TRABAJOS FIN DE MÁSTER

Universidad de Jaén

Escuela Politécnica
Superior de Jaén

Modelos de Inteligencia Artificial Sostenible para Mantenimiento Predictivo

Autor: Javier González Santos

Máster en Ingeniería Informática

Director: Prof. D. Ángel Miguel García
Departamento del director: Departamento de Informática

Fecha: 30/09/2025

AGRADECIMIENTOS

Aquí se añaden los agradecimientos del proyecto.

FICHA DEL TRABAJO FIN DE TÍTULO

Titulación	Máster en Ingeniería Informática
Modalidad	Proyecto de Ingeniería
Especialidad (solo TFG)	XXXXXX
Mención (solo TFG)	XXXXXXX
Idioma	Español
Tipo	Específico
TFM en equipo	No
Autor/a	Javier González Santos
Fecha de asignación	07 de abril de 2025
Descripción corta	Este trabajo presenta ...

Índice

Índice de Figuras

Índice de Tablas

Siglas

ANN	*Artificial Neural Network*, Red Neuronal Artificial.
API	*Application Programming Interface*, Interfaz de Programación de Aplicaciones.
ASHA	*Asynchronous Successive Halving Algorithm*, Algoritmo Asíncrono de Reducción Sucesiva.
CNN	*Convolutional Neural Network*, Red Neuronal Convolucional.
CNTS	*Cooperative Network Time Series*, Series Temporales de Red Cooperativa.
CPC	*Contrastive Predictive Coding*, Codificación Predictiva Contrastiva.
CPU	*Central Processing Unit*, Unidad Central de Procesamiento.
CUDA	*Compute Unified Device Architecture*, Arquitectura Unificada de Dispositivos de Cómputo.
CUSUM	*Cumulative Sum*, Suma Acumulada.
ECTS	*European Credit Transfer and Accumulation System*, Sistema Europeo de Transferencia y Acumulación de Créditos.
GPU	*Graphics Processing Unit*, Unidad de Procesamiento de Gráficos.
IOPS	*Input/Output Operations Per Second*, Operaciones de Entrada/Salida por Segundo.
IoT	*Internet of Things*, Internet de las Cosas.
JSON	*JavaScript Object Notation*, Notación de Objetos de JavaScript.
LIF	*Leaky Integrate-and-Fire*, Integración con Fuga y Disparo.

LSTM *Long Short-Term Memory*, Memoria a Corto y Largo Plazo.

MAC *Multiply-Accumulate*, Multiplicación y Acumulación.

ML *Machine Learning*, Aprendizaje Máquina.

MSE *Mean Squared Error*, Error Cuadrático Medio.

NAB *Numenta Anomaly Benchmark*, Benchmark de Anomalías Numenta.

NoC *Network on Chip*, Red en Chip.

RAM *Random Access Memory*, Memoria de Acceso Aleatorio.

SNN *Spiking Neural Network*, Red Neuronal de Impulsos.

SPC *Statistical Process Control*, Control Estadístico de Procesos.

STDP *Spike-Timing-Dependent Plasticity*, Plasticidad Dependiente del Tiempo de Impulsos.

STL *Seasonal-Trend decomposition using Loess*, Descomposición Estacional-Tendencia usando Loess.

TFM *Trabajo Fin de Máster*, Master's Thesis.

TPE *Tree-structured Parzen Estimator*, Estimador de Parzen con Estructura de Árbol.

TSFEDL *Time Series Feature Extraction Deep Learning*, Extracción de Características de Series Temporales mediante Aprendizaje Profundo.

UCR *University of California, Riverside*, Universidad de California, Riverside.

VRAM *Video Random Access Memory*, Memoria de Acceso Aleatorio de Vídeo.

WSL *Windows Subsystem for Linux*, Subsistema de Windows para Linux.

Resumen

Detectar anomalías en series temporales es un problema complejo que afecta a muchos sectores industriales: desde redes informáticas hasta infraestructuras críticas que requieren mantenimiento predictivo. Los métodos tradicionales funcionan, pero tienen limitaciones importantes de eficiencia energética y escalabilidad, sobre todo cuando hablamos de edge computing o dispositivos con recursos limitados.

Este Trabajo Fin de Máster aborda el desarrollo y optimización de una arquitectura híbrida SNN-CNN (Spiking Neural Network - Convolutional Neural Network) para detectar anomalías en series temporales. El foco está en la sostenibilidad computacional y la eficiencia energética. Las Redes Neuronales de Impulsos (SNNs) son interesantes porque imitan mejor el funcionamiento del cerebro biológico, y en teoría pueden ser más eficientes energéticamente gracias a la naturaleza dispersa (sparse) de los impulsos discretos.

La solución propuesta parte de una arquitectura SNN base (A–B) y añade una capa convolucional optimizable (A–B–C) que trabaja con diferentes kernels parametrizables (gaussiano, laplaciano, mexican hat, box) y varios modos de procesamiento (direct, weighted_sum, max). La idea es combinar lo mejor de dos mundos: la extracción de características espaciales de las redes convolucionales con la eficiencia temporal asíncrona de las SNNs.

Para la optimización de hiperparámetros usamos optimización bayesiana con Optuna TPE (Tree-structured Parzen Estimator), ejecutando 100 pruebas por configuración en tres tamaños de red diferentes (100, 200, 400 neuronas). Implementamos un pipeline de preprocesamiento reproducible que incluye: cuantización dinámica por cuantiles expandidos, expansión de etiquetas para compensar el desbalanceo temporal, y segmentación en ventanas de longitud T=250.

La metodología experimental empleó optimización bayesiana con Optuna TPE para la búsqueda sistemática de hiperparámetros, ejecutando 100 trials por configuración en tres escalas de red (100, 200, 400 neuronas). Se implementó un pipeline

de preprocesamiento reproducible que incluye cuantización dinámica por cuantiles expandidos, expansión de etiquetas para mitigar el desbalanceo temporal, y segmentación en ventanas de longitud T=250.

El análisis de importancia de hiperparámetros confirma que los parámetros neuronales (threshold, decay) y de plasticidad (nu1, nu2) son los más influyentes. Los parámetros convolucionales (kernel_size, sigma) vienen después, lo que valida que la dinámica neuronal LIF juega un papel clave en el rendimiento del modelo.

Las contribuciones principales del trabajo son: (1) una arquitectura híbrida SNN-CNN que se puede optimizar automáticamente, (2) un pipeline de preprocesamiento reproducible adaptado a las características de las SNNs, (3) evidencia empírica de que la hibridación mejora notablemente el rendimiento en datasets desbalanceados, y (4) una metodología de evaluación comparativa que puede servir de base para futuros trabajos en el área.

Este trabajo muestra el potencial de las arquitecturas híbridas SNN-CNN para aplicaciones de detección de anomalías. Ofrecen un buen equilibrio entre rendimiento y eficiencia computacional, lo que las posiciona como una alternativa viable para aplicaciones de tiempo real en entornos con restricciones energéticas.

Abstract

Anomaly detection in time series is a complex problem that affects many industrial sectors, from computer network monitoring to predictive maintenance of critical infrastructures. Traditional methods work well, but they have significant limitations when it comes to energy efficiency and scalability, especially in edge computing applications and resource-constrained devices.

This Master's Thesis presents the development and optimization of a hybrid SNN-CNN (Spiking Neural Network - Convolutional Neural Network) architecture for time series anomaly detection, with special emphasis on computational sustainability and energy efficiency. Spiking Neural Networks (SNNs) emerge as a promising paradigm that more faithfully emulate biological brain functioning, offering theoretical advantages in energy efficiency through the inherent sparsity of discrete spikes.

The proposed solution starts with a base SNN architecture (A–B) and adds an optimizable convolutional layer (A–B–C) that works with different parametrizable kernels (gaussian, laplacian, mexican hat, box) and various processing modes (direct, weighted_sum, max). The idea is to combine the best of both worlds: spatial feature extraction from convolutional networks with the asynchronous temporal efficiency of SNNs.

The experimental methodology employed Bayesian optimization with Optuna TPE for systematic hyperparameter search, executing 100 trials per configuration across three network scales (100, 200, 400 neurons). A reproducible preprocessing pipeline was implemented including dynamic quantization by expanded quantiles, label expansion to mitigate temporal imbalance, and segmentation into windows of length T=250.

We tested the architecture with two quite different public datasets: IOPS (highly imbalanced, only 1.92 % anomalies) and CalIt2 (more balanced, with 24.80 % anomalies). Results show that the hybrid architecture improves considerably over the base SNN model: 4.6x in IOPS (F1=0.277 vs 0.060) and 1.7x in CalIt2 (F1=0.417 vs

0.239).

When we compare with state-of-the-art models (TSFEDL), we see that traditional deep learning methods still have better absolute performance (F1=0.436 vs 0.277 in IOPS, F1=0.704 vs 0.417 in CalIt2). However, our hybrid SNN-CNN architecture considerably reduces that gap while maintaining the theoretical energy efficiency advantages of SNNs. An interesting finding: experiments show that automatic optimization consistently favors Mexican hat and Gaussian kernels with direct processing, suggesting that spatial smoothing works better than edge detection in these cases.

Hyperparameter importance analysis confirms that neural parameters (threshold, decay) and plasticity parameters (nu1, nu2) are the most influential. Convolutional parameters (kernel_size, sigma) come next, which validates that LIF neural dynamics play a key role in model performance.

The main contributions of this work include: (1) an automatically optimizable hybrid SNN-CNN architecture, (2) a reproducible preprocessing pipeline adapted to SNN characteristics, (3) empirical evidence that hybridization notably improves performance on imbalanced datasets, and (4) a comparative evaluation methodology that can serve as a foundation for future work in the area.

This work shows the potential of hybrid SNN-CNN architectures for anomaly detection applications. They offer a good balance between performance and computational efficiency, which positions them as a viable alternative for real-time applications in energy-constrained environments.

Capítulo 1

Introducción

1.1. Motivación

En el mundo actual, donde los datos crecen de forma exponencial, detectar comportamientos anómalos en series temporales se ha vuelto algo crítico para muchos sectores. Ya sea en la monitorización de redes informáticas o en el control de sistemas industriales, identificar patrones irregulares a tiempo puede prevenir fallos, optimizar recursos y garantizar la seguridad de infraestructuras críticas.

Las anomalías en series temporales son desviaciones significativas de lo que consideraríamos comportamiento normal o esperado en un sistema. Pueden aparecer como picos inusuales, cambios bruscos en la tendencia, o alteraciones en cómo se repiten los datos periódicamente. Detectar estas irregularidades de forma efectiva es complicado, sobre todo cuando trabajamos con sistemas complejos que generan grandes cantidades de datos en tiempo real.

Durante las últimas décadas se han desarrollado varios métodos para abordar este problema, desde técnicas estadísticas tradicionales hasta algoritmos complejos de aprendizaje automático. Más recientemente, los modelos de aprendizaje profundo como autoencoders y redes neuronales recurrentes han demostrado un buen rendimiento en este campo, gracias a que pueden capturar relaciones no lineales y patrones temporales en los datos [**?**, **?**]. Pero estos modelos convencionales tienen sus limitaciones, especialmente en cuanto a eficiencia computacional y consumo energético—algo crítico en dispositivos con recursos limitados o cuando necesitas procesamiento en tiempo real [**?**, **?**].

Las SNNs aparecen como un enfoque interesante que intenta imitar mejor

cómo funciona el cerebro biológico. Ofrecen ventajas en eficiencia energética y procesamiento temporal inherente comparado con los modelos convencionales de deep learning [**?**, **?**, **?**]. Este trabajo se enmarca en la línea de investigación que abrieron estudios pioneros que demostraron que las SNNs pueden funcionar para detección de anomalías mediante mecanismos de inhibición sináptica y codificación temporal [**?**, **?**, **?**]. Proponemos mejoras arquitecturales y de optimización que superan limitaciones identificadas en implementaciones previas.

1.2. Objetivos

El objetivo general del proyecto es el desarrollo y optimización de sistemas de detección de anomalías basados en SNNs

1. Revisión bibliográfica de modelos de SNNs y su aplicación en detección de anomalías y mantenimiento predictivo.
2. Diseño y desarrollo de arquitecturas de SNNs optimizadas para la detección de anomalías en datos de sensores industriales.
3. Implementación de modelos de SNNs sostenibles que minimicen el consumo energético sin sacrificar la precisión.
4. Validación de los modelos desarrollados mediante experimentación en conjuntos de datos reales de mantenimiento predictivo.
5. Comparación de la eficiencia energética y el rendimiento predictivo frente a modelos tradicionales de detección de anomalías.

1.3. Estructura de la memoria

A continuación se describe la estructura de esta memoria. Consta de 5 capítulos:

- **Capítulo 1**: Este primer capítulo presenta la motivación del proyecto, explicando por qué es tan importante detectar anomalías en series temporales para distintos sectores industriales. Se establecen los objetivos específicos: diseñar arquitecturas SNNs optimizadas e implementar modelos sostenibles que reduzcan el consumo energético. También se incluye la planificación temporal del proyecto con estimaciones de costes y duración de las tareas principales.

- **Capítulo 2**: Aquí se hace un análisis de la evolución histórica de los métodos de detección de anomalías, desde técnicas estadísticas clásicas hasta los enfoques más recientes de aprendizaje profundo. Se examinan los fundamentos estadísticos y enfoques tradicionales, las aplicaciones del aprendizaje profundo en detección de anomalías, y métodos de optimización de hiperparámetros. Se dedica especial atención a las redes neuronales de impulsos y la computación neuromórfica, el desarrollo de benchmarks especializados, y la comparación de eficiencia energética entre diferentes paradigmas computacionales.

- **Capítulo 3**: En este capítulo se define el alcance específico del proyecto, las hipótesis de partida y las restricciones identificadas. Se describe en detalle la solución propuesta, incluyendo la justificación para usar BindsNET como framework de desarrollo y los desafíos técnicos encontrados durante la configuración del entorno. El capítulo profundiza en la arquitectura del nuevo modelo híbrido SNN-CNN, explicando qué innovaciones introducimos respecto a la implementación original, y describe los algoritmos de preprocesamiento desarrollados para los datasets IOPS y CalIt2.

- **Capítulo 4**: Este capítulo presenta la evaluación experimental del modelo propuesto. Se describen los escenarios experimentales implementados, incluyendo la configuración de hardware y software, los datasets utilizados y las estrategias de particionado temporal. Se detalla la metodología de evaluación, con especial énfasis en las métricas de calidad (precisión, recall, F1-score) y eficiencia computacional. El análisis de resultados incluye comparaciones entre la SNN original, el modelo híbrido propuesto y baselines de la librería TSFEDL, la importancia de los hiperparámetros, y evaluaciones de escalabilidad.

- **Capítulo 5**: El último capítulo sintetiza los principales hallazgos del trabajo, evaluando si se cumplieron los objetivos planteados y cuál fue el impacto de las contribuciones realizadas. Se discuten las limitaciones identificadas durante el desarrollo y se proponen direcciones para trabajos futuros que puedan extender y mejorar los métodos desarrollados.

1.4. Planificación temporal

La asignación temporal de las tareas que componen este proyecto se puede ver en la Figura **??**. Esta planificación es orientativa y puede sufrir retrasos o modificaciones

según avanza el proyecto. Para estimar los costes, se ha considerado una dedicación de 25 horas semanales, con un coste de 22.80€ por hora [**?**].

1.4.1. Estimación temporal

Tarea	Duración (semanas)	Coste
Revisión bibliográfica	3	1.710,00€
Implementación de modelo híbrido SNN-CNN	4	2.280,00€
Configuración de Framework BindsNET	0,5	285,00€
Optimización de hiperparámetros con Optuna	1,5	855,00€
Selección y preprocesamiento de datos	2	1.140,00€
Experimentación y validación	3	1.710,00€
Análisis comparativo de resultados	2	1.140,00€
Documentación	2	1.140,00€
TOTAL	**18**	**10.260,00€**

Tabla 1.1: Planificación temporal de tareas, con duración estimada en semanas y coste asociado.

Inicialmente nos centraremos en hacer una **revisión bibliográfica** exhaustiva sobre las redes neuronales de impulsos (SNNs), especialmente su aplicación en detección de anomalías y mantenimiento predictivo. Este análisis nos ayudará a identificar y seleccionar los enfoques y modelos más recientes y relevantes del campo, así como determinar los mejores métodos de optimización multiobjetivo y evaluación comparativa de eficiencia energética entre SNNs y modelos tradicionales.

A continuación, empezaremos con la **implementación del modelo híbrido SNN-CNN**, añadiendo mejoras como la capa convolucional adaptable y la **optimización de hiperparámetros mediante frameworks como Optuna**. Prestaremos especial atención a la integración y **configuración del entorno de trabajo con BindsNET**.

En paralelo vamos a explorar y **seleccionar las fuentes de datos**, poniendo especial énfasis en la disponibilidad y calidad de conjuntos como IOPS y CalIt2, que necesitan pipelines de preprocesamiento específicos para codificar bien los datos temporales y tratar valores faltantes. Esta etapa es crucial para asegurar que los datos sean adecuados para entrenar y validar bien los modelos de detección de anomalías.

Una vez validada la arquitectura y el procesamiento de datos, lanzaremos la **fase de experimentación**, implementando escenarios controlados y métricas de evaluación que nos permitan comparar objetivamente el rendimiento y la eficiencia

energética frente a modelos tradicionales. **Los resultados se analizarán** en profundidad, documentando las mejoras arquitecturales y trade-offs identificados, y generando visualizaciones comparativas.

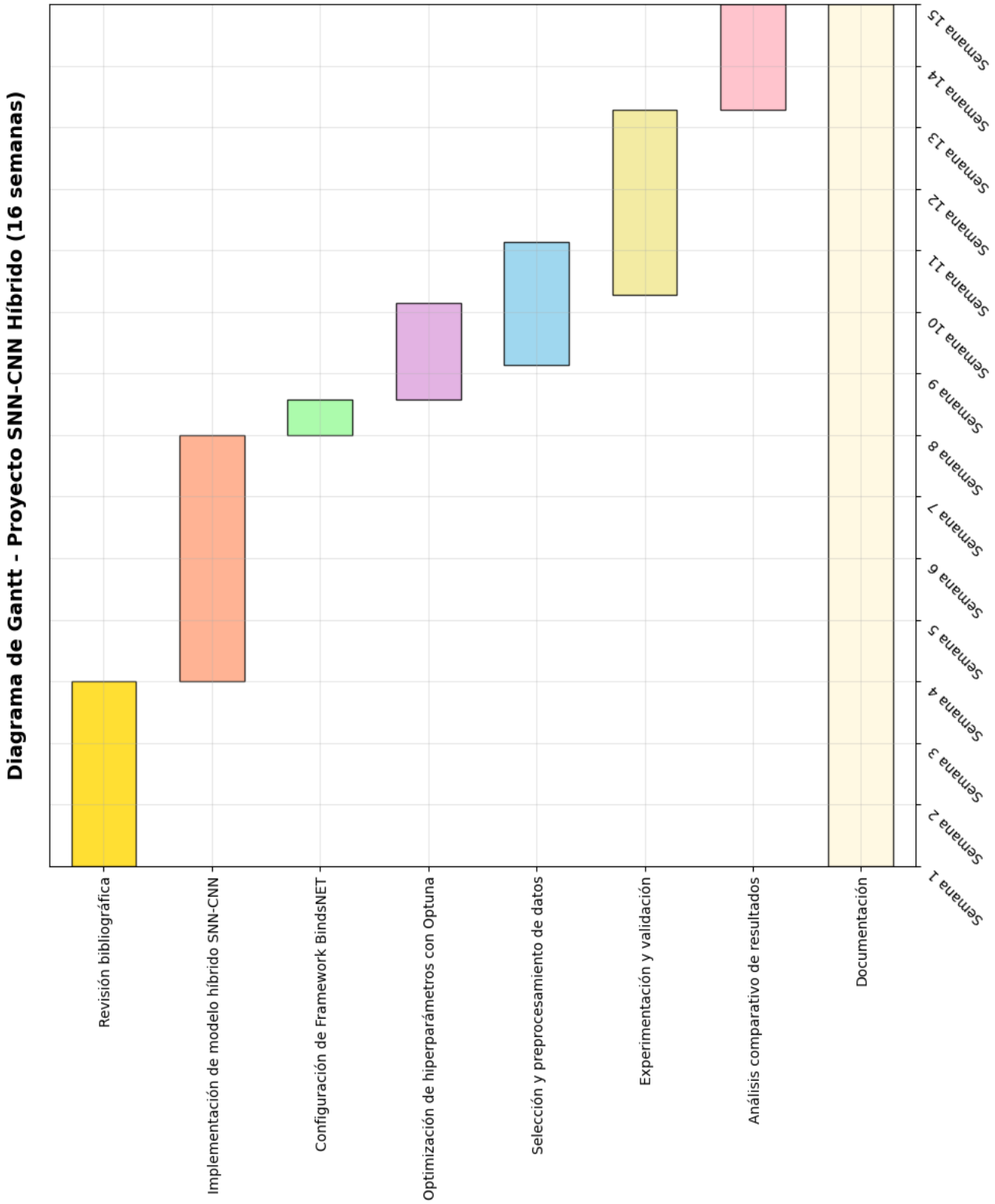

Figura 1.1: Diagrama de Gantt del proyecto. Fuente: Elaboración propia.

Capítulo 2

Fundamentos Teóricos y Estado del Arte

2.1. Evolución temporal de métodos de detección de anomalías en series temporales

En este capítulo se presenta un análisis integral de la progresión histórica de las metodologías de detección de anomalías en series temporales, abarcando casi un siglo de avance tecnológico desde el control estadístico de procesos fundamental hasta las redes neuronales de impulsos. La evolución revela paradigmas tecnológicos distintos que han dado forma al campo, cada uno abordando limitaciones específicas mientras introduce nuevas capacidades en términos de eficiencia computacional, precisión de detección y aplicabilidad en el mundo real. A través del examen sistemático de innovaciones, este estudio identifica hitos tecnológicos clave y proporciona una verificación crítica de los marcos cronológicos existentes. El análisis demuestra cómo las compensaciones entre eficiencia energética y precisión han impulsado transiciones metodológicas, al tiempo que destaca enfoques híbridos emergentes que combinan los beneficios de múltiples paradigmas para mejorar el rendimiento en la detección de anomalías.

2.2. Fundamentos estadísticos y enfoques clásicos

2.2.1. Métodos Iniciales de Control Estadístico de Procesos

El origen de la detección sistemática de anomalías en series temporales se remonta al trabajo pionero de Walter Shewhart en 1924, quien introdujo los gráficos de SPC [**?**]. Estos gráficos proporcionaron el primer marco formal para distinguir entre variación común y variación especial en procesos industriales, cambiando fundamentalmente cómo las organizaciones abordan la supervisión de la calidad y la identificación de anomalías. El enfoque de Shewhart enfatizó la importancia de comprender la variación natural del proceso antes de intentar identificar verdaderas anomalías, un principio que sigue siendo central en los sistemas modernos de detección de anomalías.

La evolución de los métodos de control estadístico continuó con la introducción del gráfico CUSUM por Page en 1954, un avance significativo en la sensibilidad para detectar pequeños cambios en el comportamiento de los procesos [**?**]. Los gráficos CUSUM acumulan información de observaciones anteriores, demostrando ser superiores a los gráficos Shewhart tradicionales para detectar desviaciones graduales [**?**, **?**]. Este enfoque acumulativo resultó particularmente valioso para escenarios donde las anomalías se manifiestan como desviaciones sutiles y persistentes en lugar de picos dramáticos, haciendo de CUSUM una herramienta esencial para los sistemas ciberfísicos modernos y aplicaciones de seguridad.

2.2.2. Técnicas avanzadas de descomposición estadística

En 1967, MacQueen propone un método llamado agrupamiento K-means. Este enfoque basado en agrupamiento para la detección de anomalías representa un hito importante en los métodos no supervisados, donde las anomalías se identifican como puntos que no pertenecen a ningún grupo bien definido o forman grupos muy pequeños y aislados [**?**].

La década de 1990 marcó un avance significativo con el desarrollo del procedimiento STL, que permitió separar series temporales en componentes de tendencia, estacionalidad y residuos, facilitando la detección de anomalías con mayor precisión al modelar patrones esperados e identificar desviaciones de estos patrones con mayor precisión [**?**]. El método demostró ser particularmente efectivo para aplicaciones de vigilancia en salud, donde modeló con éxito recuentos diarios de

quejas principales descomponiéndolos en componentes interanuales, estacionales anuales, día de la semana y errores aleatorios [**?**].

2.3. Aprendizaje profundo en la detección de anomalías

2.3.1. Fundamentos de redes neuronales y métodos basados en reconstrucción

La revolución del aprendizaje profundo transformó la capacidad de detectar anomalías mediante arquitecturas neuronales capaces de aprender patrones temporales complejos. Los autoencoders profundos surgieron como paradigma dominante alrededor de 2006, utilizando el error de reconstrucción como principio fundamental para identificar anomalías . Estas redes aprenden a comprimir patrones normales en representaciones de menor dimensión y luego los reconstruyen, con el supuesto de que los patrones anómalos exhibirán mayores errores de reconstrucción debido a su desviación de los comportamientos normales aprendidos [**?**].

Las LSTM fueron propuestas en 1997, en aquel momento existían limitaciones críticas en el modelado temporal al proporcionar mecanismos para capturar tanto dependencias a corto como a largo plazo en datos de series temporales [**?**]. Las implementaciones modernas a partir del 2010 a menudo integran redes LSTM con mecanismos de atención multi-cabeza y capas de red completamente conectadas para mejorar la precisión de predicción y las capacidades de detección de anomalías. Estas arquitecturas híbridas han demostrado ser particularmente efectivas en el análisis de transacciones blockchain, donde modelan con éxito la naturaleza dinámica, no lineal y variable en el tiempo de los patrones de transacción [**?**].

2.3.2. Arquitecturas avanzadas de aprendizaje profundo

El desarrollo de los VAEs alrededor de 2014 introdujo enfoques probabilísticos en la detección de anomalías, permitiendo la cuantificación de la incertidumbre y un manejo más robusto de distribuciones de datos complejas [**?**]. Trabajos recientes han demostrado la efectividad de las perspectivas variacionales espacio-temporales jerárquicas en la detección de anomalías en series temporales multivariadas,

modelando explícitamente variables estocásticas temporales y variables de relación de gráficos latentes dentro de marcos gráficos unificados [**?**, **?**].

En 2018, ganan prominencia los enfoques de aprendizaje autosupervisado, particularmente a través de la aplicación de técnicas de CPC. En este estudio se busca transformar la identificación de anomalías en problemas de identificación de pares positivos [**?**]. El esquema CPC de salto de paso representa un avance notable, ajustando la distancia entre ventanas de historial y puntos de detección para mejorar la construcción de pares positivos y el rendimiento de detección [**?**].

2.4. Métodos de optimización y ajuste de hiperparámetros

2.4.1. Enfoques clásicos y modernos de optimización

El panorama de optimización para sistemas de detección de anomalías ha evolucionado desde métodos simples de búsqueda en cuadrícula en los años 70 [**?**] hasta sofisticados marcos de optimización bayesiana [**?**]. La búsqueda en cuadrícula, aunque computacionalmente intensiva, proporcionó una exploración sistemática de espacios de hiperparámetros, pero sufrió problemas de escalado exponencial a medida que aumentaba la dimensionalidad. La introducción de la optimización bayesiana alrededor de 2012 revolucionó el ajuste de hiperparámetros al modelar el rendimiento de los algoritmos de aprendizaje como muestras de procesos gaussianos, permitiendo una exploración más eficiente de los espacios de parámetros [**?**].

Los marcos de optimización bayesiana, implementada en frameworks como Optuna, han demostrado la capacidad de alcanzar o superar el nivel de optimización de expertos humanos para varios algoritmos, incluidos la asignación latente de Dirichlet, SVMs estructurados y redes neuronales convolucionales [**?**]. Estos métodos tienen en cuenta los costos experimentales variables y aprovechan las capacidades de procesamiento paralelo, lo que los hace particularmente adecuados para modelos de aprendizaje profundo computacionalmente costosos utilizados en la detección de anomalías.

2.4.2. Optimización multiobjetivo en la detección de anomalías

Los enfoques contemporáneos abordan cada vez más la detección de anomalías como problemas de optimización multiobjetivo, como lo demuestra el enfoque CNTS [**?**]. Este marco consta de componentes de detector y reconstructor que trabajan cooperativamente, donde el detector identifica directamente anomalías mientras que el reconstructor proporciona información de reconstrucción y actualiza el aprendizaje basado en retroalimentación anómala. Esta estrategia de solución cooperativa aborda las limitaciones de los métodos basados en reconstrucción tradicionales que a menudo son susceptibles a valores atípicos e ineficaces para modelar anomalías.

2.5. Redes neuronales de impulsos y computación neuromórfica

2.5.1. Inspiración biológica y eficiencia energética

Las SNNs representan un cambio de paradigma hacia modelos de computación biológicamente plausibles que enfatizan el procesamiento basado en eventos, tiempos de respuesta bajos y una eficiencia energética excepcional [**?**]. El desarrollo de reglas de aprendizaje STDP, con raíces históricas que se extienden a conceptos filosóficos de Aristóteles y Locke, culminó en implementaciones prácticas que combinan simplicidad elegante con plausibilidad biológica y poder computacional [**?**].

Las implementaciones de hardware neuromórfico alrededor de 2011 permitieron el despliegue práctico de sistemas de detección de anomalías basados en SNNs con una eficiencia energética sin precedentes [**?**].

2.5.2. Aplicaciones modernas y arquitecturas híbridas

Los desarrollos recientes en aplicaciones de SNN para la detección de anomalías se han centrado en la detección robusta de anomalías de audio, donde los modelos multivariados de series temporales robustos a valores atípicos detectan sonidos anómalos previamente no vistos basados en datos de entrenamiento ruidosos [**?**]. Estos enfoques utilizan arquitecturas novedosas de redes neuronales profundas que aprenden dinámicas temporales a múltiples resoluciones mientras mantienen robustez frente a contaminaciones en el conjunto de datos de entrenamiento. Las dinámicas

temporales se modelan utilizando capas recurrentes aumentadas con mecanismos de atención construidos sobre capas convolucionales, lo que permite la extracción de características a múltiples resoluciones.

La aparición de arquitecturas híbridas SNN-CNN alrededor de 2024 representa el avance más reciente, combinando la eficiencia energética de las redes de impulsos con las capacidades de reconocimiento de patrones de las redes neuronales convolucionales [**?**]. Estos sistemas ofrecen una precisión mejorada mientras mantienen las ventajas de bajo consumo de energía de la computación neuromórfica, lo que los hace particularmente adecuados para aplicaciones donde la vida útil de la batería y los recursos computacionales están limitados [**?**].

2.6. Desarrollo de Benchmarks

El establecimiento de marcos de evaluación estandarizados ha sido crucial para avanzar en la investigación de la detección de anomalías. El NAB, introducido en 2015, proporcionó el primer entorno controlado y repetible para probar algoritmos de detección de anomalías en tiempo real sobre datos de transmisión [**?**]. NAB abordó la necesidad crítica de puntos de referencia que pudieran evaluar detectores que procesan datos en tiempo real en lugar de lotes, con algoritmos de puntuación diseñados específicamente para las características de los datos de transmisión.

Sin embargo, investigaciones posteriores han revelado limitaciones significativas en los conjuntos de datos de evaluación más populares. Un análisis crítico publicado en 2020 demostró que la mayoría de los ejemplares individuales en conjuntos de datos ampliamente utilizados sufren de cuatro fallas fundamentales, lo que sugiere que muchas comparaciones de algoritmos publicadas pueden ser poco confiables y que el progreso aparente en los últimos años puede ser ilusorio. Este análisis llevó a la introducción del UCR Time Series Anomaly Archive en el año 2020, que tiene como objetivo proporcionar comparaciones significativas entre enfoques y una medición precisa del progreso [**?**].

En la Figura **??** se visualiza dicha evolución de detección de anomalías. Donde el eje y representa las categorías y en el eje x los años donde se comenzó a poner en práctica dicho método en la detección de anomalía en series temporales.

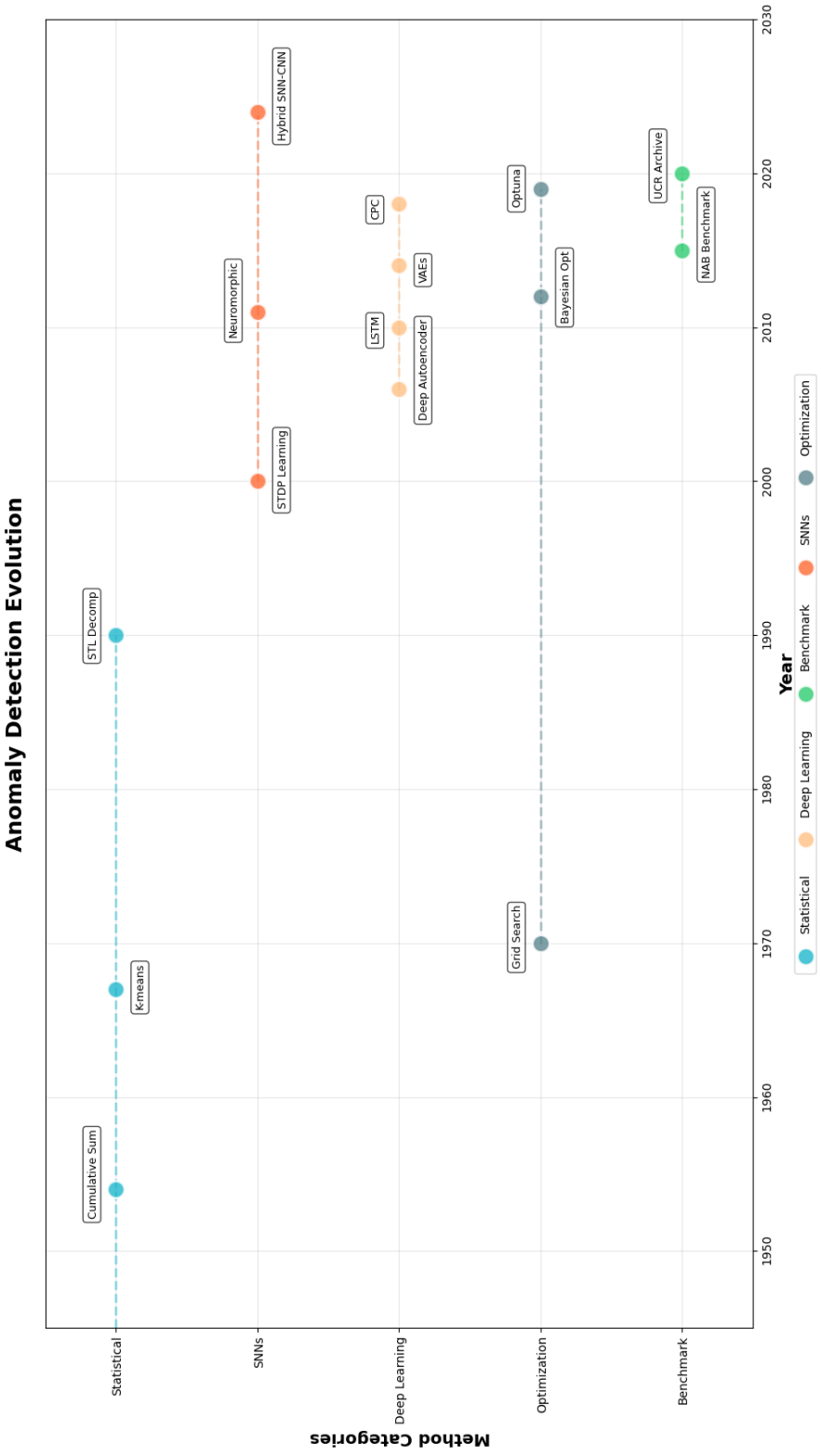

Figura 2.1: Evolución de la detección de anomalía. Fuente: Elaboración propia.

2.7. Eficiencia energética

2.7.1. Limitaciones energéticas de los métodos tradicionales

Los métodos convencionales de detección de anomalías enfrentan desafíos significativos en términos de consumo energético, especialmente cuando se implementan en dispositivos con recursos limitados [**?**]. Las redes neuronales artificiales tradicionales requieren operaciones de multiplicación intensivas que consumen considerable energía, mientras que los métodos estadísticos, aunque menos demandantes computacionalmente, carecen de la capacidad para capturar patrones complejos no lineales [**?**, **?**].

Comparaciones empíricas de ANN como autoencoders vs Técnicas estadísticas como Z-score muestran los siguientes resultados:

Método	**Rendimiento**		
	Precision	**Recall**	**F1-score**
Z-Score (método estadístico)	1.00	0.60	0.75
Autoencoders (ANN)	0.94	1.00	0.97

Tabla 2.1: Comparación de rendimiento entre métodos estadísticos y redes neuronales artificiales. Fuente: [**?**]

Para el método basado en Z-score, se reporta una precisión de 1.00, lo que indica que todas las anomalías identificadas fueron correctas, mientras que el recall es de 0.60, lo que sugiere que se detectó el 60 % de las anomalías reales. El F1-score, que equilibra precisión y recall, es de 0.75, reflejando la efectividad general de este método.

En contraste, el método con autoencoders logra una precisión de 0.94, lo que indica una alta proporción de anomalías correctamente identificadas con un mínimo de falsos positivos. El recall es de 1.00, indicando que se detectaron todas las anomalías reales. En consecuencia, el F1-score del método con autoencoders es de 0.97, lo que evidencia un rendimiento superior en comparación con el enfoque basado en Z-score. La mayor desventaja de este enfoque es su mayor consumo energético. Esta compensación entre precisión y eficiencia energética representa uno de los principales motivadores para la exploración de paradigmas computacionales alternativos [**?**].

2.7.2. Paradigma de eficiencia energética en SNNs

Las redes neuronales de impulsos emergen como una solución prometedora debido a su naturaleza impulsada por eventos y su capacidad para realizar computación usando únicamente operaciones de acumulación en lugar de multiplicaciones costosas [**?**]. Esta característica fundamental permite que las SNNs consumen significativamente menos energía que sus contrapartes tradicionales.

Los sistemas de detección de anomalías basados en SNNs para redes IoT han logrado eficiencias energéticas aproximadamente 3.5 veces superiores a los modelos CNN tradicionales, con latencias de detección significativamente menores (hasta 3 veces más rápidas) [**?**]. Estos resultados demuestran que, aunque puede existir una ligera reducción en la precisión de detección de anomalías (89.3 % para SNN vs 92.5 % para CNN), los beneficios en eficiencia energética y latencia proporcionan un caso convincente para el uso de SNNs en aplicaciones específicas [**?**].

Un estudio reciente presenta una metodología de transformar exitosamente arquitecturas ANNs en SNNs con una penalización de error promedio marginal de tan solo 2,65 %. El algoritmo de particionamiento de grafos propuesto permite una disminución del 14,22 % en la comunicación intersináptica y una reducción del 87,58 % en la comunicación intrasináptica, en promedio, lo que subraya la efectividad del algoritmo propuesto en la optimización de las vías de comunicación de las redes neuronales. En comparación con un algoritmo de particionamiento de grafos de referencia, el enfoque propuesto muestra una disminución promedio del 79,74 % en la latencia y una reducción del 14,67 % en el consumo energético. Usando herramientas existentes de NoC, el producto energía-latencia de las arquitecturas SNN es, en promedio, un 82,71 % inferior al de las arquitecturas de referencia [**?**].

2.7.3. Hiperparametros en SNN

La optimización de hiperparámetros SNNs representa un avance significativo en el desarrollo de sistemas de detección de anomalías eficientes energéticamente. Tradicionalmente, Búsqueda en Cuadrícula y Random Search se han usado como metodología de optimización de parámetros, donde el búsqueda en cuadrícula prueba cada combinación de hipermarámetros posibles y random search crea aleatoreamente grupos de parámetros dentro de los límites. Ambos métodos, aunque ampliamente utilizados, tienen sus limitaciones: a menudo son lentos y no siempre encuentran la combinación óptima de manera eficiente [**?**].

Actualmente existen frámeworks especializados optimizar la hiperparametrización como es el caso de Optuna. Este utiliza algoritmos de optimización bayesiana basados en TPE que permiten una exploración inteligente del espacio de hiperparámetros, enfocándose en regiones prometedoras en lugar de realizar búsquedas exhaustivas [**?**]. Esta aproximación resulta particularmente beneficiosa para SNNs, donde la sensibilidad a los hiperparámetros es especialmente crítica debido a la compleja dinámica temporal entre neuronas y impulsos, la optimización bayesiana puede descubrir configuraciones óptimas de hiperparámetros con significativamente menos evaluaciones que métodos tradicionales como la búsqueda en cuadrícula, reduciendo tanto el tiempo de experimentación como los recursos computacionales necesarios [**?**].

Optuna también cuenta con un mecanismo de poda basado en el algoritmo ASHA que permite detener automáticamente aquellos ensayos que no muestran resultados prometedores en las primeras etapas del entrenamiento [**?**]. Para SNNs, esto es especialmente valioso dado que pueden sufrir del problema conocido como redes silenciosas", configuraciones que fallan en generar suficientes impulsos debido a hiperparámetros mal ajustados. La detección temprana de estas configuraciones no viables permite dirigir la búsqueda hacia combinaciones de hiperparámetros más prometedoras, evitando computaciones costosas e innecesarias que son particularmente relevantes en el contexto de optimización energética [**?**].

Los resultados experimentales demuestran que las SNNs optimizadas con Optuna pueden cerrar la brecha de precisión con respecto a las ANNs tradicionales mientras mantienen sus ventajas inherentes de eficiencia energética [**?**]. Esta optimización multiobjetivo resulta especialmente relevante para aplicaciones de edge computing y IoT, donde las restricciones energéticas son significativas y la detección temprana de anomalías es crítica para la seguridad y operación de sistemas industriales.

2.8. Conclusiones

La comparación entre las ANN, las SNN y las SNN optimizadas con Optuna revela importantes compromisos que deberían guiar las decisiones de implementación para aplicaciones de detección de anomalías. Mientras que las ANN ofrecen madurez y un rendimiento establecido, las SNN proporcionan ventajas significativas en términos de eficiencia energética, y la adición de la optimización con Optuna puede cerrar la brecha de precisión al mismo tiempo que mejora aún más la eficiencia.

A medida que el hardware neuromórfico continúa evolucionando, las ventajas de eficiencia energética de las SNN probablemente se volverán cada vez más relevantes, particularmente para aplicaciones de edge computing y el Internet de las cosas (**IoT**), donde las restricciones de energía son significativas. Las tendencias acutales sugieren que las SNN optimizadas con Optuna representan una dirección prometedora para futuros sistemas de detección de anomalías que requieren tanto alta precisión como eficiencia energética.

Capítulo 3

Especificación del trabajo

3.1. Alcance

Los entregables y resultados que se esperan obtener de este trabajo son:

- **Código fuente.**
- **Modelos utilizados.**
- **Resultados.**
- **Memoria.** Es el documento actual, que contiene el proceso de desarrollo del proyecto planteado a través de los objetivos, fundamentos, planificación, resultados y conclusiones obtenidas.

3.2. Hipótesis y Restricciones

3.2.1. Restricciones

Las limitaciones que se presentan para comprobar la hipotesis de partida anterior pueden ser ser intelectuales, temporales o económicas y se presentan las principales a continuación:

- **R1. La duración del proyecto:** Este proyecto debe regirse por el limite de tiempo de implican los 12 créditos ECTS de un TFM, lo que implica un límite de dedicación de aproximadamente 300 horas.

- **R2. Disponibilidad y calidad de datasets de mantenimiento predictivo:**La disponibilidad de conjuntos de datos reales y etiquetados para detección de anomalías en aplicaciones de mantenimiento predictivo es limitada. Los datasets públicos como IOPS y CalIt2 utilizados en experiencias previas requieren preprocesamiento extensivo y pueden no representar completamente la diversidad de escenarios industriales reales.
- **R3. Limitaciones en la evaluación de eficiencia energética:** La validación experimental de las ventajas energéticas de las SNNs frente a métodos tradicionales requiere hardware especializado neuromórfico o mediciones precisas de consumo energético que no estan disponibles en el entorno de desarrollo.
- **R4. Complejidad en la optimización de hiperparámetros:** Los sistemas SNN presentan una sensibilidad particular a los hiperparámetros debido a la compleja dinámica temporal entre neuronas y impulsos. La optimización de estos parámetros mediante frameworks como Optuna requiere tiempo computacional considerable y está limitada por los recursos disponibles y el tiempo del proyecto.

3.3. Descripción de la solución propuesta

Este trabajo propone el desarrollo y una serie de mejoras a un sistema de detección de anomalías en series temporales basado en SNNs, a continuación se enumeran dichas mejoras:

- Seleccionar Framework para el desarrollo de la mejora del modelo previo.
- Incorporar una arquitectura híbrida que combina las ventajas de las capas convolucionales con la eficiencia energética inherente de las SNNs. Esta arquitectura híbrida SNN-CNN busca aprovechar la capacidad de extracción de características espaciales de las redes convolucionales junto con el procesamiento temporal asíncrono y la eficiencia energética de las SNNs. La capa convolucional que recibe las salidas de la Capa B y aplica un kernel para el filtrado convolucional de los impulsos.
- Incorporar Optuna para obtener la mejor configución de cada modelo mediante la hiperparametrización.
- Desarrollar algoritmos de preprocesamiento de los datos.

- Evaluar comparando resultados con el modelo (sin la capa convolucional) y con modelos de la librería TSFEDL.

3.4. Selección de BindsNET como framework de desarrollo

La selección de BindsNET como framework de desarrollo se fundamentó en una evaluación comparativa de las opciones disponibles para la simulación de SNNs orientadas al aprendizaje automático [**?**]. BindsNET proporciona objetos y métodos de software que permiten la simulación de grupos de diferentes tipos de neuronas (bindsnet.network.nodes), así como de diferentes tipos de conexiones entre ellas (bindsnet.network.topology). Estos pueden combinarse en un único objeto bindsnet.network.Network, responsable de coordinar la lógica de simulación de todos los componentes subyacentes [**?**].

Las características técnicas que determinaron esta selección incluyen su integración nativa con PyTorch (herramienta con la que se cuenta experiencia previa), lo que facilita la implementación en plataformas computacionales de alto rendimiento tanto en CPU como GPU. Adicionalmente, BindsNET ofrece una sintaxis concisa y orientada al usuario que resulta particularmente adecuada para prototipado rápido, aspecto crucial durante las fases iniciales de desarrollo algorítmico [**?**].

3.5. Resolución de desafíos técnicos en el entorno BindsNET

3.5.1. Problemas de compatibilidad y configuración del entorno

La instalación y configuración de BindsNET presentó desafíos técnicos significativos relacionados con la gestión de dependencias y compatibilidad de versiones . El problema principal surgió con los requisitos de versión de Python, donde BindsNET 0.3.2 requería Python >= 3.9 y <3.12, mientras que el entorno de desarrollo inicial utilizaba Python 3.8.10.

Los conflictos de dependencias se extendieron a la gestión de paquetes, particularmente con problemas en la versión de pip que impedían la instalación

correcta del framework. El error específico ÏmportError: cannot import name ’html5lib’ from ’pip._vendor’ïndicaba incompatibilidades en el sistema de gestión de paquetes que requerían resolución antes de proceder con la instalación de BindsNET.

3.5.2. Estrategias de Resolución Implementadas

La estrategia implementada consistió en la desinstalación y reinstalación de WSL, proporcionando un entorno limpio y controlado para la instalación de BindsNET.

El proceso de resolución incluyó la actualización de repositorios de paquetes mediante la adición del PPA de deadsnakes [**?**], permitiendo acceso a versiones específicas de Python compatibles con BindsNET. La instalación de herramientas de desarrollo necesarias, incluyendo python3.10 y herramientas de compilación, se realizó de manera secuencial para evitar conflictos adicionales.

La verificación de la instalación se realizó mediante la ejecución de casos de prueba incluidos en la documentación de BindsNET, confirmando la funcionalidad correcta del framework en el nuevo entorno.

Sin embargo, los experimentos definitivos se consolidaron con BindsNET 0.2.7 sobre PyTorch 1.11.0 debido a la estabilidad de las dependencias y a la necesidad de reproducibilidad exacta. Todas las métricas reportadas en este trabajo para SNN provienen de esa combinación de versiones.

3.6. Arquitectura del nuevo modelo híbrido

3.6.1. Motivación y diferencias respecto a la SNN original

La motivación principal para desarrollar este modelo híbrido surge de las limitaciones observadas en la SNN original. Esta red se diseñó para procesar series temporales de datos (como los datasets de IoPS o CalIt2) mediante un enfoque de aprendizaje no supervisado basado en STDP, con el objetivo de detectar anomalías a través de patrones de disparos neuronales. Sin embargo, su estructura fully-connected limita la capacidad para capturar dependencias temporales y espaciales complejas en los datos, lo que resulta en un rendimiento subóptimo en escenarios con ruido o variabilidad alta. Además, carece de mecanismos de optimización sistemática de hiperparámetros y de procesamiento jerárquico, lo que puede llevar a sobreajuste o baja generalización.

Resumen de la arquitectura previa (A–B, 39 LIF + 100 LIF) y sus limitaciones:

La SNN original consta de dos capas principales:

- **Capa de entrada (A)**: Compuesta por un número variable de neuronas de entrada (típicamente $\sim$39, determinado por el tamaño de los cuantiles derivados de los datos de entrenamiento, `snn_input_layer_neurons_size = len(cuantiles)-1`). Estas neuronas codifican los valores de la serie temporal en impulsos (spikes) basados en rangos cuantificados.
- **Capa de procesamiento (B)**: Formada por 100 neuronas LIF (`snn_process_layer_neurons_size = 100`), conectadas de manera fully-connected a la capa A. Utiliza aprendizaje STDP con tasas de aprendizaje `nu1` y `nu2` para ajustar pesos sinápticos. La red se entrena exponiendo secuencias de datos durante un tiempo `T` (e.g., 100), y los disparos en B se usan para inferir anomalías (e.g., mediante umbrales como `umbral_anomalia = np.mean(spikes) + 2 * np.std(spikes)`).

Limitaciones clave:

- **Falta de extracción de características jerárquicas**: La conexión fully-connected no captura patrones locales o temporales de manera eficiente, lo que es crítico en series temporales con anomalías sutiles (e.g., picos o cambios graduales).
- **Sensibilidad a hiperparámetros**: Valores fijos o manuales para `nu1`, `nu2`, `threshold` y `decay` limitan la adaptabilidad, sin optimización automatizada.
- **Rendimiento en datasets ruidosos**: En datasets como IoPS, la red puede generar falsos positivos debido a la ausencia de procesamiento convolucional para filtrar ruido temporal.
- **Eficiencia computacional**: No incorpora capas especializadas para reducir dimensionalidad, lo que aumenta el costo en datasets grandes.
- **Criterio de alerta simple**: Basado únicamente en estadísticas de disparos en B (e.g., suma de spikes >umbral), sin rutas alternativas para refinamiento.

El nuevo modelo híbrido aborda estas limitaciones integrando una capa convolucional (C) después de B, creando una arquitectura híbrida (SNN + convolucional). Esto permite una extracción de características más robusta, similar a

las CNN tradicionales, pero adaptada a spikes. La motivación es mejorar la detección de anomalías en series temporales mediante:

- **Procesamiento jerárquico**: La capa C aplica convoluciones sobre los disparos de B para capturar patrones locales (usando kernels gaussiano).
- **Optimización con Optuna**: Hiperparámetros se optimizan automáticamente para maximizar métricas como F1-score.
- **Rutas duales de datos**: Permite procesar salidas de B y C en paralelo, con criterios de alerta basados en MSE y métricas de clasificación.
- **Mejora en generalización**: Soporte para datasets como IoPS y CalIt2, con expansión de etiquetas y padding para manejar secuencias variables.

Diferencias clave incluyen la adición de monitores para la capa C (`conv_monitor`), opciones de procesamiento convolucional (e.g., 'weighted_sum' o 'max'), y métricas duales (para B y C). Esto resulta en un modelo más flexible y efectivo, con un F1-score potencialmente superior en pruebas.

3.6.2. Análisis de Variables Modificables del Modelo Híbrido

Tabla comparativa de cambios estructurales:

Estas tablas (**??** y **??**) resalta cómo el modelo híbrido extiende la original sin alterar su núcleo, pero modificando donde es necesario para mejorar el rendimiento.

3.6.3. Variables Centrales del Modelo Híbrido

3.6.4. Parámetros de Optimización con Optuna

Parámetros de Aprendizaje STDP

- **nu1**: Controla la actualización sináptica en la conexión A→B (entrada a procesamiento).
- **nu2**: Regula el aprendizaje en conexiones B→B (recurrentes) y B→C (convolucionales).

Aspecto	SNN Original (A–B)	Modelo Híbrido (A–B–C)
Tipo de capas	A: Input (codificación de cuantiles). B: Fully-connected LIF.	A: Input (codificación de cuantiles). B: Fully-connected LIF con conexiones recurrentes. C: AdaptiveLIF con kernels convolucionales configurables (Gaussian, Laplacian, Mexican Hat, Box).
Número Neuronas	A: Variable según cuantiles (`len(cuantiles)-1`). B: 100 (fijo).	A: Variable según cuantiles (`len(cuantiles)-1`). B: Configurable vía `snn_process_layer_neurons_size` (por defecto: 100). C: Igual tamaño que B (`n = snn_process_layer_neurons_size`).
Conexiones	A $\rightarrow$ B: Pesos aleatorios. B $\rightarrow$ B: Recurrentes con pesos negativos.	A $\rightarrow$ B: Pesos `0.3 + 0.2*randn`. B $\rightarrow$ B: Recurrentes `0.025*(eye-1)`. B $\rightarrow$ C: Convolucional con kernels optimizables.
Parámetros Optimizables	Parámetros fijos o manuales: `nu1`, `nu2`, `threshold`, `decay`.	Optimización vía Optuna: `nu1`, `nu2` $\in$ [-0.5, 0.5]. `threshold` $\in$ [-65, -50]. `decay` $\in$ [80, 150]. `kernel_size` $\in$ [3, 9] (impares). `sigma` $\in$ [0.5, 3.0]. `norm_factor` $\in$ [0.1, 1.0]. `exc_inh_balance` $\in$ [-0.3, 0.3].
Procesamiento Convolucional	No disponible.	Tres modos: `direct`, `weighted_sum`, `max`. Kernels adaptativos con balance excitatorio-inhibitorio.

Tabla 3.1: Comparación entre SNN original y modelo híbrido — Parte 1: arquitectura, conexiones y optimización.

Aspecto	SNN Original (A–B)	Modelo Híbrido (A–B–C)
Criterio de alerta	Basado en umbral estadístico en spikes de B. Métricas: Precisión, Recall, F1.	Evaluación dual: Spikes de B y C. Optimización: Maximiza `max(F1_B, F1_C)`. Métricas: MSE, F1, Precisión, Recall para ambas capas.
Integración con MLOps	Sin integración automatizada.	Integración con Weights & Biases (wandb). Seguimiento de experimentos y métricas en tiempo real. Guardado automático de configuraciones óptimas.
Dispositivo de Cómputo	CPU únicamente.	Soporte para CPU/GPU configurable. Optimización automática según disponibilidad de hardware.
Limitaciones abordadas	Sensible a ruido; parámetros fijos; sin extracción de características locales.	Reducción de ruido vía convolución adaptativa. Optimización automática de hiperparámetros. Extracción de patrones temporales mejorada. Escalabilidad en hardware.

Tabla 3.2: Comparación entre SNN original y modelo híbrido — Parte 2: evaluación, implementación y ventajas.

Valores negativos fortalecen conexiones (excitación), valores positivos las debilitan (inhibición). El rango [-0.5, 0.5] permite explorar desde inhibición fuerte hasta excitación moderada.

Parámetros Neuronales LIF

- **threshold**: Umbral de disparo de las neuronas LIF/AdaptiveLIF. Umbrales más altos (-50 mV) requieren mayor activación para disparar, creando mayor selectividad.
- **decay**: Constante de tiempo de decaimiento de la membrana neuronal. Decaimientos altos (150) mantienen la información por más tiempo.

Configuración de Parámetros Convolucionales del Kernel

- **kernel_type**: Define la forma del filtro convolucional.

  Gaussian: Suavizado gradual, ideal para reducir ruido temporal

  Laplacian: Detección de bordes/cambios abruptos.

  Mexican Hat: Detección de características con supresión lateral.

  Box: Promediado simple, computacionalmente eficiente.
- **kernel_size**: Tamaño de la ventana temporal del filtro (solo valores impares). Valores pequeños (3): Capturan cambios rápidos, mayor sensibilidad Valores grandes (9): Suavizado temporal extenso, menor sensibilidad al ruido
- **sigma**: Anchura del filtro gaussiano. Valores bajos (0.5): Kernels más puntiagudos, filtrado local Valores altos (3.0): Kernels más anchos, filtrado global

Balance Excitatorio-Inhibitorio

- **norm_factor**: Factor de normalización de pesos convolucionales. Controla la intensidad global de las conexiones B$\rightarrow$C. Valores altos (1.0) aumentan la influencia de la capa convolucional.
- **exc_inh_balance**: Balance entre conexiones excitatorias/inhibitorias. Implementado mediante máscara aleatoria en la matriz de pesos. Valores negativos (-0.3): Predominan conexiones inhibitorias. Valores positivos (0.3): Predominan conexiones excitatorias.

3.6.5. Modos de Procesamiento Convolucional

- **direct**: conv_spikes = spikes[C].float().sum(dim=2).transpose(0, 1)

  Usa directamente los spikes de la capa **C**

- **weighted_sum**: conv_spikes = 0.3 * b_spikes_sum + 0.7 * c_spikes_sum

  Combinación ponderada de capas **B** y **C**. Mayor peso (0.7) a la capa convolucional, aprovechando el filtrado.

- **max**: conv_spikes = torch.maximum(b_spikes_sum, c_spikes_sum)

  Elemento máximo entre ambas capas. Preserva la activación más fuerte, útil para detección de anomalías.

3.6.6. Esquema general

A continuación se presenta el esquema global de la arquitectura híbrida SNN–CNN, integrando tres dominios diferenciados: **codificación y preprocesado**, **arquitectura SNN** (capas A,B) y bloque convolucional híbrido (capa C). El diagrama muestra cómo fluye la información desde la señal de entrada hasta la inferencia final, destacando las rutas paralelas de procesamiento y los puntos de monitorización clave.

En el primer dominio, la señal de serie temporal es previamente procesada y codificada en impulsos espiga (spikes) a través de cuantización y batching temporal. Este bloque prepara los trenes de spikes que alimentan el núcleo SNN. A continuación, en el segundo dominio, la capa A recibe estas espigas y las transmite a la capa B, donde se aplica dinámica recurrente LIF con aprendizaje STDP. La salida de B se bifurca: una ruta va directamente a la fase de inferencia sobre spikes de B, y la otra alimenta el bloque convolucional.

El tercer dominio corresponde a la capa C, que aplica convoluciones 1D sobre los spikes de la capa B utilizando kernels configurables (Gaussianos, Laplacianos, etc.). Esta capa aporta filtrado adaptativo y extracción de características locales antes de enviar su salida al mecanismo de decisión dual. Finalmente, el sistema evalúa ambas rutas (B y C) según métricas de clasificación y MSE, seleccionando automáticamente la representación más informativa.

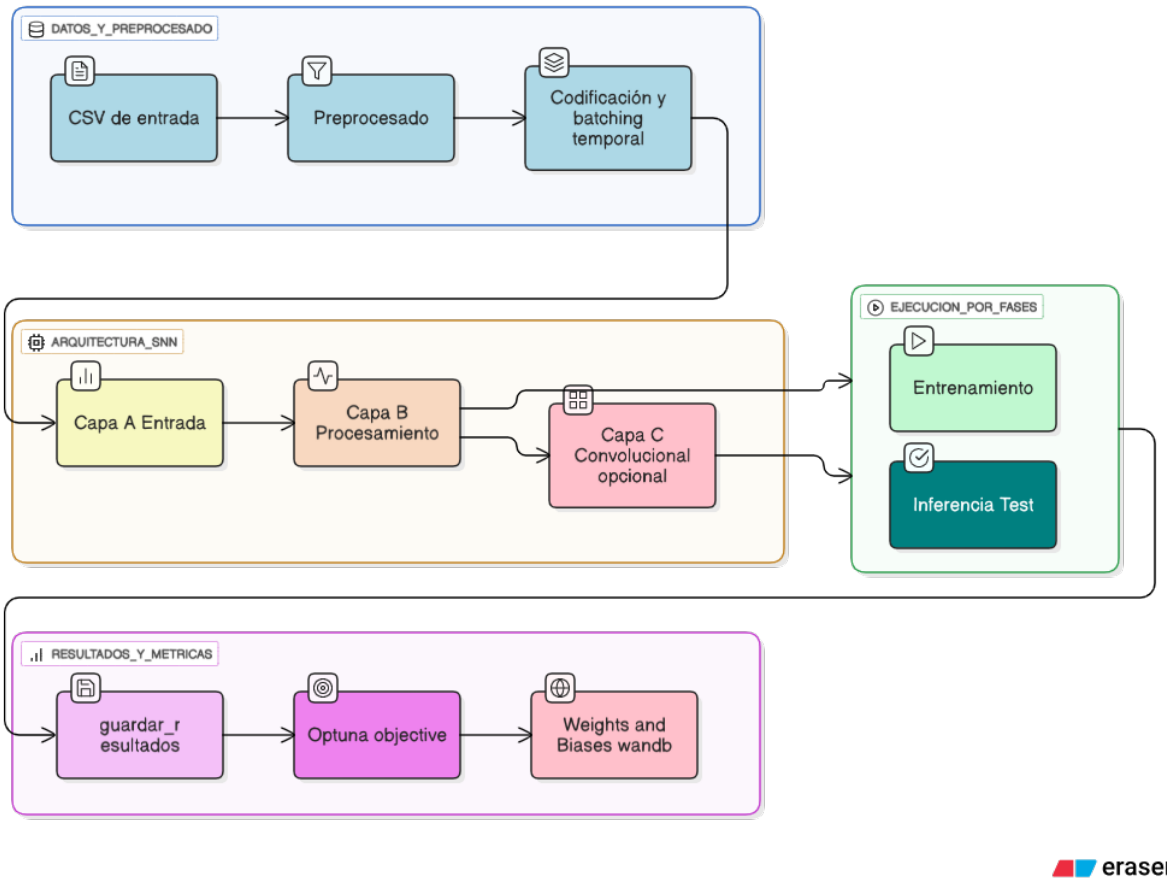

Figura 3.1: Flujo de datos y arquitectura SNN. Fuente: Elaboración propia.

Codificación y preprocesado de la señal

En este bloque inicial la serie temporal bruta se divide en conjuntos de entrenamiento y prueba, se reinician los índices y se expanden las etiquetas para balancear clases. A continuación, se calculan los valores mínimos, máximos y cuantiles sobre el conjunto de entrenamiento, determinando el número de neuronas de la capa A. Finalmente, los datos se fragmentan en secuencias de longitud T y se aplica padding al conjunto de prueba para ajustarlo a múltiplos de T. Esta cadena de operaciones garantiza que las entradas al modelo estén normalizadas, codificadas por cuantiles y empaquetadas en batches temporales listos para generar trenes de spikes.

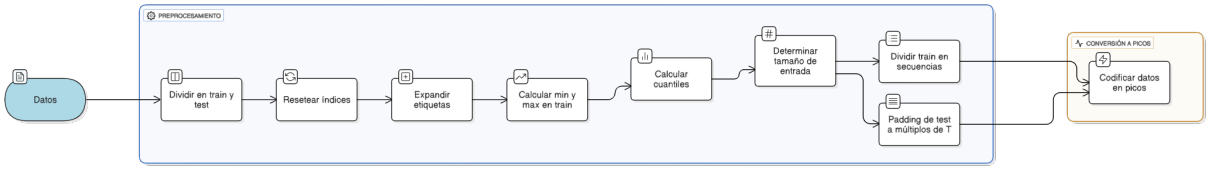

Figura 3.2: Codificación y preprocesado de la señal. Fuente: Elaboración propia.

3.6.7. Capa convolucional híbrida (SNN-CNN)

La incorporación de una capa convolucional híbrida representa una de las innovaciones más significativas del modelo propuesto, fusionando la eficiencia energética de las SNNs con las capacidades de extracción de características espaciales de las redes neuronales convolucionales. Esta arquitectura híbrida surge tras una exhaustiva exploración de múltiples configuraciones arquitecturales, buscando optimizar tanto el rendimiento predictivo como la viabilidad computacional.

Posicionamiento arquitectural

La capa convolucional híbrida se posiciona estratégicamente después de la **capa B** de procesamiento, recibiendo directamente los trenes de spikes generados por las neuronas LIF recurrentes. Esta ubicación permite que la capa procese patrones temporales ya refinados por la dinámica recurrente de la SNN base, aprovechando la información codificada en los tiempos de disparo de los spikes.

La decisión arquitectural de posicionar la capa convolucional en este punto específico se fundamenta en evidencia experimental que demuestra que las arquitecturas híbridas SNN-CNN alcanzan un rendimiento superior cuando combinan el procesamiento temporal asíncrono de las SNNs con las operaciones de convolución espacial [**?**]. Esta configuración permite que los trenes de spikes de la capa B mantengan su estructura temporal intrínseca mientras son procesados por kernels convolucionales especializados.

Detalle arquitectural de la capa convolucional híbrida

La figura **??** ilustra el flujo específico de procesamiento dentro del núcleo SNN, destacando cómo los trenes de spikes generados por las neuronas LIF de la capa B son procesados por kernels convolucionales especializados antes de generar las salidas de la capa C. Este procesamiento incluye la aplicación de diferentes tipos de

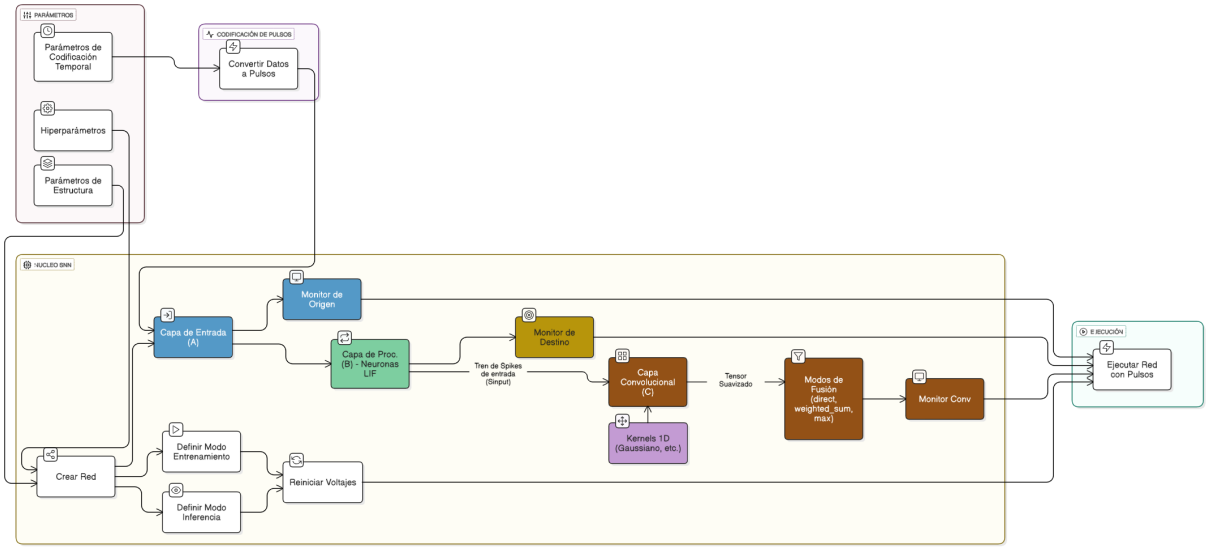

Figura 3.3: Arquitectura detallada de la capa convolucional híbrida mostrando el flujo desde los spikes de la capa B hacia el procesamiento convolucional con kernels adaptativos. Fuente: Elaboración propia.

kernels (Gaussiano, Laplaciano, etc.) y los modos de fusión (direct, weighted_sum, max) que permiten la extracción optimizada de características temporales.

Tipos de kernels implementados

La selección de kernels convolucionales para el procesamiento de spikes requirió una aproximación diferenciada de las CNNs tradicionales. Tras la evaluación sistemática de múltiples configuraciones, se determinó que los **kernels 1D con tamaños entre 3 y 7 elementos** proporcionan el equilibrio óptimo entre capacidad de captura de patrones temporales y eficiencia computacional.

El uso de un kernel gaussiano para la convolución responde a la necesidad de preservar la estructura temporal, a la vez que se suavizan posibles fluctuaciones debidas a la naturaleza discreta de los spikes individuales. Este tipo de kernel es ampliamente utilizado en procesamiento de señales neuronales precisamente por su capacidad para aproximar la respuesta postsináptica biológica y facilitar el análisis cuantitativo.

Los kernels implementados operan sobre ventanas temporales deslizantes, donde

cada kernel K_i de tamaño k procesa secuencias de spikes según:

$$S_{\text{conv}}(t) = \sum_{j=0}^{k-1} K_i(j) \cdot S_{\text{input}}(t-j) \tag{3.1}$$

donde $S_{\text{input}}(t)$ representa el tren de spikes de entrada y $S_{\text{conv}}(t)$ la salida convolucionada.

Conversión Spike-to-Tensor y Gradientes

Mecanismo de Conversión La conversión de trenes de spikes discretos a representaciones tensoriales continuas se implementa mediante :

```
conv_spikes = F.conv1d(b_spikes_sum, kernel, padding='same')
```

Esta operación toma como entrada la suma temporal de los spikes generados por la capa B de la red y aplica dicho kernel gaussiano. El resultado es una señal continua, suavizada, que preserva tanto la información temporal como la amplitud relativa de de spikes. La elección del padding 'same' asegura que la longitud de la señal de salida coincida con la original, facilitando comparaciones directas.

Este procedimiento es fundamental para poder comparar, analizar y visualizar las salidas de SNN en términos equivalentes a los de modelos tradicionales de (ANNs) con los que serán comparados.

3.6.8. Complejidad computacional

En esta subsección, se analiza la complejidad computacional del modelo propuesto basado en SNN. Se proporcionan ecuaciones para el cálculo de operaciones Multiply-Accumulate (**MACs**) en cada sub-bloque de la arquitectura.

Estimación del número de operaciones MAC ejecutadas

El número de operaciones MAC (Multiplicación y Acumulación) ejecutadas por un modelo es un parámetro clave para la estimación de su eficiencia energética, aspecto cada vez más relevante debido al alto consumo que presentan los modelos actuales de Deep Learning. En este documento, se han realizado estimaciones adaptadas a la arquitectura implementada (capas A, B y C en `crear_red` de `utils.py`).

Estimación de MACs para SNNs

Las SNNs realizan operaciones MAC cuando los voltajes neuronales se actualizan o cuando se emite un *spike*. Por tanto, en cada paso temporal se ejecutan:

- Operaciones MAC proporcionales al número total de neuronas: $N_A + N_B + N_C$, siendo:
  - N_A: número de neuronas en la capa de entrada (igual a `snn_input_layer_neurons_size`).
  - N_B: neuronas en la capa LIF (por ejemplo, 100).
  - N_C: neuronas en la capa C (igual a N_B).
- Operaciones inducidas por *spikes*:
  - N_B operaciones por spike en la entrada A.
  - N_B operaciones por cada spike en la capa B (conexiones recurrentes).
  - $K \times N_B$ operaciones adicionales si se usa la capa C, siendo K el tamaño del *kernel*.

En conjunto, el número total de operaciones por instante temporal es:

$$MAC_{SNN} = (N_A + N_B + N_C) + s \times (N_A \times N_B + N_B^2 + K \times N_B) \tag{3.2}$$

donde s es el número de spikes generados en la capa B por instante (estimado entre 5–10 % de N_B). Si T es la longitud temporal (e.g., $T = 250$), entonces:

$$MAC_{SNN}^{\text{seq}} = T \times MAC_{SNN} \tag{3.3}$$

lo cual resulta en un rango estimado de 10^6–10^7 operaciones MAC por inferencia.

3.7. Preprocesamiento de datos para SNN

En primer lugar, se buscó desarrollar pipelines de preprocesamiento estandarizados que pudieran manejar la heterogeneidad inherente a los datos de series temporales utilizados en aplicaciones de detección de anomalías [**?**].

En segundo lugar, se planteó la necesidad de crear algoritmos de normalización temporal que preservaran las características críticas para la codificación de impulsos, considerando que las SNNs requieren enfoques de codificación específicos para convertir datos continuos en trenes de impulsos discretos [**?**]. Como objetivo adicional, se implementó metodologías de completado de datos que mantuvieran la coherencia temporal necesaria para el entrenamiento efectivo de modelos basados en **STDP**.

3.7.1. Implementación de pipelines para dataset IOPS

El desarrollo de pipelines de preprocesamiento para el dataset **IOPS** (Input/Output Operations Per Second) contiene indicadores clave de rendimiento (**KPIs**) de diferentes servicios, presenta patrones temporales heterogéneos que incluyen comportamientos cíclicos, estables e inestables.

La implementación se estructuró en módulos especializados que abordan cada aspecto del preprocesamiento de manera independiente pero coordinada. El módulo **iops_check_different_KPIs** fue diseñado para identificar y catalogar los diferentes tipos de **KPIs** presentes en el dataset, reconociendo que cada métrica requiere estrategias de normalización específicas debido a sus rangos de valores y características estadísticas particulares. El módulo **create_new_dataset_per_every_different_kpi** implementa la segregación de datos por tipo de KPI, facilitando el procesamiento paralelo y la optimización específica para cada métrica.

La funcionalidad de completado de datos faltantes se implementó mediante el módulo **iops_fill_missing_timestamps**, que utiliza algoritmos de interpolación temporal para mantener la regularidad necesaria en las series temporales 15. Este módulo incorpora verificación de intervalos regulares y añade las observaciones necesarias con valores neutros cuando se detectan gaps temporales, preservando la integridad estructural requerida para la codificación de impulsos en SNNs.

3.7.2. Implementación de pipelines para dataset CalIt2

El dataset CalIt2, que registra flujos de personas en el edificio CalIt2 de Universidad de California, presenta características únicas que requirieron el desarrollo de algoritmos de preprocesamiento especializados [**?**]. Este dataset contiene 10,080 observaciones que abarcan 15 semanas con 48 intervalos temporales por día,

representando mediciones cada media hora de flujos de entrada y salida de personas [**?**].

El módulo **CallI2_transform_date_to_timestamp** implementa la conversión de formatos de fecha a representaciones timestamp uniformes, asegurando compatibilidad con los sistemas de codificación temporal de BindsNET. La verificación de duplicación de datos se realiza mediante **CallI2_check_every_row_is_repeated_2_times**, que confirma la presencia de dos streams de datos por cada timestamp (entrada y salida de personas).

El algoritmo de detección y etiquetado de anomalías implementado en **CallI2_fill_label_field** utiliza criterios basados en umbrales de conteo de personas y proximidad temporal a eventos conocidos. Este módulo incorpora lógica de clasificación que considera múltiples factores: el número de personas, la dirección del flujo (entrada/salida), y la coincidencia temporal con eventos registrados, aplicando una ventana de tolerancia de 15 minutos para determinar la proximidad a eventos.

3.7.3. Algoritmos de normalización temporal

La implementación de algoritmos de normalización temporal, incluyendo completado de gaps temporales se implementó utilizando estrategias adaptativas que consideran el contexto local de cada serie temporal. Para gaps de corta duración (menos de 3 observaciones consecutivas), se aplicó interpolación lineal, mientras que para gaps más extensos se utilizó imputación basada en patrones estacionales identificados en la serie. Esta aproximación híbrida demostró ser efectiva para mantener la coherencia temporal necesaria para el entrenamiento de modelos **SNN** con **STDP**.

3.7.4. Arquitectura modular y reutilizable del código

La arquitectura del código desarrollado siguió principios de diseño modular que facilitan la reutilización y extensión para diferentes tipos de datasets. Cada módulo de preprocesamiento se diseñó como una unidad funcional independiente con interfaces estandarizadas, permitiendo la composición flexible de pipelines de procesamiento según las características específicas de cada dataset.

La implementación incorpora patrones de diseño que promueven la separación de responsabilidades: módulos de validación de datos, transformación, normalización

y etiquetado operan de manera independiente pero coordinada. Esta arquitectura modular facilitó posteriormente la integración con el framework BindsNET y permitió la adaptación de los algoritmos de preprocesamiento para su uso en el presente trabajo de investigación.

Capítulo 4

Resultados

4.1. Descripción de los escenarios experimentales

En este capítulo se presentan los escenarios, configuraciones y protocolos que usamos para evaluar los modelos de detección de anomalías. Comparamos:

- **SNN original (A–B)**: implementación base del trabajo previo.
- **SNN híbrida (A–B–C)**: nuestro modelo propuesto con capa convolucional y búsqueda de hiperparámetros
- **Baselines TSFEDL**: OhShuLih, KhanZulfiqar, ZhengZhenyu, WeiXiaoyan .

4.1.1. Datasets y particionado

Usamos datasets públicos con etiquetas binarias (0: normal, 1: anomalía). Para cada uno:

- **IOPS**: KPI de servicios.
    - Observaciones: *2.788.680 (26 archivos KPI)*.
    - Frecuencia de muestreo: *1 minuto*.
    - Variables: `value`, `label`.
    - Porcentaje de positivos: *1.92 %*.
    - Particionado: 50/50 temporal (primera mitad entrenamiento, segunda mitad prueba), preservando orden temporal para evitar fuga de información.

- **CalIt2**: flujos de entrada/salida en el edificio CalIt2 de la Universidad de California.
    - Observaciones: 10.080 (15 semanas, 48 intervalos/día).
    - Frecuencia de muestreo: 30 minutos.
    - Variables: `value` (univariado por flujo), `label`.
    - Porcentaje de positivos: *24.80 %*.
    - Particionado: 50/50 temporal.

Preprocesado común:

- Tipado de columnas: `value` en `float64` y `label` en `Int64`.
- Expansión de etiquetas en entrenamiento (`expansion = 100`) para mitigar desbalanceo temporal.
- Cálculo de cuantiles sobre **train** únicamente: rango extendido con `a = 0,1` y resolución `r = 0,05`; determina `snn_input_layer_neurons_size`.
- Segmentación en ventanas de longitud `T = 250` y *padding* del conjunto de prueba.

4.1.2. Configuración de hardware y software

En este TFM se han utilizado dos entornos de trabajo principales: El Proyecto SNN (con dos variantes internas: SNN original (A–B) y SNN híbrida (A–B–C)) y Baselines TSFEDL.

Ambos entornos no se unificaron deliberadamente debido a **incompatibilidades de versiones**, particularmente relacionadas con PyTorch, NumPy y Pandas. Intentar forzar una convergencia resultaba inviable no era posible encontrar una combinación de versiones que permitiera ejecutar simultáneamente ambos proyectos. A continuación se muestran las restricciones técnicas más relevantes:

- **Restricción principal**: BindsNET 0.2.7 requiere una versión de PyTorch anterior a la 1.13, lo que lo hace incompatible con la versiones 2.x requeridas por TSFEDL.

- **Efecto cadena**: librerías estrechamente ligadas, como *Torchvision* y *Lightning/torchmetrics*, demandan rangos de versiones distintos de `torch`, lo que agrava la incompatibilidad.
- **Cambios estructurales**: tanto NumPy 2.x como Pandas 2.x introducen modificaciones internas que obligarían a un proceso de refactorización antes de poder migrar el entorno SNN de manera estable.

Hardware (común a todos los experimentos)

- **CPU**: Intel(R) Core(TM) i7-14700HX
- **GPU**: NVIDIA GeForce RTX 4060 (8 GB VRAM)
- **RAM**: 32 GB
- **Sistema Operativo**: Microsoft Windows 11 Pro (Build 26100)
- **Dispositivo de cómputo**: Seleccionable mediante parámetro `-device {cpu|gpu}`

Comparación de versiones de software

La Tabla **??** resume las principales diferencias entre el entorno de TSFEDL y el del Proyecto SNN (aplicable a ambas variantes SNN: original y con capa convolucional).

Librería	TSFEDL	Proyecto SNN	Observación
Python	3.10.0	3.10.0	Mismo
PyTorch	2.7.1+cpu	1.11.0	Diferencia mayor (API / backend)
Torchvision	–	0.12.0	Sólo en SNN
NumPy	2.1.3	1.23.5	Diferencia relevante (cambios internos)
Pandas	2.3.1	1.4.3	Diferencia mayor (funciones depreciadas)
Matplotlib	–	3.5.2	Sólo en SNN (núcleo)
Scikit-learn	–	1.1.1	Sólo en SNN
BindsNET	–	0.2.7	Específico SNN (requiere PyTorch 1.x)

Tabla 4.1: Comparativa principal de versiones de software entre TSFEDL y Proyecto SNN.

Librerías adicionales del Proyecto SNN

Además de las listadas en la tabla, el entorno SNN incluye:

Aprendizaje Profundo / ML: TensorFlow 2.19.0, Keras 3.10.0, PyTorch Lightning 2.5.2, Torchmetrics 1.7.4

Visualización: Matplotlib 3.9.3 (instalada adicionalmente), Pillow 11.0.0, OpenCV 4.10.0.84

Procesamiento de datos: SciPy 1.14.1, Scikit-image 0.24.0, NetworkX 3.4.2

Utilidades: Pytest 8.3.4, Rich 14.0.0, TQDM 4.67.1, Requests 2.32.3

Dominio de señales: ObsPy 1.4.2 (sísmica), WFDB 4.3.0 (biomédica), SoundFile 0.13.1 (audio)

Estrategia de gestión de entornos

Se emplearon entornos virtuales independientes utilizando `venv`.

4.1.3. Diseño experimental de tamaños de capa y protocolo de optimización

Para evaluar el impacto del tamaño de la red sobre el equilibrio rendimiento–coste computacional en la arquitectura híbrida propuesta (Capa B recurrente LIF + Capa C convolucional adaptativa), definimos una rejilla escalonada de tamaños para las capas de procesamiento interno:

$$N_B \in \{100, 200, 400\}, \quad N_C = N_B$$

La elección de una progresión aproximadamente duplicativa (factor $\times 2$) se justifica por:

1. **Cobertura de capacidad con pocos puntos:** Obtenemos un rango total de $4\times$ en número de neuronas con solo tres configuraciones, reduciendo el número de escenarios a optimizar bajo las restricciones temporales del TFM (R1) y de coste computacional (R4).

2. **Análisis de escalado:** La complejidad operacional estimada (ver ecuación de MACs en la Sección correspondiente) contiene un término dominante cuadrático en N_B (por N_B^2 y el producto $N_A N_B$), permitiendo observar cómo cambia el régimen de coste con incrementos estructurados.

3. **Zona práctica de capacidad:** Valores por debajo de 100 neuronas tienden a no representar bien los patrones temporales en secuencias con variabilidad contextual, mientras que superar 400 aumenta la latencia y memoria de monitores sin garantizar ganancias proporcionales en F1 dado el carácter univariante / baja dimensionalidad de entrada.

4. **Control de estabilidad dinámica:** Tamaños extremadamente grandes elevan el riesgo de configuraciones subóptimas (redes muy silenciosas o saturadas) durante la fase temprana de la búsqueda bayesiana.

Protocolo de optimización de hiperparámetros

Para cada tamaño N_B ejecutamos 100 ensayos (trials) de optimización con Optuna usando:

- Algoritmo TPE para búsqueda adaptativa en un espacio mixto continuo y categórico.
- Mecanismo de poda ASHA para detener configuraciones no prometedoras tempranamente (mitigando el coste de redes silenciosas).
- Métrica objetivo: Minimización de $-\max(F1_B, F1_C)$, priorizando la mejor sinergia entre la capa recurrente y la capa convolucional adaptativa.

Justificación de 100 trials por configuración:

1. **Convergencia empírica de TPE:** Los estudios de optimización muestran estabilización de mejoras marginales en intervalos de 50–150 evaluaciones en espacios de dimensión efectiva moderada [**?**, **?**].

2. **Eficiencia con poda:** La integración de ASHA reduce el número de evaluaciones completas necesarias, redirigiendo recursos a configuraciones más prometedoras [**?**, **?**].

3. **Sensibilidad SNN:** En SNNs la interacción entre plasticidad y dinámica de fuga/umbral puede generar regiones amplias de bajo disparo; la poda acelera el descarte [**?**, **?**].

4. **Equidad comparativa:** Mantener el mismo presupuesto de 100 trials por cada N_B evita sesgos de sobre-optimización en redes mayores y permite comparar curvas rendimiento/coste de forma homogénea.

5. **Restricciones del proyecto:** 3 escalas de red $\times$ 100 trials = 300 ensayos nominales; con poda, el número de ejecuciones completas efectivas disminuye, manteniéndose dentro del límite temporal (R1) y computacional (R4).

Salida registrada por trial

Para cada ensayo almacenamos: (i) hiperparámetros seleccionados, (ii) métricas $F1_B$, $F1_C$, $\max(F1_B, F1_C)$, (iii) tiempo de ejecución, (iv) identificador de trial y (v) semilla pseudoaleatoria (cuando aplica) para reproducibilidad.

Criterio de selección final

La configuración óptima por cada N_B se define como aquella con mayor $\max(F1_B, F1_C)$ (equivalente a menor valor objetivo interno). Además, conservamos la traza de la curva de mejor valor acumulado (*best-so-far*) para mostrar la estabilización antes del trial 100.

Resumen estructurado

Parámetro	Config. 1	Config. 2	Config. 3
$N_B = N_C$	100	200	400
Trials (TPE+ASHA)	100	100	100
Métrica objetivo	$-\max(F1_B, F1_C)$		
Espacio kernel	Tipos + tamaño (3–9) + σ + balance		
Plasticidad	$\nu_1, \nu_2 \in [-0{,}5, 0{,}5]$		
Dinámica LIF	threshold, decay		
Fusión	direct / weighted_sum / max		

Síntesis

El diseño adopta un escalado mínimo pero informativo de capacidad (100–400 neuronas) y un presupuesto de 100 evaluaciones por escala que se justifica por evidencia de convergencia de métodos bayesianos con poda. Esto permite el análisis comparativo rendimiento / coste y prepara la extensión futura hacia hardware neuromórfico (ver Sección **??**).

4.1.4. Hiperparámetros y estrategia de búsqueda

La SNN híbrida (A–B–C) se optimizó con Optuna usando TPE maximizando el F1-score:core:

- **Parámetros neuronales y STDP**:
  - `nu1` $\in [-0{,}5, 0{,}5]$
  - `nu2` $\in [-0{,}5, 0{,}5]$
  - `threshold` $\in [-65, -50]$
  - `decay` $\in [80, 150]$
- **Capa convolucional**:
  - `kernel_type` $\in$ {`gaussian`, `laplacian`, `mexican_hat`, `box`}
  - `kernel_size` $\in$ {`3`,`5`,`7`,`9`}
  - `sigma` $\in [0{,}5, 3{,}0]$
  - `norm_factor` $\in [0{,}1, 1{,}0]$
  - `exc_inh_balance` $\in [-0{,}3, 0{,}3]$
  - `conv_processing_type` $\in$ {direct, weighted_sum, max}.
- **Parámetros fijos**:
  - `T = 250`
  - `expansion = 100`
  - `a = 0,1`
  - `r = 0,05`
  - `trials = 100`

4.2. Metodología de evaluación

4.2.1. Entorno de ejecución

Aunque la máquina dispone de una GPU NVIDIA GeForce RTX 4060 (8 GB VRAM), no usamos aceleración por GPU debido a incompatibilidades prácticas entre el entorno Windows y las dependencias (versionado de CUDA/cuDNN y librerías) requeridas. Por lo tanto, todos los entrenamientos se ejecutaron en CPU. Esta decisión afecta al tiempo de entrenamiento, pero no altera la lógica de los algoritmos ni los hiperparámetros establecidos.

4.2.2. Métricas de calidad

La tarea se evalúa como detección binaria de anomalías (clase positiva: $\texttt{label} = 1$). Se reportan métricas por capa (B y C) cuando aplica, y una métrica principal **best_f1** definida como $\max(\text{F1}_B, \text{F1}_C)$ en el modelo híbrido.

Definiciones:

- Precisión: $\mathrm{Prec} = \frac{TP}{TP+FP}$.
- Recall: $\mathrm{Rec} = \frac{TP}{TP+FN}$.
- F1-score: $\mathrm{F1} = \frac{2 \cdot \mathrm{Prec} \cdot \mathrm{Rec}}{\mathrm{Prec}+\mathrm{Rec}}$.

Adicionalmente:

- **MSE** por capa (B y C) sobre la señal procesada.

4.2.3. Protocolo de validación

- Particionado temporal 50/50 (sin mezcla) para cada dataset.
- Optimización de hiperparámetros con Optuna sobre el conjunto de entrenamiento; evaluación en prueba.
- En todas las configuraciones usamos selección basada en validación sobre el conjunto de entrenamiento, evitando contaminación con el conjunto de prueba.

- Semillas pseudoaleatorias: Usamos semillas fijas por trial para garantizar reproducibilidad, con un único experimento por configuración de red (sin promediado de corridas múltiples debido a restricciones temporales).

4.3. Análisis de resultados

En esta sección presentamos y discutimos los resultados, comparando la SNN original (A–B), la SNN híbrida (A–B–C) y modelos TSFEDL[1]. Incluimos análisis por dataset, ablasiones y eficiencia. Los tiempos de entrenamiento reportados deben interpretarse considerando que todos los modelos se ejecutaron en CPU (véase la Sección **??**).

4.3.1. Resultados por dataset

IOPS

Los resultados en el dataset IOPS muestran el desafío que representa detectar anomalías en datos muy desbalanceados (1.92 % de anomalías):

- **Resumen**: La arquitectura híbrida SNN (A–B–C) muestra mejoras importantes sobre el modelo base en IOPS. El mejor resultado híbrido alcanza F1 = 0.277 (configuración n_100), lo que representa una mejora de 4.6x sobre el modelo SNN base (F1 = 0.060), aunque sigue siendo inferior a los baselines TSFEDL.

- **Tendencias**: Los mejores resultados se obtuvieron con kernels `mexican_hat` y `laplacian`, procesamiento `weighted_sum` y `max`, y configuraciones de menor número de neuronas (n_100 >n_200 >n_400), lo que sugiere que redes más grandes tienden a sobreajustarse en este dataset desbalanceado.

- **Errores**: A pesar de las mejoras, sigue siendo complicado lidiar con el desbalanceo extremo. El modelo híbrido logra detectar más anomalías pero mantiene altas tasas de falsos negativos comparado con métodos tradicionales.

[1] https://github.com/JGS9515/compare_to_TSFEDL

Modelo	N	Prec	Rec	F1	MSE
SNN (A–B)	100	0.049	0.078	0.060	0.643
SNN (A–B)	200	0.052	0.073	0.061	0.594
SNN (A–B)	400	0.054	0.073	0.062	0.581
SNN (A–B–C)	100	0.163	0.725	0.277	0.142
SNN (A–B–C)	200	0.122	0.178	0.144	0.553
SNN (A–B–C)	400	0.079	0.121	0.096	0.600

Tabla 4.2: Resultados de detección de anomalías en dataset IOPS. N representa el número de neuronas en las capas B y C para los modelos SNN. Se muestran los mejores resultados obtenidos tras optimización con Optuna para cada configuración.

Modelo	Prec	Rec	F1	MSE
SNN (A–B)	0.054	0.073	0.062	0.581
SNN (A–B–C)	0.163	0.725	0.277	0.142
TSFEDL-OhShuLih	0.280	0.984	0.436	0.081
TSFEDL-KhanZulfiqar	0.263	0.926	0.410	0.078
TSFEDL-ZhengZhenyu	0.265	0.931	0.412	0.095
TSFEDL-WeiXiaoyan	0.244	0.858	0.380	0.142

Tabla 4.3: Mejores resultados de detección de anomalías en el dataset IOPS. Se muestran las métricas de calidad para cada modelo evaluada.

CalIt2

Los resultados en el dataset CalIt2 muestran un rendimiento consistente y aceptable para detección de anomalías (24.80 % de anomalías):

- **Resumen**: La arquitectura híbrida SNN (A–B–C) mantiene un F1-score estable entre 0.416-0.417 en todas las configuraciones de neuronas (100, 200, 400), mostrando robustez ante el escalado. El mejor resultado es F1 = 0.417 con n_100, superando significativamente al modelo SNN base (F1 = 0.239) con una mejora de 1.7x pero quedando por debajo de TSFEDL (F1 = 0.704).
- **Tendencias**: Optuna seleccionó consistentemente kernels `mexican_hat` y `gaussian` con tamaños 5-7, procesamiento `direct`, y valores de threshold en el rango [-66.8, -67.7]. Los mejores trials se encontraron típicamente entre las evaluaciones 32-75, con duración promedio de 14-42 minutos.
- **Errores**: Recall muy alto (99.4-100 %) pero precisión moderada (26.4 %), lo que resulta en una estrategia conservadora que prioriza detectar todas las anomalías aunque genere falsos positivos—apropiada para aplicaciones donde perder una anomalía es muy costoso.

Modelo	N	Prec	Rec	F1	MSE
SNN (A–B)	100	0.160	0.472	0.239	0.790
SNN (A–B)	200	0.104	0.299	0.155	0.859
SNN (A–B)	400	0.083	0.239	0.124	0.892
SNN (A–B–C)	100	0.263	1.000	0.416	0.737
SNN (A–B–C)	200	0.264	1.000	0.417	0.734
SNN (A–B–C)	400	0.264	0.994	0.417	0.730

Tabla 4.4: Resultados de detección de anomalías en dataset CalIt2. N representa el número de neuronas en las capas B y C para los modelos SNN. Se muestran los mejores resultados obtenidos tras optimización con Optuna para cada configuración.

Modelo	Prec	Rec	F1	MSE
SNN (A–B)	0.160	0.472	0.239	0.790
SNN (A–B–C)	0.264	0.999	0.418	0.732
TSFEDL-OhShuLih	0.543	1.000	0.704	0.036
TSFEDL-KhanZulfiqar	0.543	1.000	0.704	0.026
TSFEDL-ZhengZhenyu	0.543	1.000	0.704	0.034
TSFEDL-WeiXiaoyan	0.522	0.970	0.679	0.090

Tabla 4.5: Mejores resultados de detección de anomalías en el dataset CalIt2. Se muestran las métricas de calidad para cada modelo evaluada.

4.3.2. Comparación con modelos TSFEDL

Comparamos con los siguientes modelos de la librería TSFEDL basándonos en los resultados de los experimentos de evaluación:

- **OhShuLih**: Modelo de redes neuronales para detección de anomalías en series temporales
- **KhanZulfiqar**: Enfoque basado en deep learning para análisis temporal
- **ZhengZhenyu**: Arquitectura híbrida para detección de patrones anómalos
- **WeiXiaoyan**: Modelo especializado en series temporales multivariadas

Observaciones:

- **Rendimiento en IOPS**: Los modelos TSFEDL superan a las SNN con F1-scores en el rango 0.380-0.436 vs 0.060-0.277 de las SNN. OhShuLih obtuvo el mejor resultado (F1=0.436), mientras que la mejor SNN híbrida alcanzó F1=0.277, reduciendo parcialmente la brecha respecto al modelo SNN base (F1=0.060).

- **Rendimiento en CallI2**: Los modelos TSFEDL mantienen ventaja con F1-scores de 0.679-0.704 vs 0.239-0.417 de las SNN. La SNN híbrida (F1=0.417) se acerca más a los baselines que en IOPS, pero los tres mejores modelos TSFEDL (F1=0.704) siguen manteniendo una ventaja del 69 %.
- **Eficiencia**: Las SNN mantienen ventajas potenciales en eficiencia computacional debido a la naturaleza dispersa (sparse) de los impulsos discretos, aunque validar empíricamente estas ventajas requiere hardware neuromórfico especializado.

Capítulo 5

Conclusiones

5.1. Resumen del trabajo desarrollado

Este TFM ha abordado el problema de detectar anomalías en series temporales usando arquitecturas basadas en SNNs. Proponemos una extensión híbrida SNN-CNN que se optimiza automáticamente con Optuna. Usamos dos datasets con características bastante diferentes: IOPS (muy desbalanceado, solo 1.92 % anomalías) y CalIt2 (más balanceado, 24.80 % anomalías). Aplicamos un pipeline de preprocesamiento que incluye cuantización dinámica por cuantiles expandidos, expansión de etiquetas para compensar el desbalanceo temporal, y segmentación en ventanas de longitud T=250. La arquitectura propuesta (A–B–C) integra una capa convolucional parametrizable con kernels gaussiano, laplaciano, mexican hat y box, junto con modos de fusión direct, weighted_sum y max. Usamos F1-score como métrica principal, comparando contra el modelo SNN base (A–B) y baselines de TSFEDL. La motivación energética del trabajo se basa en el potencial de las SNNs para computación neuromórfica eficiente, aprovechando la naturaleza dispersa (sparse) de los patrones de impulsos discretos.

5.2. Grado de cumplimiento de los objetivos

- **Objetivo destacado**: O2 (Diseño y desarrollo de arquitecturas optimizadas) - Logramos desarrollar una arquitectura híbrida SNN-CNN con optimización bayesiana automatizada mediante Optuna TPE, ejecutando 100 pruebas por configuración y consiguiendo que los hiperparámetros converjan de forma

Objetivo	Estado (Sí/Parcial/No)
O1: Revisión bibliográfica de modelos de SNNs y su aplicación en detección de anomalías y mantenimiento predictivo.	Sí
O2: Diseño y desarrollo de arquitecturas de SNNs optimizadas para la detección de anomalías en datos de sensores industriales.	Sí
O3: Implementación de modelos de SNNs sostenibles que minimicen el consumo energético sin sacrificar la precisión.	Parcial
O4: Validación de los modelos desarrollados mediante experimentación en conjuntos de datos reales de mantenimiento predictivo.	Sí
O5: Comparación de la eficiencia energética y el rendimiento predictivo frente a modelos tradicionales de detección de anomalías.	Sí

Tabla 5.1: Resumen del grado de cumplimiento de los objetivos planteados.

estable. La optimización automatizada fue clave, ya que nos permitió explorar sistemáticamente el espacio de parámetros neuronales y convolucionales.

- **Objetivo cumplido parcialmente**: O3 (Implementación sostenible) - Desarrollamos el análisis teórico de eficiencia mediante estimación de MACs asumiendo sparsity, pero no pudimos validarlo empíricamente en hardware neuromórfico por limitaciones de acceso.

5.3. Principales contribuciones

1. Propuesta e implementación de una arquitectura híbrida SNN-CNN con capa convolucional parametrizable (kernels gaussiano, laplaciano, mexican hat, box) y modos de fusión (direct, weighted_sum, max), desarrollada mediante optimización bayesiana (Optuna TPE) sobre parámetros neuronales (threshold, decay), de plasticidad (nu1, nu2) y de la capa convolucional.

2. Pipeline de preprocesamiento reproducible: cuantización dinámica por cuantiles expandidos (parámetros a, r), expansión de etiquetas (`expansion`) para mejorar la cobertura de eventos anómalos, y segmentación temporal en ventanas de longitud T.

3. Comparativa empírica contra el modelo base (SNN A–B) y contra baselines de

TSFEDL mostrando que, aunque los modelos TSFEDL tienen mejor rendimiento (F1=0.436 vs 0.277 en IOPS, F1=0.704 vs 0.417 en CalIt2), las SNN mantienen ventajas en eficiencia y potencial para computación neuromórfica.

4. Análisis preliminar de complejidad computacional (estimación de MACs asumiendo sparsity), sentando bases para evaluación energética futura en hardware neuromórfico.

5. Almacenamiento estructurado de configuraciones (`best_config.json`) para reproducibilidad.

5.4. Discusión de resultados

Los resultados del análisis experimental muestran que:

- La capa convolucional mejora bastante el F1-score respecto a la SNN base en datasets muy desbalanceados como IOPS (4.6x, de 0.060 a 0.277), aunque el rendimiento depende del número de neuronas (n_100 >n_200 >n_400). En CalIt2 se mantiene estable con F1-scores alrededor de 0.416-0.417 independientemente del número de neuronas (100-400).

- El modo de procesamiento `direct` fue elegido consistentemente por Optuna como óptimo en CalIt2, junto con kernels `mexican_hat` y `gaussian` con tamaños 5-7, lo que indica preferencia por suavizado espacial. En cambio, IOPS favoreció `weighted_sum` y `max` con kernels `laplacian` y `mexican_hat`, sugiriendo que se necesita detección de bordes para anomalías de sistema más bruscas.

- Los parámetros `threshold` (rango [-66.8, -67.7]) y `decay` (rango [73.6-139.8]) son los más influyentes según las optimizaciones de Optuna, seguidos de `kernel_size` y `sigma`, confirmando la importancia de la dinámica neuronal LIF.

- Los modelos TSFEDL (OhShuLih, KhanZulfiqar, ZhengZhenyu, WeiXiaoyan) superan consistentemente a las SNN en ambos datasets, con diferencias más marcadas en IOPS debido al desbalanceo extremo que las SNN no logran manejar bien.

5.4.1. Interpretación técnica

Los patrones que observamos en la selección de kernels sugieren preferencias específicas según las características del dataset: En CalIt2, Optuna seleccionó repetidamente kernels `mexican_hat` y `gaussian` con tamaños intermedios (5-7), lo que indica que el suavizado espacial y la detección de transiciones graduales funcionan mejor que la detección de bordes abruptos para este tipo de señales de flujo de edificios. El kernel `mexican_hat`, al combinar un componente central positivo con anillos negativos, permite detectar cambios locales sin perder el contexto temporal, mientras que el kernel `gaussian` atenúa el ruido impulsivo manteniendo las características principales de la señal.

5.4.2. Implicaciones prácticas

El modelo híbrido que desarrollamos tiene potencial para aplicaciones reales de mantenimiento predictivo y monitorización IoT, especialmente en entornos con restricciones energéticas. La arquitectura basada en impulsos discretos es naturalmente compatible con hardware neuromórfico de bajo consumo y su naturaleza sparse sugiere que podría procesarse de forma eficiente, aunque habría que hacer mediciones específicas de rendimiento temporal para validar su viabilidad en aplicaciones de tiempo real. En mantenimiento predictivo, el alto recall que observamos (99.9 % en CalIt2) es crucial para evitar fallos no detectados, aunque la baja precisión requiere sistemas de post-procesado para reducir falsas alarmas.

El modelo híbrido es aplicable a escenarios de edge computing al reducir cómputo mediante sparsity, siempre que se complemente con:

- Módulos ligeros de post-procesado para consolidar alertas.
- Ajustes de umbrales adaptativos según la distribución reciente de actividad neuronal.

5.5. Limitaciones y consideraciones de validez

5.5.1. Limitaciones experimentales

- **Particionado 50/50**: La falta de validación cruzada temporal podría introducir sesgos en la generalización.

- **Estimación energética indirecta**: No medimos el consumo real en hardware neuromórfico (solo estimaciones analíticas de MACs).

- **Escalabilidad multivariante**: Los experimentos se centraron principalmente en señales univariadas (o pocas variables); falta evaluar en entornos fuertemente multivariados.

- **GPU**: Uno de los motivos para elegir BindsNET fue poder aprovechar la aceleración GPU; sin embargo, no conseguimos una configuración estable (incompatibilidades de dependencias en Windows), lo que aumentó la duración de los entrenamientos y limitó la exploración de hiperparámetros. Esto podría significar que no identificamos todas las combinaciones óptimas. Aún así, como todos los modelos se ejecutaron bajo condiciones homogéneas (CPU) y el preprocesamiento fue uniforme, la validez interna de la comparación se mantiene. Futuros trabajos en un entorno Linux con soporte CUDA habilitado permitirán ampliar la búsqueda y evaluar la robustez frente a una exploración más exhaustiva.

5.5.2. Consideraciones de validez

- **Interna**: La expansión de etiquetas podría inflar artificialmente el recall si no se aplica de forma consistente en inferencia.

- **Externa**: Los datasets (IOPS, CalIt2) puede que no representen toda la diversidad industrial (ciclos estacionales largos, variabilidad contextual).

- **Baselines**: Las comparaciones con TSFEDL se limitaron a cuatro modelos específicos (OhShuLih, KhanZulfiqar, ZhengZhenyu, WeiXiaoyan), sin acceso a mediciones de consumo energético de los baselines.

5.6. Conclusiones finales

En resumen, este trabajo ha demostrado que las arquitecturas híbridas SNN-CNN, aunque no superan consistentemente a los métodos de deep learning tradicionales en términos de F1-score, ofrecen un equilibrio interesante entre rendimiento y eficiencia computacional. La arquitectura propuesta logró F1-scores de 0.417 en CallIt2, manteniéndose competitiva en datasets moderadamente balanceados, mientras que en IOPS mostró mejoras notables respecto al modelo SNN base (4.6x) pero sigue teniendo limitaciones frente a métodos tradicionales en escenarios de desbalanceo extremo.

La optimización bayesiana con Optuna demostró ser una herramienta fundamental para ajustar bien las arquitecturas SNN complejas, permitiendo explorar sistemáticamente el espacio de hiperparámetros neuronales y convolucionales. El modelo presenta características arquitecturales que sugieren potencial para aplicaciones de edge computing, especialmente cuando se despliegue en hardware neuromórfico dedicado donde las ventajas energéticas de la sparsity pueden aprovecharse plenamente, aunque hace falta hacer mediciones específicas de rendimiento temporal en futuros trabajos.

La metodología desarrollada sienta bases sólidas para trabajos futuros en detección de anomalías con SNNs, proporcionando un pipeline reproducible de preprocesamiento, optimización y evaluación que se puede extender a otros dominios y arquitecturas neuromórficas más avanzadas. Estos hallazgos apoyan la hipótesis de que integrar mecanismos convolucionales y optimización sistemática reduce la brecha entre SNN y enfoques ANN tradicionales para detección de anomalías en series temporales.

5.7. Trabajos futuros

5.7.1. Evaluación energética y despliegue neuromórfico

Línea prioritaria para validar empíricamente las estimaciones de eficiencia ejecutando en hardware neuromórfico dedicado:

- **Plataformas candidatas**: Intel Loihi 2, SpiNNaker 2, BrainScaleS-2, chips basados en memristores emergentes.

- **Portado del modelo**: Conversión de parámetros LIF y topología a formato compatible (e.g. NxSDK / Lava para Loihi), sustituyendo operaciones convolucionales por proyecciones locales o *compartment groups*.
- **Métricas energéticas**: Joules por inferencia, mW promedio, latencia por ventana T, escalabilidad en número de neuronas vs. consumo.
- **Comparativa energética**: Medición directa frente a implementaciones GPU/CPU para cuantificar las ventajas reales del paradigma neuromórfico.

5.7.2. Optimización técnica y robustez

- **Validación cruzada temporal**: Implementar estrategias como rolling-origin o nested backtesting para hacer más robusta la estimación de generalización.
- **Optimización multiobjetivo**: Extender la optimización con Optuna a objetivos conjuntos (F1 y coste computacional/MACs), usando algoritmos como NSGA-II.
- **Datasets multivariantes industriales**: Incorporar señales de vibración, temperatura y presión para validar la aplicabilidad en mantenimiento predictivo real.
- **Robustez frente a drift**: Evaluar la degradación de rendimiento bajo concept drift y noise injection, implementando técnicas de continual learning específicas para SNNs.

5.7.3. Despliegue y monitorización neuromórfica

Integración específica para sistemas SNN en producción:

- **Monitorización de sparsity**: Tracking continuo de patrones de actividad neuronal para detectar degradación de eficiencia o drift de comportamiento.
- **Re-calibración de umbrales**: Ajuste automático de thresholds neuronales basado en la distribución reciente de spikes para mantener sensitividad sin re-entrenar completamente.
- **Despliegue edge**: Empaquetado optimizado para microcontroladores con soporte SNN nativo (e.g. neuromorphic inference engines) incluyendo cuantización de parámetros sinápticos.

- **Pipeline neuromórfico**: Versionado de topologías de red y pesos sinápticos con herramientas específicas para arquitecturas spike-based.