que permiten esta transformación de tal manera que los sistemas de software resultantes puedan proporcionar información útil para tomar decisiones óptimas y elevar la productividad. Estas decisiones están presentes en diferentes sectores industriales en problemas como diagnóstico, mantenimiento predictivo, monitorización de acondicionamiento, gestión de la salud de activos, etc.

Este documento tiene como objetivo principal mostrar cómo se pueden aplicar métodos de Machine Learning y Big Data para abordar problemas industriales del mundo real y el impacto que tienen. En concreto, estas técnicas se aplicarán al caso de estudio de una máquina de corte con láser. El trabajo desarrollado a lo largo de estos últimos meses se ha focalizado en desarrollar y aplicar una solución para recoger las grandes cantidades de datos que producen los sensores de la maquinaria empleada y procesarla de tal manera que se puedan realizar mediante el empleo de redes neuronales, diagnósticos en tiempo real del estado actual de los elementos que componen la máquina industrial y así, ser capaces de poder detectar posibles fallos de los elementos que la componen.

1.1 Motivación

La elección del tema de este trabajo ha sido impulsada por una combinación de motivaciones personales y profesionales. Como he mencionado anteriormente la Industria 4.0 está transformando radicalmente diversos sectores industriales mediante la integración de tecnologías avanzadas, como el Big Data y el Machine Learning. Esta revolución representa un cambio paradigmático en la forma en que las empresas abordan la producción y la toma de decisiones.

Desde una perspectiva personal, la inmersión en el fascinante mundo de la Industria 4.0 ha sido alimentada por la apasionante búsqueda de comprender y contribuir al

avance de la tecnología. La creciente interconexión de sistemas y la explosión de datos industriales ofrecen una oportunidad única para abordar desafíos técnicos significativos. La motivación intrínseca para explorar cómo el Machine Learning y el Big Data pueden mejorar procesos industriales específicos ha sido un factor impulsor clave.

Desde un enfoque más profesional, la elección de centrarse en la aplicación de métodos de Machine Learning y Big Data en el contexto de una máquina de corte con láser responde a la necesidad imperante en la industria de optimizar la eficiencia y la fiabilidad de equipos críticos. La identificación y resolución de problemas en tiempo real, como el diagnóstico de fallos y el mantenimiento predictivo, son aspectos cruciales para la competitividad y sostenibilidad de las empresas en la era de la Industria 4.0.

La relevancia de este tema se manifiesta en la capacidad de estas tecnologías para transformar enormes cantidades de datos en conocimiento procesable. En particular, el enfoque en el caso de estudio de la máquina de corte con láser subraya la aplicabilidad práctica de las soluciones propuestas. El objetivo es desarrollar una metodología que permita no solo recoger datos, sino también procesarlos mediante redes neuronales para lograr diagnósticos en tiempo real, mejorando así la capacidad de detectar y abordar posibles fallos en los elementos de la máquina.

En resumen, la elección de este tema no solo se basa en el interés personal en la vanguardia tecnológica, sino también en la necesidad evidente de soluciones innovadoras en el ámbito industrial. Este trabajo aspira a contribuir al avance de la Industria 4.0 al aplicar de manera concreta el Machine Learning y el Big Data en un entorno industrial real, como la máquina de corte con láser, con el objetivo final de mejorar la eficiencia y la fiabilidad de los procesos industriales.

1.2 Objetivos

El presente trabajo tiene como objetivo principal aplicar de manera efectiva las tecnologías de Machine Learning y Big Data en el contexto de una máquina de corte con láser, en el marco de la Industria 4.0. Este propósito se desglosa en los siguientes objetivos específicos:

1. **Desarrollar una metodología de recopilación de datos:** Diseñar un sistema eficiente para la recopilación y almacenamiento de grandes volúmenes de datos generados por sensores y controladores de la máquina de corte con láser.

2. **Implementar técnicas de procesamiento de Big Data:** Aplicar herramientas y técnicas de Big Data para analizar y transformar los datos recopilados en información procesable, identificando patrones y tendencias relevantes.

3. **Aplicar algoritmos de Machine Learning:** Utilizar algoritmos de Machine Learning, especialmente redes neuronales, para desarrollar modelos predictivos y diagnósticos capaces de evaluar en tiempo real el estado de los componentes de la máquina y detectar posibles fallos.

4. **Realizar experimentos de inyección de fallos:** Evaluar la robustez y eficacia de la metodología y modelos desarrollados mediante la introducción controlada de diferentes tipos de fallos (ruido de alta y baja frecuencia, cabeceos) en la máquina de corte con láser en diversos escenarios operativos.

5. **Comparar resultados y evaluar eficacia:** Cuantificar y cualificar la eficacia de la metodología propuesta mediante la comparación de resultados obtenidos con y sin

la aplicación de las técnicas de Machine Learning y Big Data, destacando la mejora en la detección temprana de fallos.

6. **Desarrollar un sistema de filtrado de señales digitales:** Diseñar e implementar un sistema de filtrado digital que permita mejorar la calidad de las señales obtenidas del acelerómetro en el pórtico XYZ que se utilizará como prototipo, reduciendo el ruido y destacando las características relevantes para el análisis.
7. **Optimizar la integración de datos filtrados:** Mejorar la integración de los datos filtrados en la metodología de procesamiento de Big Data y Machine Learning, asegurando que los modelos construidos se beneficien de señales más limpias y precisas.
8. **Evaluar el impacto del filtrado en la precisión del diagnóstico:** Realizar experimentos comparativos para evaluar cómo el filtrado de señales digitales afecta la precisión de los modelos de Machine Learning en la detección de anomalías, destacando su contribución al aumento de la confiabilidad del sistema.

Estos objetivos específicos se plantean de manera clara, factible y medible, contribuyendo al logro del propósito general del trabajo y proporcionando una aproximación integral a la problemática abordada en el contexto de la Industria 4.0.

1.3 Impacto esperado

El desarrollo del producto resultante de este Trabajo de Fin de Máster (TFM) se proyecta con la visión de generar un impacto significativo en diversos usuarios y en la abordabilidad de problemas contemporáneos, alineándose con los Objetivos de Desarrollo Sostenible (ODS).

Para los **Operadores y Mantenedores de la Máquina de Corte con Láser**, la implementación de diagnósticos en tiempo real implica una optimización significativa del mantenimiento. La capacidad de anticipar y abordar problemas antes de que se conviertan en fallos críticos reduce el tiempo de inactividad no planificado, mejorando así la eficiencia operativa y la rentabilidad. Además, el acceso a información en tiempo real sobre el estado de los componentes de la máquina facilita la toma de decisiones informadas.

Los **Directivos y Responsables de Producción** se benefician de la generación de informes detallados y análisis derivados de los datos recopilados. Esta información respalda la toma de decisiones estratégicas, contribuyendo a la eficiencia global de la planta y permitiendo un enfoque más proactivo en la gestión de la producción.

Para los **Clientes de la Empresa**, la garantía de calidad se ve reforzada. La mejora en la fiabilidad de la maquinaria se traduce en una mayor calidad en los productos finales, fortaleciendo la reputación de la empresa en términos de cumplimiento de plazos y estándares de calidad, lo que se traduce en una mayor satisfacción del cliente.

Desde una perspectiva más amplia, el impacto del proyecto se extiende al ámbito de la **Sostenibilidad y el Medio Ambiente**. La capacidad de prevenir fallos y optimizar la producción contribuye a la reducción de residuos y al minimizar el consumo energético. Este enfoque alinea el proyecto con los Objetivos de Desarrollo Sostenible (ODS), especialmente con el ODS 12 (Producción y Consumo Responsables).

En resumen, este trabajo busca proporcionar beneficios tangibles para una variedad de usuarios, desde operadores y directivos hasta clientes y, de manera más amplia, contribuir a los objetivos de sostenibilidad establecidos a nivel global. La mejora en la eficiencia operativa y la calidad del producto se convierten en pilares fundamentales para un impacto positivo en el entorno industrial y más allá.

1.4 Metodología

La metodología propuesta para este trabajo está diseñada para abordar de manera eficaz los objetivos establecidos, proporcionando una guía estructurada para la realización del proyecto. La selección de esta metodología se ajusta específicamente al ámbito del trabajo, orientándolo hacia el desarrollo e implementación de soluciones en el contexto de la Industria 4.0. Se seguirá una metodología en espiral, que se caracteriza por iteraciones repetitivas a través de las siguientes etapas:

1. **Definición de objetivos específicos.** En esta fase inicial, se realiza una revisión detallada de los objetivos generales del TFM. Se desglosan estos objetivos en metas específicas y alcanzables, estableciendo así los hitos que guiarán el desarrollo del trabajo.

2. **Revisión bibliográfica y marco teórico.** Se lleva a cabo una exhaustiva revisión de la literatura existente relacionada con las tecnologías de Machine Learning, Big Data, procesamiento de señales digitales y su aplicación en el ámbito industrial. Esta revisión proporciona el contexto teórico necesario para comprender las metodologías y enfoques existentes en la detección de anomalías en maquinaria industrial.

3. **Definición de la arquitectura del sistema.** Se procede a la definición de la arquitectura del sistema, detallando las interacciones entre los diferentes componentes, como los sensores, el hardware (Arduino Mega 2560), y el software de procesamiento de datos y modelos de Machine Learning. Esta fase sienta las bases para la implementación práctica de la metodología.

4. **Desarrollo del sistema de recopilación de datos.** Se implementa un sistema eficiente de recopilación de datos utilizando el hardware seleccionado (acelerómetro). Se garantiza la correcta captura de las señales de aceleración generadas por el prototipo de la máquina de corte con láser.

5. **Procesamiento de señales digitales y filtrado.** Se aplican técnicas de procesamiento de señales digitales, incluyendo el desarrollo de sistemas de filtrado digital. La implementación en C++ facilita la extracción de muestras de aceleración y mejora la calidad de las señales para su posterior análisis.

6. **Implementación de técnicas de Big Data.** Se utilizan herramientas y técnicas de Big Data para analizar y transformar los datos recopilados en información procesable. La escalabilidad y eficiencia en el manejo de grandes volúmenes de datos se convierten en aspectos clave en esta fase.

7. **Desarrollo y entrenamiento de modelos de Machine Learning.** Se implementan y entrenan modelos de Machine Learning, especialmente redes neuronales, utilizando el lenguaje de programación Python. Estos modelos se configuran para realizar diagnósticos en tiempo real del estado de los elementos de la máquina, contribuyendo así a la detección temprana de anomalías.

8. **Experimentación y evaluación.** Se lleva a cabo una serie de experimentos que incluyen la inyección controlada de fallos en diferentes escenarios operativos. Se comparan cuantitativa y cualitativamente los resultados obtenidos con y sin la aplicación de las técnicas de procesamiento de datos y modelos de Machine Learning.

9. **Análisis y discusión de resultados.** Se analizan los resultados obtenidos en los experimentos, evaluando la eficacia de la metodología propuesta. Se discuten las

implicaciones prácticas y se establecen conclusiones derivadas de los hallazgos obtenidos.

10. **Redacción de la memoria del TFM.** Se documentan los pasos seguidos, los resultados obtenidos y las conclusiones derivadas de la investigación y experimentación.

Esta metodología en espiral proporciona un enfoque sistemático y coherente para alcanzar los objetivos planteados en el TFM, guiando el proceso de investigación y desarrollo de manera efectiva y alineada con la orientación específica del proyecto en el ámbito de la Industria 4.0.

1.5 Estructura

Este documento está estructurado en 8 capítulos. A continuación, se presenta una breve descripción de todos estos:

1. **Introducción.** En este capítulo se introduce esta memoria proporcionando la motivación que existe tras este desarrollo. Además, se listan los principales objetivos y el impacto esperado del trabajo. Para ello se ha necesitado seguir una metodología también detallada en este capítulo.

2. **Estado del arte.** En este apartado se describe el estado actual de los métodos de Machine Learning y Big Data que se aplican en la resolución de problemas industriales similares al planteado en este trabajo.

3. **Análisis del problema.** En este partiendo de los requisitos para abordar las necesidades específicas, se analizan las distintas soluciones posibles para resolver el problema dado. Así, posteriormente exponemos un análisis de la solución más apropiada a nuestro caso de estudio.

4. **Tecnología utilizada.** Se explican qué tecnologías empleadas y en qué consisten para resolver el problema planteado.

5. **Solución propuesta.** Aquí se detalla el diseño de la solución elegida para abordar el problema.

6. **Desarrollo e implementación.** Describe de forma detallada el desarollo e integración de los distintos componentes que conforman la solución final.

7. **Pruebas y resultados.** Muestra los resultados de las pruebas realizadas para verificar el correcto funcionamiento de las tareas desarrolladas.

8. **Conclusiones.** Se resumen el trabajo desarrollado y las conclusiones que se han podido extraer de toda la realización de este trabajo. También, se propone una lista de quehaceres que quedan como trabajo futuro.

Al final de esta memoria encontramos las referencias bibliográficas empleadas en el desarrollo de este trabajo. También, se incluye la relación del proyecto llevado a cabo con los ODS (Objetivos de Desarrollo Sostenible) de la Agenda 2030.

CAPÍTULO 2
Estado del arte

La aplicación de métodos de Machine Learning y Big Data en la resolución de problemas industriales ha experimentado un notable crecimiento en las últimas décadas. Antes de adentrarnos en la propuesta específica para la máquina de corte con láser, es crucial revisar la evolución tecnológica en aplicaciones similares. A continuación, se presentan algunas de las iniciativas más destacadas en el ámbito del diagnóstico industrial.

En el trabajo de Yang et al. (2019) [5], se aborda la monitorización en tiempo real de maquinaria industrial mediante el análisis de datos sensoriales. El objetivo principal de este estudio era desarrollar un sistema capaz de identificar patrones anómalos en los datos generados por sensores de maquinaria en funcionamiento. Utilizando algoritmos de Machine Learning, se logró detectar desviaciones significativas en los patrones de operación, permitiendo así prever posibles fallos o deterioros en los componentes de la maquinaria.

La contribución clave de esta investigación radica en su enfoque proactivo para el mantenimiento industrial. Al detectar anomalías en tiempo real, se facilita la intervención preventiva, evitando así tiempos de inactividad no planificados y reduciendo los costes asociados al mantenimiento correctivo. Este enfoque de monitorización en tiempo real sienta las bases para la propuesta de implementación de técnicas similares en la máquina de corte con láser, permitiendo una respuesta ágil a cualquier cambio en el estado de la maquinaria.

En el estudio de Liu et al. (2022) [6], se exploró la aplicación de técnicas avanzadas de Machine Learning para el mantenimiento predictivo en máquinas industriales. La investigación se centró en la capacidad de prever posibles fallos en componentes clave de la maquinaria, permitiendo una planificación más eficiente de las actividades de mantenimiento. Utilizando algoritmos de Deep Learning, se logró identificar patrones complejos en los datos operativos, lo que resultó en una mayor precisión en la predicción de fallos y la optimización de los intervalos de mantenimiento.

Esta investigación destaca la importancia de la anticipación en el mantenimiento industrial. Al prever el desgaste o deterioro de componentes, las empresas pueden evitar paradas no planificadas y maximizar la disponibilidad operativa de sus máquinas. La aplicación de técnicas de Machine Learning para el mantenimiento predictivo proporciona un enfoque proactivo y basado en datos, aspectos clave que se consideran en la propuesta para la máquina de corte con láser, buscando mejorar la eficiencia y la fiabilidad de la operación industrial.

En el ámbito de la monitorización y diagnóstico del estado de máquinas CNC [3], destaca el trabajo de Li et al. (2019) [7], donde se implementan técnicas avanzadas de Machine Learning para evaluar y prever el estado operativo de las herramientas de las máquinas CNC. Este estudio se centra en la detección temprana de posibles anomalías y

desgastes, el monitoreo continuo del rendimiento y la optimización de la eficiencia en la operación de máquinas CNC.

Li et al. (2019) [7] destacan la importancia de la monitorización constante para prevenir fallos inesperados y mejorar la eficacia en el uso de máquinas CNC. El uso de algoritmos de Machine Learning en este contexto permite realizar diagnósticos en tiempo real, identificar patrones sutiles que indican desviaciones del funcionamiento óptimo y mejorar la toma de decisiones para el mantenimiento predictivo.

Este enfoque en el estado operativo de las máquinas CNC complementa la propuesta del presente trabajo, proporcionando un contexto sólido y precedentes en la aplicación de técnicas de Machine Learning para mejorar la eficiencia y la fiabilidad en maquinaria industrial específica. La adaptación de estas metodologías al caso de estudio de la máquina de corte con láser refleja la continua evolución de estas tecnologías en la mejora de procesos industriales.

Dentro de las alternativas existentes, se identifican enfoques basados en reglas heurísticas y sistemas expertos, los cuales han sido utilizados en el pasado para el diagnóstico industrial. Sin embargo, la limitación de estos métodos radica en su capacidad para adaptarse a patrones complejos y cambiantes presentes en datos de maquinaria moderna.

La elección de utilizar Machine Learning y Big Data se justifica por la capacidad inherente de estas tecnologías para aprender de patrones complejos y adaptarse a cambios en tiempo real. La flexibilidad y la escalabilidad que ofrecen estas técnicas son esenciales para abordar la complejidad de los datos generados por los sensores de la máquina de corte con láser.

2.1 Crítica al estado del arte

En el análisis crítico del estado del arte, se destaca un trabajo previo desarrollado en la ETSINF [12]. Este trabajo se centra en la detección de fallos en rodamientos mediante el uso de algoritmos de Machine Learning, utilizando datos de vibraciones obtenidos de rodamientos en diferentes estados a través del Case Western Reserve University Bearing Data Center Website. El objetivo principal era entrenar y validar algoritmos para clasificar y diferenciar entre distintos fallos en los rodamientos, utilizando características extraídas de las señales vibratorias.

A pesar de la relevancia de este trabajo, se identifican limitaciones y lagunas que justifican el desarrollo del presente TFM. En primer lugar, el enfoque del trabajo anterior se concentra específicamente en rodamientos, dejando un espacio no abordado en el diagnóstico en tiempo real de maquinaria industrial más compleja, como una máquina de corte con láser.

Además, la metodología utilizada en el trabajo previo se centra en algoritmos específicos, como árboles de decisión, k vecinos más cercanos, y máquinas de vector soporte. Esta limitación plantea la necesidad de explorar enfoques más avanzados y adaptables, como las redes neuronales, para abordar la complejidad de la maquinaria industrial en estudio.

La diferencia clave con el presente TFM radica en la aplicación de técnicas más avanzadas de Machine Learning, específicamente el entrenamiento de redes neuronales. Estas técnicas permiten procesar datos de vibración en tiempo real y obtener una comprensión más profunda del estado de la máquina. Mientras que el trabajo anterior se enfoca en el análisis de rodamientos, el presente TFM se dirige hacia una aplicación más amplia en entornos industriales, introduciendo una solución más integral y avanzada para el diagnóstico en tiempo real de maquinaria compleja.

En conclusión, este análisis crítico resalta las limitaciones del trabajo previo y justifica la necesidad de desarrollar un enfoque más avanzado y adaptado a la complejidad de la maquinaria industrial, posicionando así al presente TFM como una contribución significativa para llenar las lagunas identificadas en el estado del arte.

2.2 Propuesta

La esencia de este trabajo radica en su capacidad para abordar y llenar un espacio de conocimiento tecnológico específico, destacándose por su aporte diferenciador y su potencial mejora en comparación con las soluciones ya existentes.

El ámbito de estudio se sitúa en la intersección de la Industria 4.0, el Machine Learning y el Big Data, con un enfoque específico en la aplicación de estas tecnologías al diagnóstico en tiempo real de maquinaria industrial, en particular, una máquina de corte con láser. Este espacio de conocimiento es crucial en el contexto actual de la Cuarta Revolución Industrial, donde la optimización de procesos y la toma de decisiones basada en datos son fundamentales.

Lo destacable de este trabajo es su enfoque integral. El trabajo desarrollado no aborda fragmentos específicos de la cadena de valor industrial, este TFM propone una metodología completa para la recopilación, procesamiento y aplicación de datos en tiempo real, específicamente adaptada al caso de estudio de una máquina de corte con láser.

También otro aspecto destacable es la implementación de técnicas de Machine Learning. Este trabajo se orienta a la creación de modelos que permitan un diagnóstico en tiempo real del estado actual de los elementos de la maquinaria, brindando una comprensión más profunda y proactiva de los posibles fallos. Además, la propuesta busca la eficiencia no solo en la detección de problemas, sino también en la optimización del consumo energético y la reducción de residuos.

Si bien este TFM se adentra en un espacio de conocimiento innovador, reconoce la base sólida de soluciones existentes. La propuesta se asemeja a enfoques convencionales al emplear tecnologías establecidas como el Big Data y el Machine Learning. Sin embargo, la originalidad radica en la combinación de estas tecnologías de una manera específica para abordar los desafíos únicos presentes en el diagnóstico en tiempo real de la maquinaria industrial.

En última instancia, este trabajo no solo busca introducir una aportación novedosa, sino también servir como una prueba de concepto práctica y aplicable. La combinación original de soluciones existentes, adaptadas y mejoradas para el contexto específico de una máquina de corte con láser, busca la capacidad de aplicar de forma efectiva la Industria 4.0 en entornos industriales concretos.

CAPÍTULO 3
Análisis del problema

En este capítulo, se realiza un análisis profundo del problema identificado, considerando las limitaciones y oportunidades derivadas del estado del arte. Se busca establecer un marco claro para el desarrollo del TFM, destacando las áreas de enfoque y las necesidades específicas a abordar.

El estudio del estado del arte ha revelado ciertos desafíos en la monitorización y diagnóstico de maquinaria industrial. Estos desafíos sirven como base para definir el alcance y los objetivos del TFM:

- **Monitorización Integral.** La falta de soluciones integrales que aborden la monitorización de toda la maquinaria industrial, especialmente en entornos complejos como el de una máquina de corte con láser.
- **Dependencia de Tecnologías Específicas.** La limitación de soluciones previas que dependen fuertemente de tecnologías o herramientas específicas, lo que puede obstaculizar la accesibilidad y la aplicabilidad en diversos entornos industriales.
- **Enfoque en componentes aislados.** La tendencia a concentrarse en la monitorización de componentes individuales, sin abordar la interconexión y el impacto de cada componente en el rendimiento general de la maquinaria.

El análisis del problema también revela oportunidades para la innovación y el desarrollo de soluciones que no solo superen los desafíos identificados sino que también exploren nuevas posibilidades:

- **Monitorización integral y en tiempo real.** La oportunidad de desarrollar una solución integral que permita la monitorización en tiempo real de toda la maquinaria industrial.
- **Flexibilidad en el procesamiento de datos.** La posibilidad de implementar algoritmos más flexibles, como el multiperceptrón multicapa y la red neuronal convolucional, para un procesamiento de datos adaptativo y eficaz en diversos entornos y tipos de maquinaria.
- **Optimización del mantenimiento predictivo.** La oportunidad de optimizar el mantenimiento predictivo, reduciendo costos asociados a paradas no planificadas y prolongando la vida útil de los equipos mediante soluciones más avanzadas y adaptativas.
- **Aplicación en diversos sectores industriales.** La capacidad de desarrollar una solución que sea adaptable a diversas maquinarias y sectores industriales, abriendo

oportunidades para su implementación en un amplio espectro de aplicaciones más allá de la máquina de corte con láser.

Con base en el análisis del problema, se definen requisitos detallados y concretos que deben ser satisfechos por las soluciones propuestas. Estos requisitos determinarán la solución a implementar y guiarán el desarrollo del TFM.

Requisitos funcionales:

1. Implementar un sistema de recopilación y almacenamiento de datos en tiempo real.
2. Implementar algoritmos de Machine Learning como técnicas de procesamiento digital para el análisis de los datos de vibración adquiridos en un entorno industrial.
3. Integrar la solución en el sistema operativo del prototipo de la máquina de corte con láser para la monitorización integral.
4. Automatizar el proceso de diagnóstico en tiempo real, identificando posibles fallos o anomalías.
5. Permitir la adaptabilidad y flexibilidad de la solución a diferentes maquinarias y entornos industriales.

Requisitos no funcionales:

1. Garantizar la eficiencia y velocidad en la toma de decisiones.
2. Asegurar la escalabilidad para manejar grandes volúmenes de datos.
3. Cumplir con estándares de seguridad y privacidad en el manejo de datos industriales.
4. Ser compatible con diversas plataformas tecnológicas y sistemas operativos industriales.

Se presenta el modelo conceptual en la Figura 3.1 que representa visualmente la estructura y las relaciones clave entre los componentes de la solución propuesta para la monitorización y diagnóstico en tiempo real de la máquina de corte con láser en un entorno industrial. El modelo se organiza en módulos interconectados que abordan distintos aspectos del problema identificado.

Componentes del modelo:

1. **Sistema de recopilación y procesamiento digital de los datos adquiridos.** Este módulo se encarga de la captura y almacenamiento eficiente de datos provenientes de los sensores de la máquina. Incluye la implementación de un sistema en tiempo real para garantizar la adquisición constante de información relevante. Además, se prueba a aplicar distintos filtros y procesos digitales de la señales de aceleración muestreadas para sacar las características con las que entrenar los modelos de IA.
2. **Algoritmos de Machine Learning.** Incluye algoritmos avanzados de Machine Learning, como el multiperceptrón multicapa y la red neuronal convolucional. Estos algoritmos procesan los datos recopilados y generan modelos predictivos para la detección de fallos y la monitorización en tiempo real.

3. **Integración en el sistema operativo de la máquina.** Este componente asegura la integración efectiva de la solución en el sistema operativo de la máquina de corte con láser. Implica la adaptación de la solución al entorno industrial específico y la comunicación fluida con los sistemas existentes.

4. **Automatización del Diagnóstico en Tiempo Real.** El módulo de automatización se encarga de realizar diagnósticos en tiempo real, identificando posibles fallos o anomalías en los componentes de la máquina. Este proceso se realiza de manera continua para asegurar una monitorización constante.

5. **Adaptabilidad y flexibilidad.** Garantiza que la solución sea adaptable a diferentes maquinarias y entornos industriales. Este componente permite la configuración de parámetros y modelos según las características específicas de la maquinaria a monitorizar.

Según el modo de funcionamiento las relaciones entre componentes es la siguiente. Los datos recopilados por el sistema de recopilación y procesamiento digital de señal provenientes del muestreo de las señales de aceleración, son alimentados a los algoritmos de Machine Learning para su entrenamiento. Los resultados de los algoritmos se integran en el sistema operativo de la máquina, permitiendo una comunicación bidireccional para la toma de decisiones en tiempo real. La automatización del diagnóstico utiliza los modelos generados por los algoritmos para identificar y notificar posibles fallos.

La adaptabilidad y flexibilidad se aplican en todos los módulos, permitiendo ajustes y configuraciones específicas para diferentes contextos.

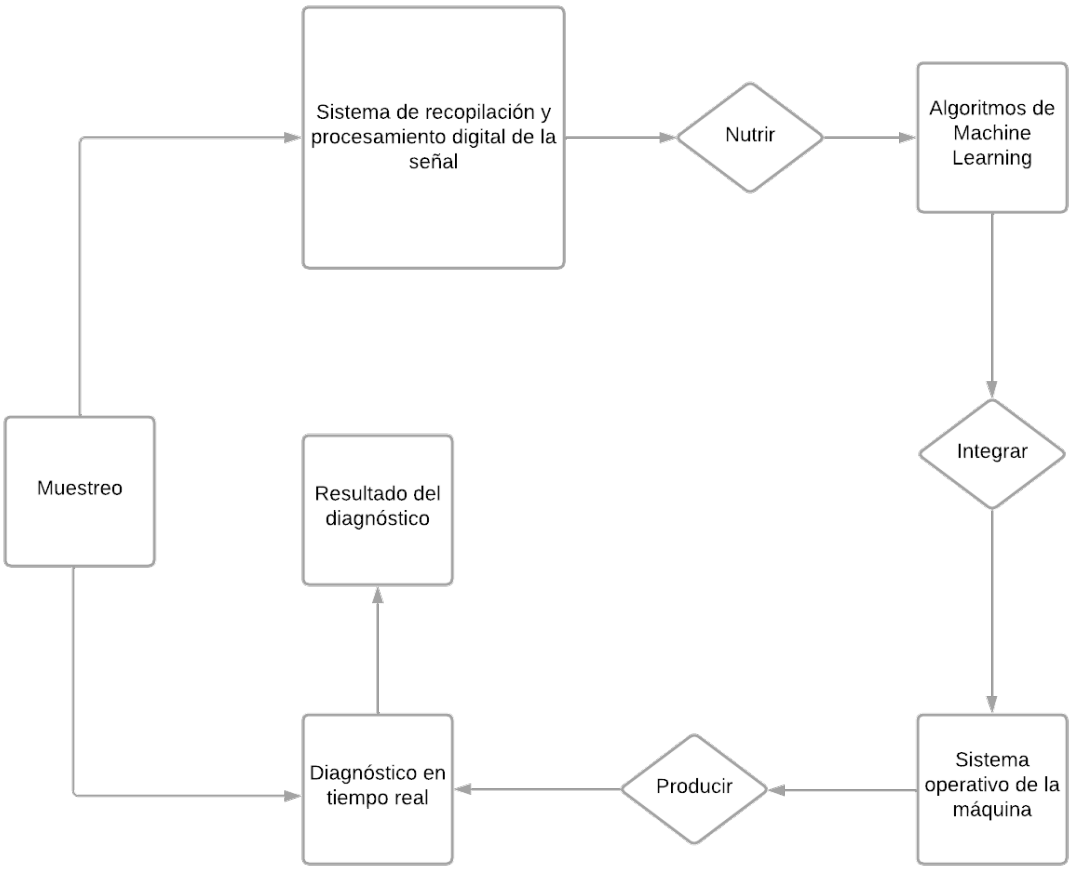


Figura 3.1: Modelado conceptual del problema.

3.1 Identificación y análisis de soluciones posibles

En la fase de identificación y análisis de soluciones, se reconocen múltiples enfoques para abordar el problema de la monitorización y diagnóstico en tiempo real de la máquina de corte con láser. Dado que los requisitos establecidos definen la dirección, se presenta un análisis detallado de las posibles soluciones, evaluando sus pros y contras, con el objetivo de establecer un criterio de selección que oriente la elección final.

1. **Solución A: Implementación de algoritmos locales.** Desarrollar algoritmos específicos para el diagnóstico de componentes locales de la máquina, abordando fallos específicos en rodamientos, sensores, y otros elementos individualmente.
   - **Pros:**
     - Enfoque detallado y especializado.
     - Posibilidad de optimización para problemas específicos.
   - **Contras:**
     - Limitado a la detección de fallos individuales.
     - Puede no capturar interacciones entre componentes.
2. **Solución B: Implementación de un sistema de redes neuronales.** Utilizar redes neuronales para el análisis de datos, permitiendo la captura de relaciones complejas entre los distintos componentes de la máquina.
   - **Pros:**
     - Capacidad para aprender patrones complejos.
     - Potencial para adaptarse a diferentes situaciones.
   - **Contras:**
     - Requiere grandes conjuntos de datos para entrenamiento.
     - Mayor complejidad en la implementación y configuración.
3. **Solución C: Integración de sensores avanzados.** Utilizar sensores más avanzados y precisos para la recopilación de datos, mejorando la calidad y cantidad de información disponible. Por ejemplo, el ADXL355 es un acelerómetro triaxial de alta precisión fabricado por Analog Devices. Este sensor es conocido por su excelente estabilidad a largo plazo y bajo ruido, lo que lo hace ideal para aplicaciones industriales que requieren una alta precisión en la medición de vibraciones y movimientos.
   - **Pros:**
     - Datos más precisos y detallados.
     - Potencial para detectar problemas incipientes.
   - **Contras:**
     - Aumento de costes asociados a la adquisición y mantenimiento de sensores.

La elección de la solución adecuada se basará en criterios fundamentales que incluyen eficacia, costes, complejidad de implementación y adaptabilidad a diferentes entornos industriales.

Tras una evaluación exhaustiva, la solución seleccionada es la **Implementación de un Sistema de Redes Neuronales**. Esta elección se fundamenta en el enfoque puramente basado en el aprendizaje automático, que busca descubrir patrones complejos.

En conclusión, este capítulo ofrece una comprensión profunda del problema identificado, destacando desafíos y oportunidades que orientarán el desarrollo del TFM. La especificación de requisitos y el modelado conceptual proporcionan una base sólida para la implementación de soluciones innovadoras y efectivas en el ámbito de la monitorización y diagnóstico en la Industria 4.0. Además, se justifica la utilización de técnicas de Machine Learning como la opción óptima para cumplir con los requisitos establecidos y ofrecer una solución adaptativa y eficiente.

CAPÍTULO 4
Tecnología utilizada

En este capítulo, se detallarán las herramientas y tecnologías empleadas durante el desarrollo del trabajo. La elección de estas herramientas se basa en consideraciones como la viabilidad, conocimiento personal, y estimación del coste de aprendizaje.

4.1 Python

Python[15] es un lenguaje de programación de alto nivel, interpretado y generalista. Su sintaxis simple y legible lo convierte en una elección popular para el desarrollo rápido y eficiente de software.

Python fue seleccionado como el lenguaje principal para la implementación de los modelos de Machine Learning. La versatilidad de Python, su amplia comunidad y la disponibilidad de bibliotecas especializadas como TensorFlow[1] y Scikit-learn[2] lo convierten en una opción robusta para tareas de aprendizaje automático.

4.2 C++

C++ es un lenguaje de programación de propósito general con un enfoque en el rendimiento y el control de bajo nivel. Es una extensión del lenguaje C con características de programación orientada a objetos.

C++[16] se empleó para la extracción de muestras de aceleración generadas por el prototipo de la máquina de corte con láser. La eficiencia y el control de bajo nivel proporcionados por C++ resultaron fundamentales para la interacción directa con el hardware y la obtención precisa de datos.

4.3 Arduino Mega 2560

Arduino Mega 2560 es un microcontrolador basado en el chip ATmega2560. Proporciona un entorno de desarrollo fácil de usar para proyectos de hardware y control.

Respecto a plataformas y hardware empleado para el desarrollo de este trabajo, el microcontrolador Arduino Mega 2560[17] se utilizó en el prototipo para simular y ejecutar los movimientos de una máquina de corte con láser 4.1. Proporcionando flexibilidad y facilidad de programación.

[1] https://www.tensorflow.org/
[2] https://scikit-learn.org/stable/

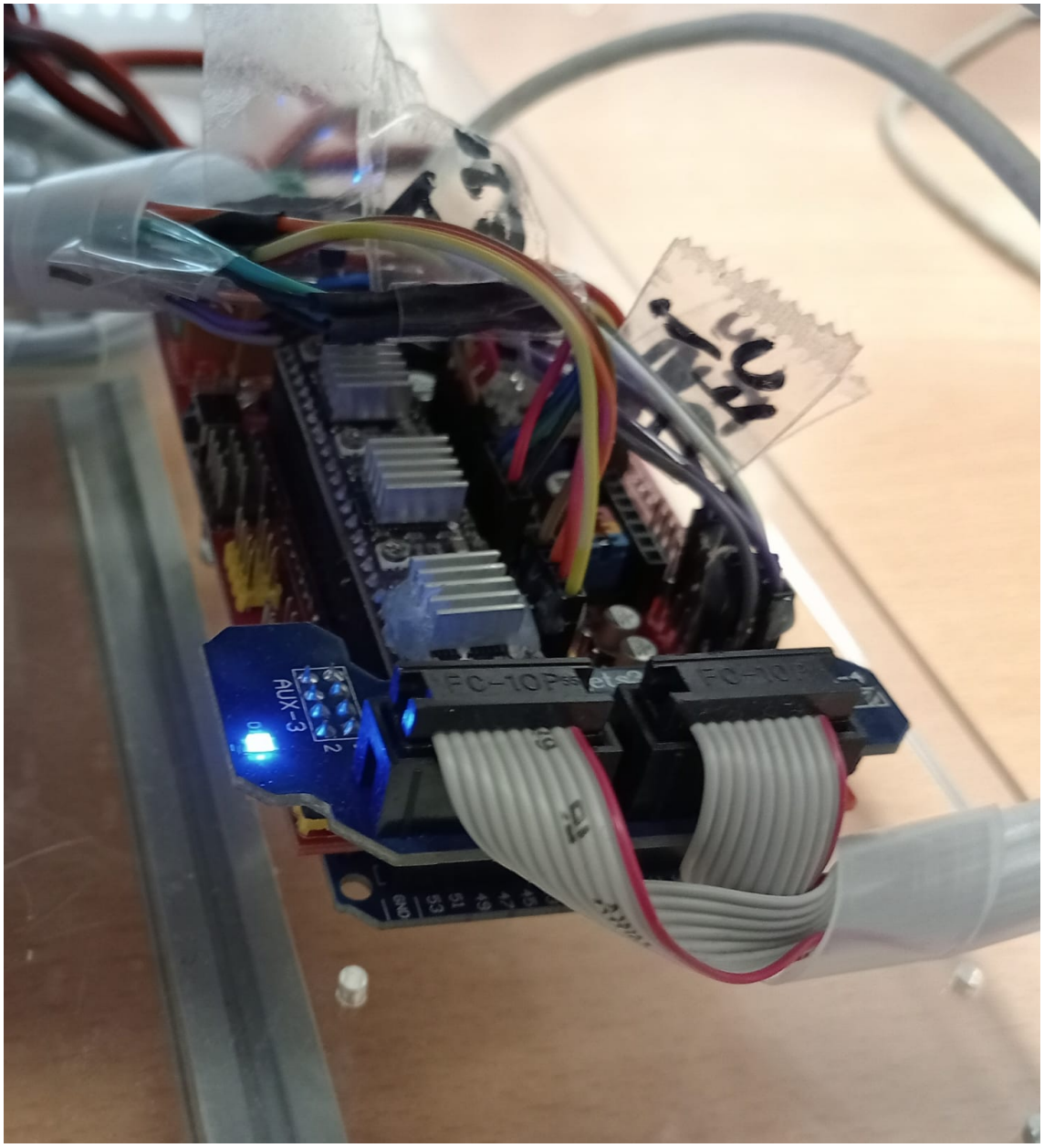


Figura 4.1: Arduino empleado en el prototipo de máquina con corte láser.

4.4 I2C

Inter-Integrated Circuit (I2C) [14] es un protocolo de comunicación de bus serie que permite la comunicación entre múltiples dispositivos periféricos con un microcontrolador o microprocesador central. Fue desarrollado por Philips Semiconductor (ahora NXP Semiconductors) en la década de 1980 y se ha convertido en un estándar de facto en la industria electrónica debido a su simplicidad y versatilidad.

El protocolo I2C utiliza dos líneas de comunicación:

1. **SDA (Serial Data Line):** Esta línea se utiliza para la transmisión de datos entre los dispositivos conectados al bus. Es bidireccional y cada dispositivo tiene una dirección única en el bus.

2. **SCL (Serial Clock Line):** Esta línea se utiliza para sincronizar la transferencia de datos entre los dispositivos. La señal en esta línea es generada por el maestro y se utiliza para sincronizar el flujo de datos en el bus.

El protocolo I2C es un protocolo de bus de corta distancia y está diseñado para la comunicación entre dispositivos dentro de un mismo circuito integrado o en una placa de circuito impreso. Se utiliza comúnmente para conectar sensores, actuadores, memorias EEPROM, convertidores analógico-digitales (ADC) y otros dispositivos periféricos a un microcontrolador o microprocesador.

Algunas características clave del protocolo I2C son su simplicidad, bajo consumo de energía y capacidad para soportar dispositivos con diferentes velocidades de transferencia de datos.

En el trabajo se ha empleado el protocolo I2C como una interfaz de comunicación para interactuar con el sensor MPU9250 y adquirir datos de aceleración necesarios para la investigación.

4.5 Acelerómetro

Un acelerómetro[18] es un sensor que mide la aceleración lineal. Detecta cambios en la velocidad y dirección de movimiento.

El acelerómetro fue fundamental para la toma de muestras de aceleración en el prototipo. Este dispositivo mide la aceleración lineal y permite capturar datos que una vez procesados se alimentan los modelos de Machine Learning.

4.6 Perceptrón Multicapa

Un perceptrón multicapa (MLP, por sus siglas en inglés, "Multilayer Perceptron") es una arquitectura de red neuronal artificial que consta de múltiples capas de neuronas (nodos) organizadas en al menos tres capas distintas (Figura 4.2): una capa de entrada, una o varias capas ocultas y una capa de salida. Es una de las arquitecturas más comunes utilizadas en aprendizaje profundo y puede abordar una variedad de tareas de aprendizaje supervisado [11].

4.6.1. Características clave del MLP

1. **Capa de Entrada.** Contiene nodos que representan las características de entrada de los datos. Cada nodo corresponde a una característica específica.

2. **Capas Ocultas.** Capas intermedias entre la capa de entrada y la capa de salida. Cada nodo en una capa oculta realiza una combinación lineal de las salidas de la capa anterior, seguida de una función de activación no lineal.

3. **Conexiones Ponderadas.** Cada conexión entre nodos tiene un peso asociado, que se ajusta durante el proceso de entrenamiento para optimizar el rendimiento de la red.

4. **Funciones de Activación.** Introducen no linealidades en la red, permitiendo al MLP aprender relaciones más complejas en los datos.

5. **Capa de Salida.** Produce la salida final de la red. El número de nodos en esta capa depende del tipo de tarea: uno para problemas de regresión y múltiples nodos para problemas de clasificación.

6. **Aprendizaje Supervisado.** Entrenado utilizando un conjunto de datos etiquetado. Se utiliza retropropagación (backpropagation) para ajustar los pesos de las conexiones y minimizar la diferencia entre las predicciones y las etiquetas reales.

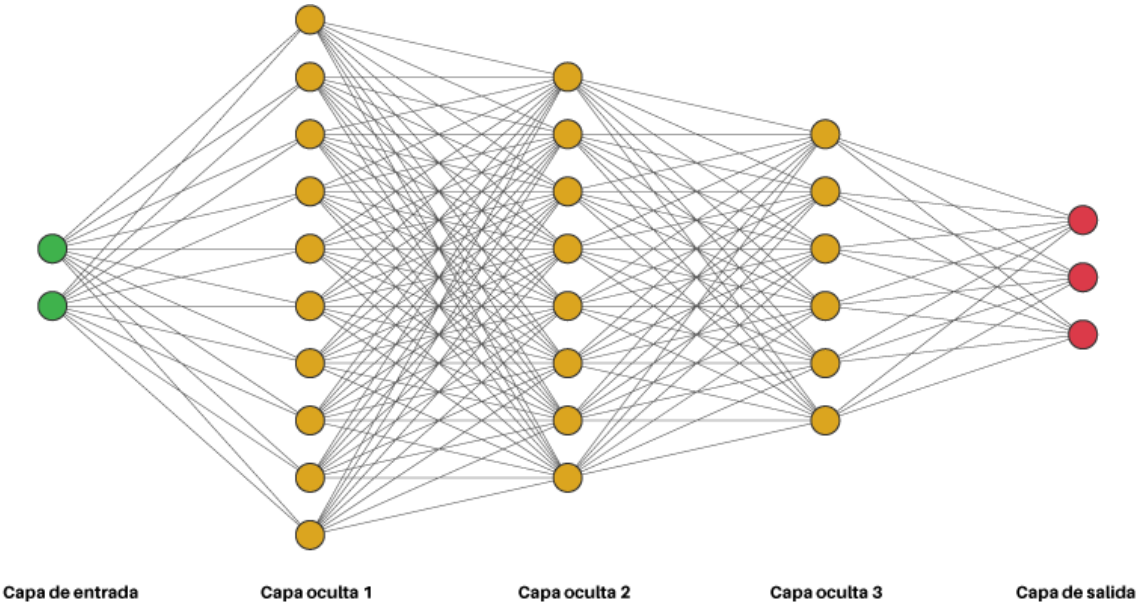


Figura 4.2: Perceptrón multicapa con tres capas ocultas (en amarillo) (Imagen de https://www.codificandobits.com/blog/que-es-una-red-neuronal/).

4.6.2. Usos Comunes del MLP

1. **Clasificación.** Clasificación binaria o multiclase. Ejemplos incluyen reconocimiento de dígitos escritos a mano, clasificación de imágenes y diagnóstico médico.

2. **Regresión.** Predicción de valores numéricos. Se utiliza en problemas como la predicción de precios de acciones, series temporales y regresión de funciones.

3. **Aproximación de Funciones.** Puede aproximar funciones no lineales y mapear relaciones complejas entre entradas y salidas.

4. **Procesamiento de Lenguaje Natural (NLP).** Se aplica a tareas de procesamiento de lenguaje natural, como clasificación de texto y modelado de lenguaje.

5. **Sistemas de Recomendación.** Utilizado en sistemas que recomiendan productos, contenido o conexiones sociales basándose en preferencias anteriores.

En resumen, el MLP es una arquitectura versátil que se utiliza en una variedad de tareas de aprendizaje supervisado. Es especialmente útil cuando se trata de problemas complejos que implican relaciones no lineales en los datos.

4.7 Red Convolucional

Una red neuronal convolucional (CNN) es un tipo de red neuronal especializada en procesar datos que tienen una estructura de cuadrícula, como imágenes. Las CNN fueron diseñadas para imitar la capacidad visual humana y han demostrado ser extremadamente eficaces en tareas relacionadas con la visión por computadora [9]. Aquí hay una explicación básica de cómo funcionan y algunos de sus usos comunes:

4.7.1. Estructura básica de una CNN

1. **Capas Convolucionales.** Estas capas aplican filtros (kernels) a regiones locales de la entrada. Los filtros detectan patrones específicos en la información, como bordes, texturas o formas. La convolución ayuda a reducir la cantidad de parámetros y mantiene la estructura espacial de la entrada.

2. **Capas de Pooling.** Estas capas reducen la dimensionalidad de la entrada mediante la reducción del tamaño de las características y la retención de la información más importante. Max pooling y average pooling son dos técnicas comunes de agrupación (pooling) para reducir la dimensionalidad de las representaciones de características y extraer las características más importantes de una región. Ambas técnicas desempeñan un papel crucial en la eficacia y eficiencia de las CNN al extraer y resumir información relevante de las características aprendidas.

3. **Capas Densas (Totalmente Conectadas).** Después de las capas convolucionales y de pooling, se pueden agregar capas densas para realizar la clasificación final. Estas capas transforman la representación espacial en una representación plana y la conectan a capas de salida (Figura 4.3).

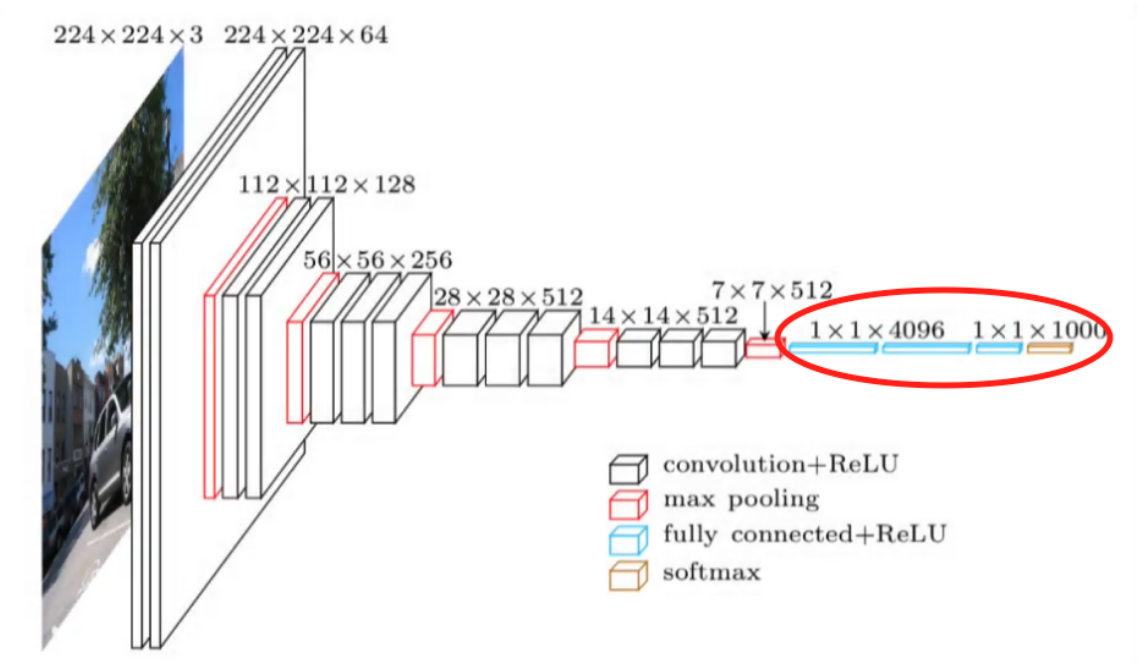


Figura 4.3: En rojo la capa de salida de una Red Convolucional (Imagen de https://www.codificandobits.com/blog/redes-convolucionales-introduccion/).

4.7.2. Proceso de Aprendizaje

Durante el entrenamiento, la red ajusta los pesos de los filtros y las conexiones para minimizar una función de pérdida. Este proceso permite a la red aprender automáticamente características y jerarquías de representación útiles para la tarea específica.

4.7.3. Usos Comunes de CNN

1. **Reconocimiento de imágenes.** Las CNN son ampliamente utilizadas para clasificar y reconocer objetos en imágenes. Ejemplos incluyen reconocimiento facial, clasificación de animales, identificación de objetos en fotografías, etc.
2. **Segmentación de imágenes.** Pueden utilizarse para dividir una imagen en segmentos y asignar etiquetas a cada segmento.
3. **Reconocimiento de patrones.** Aplicadas en campos como medicina para analizar imágenes médicas y detectar patrones anómalos o enfermedades.
4. **Procesamiento de Vídeo.** Se utilizan para analizar secuencias de video y extraer información relevante, como el seguimiento de objetos o la detección de actividades.
5. **Procesamiento de Lenguaje Natural (NLP).** Aunque inicialmente diseñadas para imágenes, las CNN también se han adaptado con éxito para tareas de procesamiento de texto en NLP.
6. **Autos de Conducción.** En el campo de vehículos autónomos, las CNN se emplean para la detección y clasificación de objetos en la carretera.
7. **Juegos y Realidad Virtual.** Para reconocimiento de gestos, expresiones faciales y objetos en entornos virtuales.

Las CNN han demostrado ser herramientas poderosas y versátiles en diversas aplicaciones debido a su capacidad para aprender representaciones jerárquicas y invariantes a la translación. Mientras que los MLP son adecuados para tareas generales de aprendizaje supervisado en datos estructurados, las CNN están diseñadas específicamente para trabajar con datos espaciales, como imágenes, y han demostrado ser altamente eficientes en tareas de visión por computadora.

En este capítulo se ha proporcionado una visión general del entorno tecnológico y nomenclatura utilizada en el desarrollo del TFM. La elección de estas herramientas se basa en la idoneidad para los objetivos específicos del proyecto, combinando eficiencia, flexibilidad y capacidad para abordar los desafíos planteados. La información próxima proporcionada sirve como guía para comprender la implementación y replicar el trabajo realizado.

CAPÍTULO 5
Solución propuesta

Después de documentar las distintas posibilidades, se ha optado por abordar el problema de monitorizar en tiempo real el estado de una máquina de corte con láser realizando una implementación del concepto explicado en la Opción B de la Sección 3.1. Como prueba de concepto de la solución propuesta, montaremos un prototipo empleando un pórtico xyz que simulará el funcionamiento de la máquina de corte. En este inyectaremos distintos tipos de fallos mediante el uso de vibradores funcionando a distintas frecuencias para simular anomálias de altas y bajas frecuencias y también, forzaremos el movimiento para que se produzcan cabeceos.

Lo primero de todo, tomaremos muestras de las vibraciones generadas en los ejes x, y, z. Así posteriormente, procesaremos las muestras para extraer las características necesarias del conjunto de datos y mejorar la capacidad de la red neuronal entrenada para aprender y generalizar a partir de los datos, así como para hacer que el modelo sea más eficiente y resistente al sobreajuste. Una vez extraídas las características del conjunto de datos, las utilizaremos para entrenar diversos mecanismos de decisión por ejemplo, a un perceptrón multicapa y una red convolucional. Así, ser capaces de detectar anomalías de altas y bajas frecuencias. Así, sacaremos el mejor modelo que nos ofrezca una alta precisión de clasificación. Obtenido el modelo más ajustado, el modelo en formato JSON se traspasará al sistema embebido donde se ejecutará en un motor de ejecución de redes neuronales hecho en C++. Además, construiremos un sistema de detección de cabeceo a partir de los resultados obtenidos.

5.1 Diseño de la solución

5.1.1. Arquitectura del sistema

En esta sección, se presenta la arquitectura del sistema diseñado para abordar los objetivos planteados en el TFM. La solución propuesta se organiza en varios bloques o subsistemas interconectados, cada uno desempeñando una función específica. A continuación, se detalla la arquitectura de alto nivel y la interacción entre los componentes esenciales.

1. **Adquisición de datos.** En el primer bloque, se encuentra el subsistema de adquisición de datos. Este componente se encarga de recopilar las muestras de aceleración provenientes del acelerómetro instalado en la máquina de corte con láser. La información recolectada es esencial para el análisis posterior y la identificación de patrones. La adquisición de datos se realiza mediante un módulo de hardware específico (acelerómetro) que se conecta al sistema de control de la máquina.

2. **Preprocesamiento de datos.** Las muestras de aceleración adquiridas se someten a un proceso de preprocesamiento en el segundo bloque. Aquí, se aplican técnicas de filtrado digital para eliminar el ruido aleatorio y resaltar las características significativas de la señal. El filtrado digital se lleva a cabo mediante un algoritmo de filtro exponencial, optimizado para preservar la información esencial mientras suprime el ruido. Además, se aplican técnicas de Big Data para extraer las características principales de las señales muestreadas.

3. **Desarrollo de los mecanismos de decisión.** En este fase aplicamos distintos algoritmos de Machine Learning para clasificar los datos de aceleración. Probamos a entrenar los modelos con las características extraídas.

4. **Evaluación y validación.** Después de aplicar los algoritmos de Machine Learning, se lleva a cabo una fase de evaluación y validación. Esto implica la división del conjunto de datos en conjuntos de entrenamiento y prueba, la evaluación del rendimiento del modelo y la recopilación de métricas relevantes como la precisión y la matriz de confusión.

El diagrama de bloques en la Figura 5.1 proporciona una representación visual de la estructura propuesta para el sistema a desarrollar.

En resumen, el sistema se diseña de manera modular, permitiendo flexibilidad y escalabilidad para adaptarse a diferentes escenarios de aplicación y requisitos específicos del usuario.

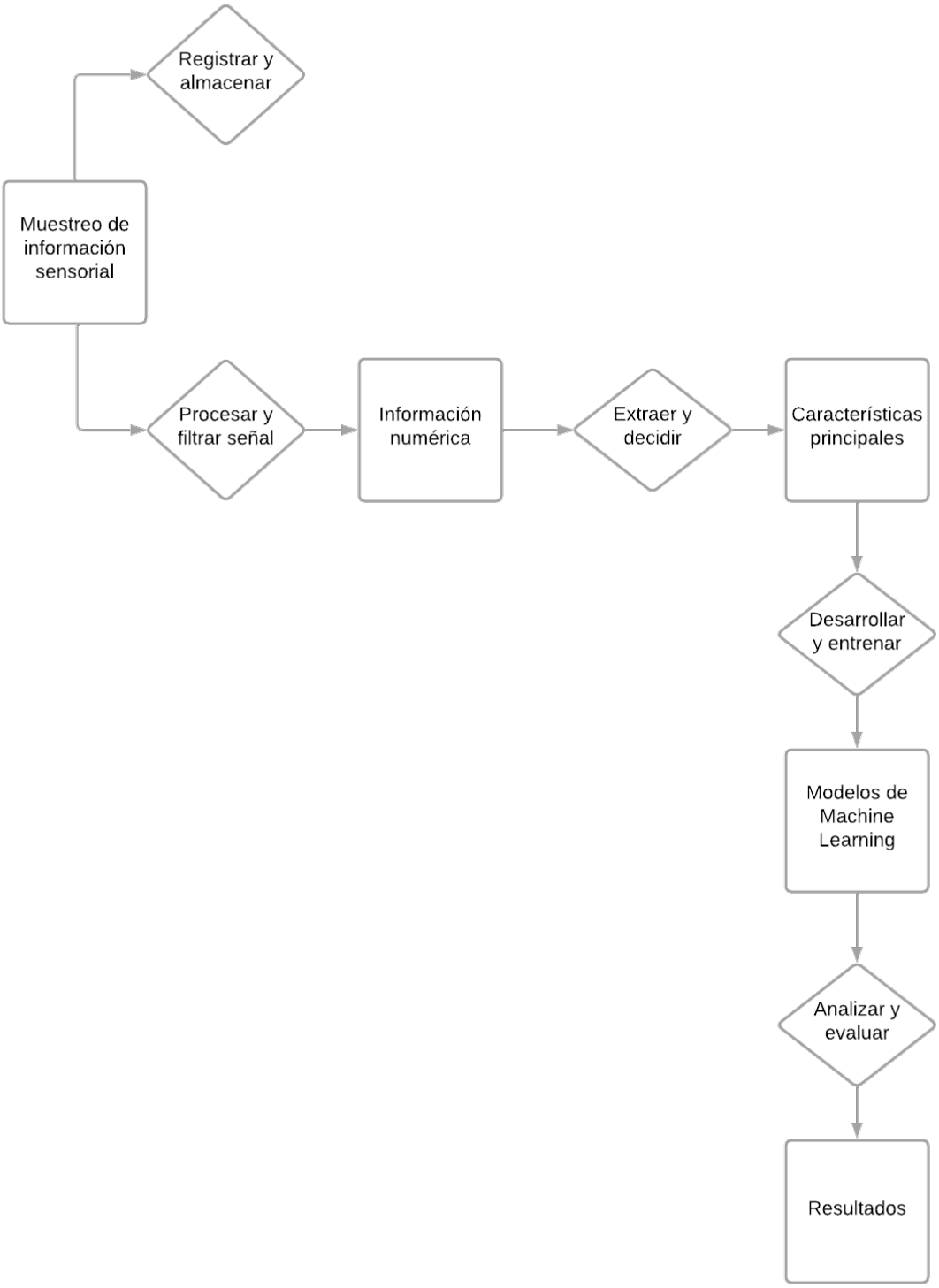


Figura 5.1: Arquitectura de la solución propuesta.

CAPÍTULO 6
Desarrollo e Implementación

En este capítulo, se describe el proceso de desarrollo y puesta en marcha de sistemas basados en inteligencia artificial para abordar el procesamiento sensorial para la detección de funcionamientos anómalos en el contexto de una máquina de corte con láser. Desde la adquisición y procesamiento de información sensorial hasta el entrenamiento de perceptrones multicapa y la implementación de redes convolucionales. Se realizará una aproximación sobre cómo aplicar estas técnicas para mejorar la eficiencia y la precisión en la detección de eventos y patrones importantes en las señales de aceleración.

6.1 Introducción al desarrollo del prototipo

En el marco del desarrollo, hemos basado el trabajo en la prueba de concepto presentada en el artículo titulado "Prediction of Motor Failure Time Using An Artificial Neural Network"[4]. Este artículo sirvió como punto de referencia para el desarrollo de nuestro propio prototipo de una máquina de corte con láser, que consiste en el empleo de un sistema de pórtico XYZ (Figura 6.1) para la toma de muestras de aceleración. El acelerómetro que mide los valores de aceleración se ha conectado (Figura 6.2) en la placa de una Raspberry Pi 5 que controla una cámara que se utiliza para la adquisición de imágenes de los materiales que están siendo cortados por la máquina.

La propuesta principal es evaluar diversas técnicas de procesamiento de información, adquirida a través de instrumentación, con el objetivo de detectar anomalías en el funcionamiento mecánico.

Como punto de partida, hemos utilizado datos obtenidos mediante el muestreo de un acelerómetro montado en un pórtico XYZ. Este pórtico se caracteriza por tener su movimiento controlado por motores paso a paso. La elección de esta configuración proporciona un entorno controlado para la realización de experimentos, inyección de fallos y recolección de datos, esencial en el desarrollo de un prototipo confiable.

El objetivo principal del trabajo es la evaluación de diversas técnicas de procesamiento de información, incluyendo el uso de procesamiento digital y modelado y entrenamiento de redes neuronales. Estas técnicas se han aplicado a los datos adquiridos por la instrumentación, con el propósito de determinar la capacidad de prever anomalías en el funcionamiento de la máquina.

El análisis realizado se enfoca en la evaluación de tres posibles anomalías que podrían afectar el funcionamiento de la máquina de corte con láser: cabeceo, aparición de altas frecuencias aparición de bajas frecuencias. Estas anomalías se seleccionaron por su relevancia y su capacidad para representar situaciones del mundo real que podrían surgir durante el funcionamiento de la máquina.

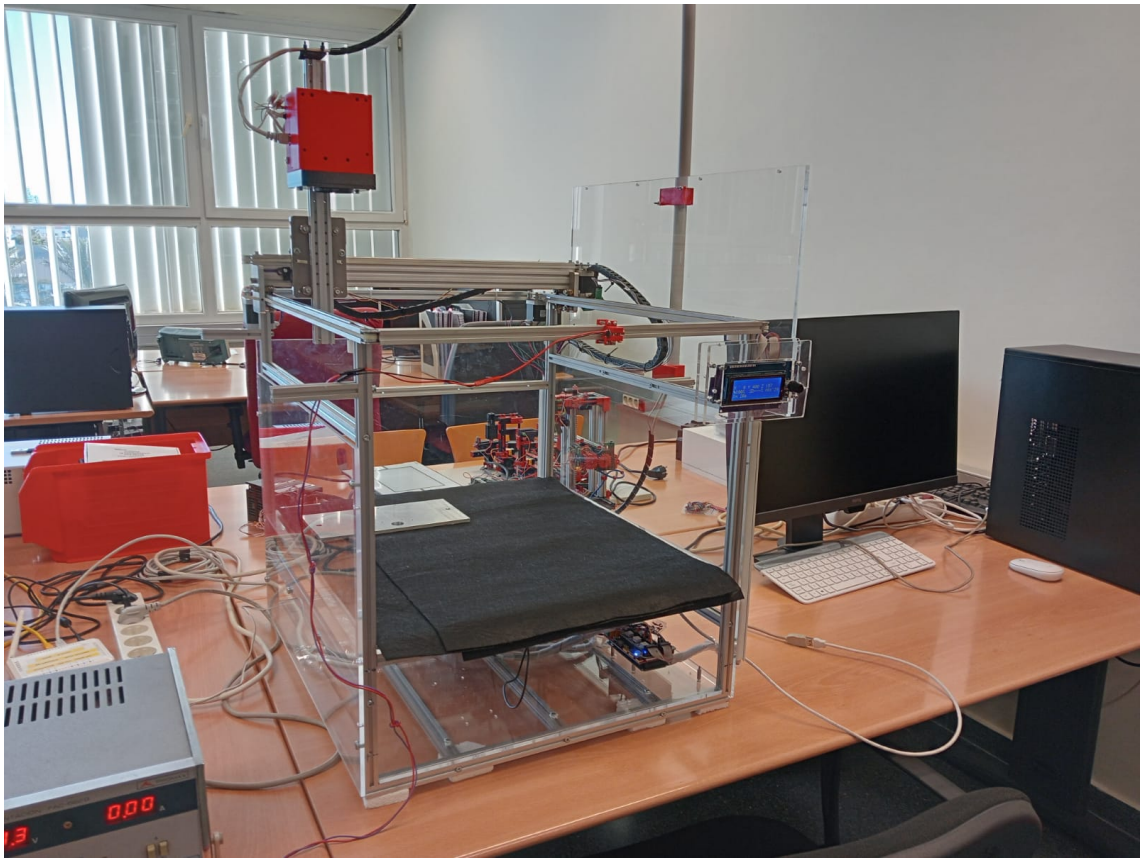
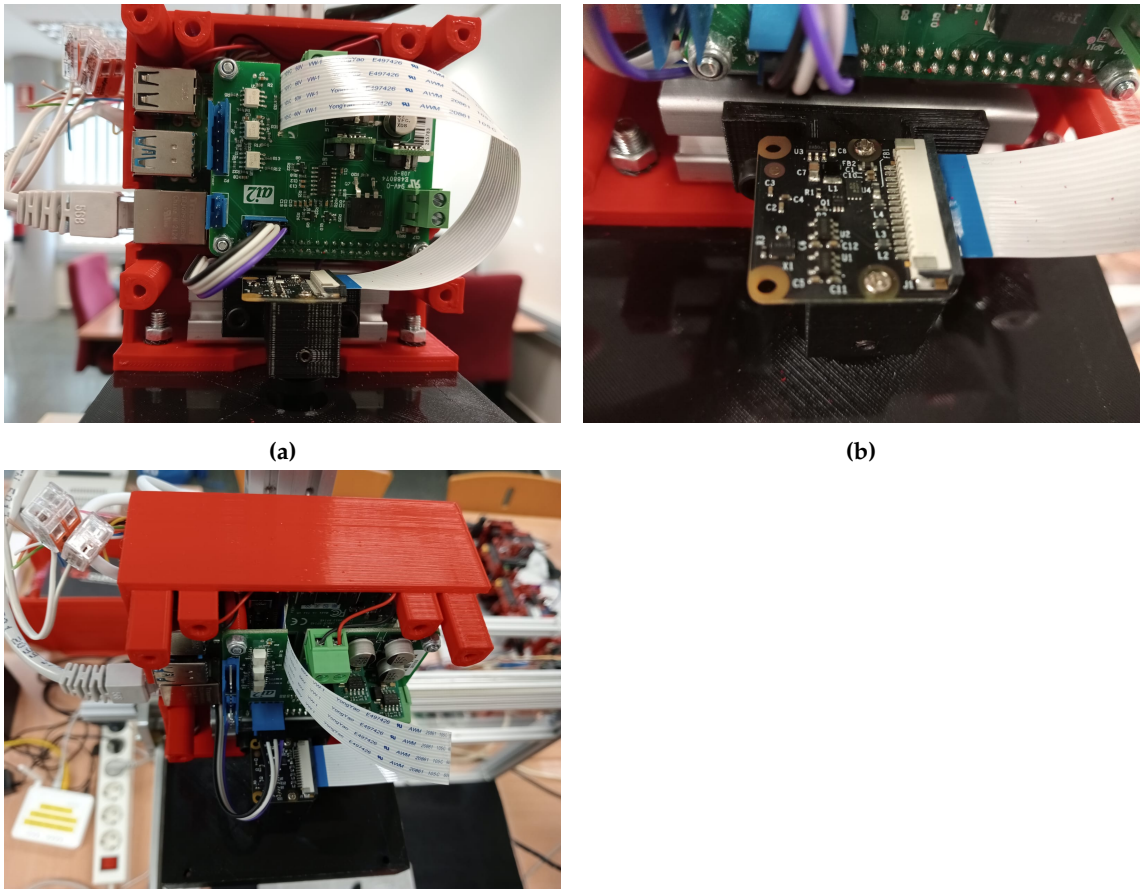


Figura 6.1: Pórtico XYZ empleado como prototipo de máquina de corte con láser.

El cabeceo se refiere a los movimientos oscilatorios no deseados de la máquina alrededor de los de movimiento X e Y. Esto puede ser causado por desequilibrios en el sistema, holguras o problemas en los motores. En la realidad, el cabeceo puede surgir debido a desgastes no uniformes, desalineaciones o incluso desequilibrios en la carga de trabajo. Analizar y detectar el cabeceo es esencial para prevenir daños a la máquina y garantizar cortes precisos. La inyección de altas frecuencias simula la presencia de ruido de alta frecuencia en la señal de aceleración. En la realidad, esto podría representar interferencias eléctricas, vibraciones no deseadas o impactos sutiles en el sistema. La capacidad de detectar y filtrar este tipo de ruido es crucial para mantener mediciones precisas y evitar malfuncionamientos durante la operación normal. La inyección de bajas frecuencias simula perturbaciones más suaves pero persistentes en la señal de aceleración. Esto podría reflejar cambios graduales en las condiciones de trabajo, como desgastes progresivos en componentes mecánicos o variaciones en la carga. La detección temprana de estas bajas frecuencias es esencial para un mantenimiento predictivo efectivo y para prevenir fallas a largo plazo.

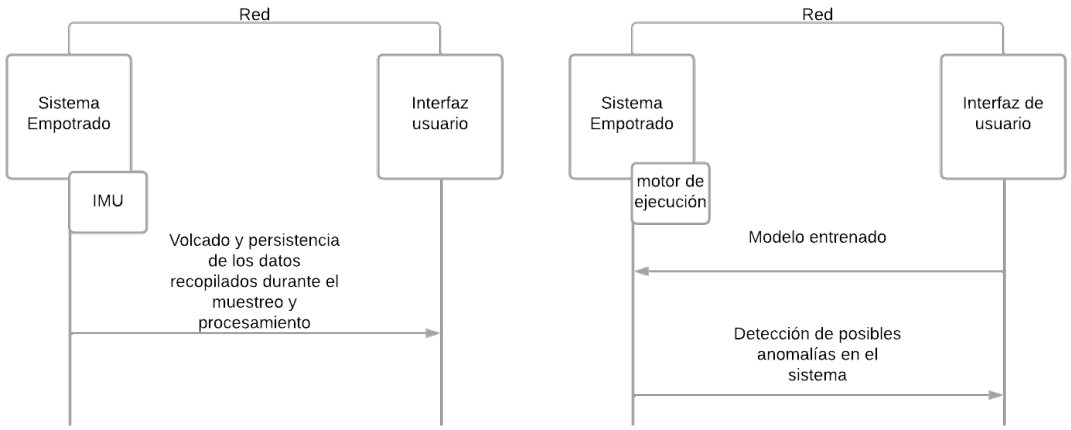
En la Figura 6.3 se puede visualizar los dos escenarios de trabajo que se pueden abordar con nuestra Raspberry Pi embebida en el pórtico. Uno en el que nos permita experimentar y extraer información para el entrenamiento de modelos (Subfigura 6.3a). Y otro que nos permita tomar decisiones en línea mediante la ejecución de esos modelos entrenados (Subfigura 6.3b). En el primer escenario nombrado se ha implementado un programa que se explicará en más detalle más adelante. Básicamente se encarga de tomar las muestras de aceleración y procesarlas para extraer las características que nos interesa como la velocidad por ejemplo y así, a continuación persistirlas para posterior análisis. En el segundo escenario entrenamos modelos de Machine Learning con las características extraídas de las señales de aceleración procesadas y persistidas durante el



(a) (b)

(c)

Figura 6.2: Sensor IMU conectado a la Raspberry Pi 5.



(a) Escenario 1 **(b)** Escenario 2

Figura 6.3: Escenarios de trabajo en el sistema empotrado.

muestreo, y lo subimos al sistema empotrado. Así, utilizando un motor de ejecución de redes neuronales podemos hacer un diagnóstico en tiempo real de las muestras que se están tomando.

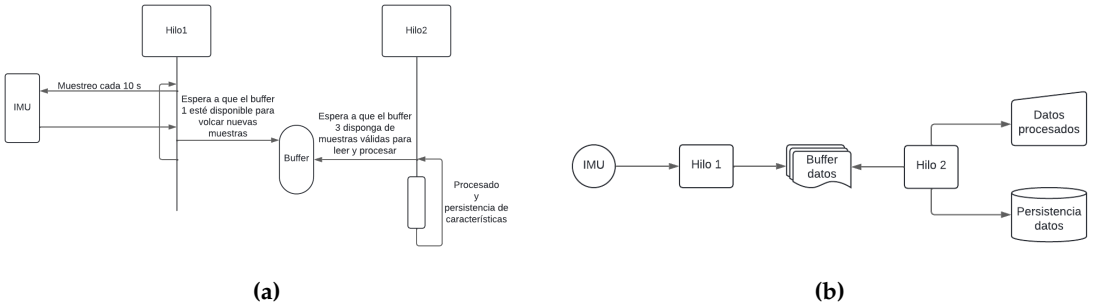
En el caso de que utilicemos el sistema empotrado para extraer información del funcionamiento del pórtico, se ha implementado un programa que interactúa con la IMU (Unidad de Medición Inercial) a través del bus I2C. Este programa lanza a ejecución un hilo que toma de forma iterativa N muestras de aceleración y las guarda en un buffer para que otro segundo hilo pueda procesar esas muestras y los resultados obtenidos persistirlos. Así por ejemplo, posteriormente poder extraer las características necesarias para entrenar un modelo (Subfigura 6.4a).

Como se muestra en la Subfigura 6.4b el hilo de ejecución Hilo1 es el encargado del muestreo de la información adquirida por el acelerómetro conectado al sistema empotrado mediante el bus I2C.

Es importante que este hilo se ejecute con la máxima resolución temporal adquiriendo muestras en instantes precisos de tiempo evitando la deriva que puediera ocasionar la aparición de latencias. Para ello se establece su política de planificación y prioridad en el nivel de Tiempo Real que implementa el núcleo de Linux. Además, se implementa la espera para el siguiente instante de muestreo sobre instantes de tiempo fijos y no mediante diferencias de tiempo. Estas dos importantes caracteristicas del Hilo1 se implementan respectivamente en las funciones "setRealTimeCurrentThreadz "sleep_until"que más adelante se verán cómo se han empleado.

El hilo de ejecución Hilo2 es el encargado del procesamineto y registro de la información suministrada por el Hilo1. En este caso, la resolución temporal no es tan importante como en el caso del Hilo1 y, por ello, no se establecene ninguna configuración especial para su planificación. De este modo, en caso de ser necesario, el Hilo1 expulsará de la ejecución al Hilo2 en favor de se ejecución.

La comunicación entre el Hilo1 y el Hilo2 se realiza mediante un mecanismo de "triple buffes"que consite en que mientras el Hilo1 toma las muestras de aceleración y las almacena en un buffer, el Hilo2 está procesando las muestras de otro buffer. Se emplea un buffer intermedio para hacer el intercambio de datos una vez que el hilo que está procesando esté listo para procesar muestras nuevas tomadas por el hilo que las toma. El pseudocódigo se muetra en la Figura 6.5.



(a) **(b)**

Figura 6.4: Procesamiento y extracción de características del muestreo de señales de aceleración del pórtico.

Algorithm 1 Adquisición y procesamiento de muestras

```
1:  Variables:
2:      bool datosListos ← False
3:      BufferSet buffers
4:  Hilo que adquiere muestras:
5:  Producir → buffers.B1
6:  lock(buffers.mtx)
7:  datosListos ← True
8:  aux ← buffers.B2
9:  buffers.B2 ← buffers.B1
10: buffers.B1 ← aux
11: notify(buffers.condListo)
12: unlock(buffers.mtx)
13: Hilo que procesa muestras:
14: while !datosListos do
15:     wait(buffers.condListo)
16: end while
17: lock(buffers.mtx)
18: aux ← buffers.B3
19: buffers.B3 ← buffers.B2
20: buffers.B2 ← aux
21: datosListos ← False
22: unlock(buffers.mtx)
23: Procesar buffers.B3
```

Figura 6.5: Pseudocódigo algoritmo de 3 buffers

Para nuestras pruebas experimentales, hemos utilizado el firmware Marlin y el lenguaje G-code para controlar el pórtico de cuatro ejes, con la torre que lleva incorporada la cámara y la IMU. Marlin, conocido por su flexibilidad y adaptabilidad, fue instalado en una placa de Arduino Mega 2560 para hacer el control de los movimientos del pórtico.

Para crear la rutina de movimiento, diseñamos secuencias en G-code que especifican las coordenadas y las trayectorias de la torre en los cuatro ejes. Estas instrucciones de G-code incluían comandos para variar la aceleración y la velocidad de la torre, permitiendo pruebas detalladas y ajustes finos en tiempo real. Durante las pruebas, la IMU registraba datos de movimiento, proporcionando un flujo constante de información para el análisis y la optimización de la rutina.

Este enfoque combinó la potencia del firmware Marlin con la precisión del G-code, permitiendo realizar movimientos complejos y capturar datos críticos para el diagnóstico y monitorización del sistema.

Además, se añadieron al pórtico dos motores vibratorios para inyectar los fallamos que deseamos monitorizar. Uno de ellos fue configurado para generar perturbaciones de bajas frecuencias y el otro para altas frecuencias. Esta configuración nos permitió simular diferentes condiciones de fallo y analizar cómo afectan al rendimiento y estabilidad del sistema, facilitando el desarrollo de técnicas de diagnóstico más precisas y efectivas.

Al abordar estas tres posibles anomalías, nuestro objetivo ha sido proporcionar un análisis exhaustivo y realista que contribuya al desarrollo de técnicas de detección de fallos robustas. La capacidad de identificar y responder a estas situaciones en tiempo real mejora significativamente la eficiencia operativa y la fiabilidad de las máquinas de corte con láser en entornos industriales.

Los resultados obtenidos a través de la aplicación de las diferentes técnicas se han comparado tanto cuantitativa como cualitativamente. Esta comparación se realiza para evaluar la eficacia de cada enfoque en la detección de anomalías. Las conclusiones derivadas de esta investigación contribuirán significativamente al avance en la detección temprana de anomalías en máquinas de corte con láser.

Las conclusiones y hallazgos derivados de este trabajo tendrán implicaciones prácticas, especialmente en el ámbito del mantenimiento predictivo y la mejora de la eficiencia operativa de máquinas de corte con láser. Se espera que estas contribuciones contribuyan al avance general en la detección temprana de problemas mecánicos, ofreciendo beneficios tangibles en términos de fiabilidad y eficiencia de estas máquinas en entornos industriales.

6.2 Muestreo de datos del acelerómetro

A continuación, se va a describir el proceso de desarrollo del programa de muestreo y las distintas modificaciones que se han ido realizando a lo largo del trabajo.

En un principio se ha implementado un programa en C++ que toma muestras de 10 segundos con un periodo de muestreo de 1 milisegundo. En la Figura 6.1 podemos acceder al programa principal que muestrea las señales de aceleración que generan los movimientos ejecutados por el pórtico.

El programa en C++ interactúa con el sensor inercial MPU9250 a través del bus I2C en la Raspberry Pi. Incluye bibliotecas para el manejo de archivos, tiempo, I2C y el sensor MPU9250, además de definir la constante **NSAMPLES** para el número de muestras a adquirir.

Se definen funciones de utilidad como **getCurrentMicroseconds()**, que devuelve el tiempo actual en microsegundos, **sleep_until(struct timespec *ts, int delay)**, para añadir un retraso y dormir hasta un nuevo tiempo, y **setRealTimeCurrentThread()**, que establece la política de planificación y la prioridad del hilo de ejecución como una tarea de tiempo real.

La función **calibrateAccelOffsets** calibra los offsets del acelerómetro, adquiriendo muestras y calculando el valor promedio de las aceleraciones en los ejes X, Y y Z, ajustando los offsets para centrar las lecturas en cero.

En la función **main**, se inicializa el bus I2C y se crea una instancia del sensor. Se comprueba la conexión y se inicializa el sensor, calibrando los offsets. Luego, se establece el hilo de ejecución en tiempo real y se inicia un bucle para adquirir muestras del sensor. En cada iteración, se obtienen las lecturas del acelerómetro y se almacenan en buffers junto con el tiempo de adquisición. Se imprime el valor de la aceleración y el tiempo transcurrido cada 1000 muestras, asegurando la frecuencia de ejecución con **sleep_until**.

En las próximas líneas realizamos un análisis de las señales vibratorias obtenidas. Al realizar este análisis simple, el objetivo que tiene éste es poder experimentalmente distinguir cuando la maquinaria está en buen estado y cuando tiene algún fallo, intentando identificar de qué fallo en concreto se trata. Para poder comparar las señales en función de si la maquinaria está en buen estado o tiene algún tipo de fallo mecánico, se han utilizado los datos recopilados variando las velocidades y aceleraciones de los motores paso a paso del pórtico entre 9000-12000 mm/s y 200-8000 mm/s², respectivamente.

Tras tomar las muestras necesarias inyectando altas frecuencias, podemos hacer un análisis de las señales vibratiorias que hemos medido con el acelerómetro.

Uno de los objetivos era conseguir un procesamiento continuo de las señales de aceleración del pórtico. Para ello hemos implementado lo que hemos denominado un algoritmo de 3 buffers. Siguiendo la idea utilizada en el procesamiento de imágenes [8], hemos modificado nuestro programa original de muestreo para que tengamos dos hilos ejecutándose.

El código proporcionado a continuación implementa un sistema para el procesamiento continuo de señales de aceleración provenientes del sensor MPU9250. Para lograr esto, se utilizan tres buffers para almacenar los datos de aceleración muestreados, lo que permite a dos hilos ejecutarse de forma concurrente para el llenado y procesamiento de estos buffers, como se ha explicado previamente.

La función sensorDataProducer (Figura 6.2), se encarga de tomar muestras de aceleración del MPU9250 y llenar los buffers correspondientes. Utiliza un hilo de ejecución para ejecutarse en paralelo con el hilo de consumo de datos. Dentro de un bucle infinito, se toman muestras de aceleración, se almacenan en el buffer actual y se realiza un cambio atómico de buffers cuando se ha llenado el actual.

La función sensorDataConsumer (Figura 6.3), procesa los datos almacenados en el tercer buffer y calcula el espectro de frecuencia de las señales de aceleración. Espera a que se haya llenado el segundo buffer antes de comenzar a procesar los datos. Una vez que se obtienen los datos, se calcula la transformada de Fourier de las señales de aceleración utilizando una tabla de senos y cosenos precalculada para acelerar el proceso.

La estructura Buffer (Figura 6.4), define un buffer que contiene arreglos para almacenar las muestras de tiempo y los valores de aceleración en cada eje (X, Y, Z).

La clase BufferSet (Figura 6.5), encapsula tres instancias de Buffer y proporciona un mecanismo de sincronización para la manipulación segura de los buffers por parte de los hilos de productor y consumidor. Utiliza un mutex y una variable de condición para controlar el acceso concurrente a los buffers compartidos por los dos hilos. El mutex

```
int main()
{
  BBB::I2C i2cBus(4);
  BBB::MPU9250Lite imu(&i2cBus, 0x68);

    unsigned long data_buffer_t0[NSAMPLES];
    unsigned long data_buffer_t1[NSAMPLES];
    float data_buffer_value_X[NSAMPLES];
    float data_buffer_value_Y[NSAMPLES];
    float data_buffer_value_Z[NSAMPLES];

  //Check connection
  cout << "Get Connection: 0x" << hex << (int)imu.getConnection()  << endl;
  //Init IMU
  imu.initialize();
  //Calibrate offsets
  calibrateAccelOffsets(&imu);
  float accx,accy,accz;
    setRealTimeCurrentThread();
    struct timespec ts;
    clock_gettime(CLOCK_MONOTONIC, &ts);
    //
    int i = 0;
  while(i<NSAMPLES){
    // Get IMU values
    unsigned long t1 = getCurrentMicroseconds();
    imu.getAccelerationXYZ(&accx, &accy, &accz);
    unsigned long t2 = getCurrentMicroseconds();
      data_buffer_t0[i] = t1;
      data_buffer_t1[i] = t2;
      data_buffer_value_X[i] = accx;
      data_buffer_value_Y[i] = accy;
      data_buffer_value_Z[i] = accz;
      //
    i = (i+1);
    if (i%1000==0) {
      float module = sqrt(accx*accx+accy*accy+accz*accz);
      cout << "ACC--> X: "<<accx<< " Y: "<<accy << " Z: "<<accz<< "       Module
          : " << module << endl;
      printf("Tiempo vuelo: %ld\n",t2-t1);
    }
    sleep_until(&ts, 1000);
  }

  //Dump data to file
      ofstream datafile;
      datafile.open ("data.txt");

    for (int i=0; i<NSAMPLES; i++) {
      datafile << data_buffer_t0[i] << " " << data_buffer_t1[i] << " " <<
          data_buffer_value_X[i] << " " << data_buffer_value_Y[i] << " " <<
          data_buffer_value_Z[i]<< endl;
    }
    datafile.close();

    cout << "IMU sampling finished" << endl;

  return 0;
}
```

Listing 6.1: Muestreo de aceleraciones.

```
void sensorDataProducer(BufferSet& buffers, BBB::MPU9250Lite& imu) {
    setRealTimeCurrentThread();
    while(true){
        struct timespec ts;
        clock_gettime(CLOCK_MONOTONIC, &ts);
        int i = 0;
        while (i < NSAMPLES) {
            unsigned long t1 = getCurrentMicroseconds();
            float accx, accy, accz;
            imu.getAccelerationXYZ(&accx, &accy, &accz);
            unsigned long t2 = getCurrentMicroseconds();
            buffers.buffer1.data_buffer_t0[i] = t1;
            buffers.buffer1.data_buffer_t1[i] = t2;
            buffers.buffer1.data_buffer_value_X[i] = accx;
            buffers.buffer1.data_buffer_value_Y[i] = accy;
            buffers.buffer1.data_buffer_value_Z[i] = accz;
            i++;
            sleep_until(&ts, 2000);
        }
        {
            std::unique_lock<std::mutex> lock(buffers.mtx);
            Buffer temp = buffers.buffer1;
            buffers.buffer1 = buffers.buffer2;
            buffers.buffer2 = temp;
            buffers.datosListos = true;
        }
        buffers.cond.notify_all();
    }
}
```

Listing 6.2: Función sensorDataProducer.

(mtx) se bloquea y desbloquea según sea necesario para garantizar la exclusión mutua. La variable de condición (cond) se utiliza para notificar al hilo que ejecuta procesa las muestras tomadas cuando hay datos disponibles en los buffers.

La función lookuptable (Código 6.6), se encarga de precalcular una tabla de senos y cosenos para agilizar los cálculos de la transformada de Fourier utilizada en el cálculo del espectro de frecuencia de las señales de aceleración.

6.3 Procesamiento de un conjunto de datos

Aquí hay varias razones por las cuales se realiza este procesamiento:

- **Representación más adecuada.** Los datos crudos pueden contener información redundante o irrelevantes para la tarea en cuestión. Extraer características significa transformar los datos en una representación más adecuada y significativa para el problema específico que la red neuronal está destinada a resolver.

- **Reducción de dimensionalidad.** Algunos conjuntos de datos pueden ser de alta dimensionalidad, lo que significa que contienen muchas características o variables. Esto puede llevar a hacer que el modelo sea más complejo y computacionalmente costoso de entrenar. La extracción de características puede ayudar a reducir la dimensionalidad al centrarse en las características más relevantes.

- **Mejora del rendimiento.** Al seleccionar y procesar características relevantes, se espera que la red neuronal sea capaz de aprender patrones más efectivamente y,

```
void sensorDataConsumer(BufferSet& buffers, std::vector<std::vector<double>>&
    cos_table, std::vector<std::vector<double>>& sin_table) {
  while(true){
        unique_lock<std::mutex> lock(buffers.mtx);
        buffers.cond.wait(lock, [&]() { return buffers.datosListos; });
        Buffer temp = buffers.buffer2;
        buffers.buffer2 = buffers.buffer3;
        buffers.buffer3 = temp;
        lock.unlock();
        buffers.datosListos = false;
        double xEspectroX[ESPECTRO_DIM];
        double xEspectroY[ESPECTRO_DIM];
        double yEspectroX[ESPECTRO_DIM];
        double yEspectroY[ESPECTRO_DIM];
        double zEspectroX[ESPECTRO_DIM];
        double zEspectroY[ESPECTRO_DIM];
        if (ESPECTRO_DIM > 0) {
            for(int i=0; i<ESPECTRO_DIM; i++) {
                xEspectroX[i]=0; xEspectroY[i]=0;
                yEspectroX[i]=0; yEspectroY[i]=0;
                zEspectroX[i]=0; zEspectroY[i]=0;
                for (int j=0; j<NSAMPLES; j++) {
                        xEspectroX[i]+=buffers.buffer3.data_buffer_value_X[j] *
                            cos_table[i][j];
                        xEspectroY[i]+=(-1) * buffers.buffer3.
                            data_buffer_value_X[j] * sin_table[i][j];
                        yEspectroX[i]+=buffers.buffer3.data_buffer_value_Y[j] *
                            cos_table[i][j];
                        yEspectroY[i]+=(-1) * buffers.buffer3.
                            data_buffer_value_Y[j] * sin_table[i][j];
                        zEspectroX[i]+=buffers.buffer3.data_buffer_value_Z[j] *
                            cos_table[i][j];
                        zEspectroY[i]+=(-1) * buffers.buffer3.
                            data_buffer_value_Z[j] * sin_table[i][j];
                }
                xEspectroX[i]=xEspectroX[i]/NSAMPLES; ///modulo_max;
                xEspectroY[i]=xEspectroY[i]/NSAMPLES;
                yEspectroX[i]=yEspectroX[i]/NSAMPLES;
                yEspectroY[i]=yEspectroY[i]/NSAMPLES;
                zEspectroX[i]=zEspectroX[i]/NSAMPLES;
                zEspectroY[i]=zEspectroY[i]/NSAMPLES;
            }
        }
    }
}
```

Listing 6.3: Función sensorConsumer.

```
struct Buffer {
    unsigned long data_buffer_t0[NSAMPLES];
    unsigned long data_buffer_t1[NSAMPLES];
    float data_buffer_value_X[NSAMPLES];
    float data_buffer_value_Y[NSAMPLES];
    float data_buffer_value_Z[NSAMPLES];
    int count = 0;
};
```

Listing 6.4: Estructura Buffer

```
class BufferSet {
  public:
    Buffer buffer1;
    Buffer buffer2;
    Buffer buffer3;
    bool datosListos = false;
    std::mutex mtx;
    std::condition_variable cond;
};
```

Listing 6.5: Clase BufferSet.

```
void lookuptable(std::vector<std::vector<double>>& cos_table, std::vector<std::
    vector<double>>& sin_table) {
    const double PI=3.141592653589793;
    for (int i = 0; i < ESPECTRO_DIM; ++i) {
        for (int j = 0; j < NSAMPLES; ++j) {
            double angle = 2 * PI / NSAMPLES * i * j;
      cos_table[i][j] = cos(angle);
            sin_table[i][j] = sin(angle);
        }
    }
}
```

Listing 6.6: Función lookuptable.

por lo tanto, mejorar su rendimiento en la tarea específica. Las características bien elegidas pueden capturar aspectos clave de los datos que facilitan la generalización a nuevas instancias.

- **Regularización y prevención del sobreajuste.** Al limitar el número de características o al aplicar técnicas de extracción de características, se puede reducir el riesgo de sobreajuste. El sobreajuste ocurre cuando un modelo se ajusta demasiado a los detalles específicos del conjunto de entrenamiento y no generaliza bien a nuevos datos. La extracción de características puede ayudar a evitar la inclusión de características irrelevantes que podrían conducir al sobreajuste.

- **Eficiencia computacional.** La extracción de características también puede mejorar la eficiencia computacional al reducir la cantidad de datos y operaciones que la red neuronal debe procesar durante el entrenamiento y la inferencia.

En nuestro caso de estudio a parte de registrar los datos crudos de aceleración, hemos hecho un procesamiento digital de estas señales para aplicar técnicas que nos permitan tomar ciertas decisiones sobre esos datos. Queremos detectar posibles anomalías de altas y bajas frecuencias por ello hemos hecho un análisis de frecuencia de las señales. Además, queremos detectar el cabeceo para ello hemos integrado los valores de aceleración muestreados, separado los puntos de intéres donde se producía el cabeceo y eliminado la tendencia de las señales para calcular la sobreoscilación producida.

Hemos hecho un análisis en frecuencia aplicando la Transformada de Fourier que más adelante explicaremos y mostraremos los resultados obtenidos. Así, podemos reconocer los casos en los que se estén produciendo anomalías de bajas o altas frecuencias, analizando los armónicos de las señales de aceleración de la máquina.

Otros datos que hemos calculado ha sido la velocidad por integración. De este modo, hemos sido capaces de caracterizar el cabeceo de la máquina por cambios en la varianza en los puntos que se reconocían como posibles puntos de sobreoscilación. Para ello hemos

tenido que eliminar las tendencias de las señales de la velocidad por regresión lineal (Figura 6.6) y así, conseguimos calcular la sobreoscilación producida (Código 6.7).

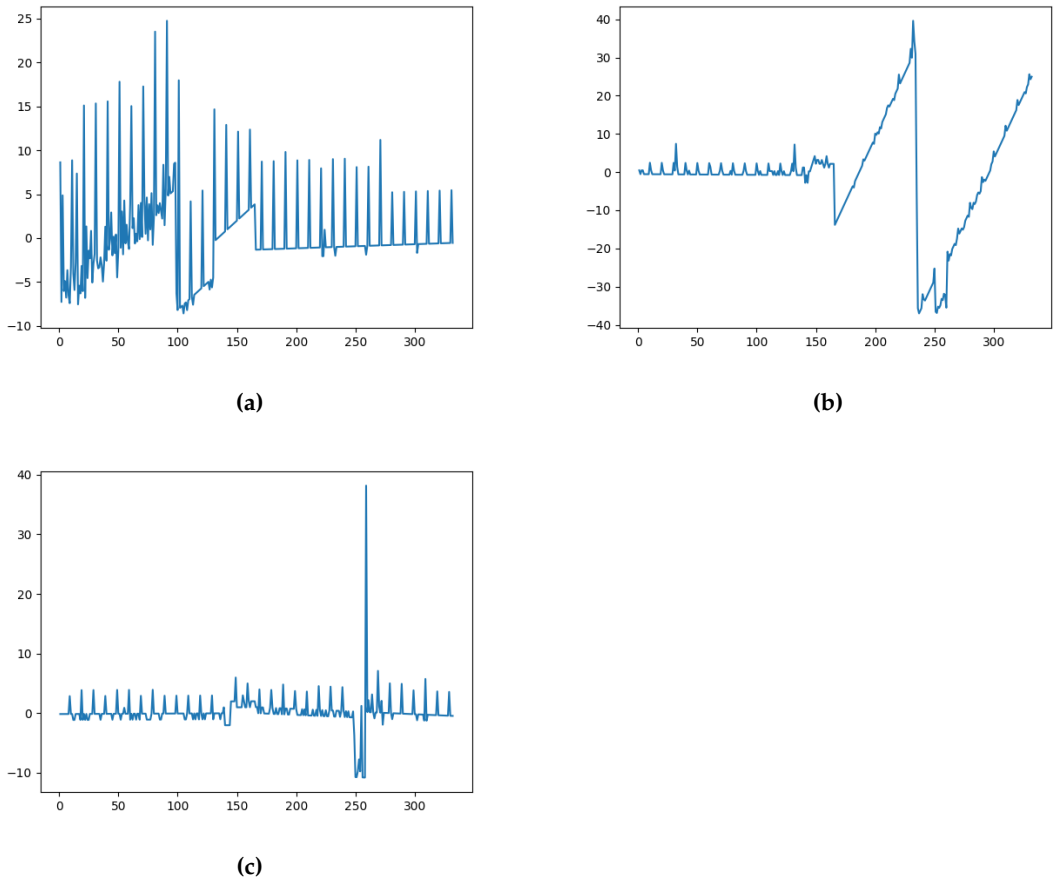


Figura 6.6: Cálculo de la sobreoscilación mediante el cambio de varianza de la señal de velocidad.

Para identificar los cambios bruscos en la velocidad de la señal muestreada y detectar los cabeceos de la máquina, hemos calculado la velocidad absoluta en los ejes X e Y a partir de la integración de la señal de aceleración. En el código proporcionado, la función sensorDataConsumer (Figura 6.8) se encarga de este cálculo en tiempo real de muestreo. A continuación, se explica el proceso detallado.

En el fragmento de código (Figura 6.8), la velocidad se calcula mediante la integración de la aceleración en cada intervalo de tiempo. La aceleración se multiplica por 9.81 para convertirla a metros por segundo al cuadrado, y luego se multiplica por el tiempo transcurrido entre muestras para obtener el cambio en la velocidad. Esto se suma a la velocidad anterior para obtener la velocidad actual.

Este enfoque nos permite seguir la evolución de la velocidad en cada eje (X e Y) y detectar cambios bruscos que podrían indicar cabeceos de la máquina. El proceso se repite en cada iteración del bucle mientras se reciben datos nuevos. La velocidad calculada se almacena en vectores para su posterior análisis y aplicación de un sistema de decisión.

Hasta ahora, los datos se han centrado principalmente en eventos donde la aceleración excede ciertos umbrales predefinidos, indicando cabeceo significativo. Sin embargo, también es crucial considerar las situaciones en las que el cabeceo es mínimo o inexistente, así como evaluar la evolución de las señales en diferentes condiciones de aceleración.

En las pruebas donde no se detecta cabeceo debido a la ausencia de aceleraciones bruscas, no se identifican intervalos de interés para el análisis. Esta situación es impor-

```
int delta = NSAMPLES/30; // delta inferior (333 puntos aprox.)
    for(int i=0; i< indices_sobreoscilacion_x.size(); i++){
        // Calcular el intervalo alrededor del indice de sobreoscilacion
        int lower_bound = indices_sobreoscilacion_x[i] -  delta/2; // Limite
            inferior del intervalo
        int upper_bound = indices_sobreoscilacion_x[i] +  delta/2; // Limite
            superior del intervalo
        // Asegurarse de que los limites estan dentro de los limites del vector
        lower_bound = max(0, lower_bound); // No debe ser menor que 0
        upper_bound = min(NSAMPLES - 1, upper_bound); // No debe ser mayor que
            NSAMPLES - 1

        double vEntrada = vx[lower_bound]; //valor de velocidad en principio de
             intervalo
        double vSalida = vx[upper_bound];  //valor de velocidad en final de
            intervalo
        // Archivos con fenomeno de cambio aislado
        // Calculo las diferencias con respecto al principio y fin del
            intervalo
        double vectorDifSalida[delta];
        int j = lower_bound;
        int k = 0;
        while((j+1)<=upper_bound){
            j++;
            k++;
            double dvEntrada = (vx[j] - vEntrada);
            double dvSalida = (vx[j] - vSalida);
            vectorDifSalida[k] = dvSalida; // me guardo solo la difsalida para
                calcular la desviacion tipica
        }
        //Calculo media
        double mediaAntes = 0;
        double mediaLuego = 0;
        for (int ndx=0; ndx<delta; ndx++) {
            if (ndx < delta/2) {
                mediaAntes += vectorDifSalida[ndx];
            }
            if (ndx > delta/2) {
                mediaLuego += vectorDifSalida[ndx];
            }
        }
        mediaAntes /= (delta/2);
        mediaLuego /= (delta/2);
        cout<< "MediaAntes: " << mediaAntes << endl;
        cout<< "MediaLuego: " << mediaLuego << endl;
        //Calculo desviacion tipica
        double sigmaAntes = 0;
        double sigmaLuego = 0;
        for (int ndx=0; ndx<delta; ndx++) {
            if (ndx < delta/2) {
                sigmaAntes += (vectorDifSalida[ndx] - mediaAntes) * (
                    vectorDifSalida[ndx] - mediaAntes);
            }
            if (ndx > delta/2) {
                sigmaLuego += (vectorDifSalida[ndx] - mediaLuego) * (
                    vectorDifSalida[ndx] - mediaLuego);
            }
        }
        sigmaAntes = sqrt(sigmaAntes/(delta/2));
        sigmaLuego = sqrt(sigmaLuego/(delta/2));
```

Listing 6.7: Detección y cálculo de la sobreoscilación.

```
void sensorDataConsumer(BufferSet& buffers, std::vector<std::vector<double>>&
    cos_table, std::vector<std::vector<double>>& sin_table, int tipoFallo) {
    // Variables declaration
    double velocity_x = 0; // Initial velocity
    double velocity_y = 0;
    // Other objects and variables needed
    while(true){
        // Wait for data to be ready
        unique_lock<std::mutex> lock(buffers.mtx);
        buffers.cond.wait(lock, [&]() { return buffers.datosListos; });
        Buffer temp = buffers.buffer2;
        buffers.buffer2 = buffers.buffer3;
        buffers.buffer3 = temp;
        lock.unlock();
        buffers.datosListos = false;
        // Process data
        if (ESPECTRO_DIM > 0) {
            // Other needed processes
            for(int i=1; i<ESPECTRO_DIM; i++) {
                // Calculate velocity in axis x and y
                velocity_x += 9.81 * accx * (t1 - buffers.buffer3.
                    data_buffer_t0[i-1]) / 1000000.0;
                velocity_y += 9.81 * accy * (t1 - buffers.buffer3.
                    data_buffer_t0[i-1]) / 1000000.0;

                vx[i] = velocity_x;
                vy[i] = velocity_y;
            }
        }
    }
}
```

Listing 6.8: Cálculos de la velocidad en el plano conformado por los ejes XY a partir de la integración de la aceleración en esos ejes.

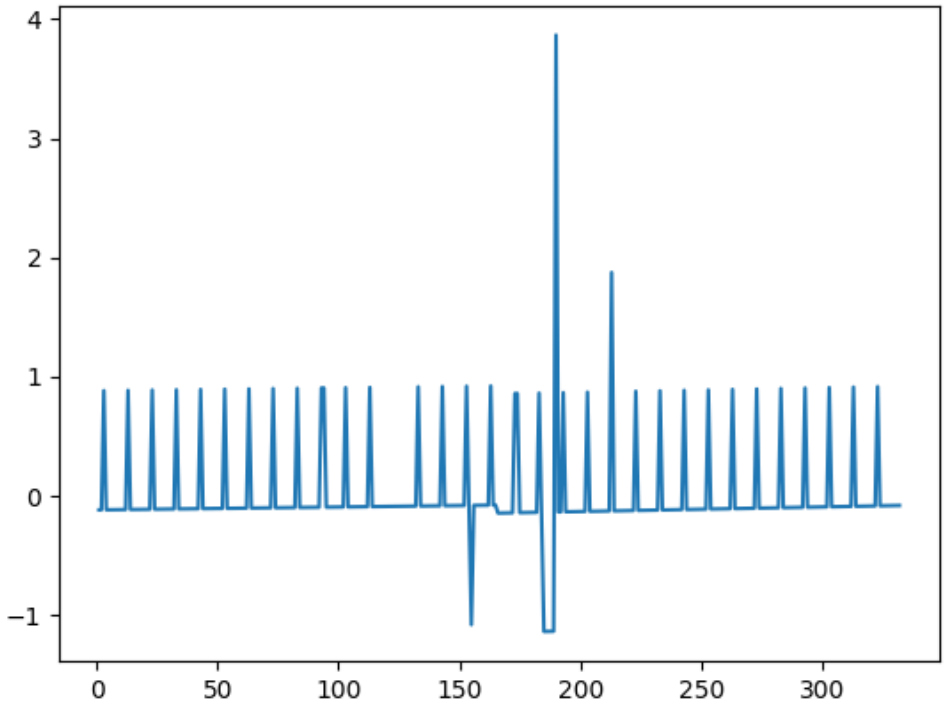


Figura 6.7: Análisis de la varianza de la señal de velocidad en ausencia de cabeceo.

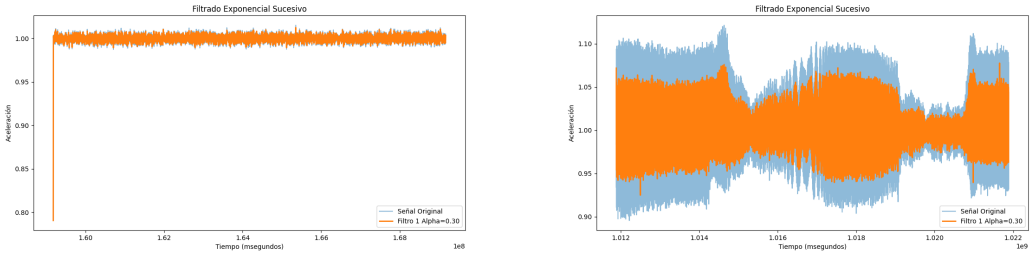
tante para establecer una línea base de comportamiento normal de la máquina, donde las señales de aceleración permanecen dentro de un rango estable. Este tipo de datos es fundamental para contrastar y validar los resultados obtenidos en condiciones anómalas.

Al analizar los datos, se observa que con mayor aceleración, las gráficas muestran cambios más pronunciados, mientras que con menor aceleración o ausencia de aceleración, los cambios son más sutiles (Figura 6.7).

En esta sección, hemos aprendido que para entender mejor la evolución de las señales de aceleración en diferentes condiciones, se deben realizar múltiples experiencias con varios niveles de aceleración. Al analizar los datos, se observa que con mayor aceleración, las gráficas muestran cambios más pronunciados, mientras que con menor aceleración, los cambios son más sutiles. En ausencia de aceleración, no se detectan puntos de interés para el análisis. Este análisis detallado permite correlacionar el nivel de aceleración con la magnitud del cabeceo, proporcionando una herramienta valiosa para predecir y diagnosticar estos eventos.

6.4 Filtrado digital del ruido aleatorio de una señal.

En esta sección, se presenta el diseño para aplicar técnicas de filtrado digital a muestras de aceleración obtenidas mediante un acelerómetro y el desarrollo de estas técnicas mediante Python. La aplicación de filtros digitales es esencial para mejorar la calidad de las mediciones, especialmente cuando las señales capturadas están afectadas por ruido ambiental. En este contexto, se exploran dos tipos de filtros: el filtro exponencial y el filtro de promedio de ventana deslizante centrada. Este análisis se ha hecho previo al registro



(a) **(b)**

Figura 6.8: Comparación de la aplicación del filtro exponencial sobre la señal sin inyección de perturbaciones a) y cuando se añaden perturbaciones b).

de datos para determinar qué tipo de procesamiento y valores serían los más adecuados de procesar.

Las señales provenientes de sensores, como un acelerómetro, a menudo están expuestas a interferencias y ruido ambiental. Este ruido puede distorsionar las mediciones y dificultar la identificación precisa de eventos o patrones importantes en la señal. La aplicación de filtros digitales permite eliminar o reducir este ruido, mejorando así la fiabilidad y utilidad de las mediciones.

Se ha implementado en Python la aplicación de dos técnicas de filtrado digital: el filtro exponencial y el filtro de promedio de ventana deslizante centrada. Primero de todo, se tomaron muestras de aceleración con la máquina en reposo, tanto sin perturbaciones como añadiendo de perturbaciones.

Filtro exponencial

El filtro exponencial es una técnica de filtrado que asigna diferentes pesos a los puntos de datos en función de su antigüedad. La fórmula del filtro exponencial es 6.1:

$$y[i] = \alpha \cdot y[i-1] + (1-\alpha) \cdot x[i] \tag{6.1}$$

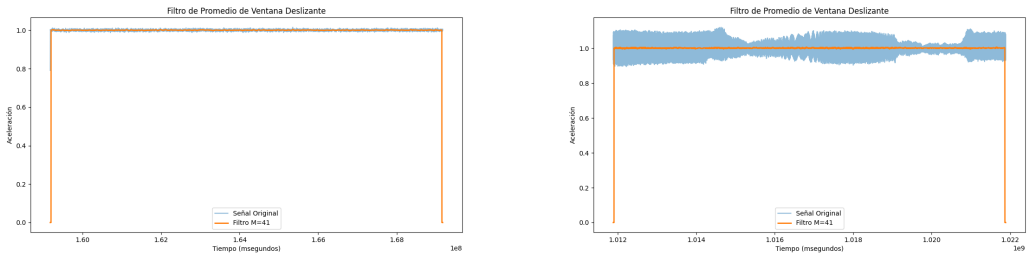
Donde:

- $y[i]$ es la salida del filtro en el instante i.
- $x[i]$ es la señal de entrada en el instante i.
- α es el factor de recuerdo, con valores entre 0 y 1.

La elección de α determina la rapidez con la que el filtro responde a cambios en la señal. Cuando aumentamos el valor de α, el filtro da más peso a los datos más recientes y menos a los antiguos. Por lo tanto, un α más alto significa una respuesta más rápida pero con una mayor susceptibilidad al ruido. Se experimentó con diferentes valores de α y se seleccionó α=0.3 (Figura 6.8) para mantener un equilibrio entre el filtrado del ruido y la preservación de las características de la señal original.

Filtro de promedio de ventana deslizante centrada

El filtro de promedio de ventana deslizante centrada calcula la media de un conjunto de puntos de datos dentro de una ventana móvil. La fórmula es 6.2:



(a) **(b)**

Figura 6.9: Comparación de la aplicación del filtro de promedio con M= 41 sobre la señal sin inyección de perturbaciones a) y cuando se añaden perturbaciones b).

$$y[i] = \frac{1}{M} \sum_{j=-(M-1)/2}^{M-1} x[i+j] \tag{6.2}$$

Donde:

- $y[i]$ es la salida del filtro en el instante i.
- $x[i]$ es la señal de entrada en el instante i.
- M es el tamaño de la ventana deslizante.

Se experimentó con diferentes valores de M (desde 11 hasta 41, en incrementos de 10). La elección de M en el filtro de promedio se vincula directamente con el periodo de muestreo de 1 milisegundo. Optamos por valores de M que son múltiplos del periodo de muestreo para asegurar que la ventana deslizante abarque intervalos completos. Dado que utilizamos una ventana deslizante centrada, M se selecciona como un valor impar para evitar desfases temporales. Sin embargo, este filtro resultó ser demasiado agresivo, comprometiendo la información esencial de la señal (Figura 6.9).

6.5 Desarrollo de un sistema de decisión simple para la detección de cabeceo

En el contexto de la inteligencia artificial (IA), un sistema de decisión es un componente o conjunto de algoritmos diseñados para tomar decisiones automáticamente en función de datos y reglas predefinidas. Estos sistemas se utilizan para abordar problemas complejos que involucran la toma de decisiones, y pueden ser parte de sistemas más grandes de IA.

Un sistema de decisión en IA generalmente implica los siguientes componentes:

1. **Entrada de Datos.** Recopila datos relevantes del entorno o del usuario para su análisis.
2. **Procesamiento de Datos.** Aplica algoritmos y técnicas de aprendizaje automático para analizar y procesar los datos.
3. **Modelo de Decisión.** Utiliza modelos predictivos, reglas o algoritmos para tomar decisiones basadas en los datos procesados.

4. **Salida de Decisiones.** Proporciona el resultado de la decisión, que puede ser una acción específica, una recomendación, o simplemente una clasificación.

En nuestro trabajo nos puede interesar aplicar sistemas de decisiones para detectar en tiempo real anomalías o cabeceos como hemos visto en el apartado 6.1.

Existen varios tipos de sistemas de decisión utilizados en la inteligencia artificial, algunos de los cuales incluyen:

1. **Sistemas de Reglas.** Utilizan reglas lógicas predefinidas para tomar decisiones. Si-entonces reglas son comunes en este enfoque.
2. **Redes Neuronales.** Modelos inspirados en la estructura y función del cerebro humano que pueden aprender patrones complejos a partir de datos.
3. **Máquinas de Soporte Vectorial (SVM).** Algoritmos de aprendizaje supervisado que pueden clasificar datos en categorías.
4. **Árboles de Decisión.** Estructuras de árbol que representan decisiones y sus posibles consecuencias.
5. **Lógica Difusa.** Permite manejar la incertidumbre en la toma de decisiones asignando grados de pertenencia a las categorías.
6. **Algoritmos de Aprendizaje Reforzado.** Aprenden a través de la interacción con el entorno, recibiendo retroalimentación en forma de recompensas o castigos.

El tipo de sistema de decisión utilizado dependerá del problema específico que esté siendo abordado y de los datos disponibles. En muchos casos, se combinan varios enfoques para lograr un sistema más robusto y eficiente. Además, con el avance de la investigación en inteligencia artificial, se desarrollan constantemente nuevos métodos y enfoques para mejorar la capacidad de toma de decisiones de los sistemas de IA.

En este apartado queremos estudiar la detección de cuándo la máquina ha producido un cabeceo. En este caso de estudio no es necesario construir ninguna red neuronal, desarrollaremos un sistema de decisión simple para llevar a cabo la tarea de detección del cabeceo.

Un sistema de decisión simple se caracteriza por depender de reglas lógicas claras y predefinidas, como las declaraciones "si-entonces". Su toma de decisiones se basa en condiciones y acciones específicas, y las reglas son transparentes y fáciles de entender. Estos sistemas no incorporan aprendizaje automático y, por lo tanto, no cambian con el tiempo o la experiencia. Además, son eficientes en términos de recursos computacionales.

La decisión tomada depende del procesamiento de los datos que se realice. Como se ha explicado en el apartado 6.3, para detectar los instantes de cabeceo en la máquina hemos tomado los valores de aceleración y a partir de estos aplicado un sistema de decisión que dado un instante si la sobreoscilación es mayor a la media entonces se tomaba como un posible punto de cabeceo. Para calcular dicha sobreoscilación y poder corrobar que es realmente un punto de cabeceo se han integrado los valores de acleración para obtener la velocidad por regresión lineal, eliminando las tendencias.

En cambio, una red neuronal es un modelo más complejo inspirado en la estructura del cerebro humano. Su arquitectura incluye capas de nodos interconectados, lo que permite el procesamiento de información a través de múltiples capas. Una de sus principales fortalezas radica en su capacidad para aprender patrones complejos a partir de datos durante el entrenamiento. Las redes neuronales son adaptables a nuevos datos y

situaciones, y tienen la capacidad de generalizar patrones aprendidos a partir de datos de entrenamiento a nuevos datos. Sin embargo, el entrenamiento y la ejecución de redes neuronales pueden requerir más potencia computacional en comparación con sistemas de decisión simples, especialmente para modelos grandes y complejos. Además, las redes neuronales pueden capturar relaciones más complejas y aprender representaciones abstractas de datos. La elección entre un sistema de decisión simple y una red neuronal depende del tipo de tarea y de los requisitos específicos del problema a abordar. Las redes neuronales las hemos aplicado para distinguir las frecuencias que producen en un funcionamiento "normal.º .ªnormal".

En el contexto de la detección de cabeceo, el desarrollo de un sistema de decisión simple implica la creación de reglas lógicas predefinidas para identificar el comportamiento específico asociado con el cabeceo. Estas reglas pueden basarse en características observables, como movimientos repetitivos de la cabeza o patrones de comportamiento.

En el libro de Gregory S. Parnell [10], se abordan enfoques y metodologías para la toma de decisiones en sistemas complejos. Aunque el libro no se centra específicamente en la detección de cabeceo, proporciona una base teórica y práctica valiosa para entender los principios generales de la toma de decisiones en el ámbito de la ingeniería de sistemas.

En este caso, el desarrollo del sistema de decisión simple puede aprovechar algunos conceptos discutidos en el libro, como la importancia de definir claramente los criterios de decisión y establecer reglas lógicas para guiar el proceso de toma de decisiones.

Por ejemplo, podemos establecer reglas lógicas como:

1. **Regla de Movimiento Repetitivo.** Si se detectan movimientos repetitivos de la cabeza en un periodo de tiempo específico, entonces clasificar como cabeceo.

2. **Regla de Patrón de Velocidad.** Si se observa un cambio brusco en la velocidad de movimiento de la cabeza, entonces clasificar como cabeceo.

3. **Regla de Duración.** Si la duración de los movimientos de la cabeza supera un umbral predefinido, entonces clasificar como cabeceo.

Estas reglas proporcionan un marco lógico y transparente para la detección de cabeceo, siguiendo el enfoque de un sistema de decisión simple.

En el desarrollo del sistema de decisión simple para la detección de cabeceo, aplicaremos reglas lógicas claras y predefinidas, utilizando la segunda regla mencionada (*Regla de Patrón de Velocidad*) anteriormente. Esta regla se centra en el patrón de velocidad de movimiento de la cabeza como un indicador potencial de cabeceo.

Esta regla implica que, al analizar los datos de entrada sobre la velocidad de movimiento de la cabeza, si se identifica un cambio significativo que sugiere un movimiento repetitivo característico del cabeceo, el sistema emitirá una decisión de clasificar la actividad como cabeceo.

Al aplicar esta regla en nuestro sistema de decisión, estamos incorporando un criterio específico y transparente para identificar el cabeceo. Este enfoque es consistente con la naturaleza de un sistema de decisión simple, donde las reglas lógicas predefinidas guían la toma de decisiones sin depender del aprendizaje automático o la adaptabilidad a nuevas situaciones.

```
float aceleracion_media = suma_aceleracion / ESPECTRO_DIM;
for (int i = 0; i < ESPECTRO_DIM; ++i) {
    if (abs(acx[i]-aceleracion_media) > 0.5) {
        if(indices_sobreoscilacion_x.empty() || (!indices_sobreoscilacion_x.
            empty() && xs[i]-xs[indices_sobreoscilacion_x.back()]>threshold)){
            indices_sobreoscilacion_x.push_back(i);
        }
    }
}
```

Listing 6.9: Comparación con la aceleración media.

6.5.1. Ejemplo de aplicación del sistema de decisión

En la Figura 6.10 se puede visualizar los datos de velocidad acumulada a lo largo del tiempo. Esta visualización proporciona una representación gráfica clara de cómo varía la velocidad en los ejes X e Y en respuesta a los cabeceos de la máquina.

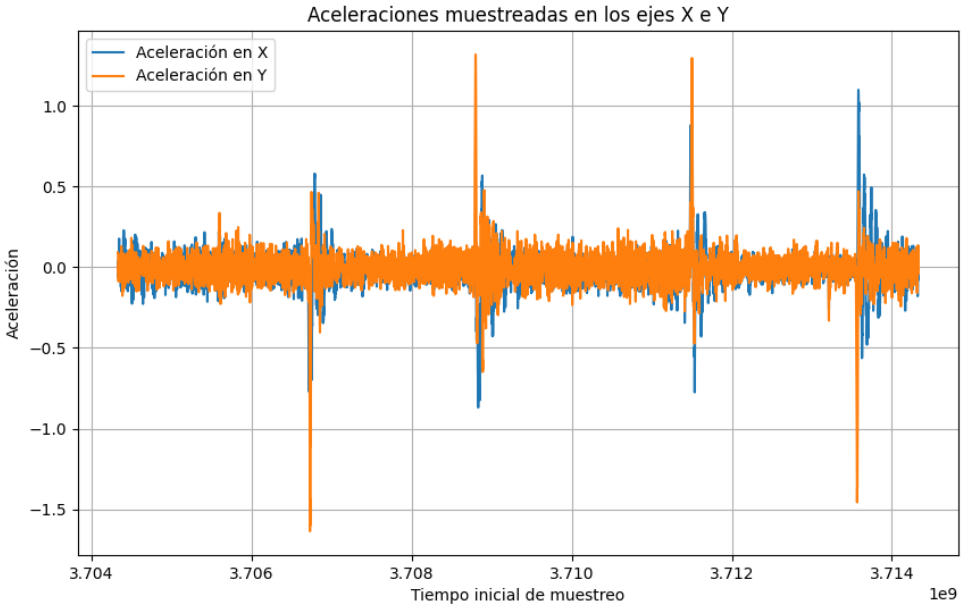
A continuación, se desarrollará un sistema de decisión simple para la detección de cabeceos en tiempo real utilizando la información de velocidad obtenida en este proceso. Este sistema de decisión se basará en reglas lógicas predefinidas para identificar patrones característicos asociados con los cabeceos de la máquina.

En la Figura 6.9 se calcula la aceleración media y compara cada valor de aceleración con esta media para detectar posibles sobreoscilaciones. Si la diferencia entre el valor de aceleración y la media es mayor que 0.5, se considera una posible sobreoscilación y se guarda su índice en el vector indices_sobreoscilacion_x.

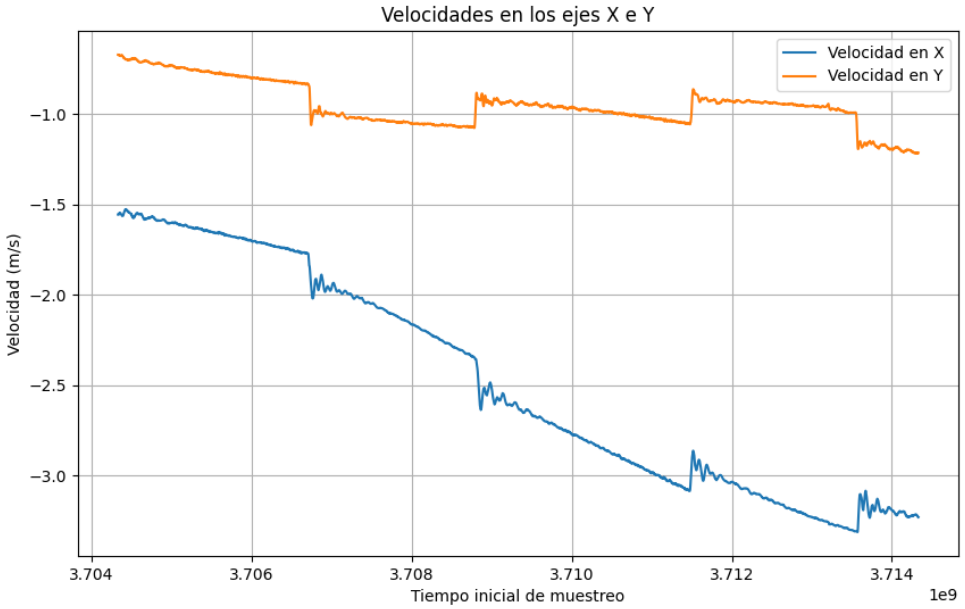
Para cada índice de sobreoscilación detectado en el paso anterior, se calcula un intervalo alrededor de este índice. Dentro de este intervalo, se calculan las diferencias de velocidad con respecto a los valores de velocidad en los extremos del intervalo (velocidad de entrada y salida). Se realiza un análisis simple de estas diferencias para calcular la media y la desviación estándar tanto antes como después del cambio. A continuación, se calcula la diferencia entre la desviación estándar de la velocidad antes y después del cambio (Figura 6.10). Al calcular la diferencia entre las desviaciones estándar antes y después del cambio, puedes observar cuánto cambia la variabilidad de la velocidad alrededor de los eventos de sobreoscilación. Una diferencia significativa indica que la sobreoscilación tiene un impacto en la estabilidad de la velocidad.

6.6 Entrenamiento de un perceptrón multicapa

Para entrenar un perceptrón multicapa con Python y sus librerías de Machine Learning, como scikit-learn y TensorFlow, se sigue un proceso estructurado. En primer lugar, se importan las bibliotecas necesarias, como NumPy para manipulación de datos y matplotlib para visualización. Luego, se carga el conjunto de datos y se preprocesa según las necesidades del modelo. Después, se construye el modelo de perceptrón multicapa utilizando la API de TensorFlow. Se definen la arquitectura de la red, la función de activación y otros hiperparámetros relevantes. Posteriormente, se realiza el entrenamiento del modelo utilizando los datos de entrenamiento. Durante este proceso, el perceptrón multicapa ajusta sus pesos para minimizar la función de pérdida. Finalmente, se evalúa el rendimiento del modelo en un conjunto de datos de prueba para verificar su capacidad de generalización. Este enfoque permite aprovechar la potencia y flexibilidad de las librerías de Machine Learning en Python para entrenar perceptrones multicapa de manera efectiva y realizar tareas como clasificación con facilidad.



(a)



(b)

Figura 6.10: Visualización de los cambios en las velocidades producidos por los cabeceos.

```
double mediaAntes = 0;
double mediaLuego = 0;
for (int ndx=0; ndx<delta; ndx++) {
    if (ndx < delta/2) {
        mediaAntes += vectorDifSalida[ndx];
    }
    if (ndx > delta/2) {
        mediaLuego += vectorDifSalida[ndx];
    }
}
mediaAntes /= (delta/2);
mediaLuego /= (delta/2);
//Calculo desviacion tipica
double sigmaAntes = 0;
double sigmaLuego = 0;
for (int ndx=0; ndx<delta; ndx++) {
    if (ndx < delta/2) {
        sigmaAntes += (vectorDifSalida[ndx] - mediaAntes) * (vectorDifSalida[
            ndx] - mediaAntes);
    }
    if (ndx > delta/2) {
        sigmaLuego += (vectorDifSalida[ndx] - mediaLuego) * (vectorDifSalida[
            ndx] - mediaLuego);
    }
}
sigmaAntes = sqrt(sigmaAntes/(delta/2));
sigmaLuego = sqrt(sigmaLuego/(delta/2));
double diferencia_sigma = sigmaLuego - sigmaAntes;
```

Listing 6.10: Cáculo de la desviación típica en la velocidad.

Además, como parte de este estudio, se entrenaron dos perceptrones multicapa para abordar diferentes aspectos del funcionamiento de la máquina de corte con láser. En primer lugar, se diseñó un perceptrón para detectar las altas frecuencias en comparación con el funcionamiento normal. Posteriormente, se llevó a cabo un segundo entrenamiento para identificar la presencia o ausencia de bajas frecuencias. El objetivo de entrenar estas redes neuronales es para ser capaces de diferenciar los casos de funcionamiento "normal"de la maquinaria frente al funcionamiento .ªnormal"de ésta.

Para fortalecer la evaluación del perceptrón multicapa, se aplica la técnica de k-fold cross-validation. Esta estrategia divide el conjunto de datos en k particiones (o "folds"). Luego, el modelo se entrena k veces, utilizando k-1 folds como datos de entrenamiento en cada iteración y el fold restante como conjunto de validación. Este proceso se repite hasta que cada fold ha sido utilizado como conjunto de validación exactamente una vez. La utilización de k-fold cross-validation proporciona varias ventajas: en primer lugar, contribuye a una evaluación más robusta del modelo al reducir la sensibilidad a la partición inicial de los datos. Además, permite obtener estadísticas más confiables sobre el rendimiento del modelo, ya que se promedian los resultados de k entrenamientos diferentes. Este enfoque ayuda a detectar posibles problemas de sobreajuste o subajuste y garantiza que el perceptrón multicapa tenga un rendimiento generalizado y sólido en distintos subconjuntos de datos. La implementación de k-fold cross-validation en conjunto con las librerías de Machine Learning en Python añade una capa adicional de confianza en la evaluación del modelo.

En la figura 6.11 podemos ver un extracto del código en Python para construir y entrenar un perceptrón multicapa.

Se plantea una arquitectura de modelo simple, computesta por:

- **Capa de entrada con NFEATURES nodos**, donde NFEATURES se corresponden al número de características extraídas de nuestro dataset preprocesado.
- **1 capa oculta de NEURONS_HIDDEN_LAYER nodos,** esta capa densa tendrá una función de activación de tipo ReLu, ya que se ha demostrado que esta función funciona bien en redes neuronales.
- **Capa de salida de NCLASSES nodos**, que coinciden con el número de clasificaciones posibles (en este caso 2). La función de activación que se emplea en la capa de salida es una función softmax. La función Softmax hace que la salida en conjunto sume 1, por lo que la salida puede interpretarse como probabilidades. El modelo hará su predicción según la opción que tenga la mayor probabilidad.

Para compilar nuestro modelo, utilizaremos los siguientes tres parámetros:

- **Función de pérdida.** Utilizaremos **categorical_crossentropy**, que es la opción más común para la clasificación. Una puntuación más baja indica que el modelo está funcionando mejor.
- **Métricas.** La métrica de **accuracy** nos permitirá ver la precisión en los datos de validación cuando entrenemos el modelo.
- **Optimizador.** Utilizaremos **adam**, para muchos casos de uso es generalmente un buen optimizador.

6.7 Entrenamiento de una red convolucional

Con los avances en el aprendizaje profundo y las redes neuronales, ahora es posible crear redes simples pero poderosas capaces de realizar tareas de clasificación, pronóstico y detección de objetos que anteriormente eran altamente complejas e incluso inviables. En esta sección, se detalla el proceso de entrenamiento y validación de dos redes convolucionales diseñadas para discernir situaciones específicas en el funcionamiento de la máquina de corte con láser: la presencia de altas frecuencias y la ocurrencia de bajas frecuencias. Inspirados en el trabajo de Christian [13], que demuestra la aplicabilidad de las CNN en problemas no relacionados con la visión por computadora, como la clasificación de sonidos a través de espectrogramas, hemos adaptado estas técnicas a nuestro estudio.

Hemos creado una representación visual de cada muestra de aceleración para identificar y extraer características clave. Siguiendo la metodología de Christian [13], generamos espectrogramas a partir de las señales de aceleración muestreadas. Los espectrogramas, visualizaciones del espectro de frecuencia en función del tiempo, proporcionan información valiosa para el análisis de señales temporales. Para obtener estos espectrogramas en Python, empleamos bibliotecas como matplotlib y scipy.

Siguiendo los pasos del artículo de Christian [13], convertimos los valores de aceleración en espectrogramas en el proceso de preparación de datos. En lugar de utilizar directamente las secuencias temporales de aceleración, se opta por representarlas en forma de espectrogramas. Esto se logra mediante la aplicación de la función de espectrograma de la biblioteca scipy.signal. El espectrograma divide la señal de aceleración en segmentos, calcula la Transformada de Fourier de cada segmento y visualiza la intensidad de la señal en el dominio de la frecuencia a lo largo del tiempo. Estos pasos incluyen también la definición de parámetros como la frecuencia de muestreo y el tamaño de ventana.

```
import numpy as np
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Dense, Activation
from sklearn.model_selection import train_test_split, KFold

def train_and_evaluate(X_train, y_train, X_test, y_test, epochs,
    neurons_per_layer):
    model = Sequential()
    model.add(Dense(neurons_per_layer, input_shape=[X_train.shape[1]],
        activation='relu'))
    model.add(Dense(neurons_per_layer, activation='relu'))
    model.add(Dense(NCLASSES, activation='softmax'))
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['
        accuracy'])

    model.fit(X_train, y_train, epochs=epochs, verbose=0)
    return model

def train_and_evaluate_kfold(X, y, epochs, neurons_per_layer, n_splits=5):
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=42)
    accuracies = []

    # Convertir X e y a arrays de numpy
    X = np.array(X)
    y = np.array(y)

    for train_index, test_index in kf.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]

        model = train_and_evaluate(X_train, y_train, X_test, y_test, epochs,
            neurons_per_layer)

        _, accuracy = model.evaluate(X_test, y_test, verbose=0)
        accuracies.append(accuracy)

    return np.mean(accuracies)

if __name__ == "__main__":
    print("********************** NN model compilation
        ***************************")
    modelo_keras = Sequential()
    modelo_keras.add(Dense(NEURONS_HIDDEN_LAYER, input_shape=[NFEATURES],
        activation='relu'))
    modelo_keras.add(Dense(NEURONS_HIDDEN_LAYER, activation='relu'))
    modelo_keras.add(Dense(NCLASSES, activation='softmax'))
    modelo_keras.compile(optimizer='adam', loss='categorical_crossentropy',
        metrics=['accuracy'])

    print("********************** Training with k-fold cross-validation
        ************************")
    mean_accuracy = train_and_evaluate_kfold(X_tr, y_tr, EPOCHS,
        NEURONS_HIDDEN_LAYER)
    print("Mean accuracy using k-fold cross-validation: {:.2f}%".format(
        mean_accuracy * 100))
```

Listing 6.11: Construcción y entrenamiento perceptrón multicapa.

La elección de un tamaño de ventana de 10,000 puntos y una frecuencia de muestreo de 1,000 Hz se basa en consideraciones específicas. El tamaño de la ventana (nperseg) se determina para capturar patrones locales en la señal de aceleración. En este caso, 10,000 puntos proporcionan una resolución temporal suficiente para analizar eventos locales en la secuencia temporal de 10 segundos. El solapamiento entre ventanas (noverlap) se establece en la mitad de la longitud de la ventana para equilibrar la resolución temporal y frecuencial. La elección de una frecuencia de muestreo de 1,000 Hz se alinea con la rapidez de variación esperada en la señal de aceleración, garantizando que la información relevante no se pierda durante el proceso de discretización.

Estos espectrogramas resultantes, que representan la distribución de la intensidad de la señal en el dominio de la frecuencia a lo largo del tiempo, se convierten en entrada para la red convolucional, permitiendo que el modelo capture patrones complejos y características temporales en el dominio del espectro durante el entrenamiento, en un enfoque análogo al procesamiento de imágenes. Un ejemplo de la imagen que se genera se muestra en la Figura 6.11.

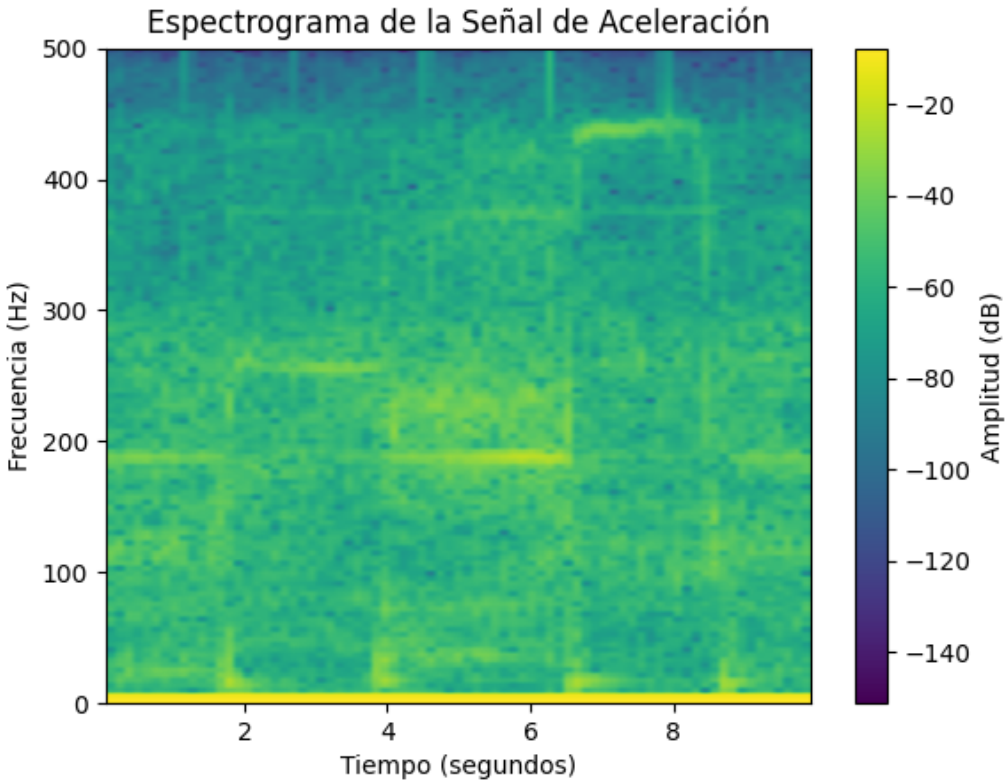


Figura 6.11: Transformación de la señal de aceleración a espectograma para entrenar una red convolucional.

A continuación, se proporciona el código implementado en Python para realizar la preparación del conjunto de datos para el entrenamiento y validación del modelo, en el que se aborada el problema de la clasificación entre el funcionamiento normal y la presencia de altas frecuencias.

En 6.12 se carga las imágenes desde el directorio y crea conjuntos de entrenamiento y validación utilizando tf.keras.preprocessing.image_dataset_from_directory.

Además, se define una función para aplicar transformaciones como la rescalación y la rotación a los conjuntos de datos de entrenamiento y validación. La función **prepare(ds,**

```
# Make a dataset containing the training spectrograms
train_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    batch_size=BATCH_SIZE,
    validation_split=0.1,
    directory=OUTPUT_DIR,
    shuffle=True,
    color_mode='rgb',
    image_size=(IMAGE_HEIGHT, IMAGE_WIDTH),
    subset="training",
    seed=0
)

# Make a dataset containing the validation spectrograms
valid_dataset = tf.keras.preprocessing.image_dataset_from_directory(
    batch_size=BATCH_SIZE,
    validation_split=0.1,
    directory=OUTPUT_DIR,
    shuffle=True,
    color_mode='rgb',
    image_size=(IMAGE_HEIGHT, IMAGE_WIDTH),
    subset="validation",
    seed=0
)
```

Listing 6.12: Creación de conjuntos de entrenamiento y validación de una red convolucional.

augment=False) 6.13 desempeña un papel fundamental en el preprocesamiento de los conjuntos de datos de entrenamiento y validación utilizados para entrenar un modelo de clasificación de imágenes. A continuación, desglosaré los motivos detrás de las transformaciones aplicadas, como la rescalación, la rotación y el volteo aleatorio.

La rescalación (**Rescaling(1./255)**) se realiza para normalizar los valores de píxeles en las imágenes. Muchas veces, las imágenes se representan con intensidades de píxeles en el rango de [0, 255]. Dividir cada valor de píxel por 255 coloca los datos en una escala más manejable, entre 0 y 1. Esta normalización facilita la convergencia de los algoritmos de optimización durante el entrenamiento del modelo. En esencia, la rescalación ajusta la escala de los datos para mejorar la eficiencia del proceso de aprendizaje.

La rotación y el volteo aleatorio (**RandomFlip** y **RandomRotation**) introducen variabilidad en los datos de entrenamiento. Esta variabilidad es crucial para mejorar la capacidad del modelo para generalizar a nuevas instancias. La rotación permite al modelo aprender patrones desde diferentes perspectivas, mientras que el volteo aleatorio simula diversas orientaciones y configuraciones en las imágenes. Estas transformaciones, aplicadas de manera aleatoria, imitan situaciones del mundo real y mejoran la capacidad del modelo para adaptarse a la diversidad en los datos de entrada.

En el código proporcionado 6.14, se define un modelo de red neuronal convolucional (CNN) secuencial para la clasificación de imágenes. A continuación, explicaré cada parte del bloque de definición del modelo.

Es importante destacar que estas transformaciones se implementan de manera específica para conjuntos de datos de imágenes y están diseñadas para preservar las etiquetas de clase asociadas con las imágenes transformadas. En otras palabras, si una imagen se rota o voltea, su etiqueta de clase se ajusta en consecuencia. Esto garantiza que la información de la clase se mantenga coherente a pesar de las transformaciones aplicadas.

El parámetro **augment** en la función **prepare** permite controlar si se aplican o no transformaciones de aumentación. Cuando **augment** es **True**, se aplican rotaciones y volteos aleatorios durante el entrenamiento para enriquecer el conjunto de datos con variabili-

```python
# Function to prepare our datasets for modelling
def prepare(ds, augment=False):
    # Define our one transformation
    rescale = tf.keras.Sequential([tf.keras.layers.experimental.preprocessing.
        Rescaling(1./255)])
    flip_and_rotate = tf.keras.Sequential([
        tf.keras.layers.experimental.preprocessing.RandomFlip("
            horizontal_and_vertical"),
        tf.keras.layers.experimental.preprocessing.RandomRotation(0.2)
    ])

    # Apply rescale to both datasets and augmentation only to training
    ds = ds.map(lambda x, y: (rescale(x, training=True), y))
    if augment: ds = ds.map(lambda x, y: (flip_and_rotate(x, training=True), y)
        )
    return ds

train_dataset = prepare(train_dataset, augment=False)
valid_dataset = prepare(valid_dataset, augment=False)
```

Listing 6.13: Aplicación de transformaciones en los conjuntos de datos de entrenamiento y validación de una red convolucional.

dad adicional. Por otro lado, cuando **augment** es **False**, solo se realiza la rescalación para mantener la coherencia entre los conjuntos de entrenamiento y validación, sin introducir variaciones adicionales.

Estas transformaciones son esenciales en el preprocesamiento de imágenes, ya que contribuyen significativamente a mejorar la robustez y la capacidad de generalización de los modelos de aprendizaje automático.

En el código proporcionado 6.14, se define un modelo de red neuronal convolucional (CNN) secuencial para la clasificación de imágenes.

Se define la capa de entrada de la red neuronal, especificando las dimensiones de las imágenes que se utilizarán para el entrenamiento (altura, ancho y canales de color).

Se utilizan tres capas convolucionales para extraer características de las imágenes. Cada capa Conv2D utiliza un kernel de tamaño 3x3 y la función de activación ReLU para introducir no linealidades.

Además, se insertan capas de normalización de lotes después de cada capa convolucional. La normalización de lotes ayuda a estabilizar y acelerar el entrenamiento al normalizar la entrada de cada capa.

Se aplican capas de MaxPooling después de cada par de capas convolucionales. Así, se reduce la dimensionalidad espacial de las representaciones, conservando las características más importantes.

La capa de aplanamiento (**Flatten**) transforma la salida de las capas convolucionales a un vector plano, preparándolo para ser utilizado en capas completamente conectadas.

Se utilizan dos capas densas (totalmente conectadas) para realizar la clasificación final. La primera capa densa tiene 256 unidades con activación ReLU.

Se inserta otra capa de normalización de lotes seguida de una capa de Dropout para regularizar y prevenir el sobreajuste.

La última capa densa tiene tantas unidades como clases en el problema de clasificación, y utiliza la función de activación softmax para obtener probabilidades que suman 1, indicando la probabilidad de pertenencia a cada clase.

```
# Build the model
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Input(shape=(IMAGE_HEIGHT, IMAGE_WIDTH, N_CHANNELS)))
model.add(tf.keras.layers.Conv2D(32, 3, strides=2, padding='same', activation='
    relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Conv2D(64, 3, padding='same', activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Conv2D(128, 3, padding='same', activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dense(256, activation='relu'))
model.add(tf.keras.layers.BatchNormalization())
model.add(tf.keras.layers.Dropout(0.5))
model.add(tf.keras.layers.Dense(N_CLASSES, activation='softmax'))
```

Listing 6.14: Definición del modelo de una red convolucional.

```
# Compile model
model.compile(
    loss='sparse_categorical_crossentropy',
    optimizer=tf.keras.optimizers.RMSprop(),
    metrics=['accuracy'],
)
```

Listing 6.15: Compilación del modelo de una red convolucional.

Una vez definido el modelo, se compila 6.15 el modelo especificando la función de pérdida (**sparse_categorical_crossentropy**), ideal para problemas de clasificación con etiquetas enteras. El optimizador (**RMSprop**), conocido por ajustar las tasas de aprendizaje de manera adaptativa, superando problemas de inestabilidad y subóptimas tasas fijas. La métrica seleccionada es la precisión (**accuracy**), común en problemas de clasificación, que evalúa la fracción de predicciones correctas durante el entrenamiento y evaluación del modelo. Esta configuración proporciona una base sólida para el entrenamiento y evaluación de la red neuronal en el conjunto de datos.

Así, cuando hemos compilado el modelo implementamos una validación cruzada k-fold estratificada con 5 pliegues para evaluar el modelo de la red convolucional en el conjunto de datos proporcionado 6.16. En cada iteración del bucle, se dividen los datos en conjuntos de entrenamiento y prueba, y el modelo se entrena durante 10 épocas. La validación se realiza en el conjunto de prueba, y se recopilan métricas como la precisión y la matriz de confusión. Estas métricas se almacenan para cada fold y se utilizan para evaluar la variabilidad del rendimiento del modelo en diferentes conjuntos de prueba, proporcionando una evaluación robusta de su desempeño general. Sin embargo, debido a la limitación del conjunto de datos, los resultados de precisión no alcanzan un rendimiento óptimo.

```
# Configurar k-fold cross-validation
kfold = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

# Listas para almacenar resultados
accuracy_scores = []
confusion_matrices = []

# Iterar sobre los folds
for train_index, test_index in kfold.split(X, y_encoded):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y_encoded[train_index], y_encoded[test_index]

    # Calcular pesos de clase para abordar el desequilibrio de clases
    class_weights = class_weight.compute_class_weight('balanced', np.unique(
        y_train), y_train)

    # Entrenar el modelo
    history = model.fit(X_train, y_train, epochs=10, validation_data=(X_test,
        y_test), class_weight=dict(enumerate(class_weights)))

    # Evaluar el modelo en el conjunto de prueba
    _, accuracy = model.evaluate(X_test, y_test)
    accuracy_scores.append(accuracy)

    # Obtener predicciones y matriz de confusion
    y_pred = model.predict_classes(X_test)
    cm = confusion_matrix(y_test, y_pred)
    confusion_matrices.append(cm)
```

Listing 6.16: Entrenamiento y validación de la red convolucional.

6.8 Automatización del entrenamiento de modelos de Machine Learning

Una vez automatizado los procesos de adquisición de datos, preprocesamiento y etiquetado de estos. Vamos a automatizar la parte de entrenamiento con modelos de Machine Learning. Para ello generaremos unos scripts en Python que tomarán el conjunto de datos procesados y se lo pasará al modelo que queramos entrenar y validar. Así, simplemente tendremos que cambiar o modificar la estructura del modelo que queramos hacer uso. Los datos que le estamos pasando a los modelos serán la transformada de Fourier computada a partir de los datos de aceleración medidos en los ejes x, y, z en nuestro pórtico.

A continuación, se proporciona las partes del script que automatizan el proceso de entrenamiento y evaluación de un modelo cualquiera de Machine Learning para la clasificación, utilizando los datos etiquetados proporcionados en el archivos CSV.

En la Figura 6.17, se lee un archivo CSV que contiene los datos etiquetados, divide estos datos en conjuntos de entrenamiento y prueba, y prepara las etiquetas de clase para su uso en el modelo que queramos entrenar. En el código se ve cómo se convierten los arrays numpy a listas de Python que son las que se usan como entrada de Keras. Cada elemento del vector de clasificación "np_y"(1, 2 o 3) lo convertiremos en un vector de dimensión NCLASSES (3) que indica la confianza de pertenencia a cada clase.

Luego, en la Figura 6.18 se puede visualizar cómo se carga el modelo desde el archivo h5. Se realiza predicciones en el conjunto de datos de prueba y evalúa el rendimiento del modelo calculando la tasa de éxito. Además, calcula y muestra la matriz de confusión para una evaluación más detallada del rendimiento del modelo.

```
df = pd.read_csv(dataFilename, sep=' ')
    NCOLUMNAS = len(df.columns);
    NFEATURES = NCOLUMNAS - 2;

    np_y = df.iloc[:,[0]].to_numpy().reshape(len(df));
    np_X = df.iloc[:,range(1,NFEATURES+1)].to_numpy();

    # Split into training and testing data
    X_train, X_test, y_train, y_test = train_test_split(np_X, np_y, test_size
        =0.50, random_state=100)

    y_t1 = []
    for i in y_train:
        if i==1:
            y_t1.append([1.0,0.0,0.0]);
        if i==2:
            y_t1.append([0.0,1.0,0.0]);
        if i==3:
            y_t1.append([0.0,0.0,1.0]);
    X_train = X_train.tolist()
    X_test = X_test.tolist()
```

Listing 6.17: Automatización del entrenamiento de un modelo de Machine Learning.

```
    modelo_keras = load_model(modelFilename)

    predicciones = modelo_keras.predict(X_test)
    aciertos = 0
    res = 0;
    for p in predicciones:
        if (p[0] >= p[1] and p[0] >= p[2]):
            res = 1;
        elif (p[1] >= p[2] and p[1] >= p[0]):
            res = 2;
        elif (p[2] >= p[1] and p[2] >= p[0]):
            res = 3;
        if y_test[indx] == res:
            aciertos = aciertos + 1;
        indx = indx +1;
    ratio = aciertos/len(predicciones);
    print("************************************************************");
    print("Using model ",modelFilename, "over the dataset ", dataFilename, "
        Success ratio: ", ratio, " in ", len(predicciones), " processed.")
    print("************************************************************");

    # confusion matrix
    cm = confusion_matrix(y_test,predict_y)
```

Listing 6.18: Automatización de la validación en el entrenamiento de un modelo de Machine Learning.

CAPÍTULO 7
Pruebas y Resultados

En este capítulo presentaremos los resultados obtenidos tras aplicar las distintas técnicas desarrolladas en el Capítulo 6. Durante el desarrollo de nuestro estudio, nos enfrentamos al desafío de analizar y comprender las señales de aceleración en un entorno donde el ruido aleatorio era una fuente significativa de interferencia. Para abordar esta problemática, implementamos diversas estrategias de filtrado digital con el objetivo de mejorar la calidad de las mediciones y facilitar la detección de eventos anómalos.

Además, exploramos el rendimiento de modelos de aprendizaje automático, específicamente perceptrones y redes convolucionales, para la detección y clasificación de patrones en las señales de aceleración. A continuación, analizaremos en detalle los resultados obtenidos y discutiremos las implicaciones de estos hallazgos en el contexto de nuestro estudio.

7.1 Ejemplo de datos experimentales obtenidos

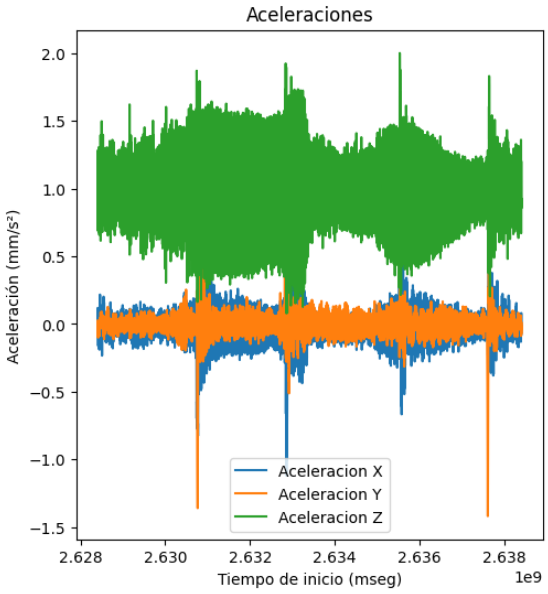
A continuación, vamos a analizar los datos experimentales obtenidos en la toma de muestras de aceleración en la sección 6.2.

Las Figuras 7.1a, 7.1b y 7.1c muestran el cambio de aceleración sobre el tiempo de las señales obtenidas a una velocidad y aceleración de 9000 mm/s y 6000 mm/s², respectivamente. En las tres Figuras se puede analizar cómo varia la aceleración en el caso de que no se hayan generado ningún tipo de perturbación (Figura 7.1c), se hayan producido perturbaciones de altas frecuencias (Figura 7.1a) o de bajas frecuencias (Figura 7.1b).

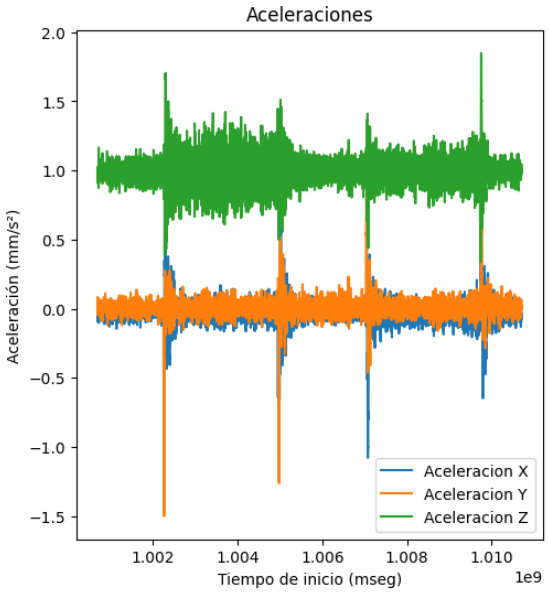
Si hacemos zoom sobre las señales (Figuras 7.2c y 7.2a), podemos apreciar como la amplitud de la aceleración es mucho mayor en el caso de que se produzca alguna perturbación de altas frecuencias sobre la máquina respecto de la señal de la maquinaria funcionando correctamente. Por como está dispuesto el motor que añade las vibraciones sobre el pórtico, éstas se pueden ver sobre el eje z. En el caso de que se produzcan perturbaciones de bajas frecuencias, también podemos a simple vista apreciar que la aceleración aumenta, aunque no tanto como en el caso de altas frecuencias (Figura 7.2b).

Pero tan solo con la señal original, es difícil apreciar a simple vista el cabeceo de la maquinaria. Para ello se van a analizar las señales más profundamente, para ello visualizaremos las señales en el dominio de la velocidad.

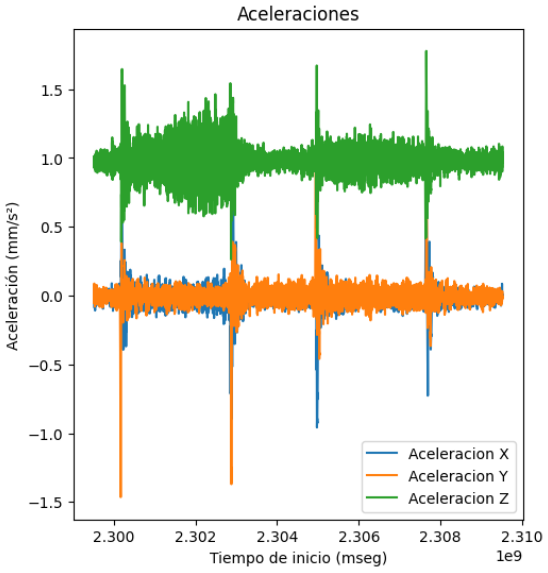
A una velocidad de 9000 mm/s y y aceleración de 6000 mm/s², podemos ver en las Figuras 7.3a y 7.3b que cuando se va a producir el cabeceo de la maquinaria, la velocidad se decrementa drásticamente.



(a)



(b)



(c)

Figura 7.1: Ejemplos de señal completa con inyección de altas frecuencias completa a), con inyección de bajas frecuencias completa b) y de la maquinaria en buen estado c).

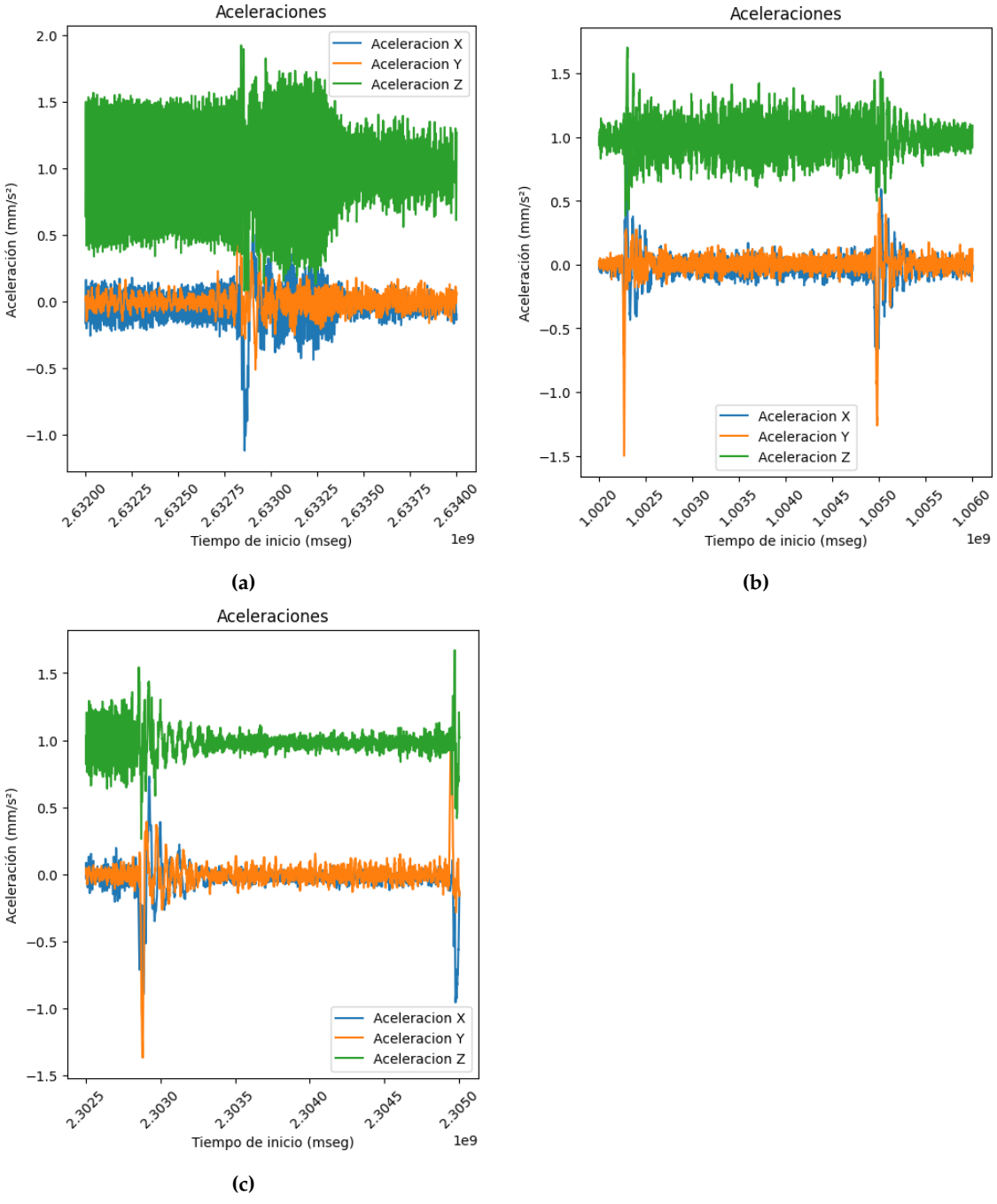
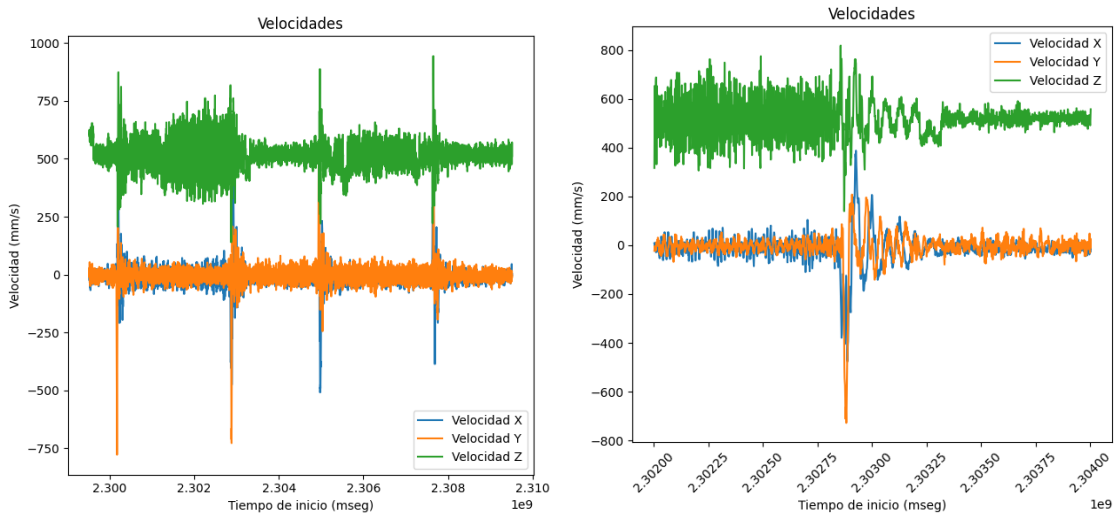


Figura 7.2: Ejemplos de señal ampliada con inyección de altas frecuencias a), con inyección de bajas frecuencias b) y de la maquinaria en buen estado c).



(a) **(b)**

Figura 7.3: Gráficas de la velocidad de una señal a) y ampliación de la velocidad de la señal muestreada b).

7.2 Evaluación del filtrado del ruido aleatorio de una señal

En la sección 6.4 tomamos muestras de aceleración con la máquina en reposo. Tras aplicar ambos filtros a las muestras, se observó que el filtro exponencial lograba un filtrado eficiente del ruido ambiental, preservando las características fundamentales de la señal. La elección de $alpha$ fue crucial, ya que un valor demasiado alto resultó en una respuesta excesivamente rápida pero con mayor susceptibilidad al ruido. Se seleccionó $alpha$=0.3 para equilibrar la respuesta y la supresión de ruido.

A continuación, tomamos muestras de aceleración con la máquina en movimiento, tanto añadiendo perturbaciones como sin añadirlas. Esto nos ha permitido evaluar la eficacia del filtrado en condiciones tanto estáticas como en dinámicas.

Para verificar la capacidad del filtro exponencial para detectar perturbaciones, se aplicó la Transformada de Fourier sobre la señal muestreada. Esto permitió identificar perturbaciones tanto a altas como a bajas frecuencias durante el movimiento de la máquina (Figura 7.4).

El filtro exponencial implementado en Python demostró ser eficaz en la eliminación del ruido aleatorio de las muestras de aceleración, permitiendo una identificación clara de perturbaciones en la señal. Este enfoque ofrece una mejora significativa en la calidad de las mediciones y proporciona una base sólida para el análisis de datos en entornos ruidosos. La elección cuidadosa de $alpha$ en el filtro exponencial es crucial para lograr un equilibrio óptimo entre el filtrado del ruido y la preservación de la información relevante de la señal. A medida que $alpha$ aumenta, la respuesta del filtro se vuelve más rápida, pero se vuelve más sensible al ruido.

7.3 Validación de un sistema de decisión simple para la detección de cabeceo

La implementación del sistema de decisión explicado en la sección 6.5 sigue un enfoque claro y transparente, basándose en reglas lógicas predefinidas para identificar el cabeceo sin depender de técnicas de aprendizaje automático. El umbral utilizado propor-

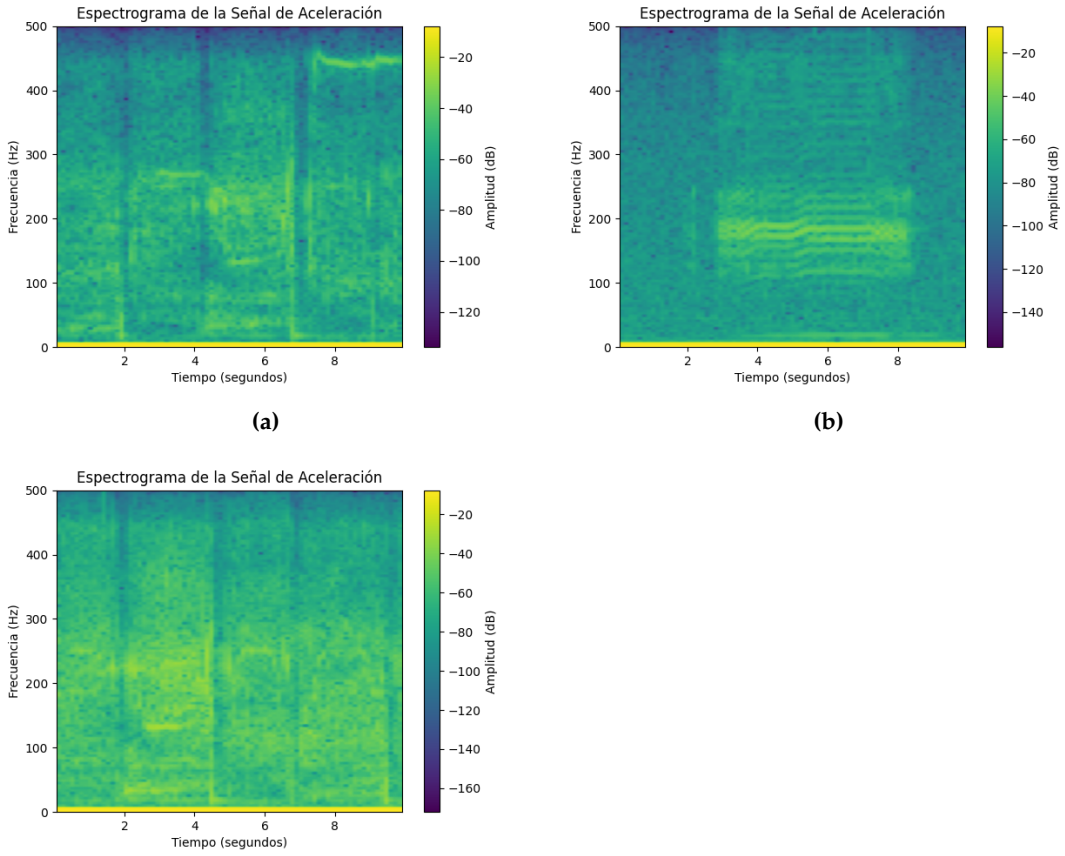


Figura 7.4: Comparación de frecuencias e intensidades de la señal filtrada inyectando frecuencias altas a), inyectando frecuencias bajas b) y sin inyectar ningún tipo de perturbación c).

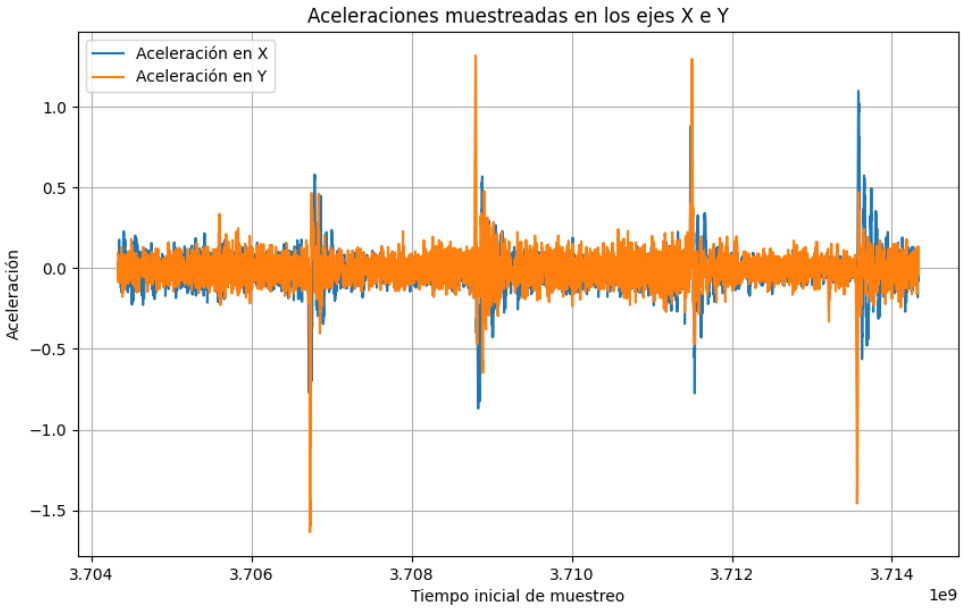


Figura 7.5: Visualización de los cabeceos en la señal de aceleración.

ciona flexibilidad para ajustar la sensibilidad del sistema según las necesidades específicas del problema.

Instante	Diferencia
3706725814	0.0147822
3708789813	0.0183688
3711493814	0.0143531
3713556813	0.0154521

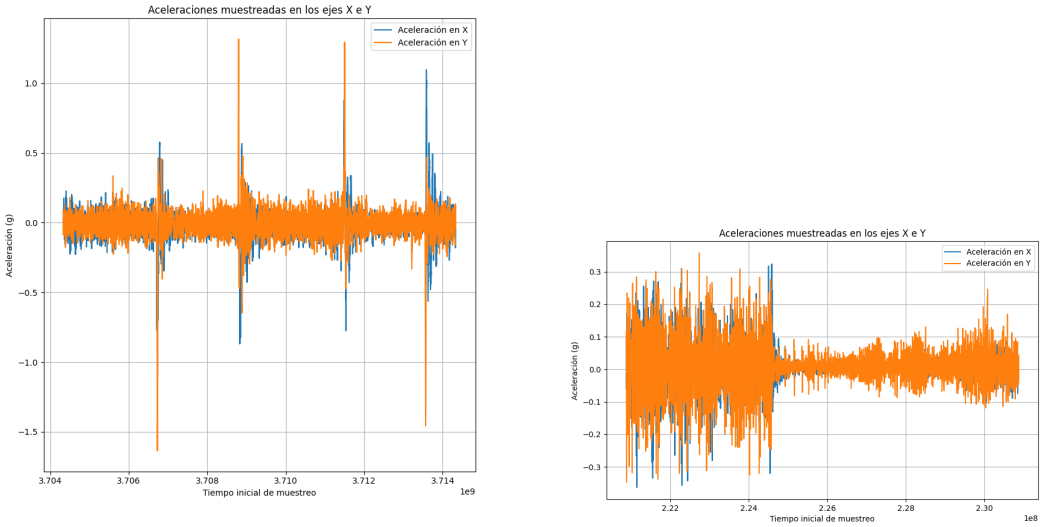
Tabla 7.1: Tabla con instantes de tiempo extraídos en la detección de cabeceo.

Todos los instantes de tiempo donde se han detectado que se ha producido un cabeceo se han almacenado en un archivo.

Mediante un análisis de este archivo, se puede verificar de manera efectiva la precisión del sistema de detección. Específicamente, en la Tabla 7.1 los instantes de tiempo registrados en el archivo corresponden a aquellos momentos en los cuales la señal de aceleración (Figura 7.5), indica que ha ocurrido un cabeceo.

Esta estrategia permite una validación visual y manual de los resultados, ya que los instantes registrados en el archivo pueden ser comparados directamente con la señal ampliada para corroborar la correspondencia con los eventos de cabeceo detectados.

Hemos comprobado que el sistema de decisión detecta realmente cabeceos y no puntos donde se aumentan la aceleración. Se han hecho diferentes valores de acleración y velocidad máxima en el movimiento del pórtico donde no se produjesen cabeceos. Así podemos diferenciar escenarios donde se producen cabeceos (Figura 7.6a) y en los que no (Figura 7.6b).



(a) **(b)**

Figura 7.6: Diferencias entre el escenario donde se produce cabeceo y donde solo hay un aumento en la aceleración del movimiento.

7.4 Validación de un perceptrón multicapa

Tras entrenar el modelo desarrollado en la sección 6.6 evaluaremos su precisión y se presentará la matriz de confusión del algoritmo (Figura 7.7a). Una matriz de confusión es una herramienta esencial en la evaluación de modelos de clasificación en Machine Learning. Te permite analizar el rendimiento del modelo al comparar las predicciones con los resultados reales. En la matriz de confusión, los elementos diagonales representan las predicciones correctas, mientras que los elementos fuera de la diagonal indican errores. Con esta información, puedes calcular métricas como precisión, sensibilidad, especificidad y la tasa de falsos positivos. En resumen, la matriz de confusión proporciona una visión detallada y cuantitativa del rendimiento de tu modelo de cómo el perceptrón multicapa realiza sus predicciones, permitiendo ajustes y mejoras según sea necesario.

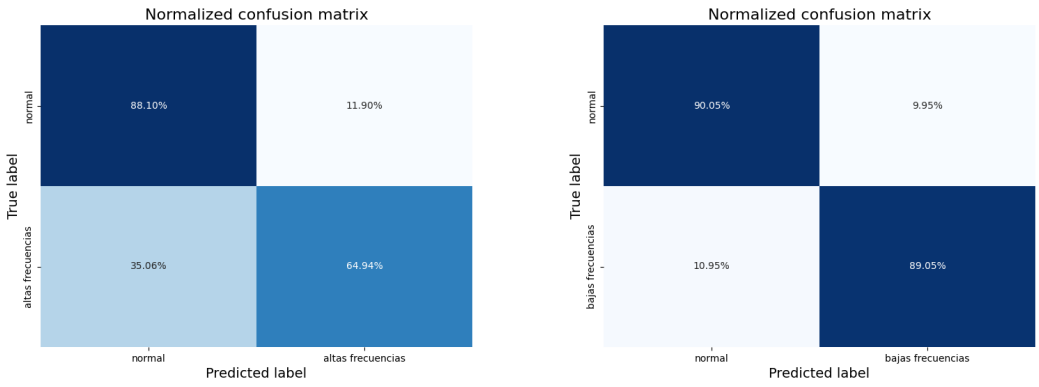
Después de 10 épocas y de entrenar y validar el modelo con el método de validación cruzada k-fold con k=5, obtenemos una precisión promedia del 99.81 %.

El análisis de la matriz de confusión (Figura 7.7a) proporciona una evaluación detallada del rendimiento del perceptrón multicapa en la tarea de clasificación. A pesar de la impresionante precisión promedio del 99.81 %, es esencial profundizar en los resultados específicos de la matriz para una comprensión más completa.

En cuanto a las muestras etiquetadas como Normal (valor 2 en el eje x), el modelo ha demostrado una precisión del 88.10 %. Esto significa que el 88.10 % de las veces, el perceptrón ha acertado al predecir situaciones de funcionamiento normal. Esta métrica destaca la capacidad del modelo para identificar correctamente eventos normales.

En el caso de las muestras etiquetadas como Altas Frecuencias (valor 3 en el eje x), el perceptrón ha logrado una precisión del 64.94 %. Aunque esta cifra es ligeramente menor que la precisión para la clase Normal, aún indica un rendimiento sólido en la detección de situaciones de Altas Frecuencias.

Es importante examinar los errores cometidos por el modelo. Los Falsos Positivos (FP), que representan el 11.90 % de las predicciones incorrectas, indican casos en los que



(a) **(b)**

Figura 7.7: Matriz de confusión del perceptrón multicapa entrenado para discernir el escenario donde se producen altas frecuencias a) y para discernir el escenario donde se producen bajas frecuencias b).

el modelo ha predicho Altas Frecuencias cuando el escenario era realmente de funcionamiento normal. Este tipo de error, puede tener implicaciones específicas según el contexto de la aplicación. En nuestro caso de estudio: detenciones innecesarias de la máquina, desgaste innecesario de componentes, ineficiencias operativas, confianza en el sistema de monitoreo, etc. Por otro lado, los Falsos Negativos (FN), que constituyen el 35.06 % de los errores, señalan situaciones en las que el modelo ha predicho incorrectamente Normal en presencia de Altas Frecuencias.

Continuando con la explicación proporcionada, ahora añadiré el análisis de los resultados obtenidos al entrenar otro perceptrón para discernir los casos en los que se inyectan bajas frecuencias y en los que no.

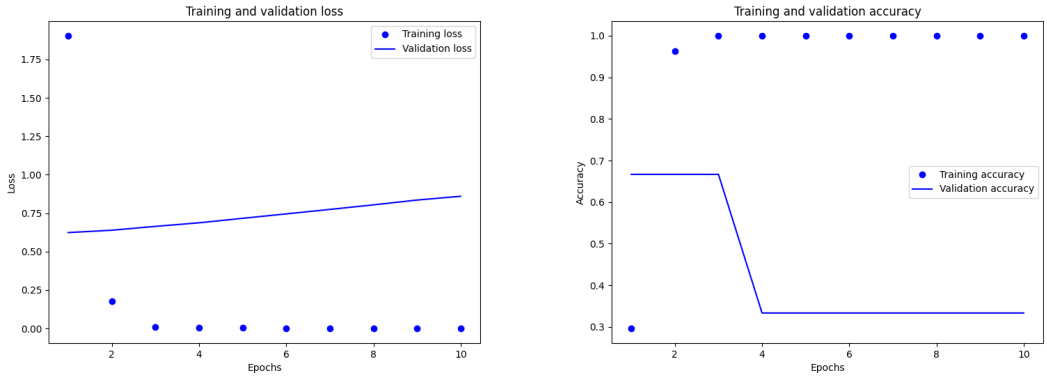
El segundo perceptrón, destinado a la detección de bajas frecuencias, demostró un rendimiento destacado con una precisión media del 99.92 %. Sin embargo, es crucial analizar detenidamente los resultados obtenidos para comprender la efectividad y limitaciones del modelo.

Como podemos observar en la matriz de confusión de la Figura 7.7b, en las predicciones de bajas frecuencias, se observó que el 9.95 % de las veces el modelo indicó la presencia de bajas frecuencias cuando, de hecho, el escenario era de funcionamiento normal.

En contraste, el modelo también mostró un comportamiento similar del 9.95 % al predecir la ausencia de bajas frecuencias cuando, de hecho, se estaban inyectando. Estos casos de Falsos Negativos resaltan situaciones en las que el modelo no logró detectar la presencia de bajas frecuencias, lo cual puede ser crítico dependiendo de las consecuencias asociadas. En el caso de la máquina de corte con láser: riesgo de daño a la máquina, calidad del corte deficiente, pérdida de precisión, inestabilidad del proceso de corte, aumento del riesgo de accidentes, etc.

Además, al analizar las predicciones en los casos en los que se predijo un buen funcionamiento, se encontró que el modelo acertó en un 90.05 % de las ocasiones. No obstante, el restante porcentaje representó predicciones incorrectas, mostrando áreas donde el modelo podría beneficiarse de ajustes adicionales para mejorar su capacidad de generalización.

Este enfoque dual de entrenar dos perceptrones distintos permite abordar de manera más específica y especializada los diferentes aspectos del comportamiento de la máqui-



(a) **(b)**

Figura 7.8: Gráficas que muestran la tasa de pérdidas en el entrenamiento y validación de la red convolucional para discernir cuando se generan frecuencias altas a) y la precisión en el entrenamiento y validación de la red convolucional para discernir cuando se generan frecuencias altas b).

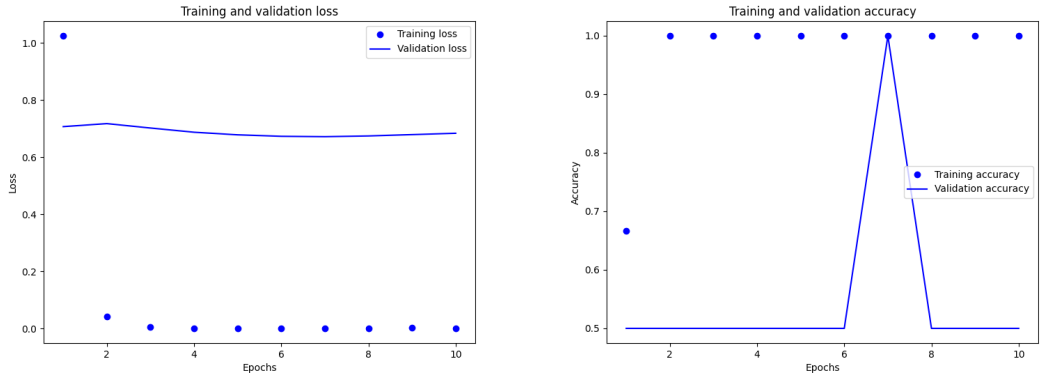
na de corte con láser. Aunque cada modelo ha demostrado una alta precisión, el análisis detallado de los resultados resalta la importancia de comprender las características específicas de cada tarea y optimizar los modelos en consecuencia.

En resumen, a pesar de la alta precisión general, el análisis detallado de la matriz de confusión destaca áreas específicas donde el modelo puede beneficiarse de ajustes adicionales. La reducción de Falsos Positivos y Falsos Negativos podría ser un enfoque clave para mejorar aún más la capacidad del perceptrón multicapa, asegurando un rendimiento óptimo en contextos de clasificación más específicos.

7.5 Validación de una red convolucional

Dado que las redes convolucionales requieren conjuntos de datos significativos para un entrenamiento efectivo, nuestro estudio se enfrenta a la limitación de un conjunto de datos relativamente pequeño en comparación con las demandas de una red convolucional típica. Esta limitación se traduce en una baja precisión y un error elevado después del entrenamiento y la validación del modelo (Figuras 7.8a y 7.8b). Como se observa en las gráficas correspondientes, se evidenció una tasa de error considerablemente elevada y una baja tasa de precisión durante la validación del modelo. Este fenómeno se atribuye a las limitaciones en la cantidad de muestras disponibles para entrenar el modelo. La generación de un conjunto de datos más amplio es esencial para mejorar la capacidad del modelo para generalizar patrones en los datos, lo cual se ve afectado por la falta de variabilidad y representatividad en el conjunto de datos actual. A pesar de estas limitaciones, se realizaron esfuerzos para abordar el desequilibrio de clases mediante el cálculo de pesos de clase. Las conclusiones derivadas de este análisis subrayan la importancia de disponer de recursos adecuados para la recopilación de datos, lo cual impacta directamente en la calidad y eficacia de los modelos de aprendizaje automático.

Para abordar la clasificación entre el funcionamiento normal y la ocurrencia de bajas frecuencias, se implementó otra red convolucional. Siguiendo un enfoque similar, se generaron espectrogramas como entrada para la red. Se observaron los mismos desafíos significativos en este escenario también debido a la falta de variabilidad y representación en el conjunto de datos.



(a) **(b)**

Figura 7.9: Gráficas que muestran la tasa de pérdidas en el entrenamiento y validación de la red convolucional para discernir cuando se generan frecuencias bajas a) y la precisión en el entrenamiento y validación de la red convolucional para discernir cuando se generan frecuencias bajas b).

El modelo se entrenó con la misma arquitectura y configuración que la red anterior. No obstante, la limitación de datos resultó en una baja precisión final del 50 % (Figura 7.9a) y una tasa de pérdidas del 68.37 % (Figura 7.9b). La matriz de confusión indicó que el modelo tuvo dificultades tanto en la identificación de las bajas frecuencias como en la diferenciación precisa del funcionamiento normal.

En resumen esta sección proporciona una visión detallada del proceso de entrenamiento y validación de la red convolucional, destacando los desafíos asociados con conjuntos de datos pequeños. La baja precisión y la elevada tasa de error destacan las limitaciones de los modelos en la clasificación de frecuencias en este contexto específico. A pesar de las limitaciones, se extraen lecciones valiosas sobre la aplicabilidad de las CNN en la clasificación de datos de aceleración. Se recomienda la expansión del conjunto de datos y la exploración de estrategias adicionales, como el aumento de datos, para mejorar la capacidad del modelo en la identificación de patrones complejos en las señales de aceleración.

7.6 Validación algoritmo de 3 buffers para el procesamiento continuo

Una vez implementado el algoritmo de 3 buffers, comprobamos que se siguen cumpliendo los periodos de muestreo de 1 ms (Figura 6.11).

Este diseño proporciona un método eficiente y seguro para el procesamiento continuo de señales de aceleración, permitiendo detectar perturbaciones tanto cuando la máquina está en marcha (Figura 7.12) como cuando está detenida (Figura 7.11).

7.7 Validación de la automatización del entrenamiento de modelos de Machine Learning

Con los scripts en Python que nos permitirá automatizar el proceso de extracción de los archivos .csv las características etiquetadas y el proceso de entrenamiento y valida-

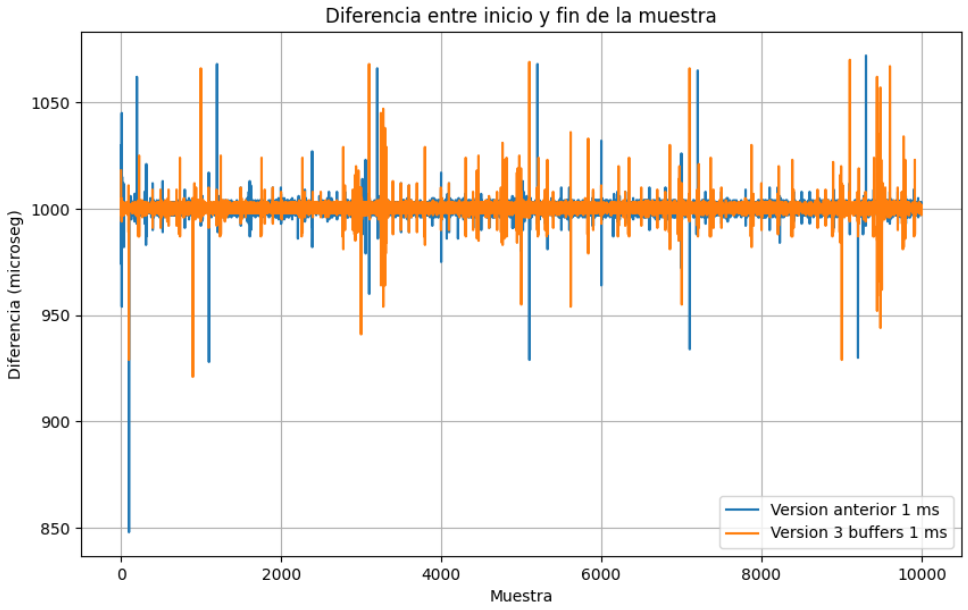


Figura 7.10: Comprobación algoritmo 3 buffers cumple el plazo de muestreo de 1 ms.

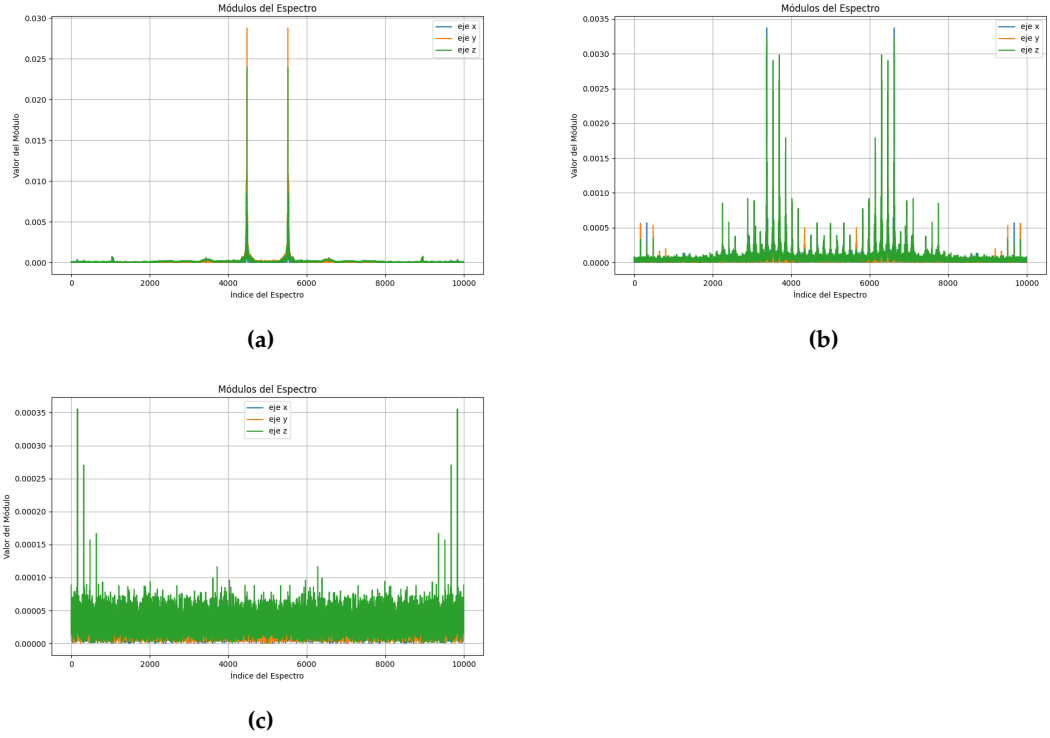


Figura 7.11: Comparación del espectro de frecuencias de la señal de aceleración inyectando frecuencias altas a), inyectando frecuencias bajas b) y sin inyectar ningún tipo de perturbación c), con la máquina detenida.

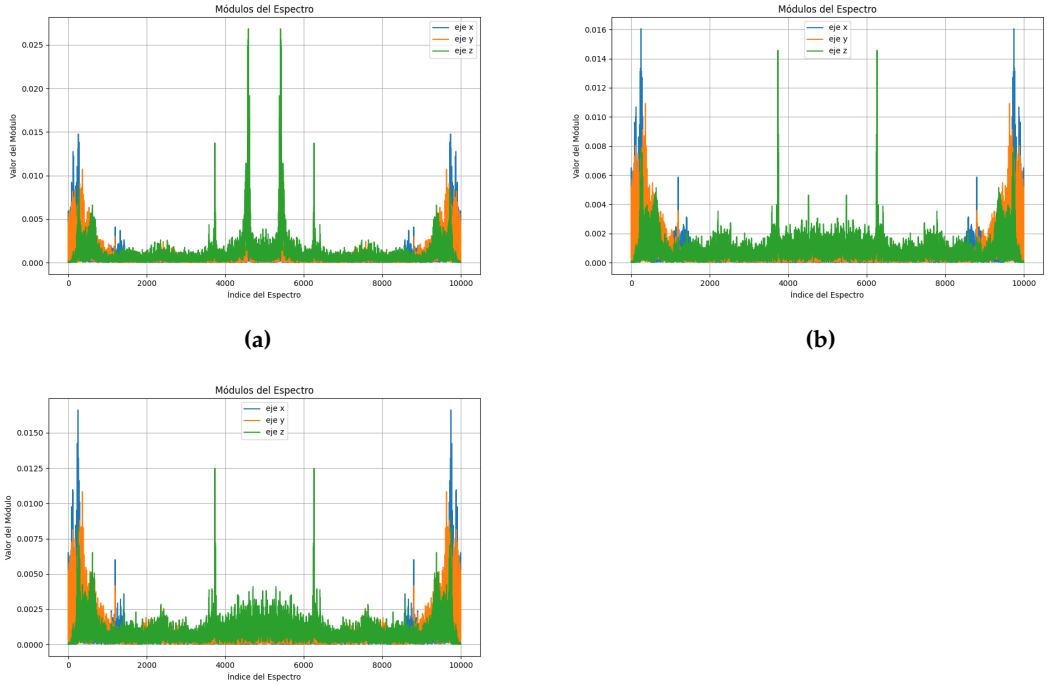
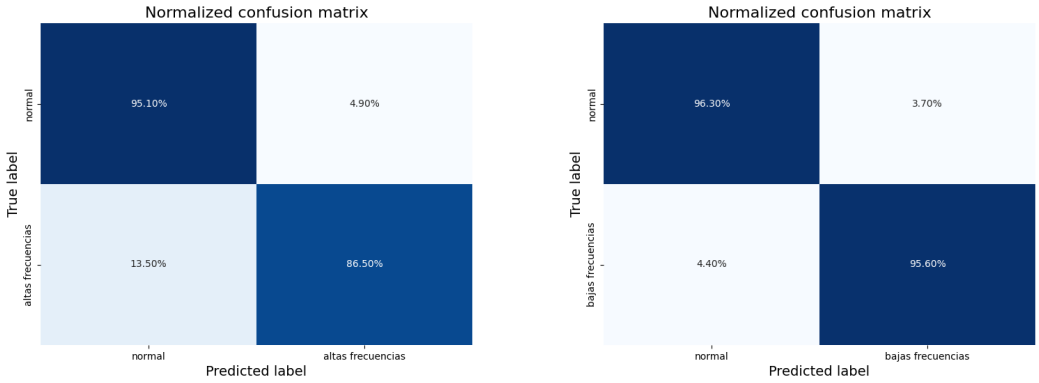


Figura 7.12: Comparación del espectro de frecuencias de la señal de aceleración inyectando frecuencias altas a), inyectando frecuencias bajas b) y sin inyectar ningún tipo de perturbación c), con la máquina en movimiento.

ción de un modelo de Machine Learning. Vamos a construir y entrenar un modelo de perceptrón multicapa, y evaluar su precisión de predicción.

En la figura 7.13 podemos ver que el entrenamiento de los perceptrones multicapa con los datos de aceleración preprocesados en ambos estudios han alcanzado precisiones de predicción óptimas.

Con todo esto podemos concluir que el trabajo desarrollado a lo largo de estos meses ha dado sus frutos. Partiendo de un concepto inicial, hemos conseguido poner en marcha y desarrollar nuestra prueba de concepto sobre un prototipo real.



(a) **(b)**

Figura 7.13: a) Matriz de confusión del perceptrón multicapa para detectar altas frecuencias y b) matriz de confusión del perceptrón multicapa para detectar bajas frecuencias, de forma continua.

CAPÍTULO 8
Conclusiones

Durante este estudio, nos hemos adentrado en el apasionante mundo del análisis de señales de aceleración en entornos industriales, donde el ruido y las perturbaciones representan un desafío constante. A través de la implementación de diversas técnicas de procesamiento de señales y aprendizaje automático, hemos explorado cómo mejorar la detección y clasificación de eventos anómalos en estas señales.

La aplicación de técnicas de filtrado digital, como el filtro exponencial, ha demostrado ser efectiva para eliminar el ruido aleatorio de las señales de aceleración. La selección cuidadosa de los parámetros del filtro es crucial para lograr un equilibrio óptimo entre la supresión del ruido y la preservación de la información relevante de la señal.

La implementación de un sistema de decisión basado en reglas lógicas ha proporcionado una manera transparente y efectiva de detectar eventos como el cabeceo en las señales de aceleración. La validación manual de los resultados ha demostrado la precisión del sistema y su capacidad para identificar eventos de interés.

Tanto los perceptrones multicapa como las redes convolucionales han mostrado prometedores resultados en la clasificación de patrones en las señales de aceleración. Si bien los perceptrones ofrecen una alta precisión en la detección de diferentes clases de eventos, las redes convolucionales muestran un potencial aún mayor cuando se dispone de conjuntos de datos más amplios y variados.

La principal limitación de este estudio radica en la disponibilidad de datos. La calidad y efectividad de los modelos de aprendizaje automático se ven directamente afectadas por la cantidad y variedad de muestras disponibles para el entrenamiento. Se recomienda la recopilación de un conjunto de datos más amplio y diverso para mejorar la capacidad de generalización de los modelos.

La automatización del proceso de entrenamiento de modelos de Machine Learning a través de scripts en Python ofrece una manera eficiente de gestionar grandes cantidades de datos y realizar experimentos con diferentes arquitecturas de modelos. Esta automatización simplifica el proceso de desarrollo y evaluación de modelos, permitiendo una rápida iteración y optimización.

En resumen, este estudio representa un paso significativo hacia la mejora de los sistemas de monitoreo y diagnóstico de maquinaria industrial basados en señales de aceleración. Si bien hemos logrado avances significativos en la detección y clasificación de eventos anómalos, queda aún mucho por explorar y mejorar en términos de la robustez y generalización de los modelos desarrollados. Con un enfoque continuo en la recopilación de datos de alta calidad y la experimentación con técnicas más avanzadas de aprendizaje automático, podemos seguir avanzando hacia sistemas de monitoreo más precisos y eficientes en entornos industriales exigentes.

8.1 Relación del trabajo desarrollado con los estudios cursados

El trabajo desarrollado en este TFM está estrechamente relacionado con varias asignaturas del máster en Automática e Informática Industrial. En primer lugar, la asignatura de Programación de Sistemas proporcionó los fundamentos sobre el funcionamiento del sistema operativo, así como habilidades prácticas en el desarrollo de aplicaciones en entornos Linux utilizando llamadas al sistema y programación en lenguaje C. Estos conocimientos son fundamentales para entender y trabajar con sistemas empotrados basados en Linux, como los utilizados en el proyecto.

Por otro lado, la asignatura de Interfaces Físicos y Sistemas Empotrados amplió el enfoque hacia el desarrollo de aplicaciones que interactúan estrechamente con el entorno físico, utilizando sensores y actuadores. Los conceptos aprendidos sobre arquitectura de sistemas empotrados, programación en entornos concurrentes y de tiempo real, así como el manejo de entradas y salidas digitales y analógicas, son directamente aplicables al diseño y desarrollo de sistemas de monitorización y control utilizados en el proyecto.

Además, la asignatura de Redes y Sistemas Distribuidos para Control proporcionó los conocimientos necesarios sobre protocolos de comunicación y sistemas distribuidos. Estos conocimientos también son esenciales en el contexto de la digitalización industrial y la Industria 4.0.

Por último, la asignatura de Implementación de Sistemas de Control abordó aspectos prácticos relacionados con la implementación de sistemas de control en la plataforma de laboratorio. Aunque el enfoque principal de esta asignatura es diferente al del proyecto, los conceptos básicos de control y la experiencia en la implementación de sistemas en tiempo real son relevantes para el diseño e implementación de algoritmos de control utilizados en el proyecto.

En resumen, el trabajo realizado en este TFM se apoya en los conocimientos adquiridos en asignaturas como Programación de Sistemas, Interfaces Físicos y Sistemas Empotrados, Redes y Sistemas Distribuidos para Control, e Implementación de Sistemas de Control. Estas asignaturas proporcionaron las bases teóricas y prácticas necesarias para abordar con éxito el desarrollo del proyecto, demostrando la aplicación práctica de los conocimientos adquiridos durante el máster en Automática e Informática Industrial.

8.2 Trabajos futuros

En trabajos futuros, se pueden abordar diversas áreas para mejorar y ampliar el estudio realizado. Una opción relevante sería la ampliación del conjunto de datos utilizado en el análisis. Al contar con más datos provenientes de diferentes máquinas y entornos industriales, se podría enriquecer el conjunto actual, lo que probablemente mejoraría la capacidad de generalización de los modelos de aprendizaje automático.

Además, sería interesante explorar otras técnicas avanzadas de aprendizaje automático que no se hayan considerado en este estudio. Por ejemplo, se podrían investigar arquitecturas de redes neuronales como las redes recurrentes o las redes neuronales convolucionales bidireccionales, las cuales podrían capturar mejor las relaciones temporales presentes en las señales de aceleración.

Otro aspecto a considerar sería mejorar la eficiencia del procesamiento de señales. Esto podría lograrse mediante la investigación y aplicación de métodos más avanzados de filtrado y detección de eventos anómalos, como técnicas de filtrado adaptativo o algoritmos de procesamiento de señales más sofisticados.

La integración de datos provenientes de sensores adicionales, como sensores de temperatura o vibración, también podría ser beneficioso. Combinar estos datos con las señales de aceleración en los modelos de aprendizaje automático podría proporcionar una visión más completa del estado de la maquinaria industrial y mejorar la capacidad de detección y diagnóstico de fallas.

Además, sería importante explorar la viabilidad de implementar los modelos desarrollados en sistemas de monitoreo en tiempo real. Esto requeriría optimizar los modelos para el desempeño en tiempo real y asegurar su integración con sistemas de monitoreo existentes en entornos industriales.

Es crucial evitar el sobreajuste de los modelos y optimizar adecuadamente los hiperparámetros para garantizar su generalización a nuevos datos. Sería recomendable realizar experimentos adicionales para encontrar la combinación óptima de parámetros que maximice el rendimiento de los modelos en diferentes escenarios.

Además, se podría investigar en detalle las causas detrás de los errores de clasificación, tanto falsos positivos como falsos negativos, para identificar posibles mejoras en las técnicas de preprocesamiento de datos y en el diseño de los modelos. Esto podría incluir la identificación de características específicas de las señales que conducen a errores de clasificación y la implementación de estrategias para mitigarlos.

Los métodos y técnicas desarrollados en este estudio podrían aplicarse a otras áreas industriales más allá de la monitorización de maquinaria. Por ejemplo, podrían adaptarse para el diagnóstico de fallas en sistemas eléctricos, la detección de intrusiones en sistemas de seguridad, o la optimización de procesos industriales.

Finalmente, fomentar la colaboración interdisciplinaria con expertos en ingeniería de control, mantenimiento predictivo y seguridad industrial podría enriquecer el enfoque presentado en este estudio. La combinación de conocimientos en diferentes áreas podría conducir a soluciones más robustas y efectivas para los desafíos en el monitoreo de maquinaria industrial.

Bibliografía

[1] Henning Kagermann. Recommendations for implementing the strategic initiative industrie 4.0: Final report of the industrie 4.0 working group. *Munchen: Acatech*, 2013.

[2] Tom Breur. Big data and the internet of things. *Journal of Marketing Analytics*, 3, 2015.

[3] Paulo Augusto Sherring da Rocha and Rogério Diogne de Silva e Souza and Maria Emília de Lima Tostes. Prototype CNC machine design. *2010 9th IEEE/IAS International Conference on Industry Applications - INDUSCON 2010*, 1-5, 2010.

[4] Gustavo Scalabrini Sampaio, Arnaldo Rabello de Aguiar Vallim Filho, Leilton Santos da Silva. When computers were women. *Sensors* , 19:(19):4342, 2019.

[5] Chunzhen Yang and Jingquan Liu and Yuyun Zeng and Guangyao Xie. Real-time condition monitoring and fault detection of components based on machine-learning reconstruction model. *Renewable Energy*, 133, 433-441, 2019.

[6] Changchun Liu and Haihua Zhu and Dunbing Tang and Qingwei Nie and Tong Zhou and Liping Wang and Yejia Song. Probing an intelligent predictive maintenance approach with deep learning and augmented reality for machine tools in IoT-enabled manufacturing. *Robotics and Computer-Integrated Manufacturing*, 77, 102357, 2022.

[7] Zhixiong Li and Rui Liu and Dazhong Wu. Data-driven smart manufacturing: Tool wear monitoring with audio signals and machine learning. *Journal of Manufacturing Processes*, 48, 66-76, 2019.

[8] James Williams and Ilya Mikhelson. Triple frame buffer FPGA implementation. *HardwareX*, 5:2468-0672, 2019.

[9] S. Albawi, T. A. Mohammed and S. Al-Zawi. Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*, 1-6, 2017.

[10] Gregory S. Parnell, Patrick J. Driscoll, Dale L. Henderson. *Decision Making in Systems Engineering and Management*. Wiley, 2011.

[11] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1994.

[12] Diseño de algoritmos de diagnóstico de fallos en rodamientos basados en métodos de machine learning aplicados al análisis de vibraciones. Ortiz Pérez, A. (2021). Consultado en http://hdl.handle.net/10251/174233.

[13] Classify MNIST Audio using Spectrograms/Keras CNN. Consultado en https://www.kaggle.com/code/christianlillelund/classify-mnist-audio-using-spectrograms-keras-cnn.

[14] Understanding the I2C Bus. Consultado en https://www.ti.com/lit/an/slva704/slva704.pdf.

[15] About Python. Consultado en https://www.python.org/about/.

[16] C++ Reference. Consultado en https://en.cppreference.com/w/.

[17] Arduino - Home. Consultado en https://www.arduino.cc/.

[18] What is an accelerometer. Consultado en https://www.analog.com/en/resources/glossary/accelerometer.html.

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenible	**Alto**	**Medio**	**Bajo**	**No procede**
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**				X
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**	X			
ODS 10. **Reducción de las desigualdades.**				X
ODS 11. **Ciudades y comunidades sostenibles.**				X
ODS 12. **Producción y consumo responsables.**	X			
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**				X
ODS 17. **Alianzas para lograr objetivos.**				X

La relación entre mi TFM y los Objetivos de Desarrollo Sostenible (ODS) es un aspecto fundamental para comprender el impacto y la relevancia de mi investigación en el contexto global de desarrollo sostenible. Mi TFM se enfoca en el desarrollo de un sistema de diagnóstico en tiempo real para una máquina de corte con láser, con el objetivo de optimizar el mantenimiento, mejorar la eficiencia operativa y reducir el impacto ambiental. A continuación, reflexionaré sobre cómo mi TFM se relaciona con los ODS, centrándome en aquellos que considero más relevantes: el ODS 9 (Industria, Innovación e Infraestructura) y el ODS 12 (Producción y Consumo Responsables).

El ODS 9 busca construir infraestructuras resilientes, promover la industrialización inclusiva y sostenible, y fomentar la innovación. Mi TFM contribuye a este objetivo al desarrollar un sistema de diagnóstico avanzado para una máquina industrial específica, en este caso, una máquina de corte con láser. Esta máquina es una parte integral de la infraestructura industrial en muchos sectores, incluidos la manufactura, la construcción y la automoción. Optimizar su funcionamiento y prolongar su vida útil mediante un mantenimiento predictivo no solo mejora la eficiencia operativa de la máquina, sino que también garantiza la continuidad y fiabilidad de las operaciones industriales en general.

Además, el aspecto de la innovación es central en mi TFM, ya que implica el desarrollo e implementación de tecnologías de vanguardia, como sistemas de monitoreo en tiempo real, algoritmos de análisis de datos y técnicas de aprendizaje automático. Estas innovaciones son cruciales para mejorar la competitividad de las empresas en un entorno industrial en constante evolución y para impulsar el progreso tecnológico hacia formas más eficientes y sostenibles de producción.

Por otro lado, el ODS 12 se centra en garantizar patrones de producción y consumo sostenibles. Mi TFM se alinea estrechamente con este objetivo al abordar la eficiencia y la responsabilidad en la producción industrial. La implementación de un sistema de diagnóstico en tiempo real permite una gestión más eficiente de los recursos al prevenir fallos y minimizar el desperdicio de materiales y energía asociado con el mantenimiento no planificado. Al anticipar y abordar problemas antes de que se conviertan en fallos críticos, se reducen los tiempos de inactividad no planificados y se optimiza el uso de recursos, lo que contribuye a una producción más sostenible.

Además, al proporcionar información detallada sobre el estado de la máquina y sus componentes, el sistema de diagnóstico en tiempo real facilita la toma de decisiones informadas en relación con el mantenimiento y la planificación de la producción. Esto permite una gestión más eficaz de los recursos y una mayor transparencia en las operaciones industriales, lo que a su vez fomenta prácticas de producción más responsables y sostenibles.

En resumen, mi TFM se relaciona estrechamente con los ODS 9 y 12 al contribuir a la construcción de infraestructuras resilientes, promover la industrialización sostenible, fomentar la innovación y garantizar patrones de producción y consumo responsables. Al desarrollar e implementar un sistema de diagnóstico en tiempo real para una máquina de corte con láser, mi investigación busca mejorar la eficiencia operativa, prolongar la vida útil de los equipos industriales y reducir el impacto ambiental asociado con la producción industrial. En última instancia, mi objetivo es contribuir al avance hacia un modelo de desarrollo más sostenible y equitativo, en línea con la Agenda 2030 para el Desarrollo Sostenible de las Naciones Unidas.