UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

DETECCIÓN E IDENTIFICACIÓN DE "SOFT FAILURES" EN ENLACES ÓPTICOS EN BASE A MACHINE LEARNING

POR

Hernán Andrés Orellana Espinoza

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil en Telecomunicaciones.

Profesor Guía
Gabriel Alejandro Saavedra Mondaca

Concepción,
5 de agosto de 2024

Agradecimientos

En primer lugar, quiero expresar mi agradecimiento a la vida por estos siete años de universidad, que han sido una experiencia inolvidable llena de altos y bajos, nuevas amistades y, sobre todo, valiosos aprendizajes que me guiarán y me permitirán crecer en mi vida profesional.

Agradezco profundamente a mi familia, especialmente a mis padres, Pedro y Uberlinda. Gracias a su esfuerzo, apoyo y confianza, pude superar las dificultades económicas y emocionales que conlleva ser estudiante universitario. Sin su respaldo, no habría logrado este objetivo de la misma manera. También quiero agradecer a mi hermana Daniela, quien, a pesar de la distancia, siempre estuvo a mi lado, especialmente en este último año. No puedo olvidar a mis familiares en Concepción, quienes me brindaron compañía, apoyo y buenos momentos durante mi tiempo como foráneo.

Un agradecimiento especial a mi amada Javiera, por su incondicional apoyo emocional durante los últimos años de mi carrera y por ser una compañera excepcional en la vida.

Quiero extender mi gratitud a mi profesor guía, Gabriel Saavedra, por su valioso apoyo y disposición para que pudiera escribir esta Memoria de Título en el menor tiempo posible. También agradezco a todos los profesores y empleados de la facultad, quienes contribuyeron a mi crecimiento académico y personal. Esta memoria de título contó con financiamiento a través del proyecto Fondecyt Regular 1231826.

Finalmente, pero no menos importante, agradezco a la gloriosa Tuna de la Facultad de Ingeniería de la Universidad de Concepción. Su espíritu colorido, sus viajes, fiestas y tradición han sido una parte fundamental de este proceso. Pertenecer a esta gran hermandad es una de las experiencias de las que me siento más orgulloso, y me ha brindado los mejores recuerdos de mi vida.

Gracias por todo

Resumen

Esta memoria de título se centra en la investigación, desarrollo e implementación de un algoritmo para la detección e identificación de Soft Failures en enlaces ópticos, utilizando técnicas de aprendizaje automático (Machine Learning) y algoritmos matemáticos. El objetivo principal es encontrar alternativas más eficientes y rápidas para llevar a cabo esta tarea. Para alcanzar este objetivo, se realizó una investigación exhaustiva sobre el origen de estos fallos, su comportamiento en el espectro óptico y las técnicas académicas actuales para abordar este problema.

El estudio se divide en cuatro partes: la primera se ocupa del montaje de un conjunto de laboratorio capaz de reproducir estos fallos; la segunda, de la creación de un conjunto de datos lo suficientemente amplio para realizar estudios, abarcando tanto datos experimentales como simulados. La tercera parte se enfoca en la detección de Soft Failures mediante el análisis de la variación del bit error rate (BER) y su comportamiento en escenarios normales y en presencia de anomalías. Finalmente, la cuarta parte aborda la identificación de un Soft Failure a partir del análisis del espectro óptico.

Un aspecto clave de este trabajo fue la elección de los modelos más adecuados, tanto para la detección como para la identificación, que permitan obtener resultados precisos y adaptarse a las diversas situaciones que se presentan en la práctica.

Los resultados de esta investigación indican que, en la etapa de detección, el modelo basado en autoencoder logra una precisión del 100 % al identificar datos anómalos en comparación con datos en condiciones normales de operación. Posteriormente, el modelo fue sometido a una revalidación al agregar ruido gaussiano a las muestras de BER normales y, al compararlo con otros modelos matemáticos, mostró una mayor sensibilidad y, por ende, un mejor rendimiento en presencia de estos nuevos datos. Finalmente, en la etapa de identificación, una red neuronal convolucional en una dimensión alcanzó una precisión del 100 % en la clasificación de Soft Failures y demostró un rendimiento superior frente a otros modelos de Machine Learning en distintos escenarios donde se modificaron algunos parámetros del enlace.

Abstract

This thesis focuses on the research, development, and implementation of an algorithm for the detection and identification of Soft Failures in optical links, using machine learning techniques and mathematical algorithms. The main goal is to find more efficient and faster alternatives for carrying out this task. To achieve this goal, a thorough investigation was conducted into the origins of these failures, their behavior in the optical spectrum, and current academic techniques for addressing this problem.

The study is divided into four parts: the first deals with setting up a laboratory environment capable of recreating these failures; the second involves creating a sufficiently large dataset for conducting studies, incorporating both experimental and simulated data. The third part focuses on detecting a Soft Failure by analyzing the variation in the bit error rate (BER) and its behavior in normal conditions and in the presence of anomalies. Finally, the fourth part addresses the identification of a Soft Failure through optical spectrum data analysis.

A key aspect of this work was the selection of suitable models for both detection and identification, which would yield accurate results and adapt to the diverse situations encountered in practice.

The results of this research indicate that, in the detection stage, the autoencoder-based model achieves 100 % accuracy in identifying anomalous data compared to data under normal operating conditions. Subsequently, the model underwent revalidation by adding Gaussian noise to the normal BER samples. When compared to other mathematical models, it showed greater sensitivity and, therefore, better performance in the presence of these new data. Finally, in the identification stage, a one-dimensional convolutional neural network achieved 100 % accuracy in classifying Soft Failures and demonstrated superior performance compared to other Machine Learning models in various scenarios where some link parameters were changed.

Índice General

Índice de Figuras

Índice de Tablas

Siglas

ASE Amplified Spontaneous Emission

BER Bit Error Rate

CD Chromatical Dispersion

CI Channel Interference

FO Fibra(s) Óptica(s)

FS Filter Shift

FT Filter Tightening

HF Hard Failure

IASEN Incremented ASE Noise

OSA Optical Signal Analyzer

OSNR Optical Signal Noise Ratio

PSD Power Spectral Density

ROP Received Optical Power

SF Soft Failure

SFD Soft Failure Detection

SFI Soft Failure Identification

1. Introducción

Las telecomunicaciones han emergido como protagonistas indiscutibles en la vida cotidiana de las personas a nivel mundial. En la era actual, el acceso a diversos sistemas dentro de este campo es crucial tanto para la industria como para el usuario promedio. Desde la introducción del internet en el siglo XX, se ha desatado una revolución sin precedentes en la sociedad moderna. Aproximadamente seis décadas después de su surgimiento en Estados Unidos, más del 60 % de la población mundial está conectada a la red [1], equiparando la conexión a internet desde dispositivos digitales con servicios básicos como la electricidad y el agua en nuestros hogares.

Para satisfacer la creciente demanda de datos a nivel global, las comunicaciones por fibra óptica se han convertido en la opción preferida de las grandes empresas debido a su capacidad para transportar grandes volúmenes de información a largas distancias sin necesidad de repetidores [2]. Este desarrollo ha impulsado significativas inversiones en infraestructura de redes ópticas, alcanzando cientos de millones de dólares. Por ejemplo, en Chile, se planea desplegar más de 14,800 kilómetros de fibra óptica para conectar América del Sur con Asia-Pacífico, con una inversión estimada de US 55 millones [3]. Sin embargo, este crecimiento también aumenta la complejidad y la probabilidad de fallos en los sistemas, representando pérdidas económicas y problemas legislativos. En respuesta, muchos países están implementando regulaciones para garantizar la conectividad ininterrumpida, lo que hace imperativo contar con sistemas de vigilancia de enlaces para detectar y prevenir fallos, asegurando la integridad y eficiencia de las redes de comunicaciones.

El avance de las redes ópticas elásticas ha incrementado la dinamicidad y flexibilidad de los sistemas de comunicación óptica, siendo esencial para las empresas de telecomunicaciones mantener la operatividad continua de estas redes para garantizar la calidad de servicio (QoS) y cumplir con contratos tanto con clientes como con entidades gubernamentales. No obstante, este progreso también ha aumentado la susceptibilidad a fallos en los enlaces, que pueden clasificarse en “Hard Failures” y “Soft Failures”. Los “Hard Failures” implican una interrupción abrupta del enlace, mientras que los “Soft Failures” afectan gradualmente la calidad o rendimiento de la red sin interrumpir completamente el servicio [4].

2. Definición del problema

El problema que aborda esta investigación es la falta de un basado en Software que pueda detectar e identificar un Soft Failure. Hoy en día, las empresas de telecomunicaciones, para garantizar que los recursos empleados en el enlace puedan tener un desempeño correcto, despliegan técnicos que revisan el estado del sistema inyectando una señal de prueba en el transmisor y leyendo el bit error rate (BER) en el receptor. Como sea el caso, este tipo de pruebas y análisis usualmente necesitan de una intervención humana ya que se necesita reiterar el proceso un número determinado de veces para determinar la causa del fallo, proceso que puede demorar días enteros [13]. Es por esto que se requiere de un modelo que pueda realizar un análisis y toma de decisiones para etapas de detección e identificación de un SF en basado en código computacional. La ausencia de una herramienta adecuada para detectar y diagnosticar estas fallas de manera oportuna puede llevar a tiempos de inactividad prolongados y costosos, así como a una disminución en la calidad del servicio para los usuarios finales, por lo tanto, la elaboración de un algoritmo que sea eficiente en este proceso es crucial.

2.1. Objetivos

2.1.1. Objetivo general

Diseño e implementación de algoritmos de monitoreo para detección e identificación de fallos en enlaces ópticos utilizando técnicas de Machine Learning.

2.1.2. Objetivos específicos

- Montaje en laboratorio de un enlace óptico de larga distancia
- Creación del set de datos para entrenamiento de algoritmos de Machine Learning.
- Desarrollo e implementación de autoencoders para la detección de “Soft Failures”.

- Desarrollo e implementación de redes neuronales convolucionales en una dimensión para identificación de “Soft Failures”.

2.2. Metodología, Alcances y limitaciones

2.2.1. Metodología

- **Objetivo:** Montaje de Setup de laboratorio que simule un enlace óptico de larga distancia. La instalación del setup de laboratorio será implementada a partir de la sección de la revisión bibliográfica en donde se exploran las diversas arquitecturas de estos. Para cumplir este objetivo, se debe tener en cuenta la disponibilidad de un computador, fibras ópticas, cables Ethernet, EDFA(s), VOA(s), un transmisor y receptor (Switch Cassini), analizador de espectro y filtros ópticos (WSS).

- **Objetivo:** Creación del set de datos para entrenamiento de algoritmos de Machine Learning.

  Para este objetivo primero se elaborará un código en Python utilizando la plataforma visual basic. Este código tiene contemplada la conexión mediante cable ethernet a un analizador de espectro óptico y un switch óptico “Cassini” disponibles en el laboratorio de Optoelectrónica de la Facultad de Ingeniería. La idea de conectar a ambos dispositivos al mismo tiempo es la recolección de datos en forma reiterativa y simultánea para tener una correlación entre ellos y analizar qué parámetros son útiles para el desarrollo de los posteriores algoritmos. Para completar este objetivo, luego de elaborar este código en Python es necesario obtener los datos necesarios para entrenar los modelos de Machine Learning, para esto se debe crear una base de datos lo suficientemente extensa para el set de entrenamiento, validación y prueba considerando escenarios óptimos de operación y escenarios en donde la calidad del enlace se vea reducida. Estos datos contendrán información sobre la variación del BER y la información del espectro óptico.

- **Objetivo:** Desarrollo y análisis de viabilidad de implementar autoencoders para la detección temprana de soft failures.

  Luego de haber cumplido el objetivo (2), ya con el set de datos obtenido, es necesario programar un código en lenguaje Python que tenga las librerías necesarias para la confección de un algoritmo de detección basado en autoencoders. Para esto, se utilizará la platafor-

ma Google Colab que soporta este lenguaje y tiene a disposición las GPU(s) de Google para el entrenamiento rápido de este algoritmo. Se evaluará el desempeño del autoencoder mediante las métricas revisadas en [4, 14].

- **Objetivo:** Desarrollo y análisis de viabilidad de redes neuronales convolucionales para la etapa de identificación de los soft failures.

  Al igual que para la etapa de detección, se utilizará la plataforma de Google Colab, con datos del espectro óptico para confeccionar un algoritmo basado en Redes Neuronales Convolucionales (CNN) y evaluar mediante métricas su precisión.

2.2.2. Alcances

Los aportes que puede entregar este trabajo son enumerados a continuación:

- **Mejora la fiabilidad de la red:** Al desarrollar un algoritmo efectivo que sea capaz de detectar e identificar soft failures, puede mejorar la fiabilidad de la red óptica al identificar y abordar problemas antes de que impacten en el servicio al cliente.
- **Reducción de tiempo de inactividad:** La detección temprana de problemas puede reducir significativamente el tiempo de inactividad de la red, lo que a su vez mejora la experiencia del usuario final y reduce los costos asociados con interrupciones no planificadas.
- **Optimización de recursos:** Al desarrollar estos algoritmos, se automatizan tareas que antiguamente las realizaban humanos, reduciendo pérdidas de dinero y tiempos de reparación.
- **Facilitación de la planificación a largo plazo:** El análisis de datos recopilados a través del monitoreo de Soft Failures puede proporcionar información útil para la planificación a largo plazo de la infraestructura de la red, ayudando a anticipar y mitigar posibles problemas futuros.
- **Implementación de software:** Con los algoritmos construidos y un set de datos extenso, puede ser posible la confección de un software que sea capaz de monitorear un enlace real para que se garantice la calidad de servicio.

2.2.3. Limitaciones

Las limitaciones que puede tener este estudio se enumeran a continuación:

- **Disponibilidad de Datos:** Si bien se planea trabajar con un set de datos extenso, la efectividad de este estudio puede depender en gran medida de la disponibilidad y calidad de los datos de telemetría óptica. Si los datos son incompletos, inexactos o difíciles de obtener, podría limitar la precisión y la utilidad de los resultados.
- **Implementación en situaciones reales:** El enfoque de este estudio se dirigirá hacia el desarrollo de algoritmos utilizando datos generados en un entorno de laboratorio. Esto implica que todos los componentes utilizados, como láseres, fibras ópticas y analizadores de espectros, son utilizados exclusivamente para simular fallos, no que sean elementos de la red que realmente estén fallando. Esta circunstancia podría presentar desafíos en la implementación práctica de estos algoritmos en situaciones del mundo real, donde se emplea una gama más amplia de dispositivos y equipos.
- **Disponibiliad y limitaciones de los equipos:** Es posible que el laboratorio cuente con equipamiento que no sea capaz de cumplir a cabalidad la recreación de Soft Failures. Es por esto que se debe explicitar qué tipos de fallos existen y cuáles fueron recreados exitosamente.

3. Marco Teórico

3.1. Introducción

Con el avance de las redes ópticas elásticas, la dinamicidad y flexibilidad de los sistemas de comunicaciones a nivel óptico están en constante aumento. La operatividad continua de estas redes es esencial para las empresas de telecomunicaciones, ya que garantiza la calidad de servicio (QoS) y previene problemas asociados con contratos tanto con clientes como con entidades gubernamentales. Sin embargo, este progreso tecnológico también ha hecho que las redes ópticas sean más susceptibles a fallos en los enlaces.

En términos generales, los fallos en la red pueden clasificarse en dos categorías principales: "Hard Failure" y "Soft Failures". Un "Hard Failure" implica una falla completa en algún enlace óptico, como un corte en la fibra óptica o un desastre natural que interrumpe abruptamente la comunicación entre nodos de la red. Por otro lado, un "Soft Failure" se refiere a un tipo de fallo que no interrumpe completamente el servicio, pero puede afectar la calidad o el rendimiento de la red, como la ausencia o mal posicionamiento de un componente. [4]. Haciendo una analogía con las redes públicas de agua potable y electricidad, es posible notar que cuando alguno de estos servicios está cerca de sufrir un corte, presenta ciertos "síntomas". Por ejemplo, en el sistema eléctrico se puede observar una caída progresiva del voltaje que puede culminar en un corte del suministro en una zona determinada, mientras que, en el sistema de agua potable, la disminución de la presión del agua en una red pública puede también derivar en un corte del suministro para varias viviendas. De manera similar, en las redes ópticas, es crucial contar con sistemas de monitoreo, análisis e identificación de posibles fallos potenciales, dada la importancia creciente de Internet para una amplia gama de aplicaciones y la gran contingencia que implica su uso.

La bibliografía cuenta con numerosas publicaciones que abordan este tema, donde se han implementado diversas técnicas, desde técnicas que utilizan el Machine Learning e incluso algoritmos matemáticos avanzados para detectar y posteriormente identificar las causas de posibles "Soft Failures". Por lo tanto, en esta revisión bibliográfica se realizará un análisis exhaustivo de la literatura disponible sobre esta temática, con el objetivo de identificar los métodos ya implementados y evaluar nuevas propuestas que puedan contribuir a la solución de este problema.

3.2. Definición de un Soft Failure

La literatura disponible aborda dos tipos de fallos en las redes: “hard failures” (HF) y “soft failures” (SF). Los HF, como cortes en fibras ópticas o interrupciones eléctricas, interrumpen la comunicación abruptamente y son fáciles de reparar. En contraste, los SF deterioran gradualmente el rendimiento de la red, y su detección es compleja porque pueden evolucionar a HF si no se tratan a tiempo [5]. La identificación temprana de fallos, como señalan [12], reduce los tiempos de reparación. Actualmente, para garantizar un buen desempeño del enlace, se inyecta una señal de prueba y se mide el BER, proceso que requiere intervención humana [13]. La detección automática de SF en un sistema de monitoreo continuo es esencial [8]. Además, representaciones del comportamiento del BER en presencia de SF ayudan a comprender mejor estos fallos [11].

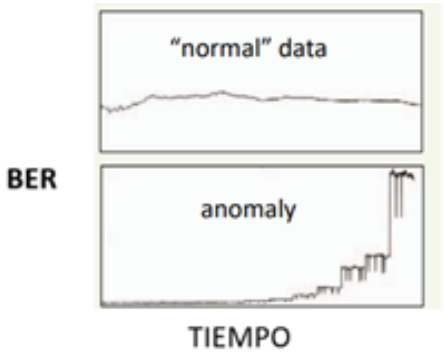


Fig. 3.1: Normal Data v/s Anomalía. Fuente: [11]

3.3. Definición de parámetros y monitoreo de enlaces

Para identificar cualquier tipo de “soft failure” que se manifieste gradualmente, es esencial medir ciertos parámetros ópticos que pueden indicar la presencia de dicha falla. A continuación, se definen los distintos parámetros que pueden ser medidos en una red óptica para facilitar el monitoreo y la detección de fallos:

- *Recieved Optical Power (ROP):* También conocido como potencia óptica RX, es la potencia óptica de recepción en la entrada del transceptor óptico. Sus unidades son en (dB) y la mayoría de los transceptores o medidores de potencia lo miden con una referencia de 1 [mW], por lo tanto, la lectura final sería en [dBm] [6].

- *Bit Error Rate (BER):* Según los autores en [7], es el parámetro clave a la hora de monitorizar las prestaciones de un sistema de comunicaciones digitales en general. Este parámetro fija la calidad de la señal digital y tiene un impacto directo en factores como el ruido, efectos no lineales y la dispersión. Los errores en los bits suceden cuando el encargado de detectar la señal convierte las señales ópticas en señales eléctricas.

- *Optical Signal-Noise Reason (OSNR):* Posteriormente en [7], se indica que las medidas de BER necesitan fotodetectar la señal en el caso de una red óptica, por lo tanto, requiere un parámetro alternativo conocido como OSNR. Este parámetro se define como la relación entre la potencia media de la señal en el espectro óptico y la potencia del piso de ruido distribuido en todo el ancho de banda medido. Sus medidas están en decibelios [dB], y su fórmula matemática es:

$$\text{OSNR [dB]} = 10 \log_{10} \left(\frac{S}{N} \right) \tag{3.1}$$

Donde S es la potencia media de la señal y N es la potencia media de ruido en el ancho de banda medido. La figura 3.2 indica gráficamente cómo calcula el OSNR un analizador de espectros ópticos (OSA) de manera general:

Fig. 3.2: Representación gráfica de una señal en un espectro óptico. Fuente: [7]

La potencia de la señal (S) se obtiene restando el valor peak del piso de ruido. El valor del piso de ruido de fondo puede obtenerse mediante interpolación cuadrática. El OSNR se relaciona con el BER y depende de la distancia y la modulación; mientras más amplificadores tenga la fibra, más ruido agregan y más afectan los valores de OSNR. La figura 3.3 representa la relación entre el BER, el SNR y los formatos de modulación de un transmisor óptico coherente con datos de laboratorio.

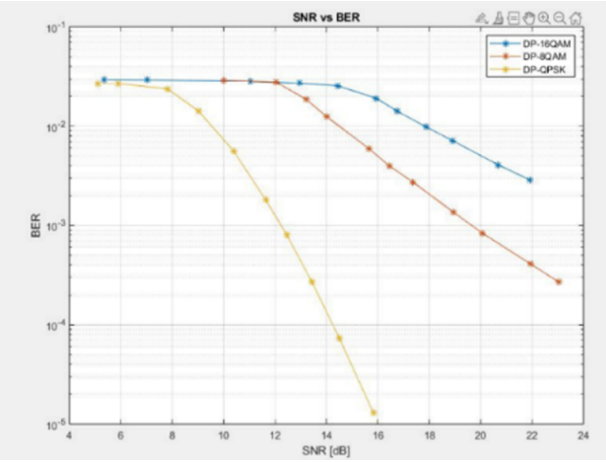


Fig. 3.3: Relación SNR vs BER para distintos formatos de modulación. Fuente: [8]

Monitorear la capa física es esencial para cumplir los SLA (Service Level Agreement) y, en caso de fallos o degradaciones (como fallos en el láser o mala configuración de los filtros), localizar los elementos defectuosos y tomar medidas para preservar los servicios. La información de monitores de potencia puede combinarse con la monitorización de transpondedores emergentes basados en detección coherente, que pueden medir parámetros como BER, pre-FEC BER, dispersión cromática y OSNR [15]. Según [4] y [12], el espectro óptico contiene mucha información sobre el estado de un enlace óptico. Además, el espectro digital obtenido mediante la Transformada Rápida de Fourier (FFT) en el muestreo digital es útil para detectar e identificar fallos suaves (SF), proporcionando información valiosa sobre la simetría del espectro y el nivel de ruido.

Definir los parámetros y formas de medición es vital ya que da contexto a la definición de cada tipo de SF en un enlace, lo que se detallará en la siguiente sección.

3.4. Tipos de Soft Failure

Los autores en [4] indican que los s.f. suelen tener su origen en el envejecimiento, desajuste o mal funcionamiento de los componentes de la red. A continuación, se definirá y representarán gráficamente, en el espectro digital, las cuatro fallas comunes de los SF.

- *Channel Interference:* En la Figura 3.4.a se describe una falla de tipo SF donde la interferencia entre canales ópticos causa problemas. Debido a efectos no lineales en la fibra,

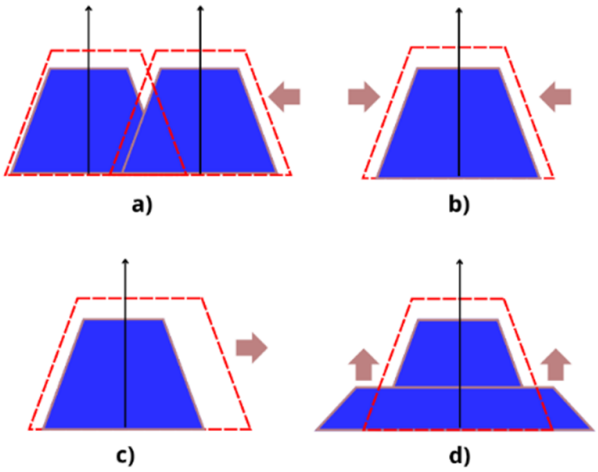


Fig. 3.4: a) Channel Interference, b) Filter Tightening, c) Filter Shift, d) IASEN. Fuente: Autoría propia.

espectros de señales distintas pueden solaparse. Según [4, 15], esta SF ocurre cuando la conexión óptica invade la de su vecino, mientras que [15] lo denomina *Signal Overlap*.

- *Filter Tightening (FT):* En la Figura 3.4.b se ilustra una falla conocida como "tight filtering", que sucede cuando los filtros ópticos se ajustan en exceso, provocando distorsiones en la transmisión de datos. Este problema surge durante el mantenimiento del sistema óptico. Según [4, 11], el "tight filteringçorta simétricamente ambos lados del espectro, agudizando el borde. [15] menciona esta falla de manera general como "tight filtering", relacionada con la imprecisión en la anchura de los filtros, abarcando cuatro tipos de fallas.

- *Filter Shift (FS):* En la Figura 3.4.c se presenta una falla conocida como "filter shift", causada por el desplazamiento o alteración de los filtros ópticos en el sistema. Estos filtros, cruciales para el procesamiento de señales, pueden distorsionar la señal si se desplazan, afectando la calidad de la comunicación óptica. [11] señala que el "filter shiftreduce la simetría del espectro óptico, cortando la energía del espectro de un lado, mientras que [4] confirma que un lado del espectro se filtra mientras el otro permanece intacto.

- *Increased ASE Noise:* En la Figura 3.4.d se trata el ruido de emisión espontánea amplificada (ASE), común en enlaces ópticos de larga distancia con amplificadores ópticos como los EDFA's [16]. Cada amplificador puede añadir ruido, elevando el piso de ruido y reduciendo el OSNR a medida que la señal avanza. Si un amplificador aumenta el ruido de forma gradual, se denomina Incremento del Ruido ASE (IASEN), lo que degrada la calidad de la señal, afectando el OSNR y el BER. Según [12], el IASEN eleva gradualmente el ruido del espectro mientras que el espectro de la señal permanece casi inalterado.

En resumen, las publicaciones [4, 11, 12, 15] desarrollan esta problemática con una mayor profundidad para CI, FT y FS. Mientras que en [11, 12] se ahonda un poco más acerca del incremento del piso de ruido o IASEN por sus siglas en inglés.

Existen otras deficiencias del enlace que afectan directamente a parámetros como la potencia recibida, dispersión cromática, BER y OSNR y tienen un comportamiento ruidoso a través del tiempo.

- **Ruido de amplitud (Intensity Noise)** Variaciones en la corriente entregada al láser puede desencadenar en cambios de temperatura dentro de un láser, lo que finalmente termina el variaciones rápidas en potencia de salida, pequeños desplazamientos en frecuencia (longitud de onda) y umbral de emisión del láser [30]. Esto directamente en el espectro óptico no tiene efectos significativos, sin embargo en el tiempo afecta directamente al BER y al OSNR por ese cambio en potencia.

  Es posible modelar este tipo de comportamiento ruidoso añadiendo un tipo específico de ruido llamado ruido Gaussiano aditivo circularmente simétrico complejo con media cero. Para enlaces de transmisión larga distancia que usan detección coherente y no compensan la dispersión durante la transmisión, esta modelación es válida. [31]

3.5. Revisión Bibliográfica

3.5.1. Dual-Stage Soft Failure Detection and Identification for Low-Margin Elastic Optical Network by Exploiting Digital Spectrum Information [4]

En [4], el proceso se divide en dos etapas: “Soft Failure Detection” (SFD) y “Soft Failure Identification” (SFI).

SFD:

1. **SFD-I**: Monitorea el pre-FEC BER y el ROP para detectar anomalías usando un algoritmo basado en distribución gaussiana. La calidad del algoritmo se degrada si los datos no siguen una distribución gaussiana, pero es suficiente para enviar una alarma.

2. **SFD-II**: Usa un algoritmo de Machine Learning (OC-SVM) para distinguir fallos verdaderos de falsos, analizando el espectro óptico.

Se evaluó el rendimiento usando las métricas FPR (False Positive Rate) y FNR (False Negative Rate):

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{3.2}$$

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} \tag{3.3}$$

Se obtuvieron valores de 0.42 % y 1.47 % respectivamente, tomando en cuenta el ROP, pre-FEC BER, y el nivel de ruido de la señal óptica.

SFI:

1. Se trata como un problema de clasificación multiclase usando SVM.
2. El SVM se entrenó con datos del espectro digital (70 % de 720 muestras) para cuatro tipos de fallos: FS, FT, IASEN y CI. El otro 30 % se utilizó para pruebas.
3. La precisión del modelo se optimiza ajustando el parámetro C, logrando cerca del 99 % de precisión con datos de ROP, pre-FEC BER, y el nivel de ruido.

Datos Importantes:

- Para SFD-II y SFI, se necesitan entre 6000 y 9000 puntos para el cálculo de la FFT para alcanzar el 100 % de precisión.

Desafíos y Conclusión:

- La obtención de datos confiables para los modelos de ML es un desafío para la implementación práctica en redes ópticas reales.
- Se destaca la importancia de un modelo YANG para centralizar la información y contar con un dataset mayor y confiable.

3.5.2. Machine-Learning-Based Soft-Failure Detection and Identification in Optical Networks [11]

En [11], al igual que en [4], el proceso se subdivide en dos etapas: DET-F (detección) e IDENT-F (identificación). La etapa de detección se basa únicamente en el valor de BER, utilizando algoritmos de Machine Learning como Binary Support Vector Machine (SVM), Random Forest (RF), Multiclass SVM y una red neuronal de una capa escondida. Estos algoritmos son semi-supervisados, entrenados solo con datos normales de BER.

Para la etapa de detección, se define un tiempo de muestreo T_{BER} de 22 segundos, con el Binary-SVM mostrando el mejor rendimiento entre 22, 44, 66, 88 y 110 segundos.

En la etapa de identificación, se mantiene el BER como parámetro y se reduce T_{BER} de 22 a 3 segundos para mejorar la capacidad de identificación de una red neuronal multicapa. La ventana total de tiempo para lograr buenos resultados en precisión es de 15 minutos.

3.5.3. Learning from the Optical Spectrum: Soft- Failure Identification and Localization [Invited] [12]

En [12], el estudio se enfoca en el uso de analizadores de espectro ópticos (OSA) para la detección de soft failures (SF), destacando la importancia del análisis espectral para mejorar el desempeño de la red. El dataset se construye a partir de pares frecuencia-potencia, extrayendo características como simetría, nivel de ruido y frecuencia central. Las herramientas de Machine Learning (ML) mejoran la eficiencia operativa del algoritmo.

Un módulo de clasificación, Signal Spectrum Verification (SSV), se utiliza para analizar las características espectrales y detectar desconfiguraciones. Este clasificador multiclase en forma de árbol de decisiones categoriza los estados en: Normal, desplazamiento del láser o fallo del filtro. Los estados fuera de “Normal” se consideran SF en potencia. En caso de “fallo del filtro”, un clasificador SVM adicional determina si es causado por SF conocidos como FS o FT.

El estudio también aborda la clasificación de SF relacionados con filtros. Se desarrolla el algoritmo FEELING (FailurE cause Localization for optIcal NetworkinG) para distinguir entre fallos reales y efectos normales derivados de filtros en cascada. Aunque la detección funciona, la identificación puede enfrentar problemas debido a la dificultad de los algoritmos de ML para

distinguir entre fallos reales y señales deterioradas.

Los resultados indican que los clasificadores SVM ofrecen una precisión del 95 %. Sin embargo, con el aumento del número de elementos en la red (como WSS), se requiere una mayor complejidad en el algoritmo para mantener el rendimiento.

3.5.4. Soft Failure Localization in Elastic Optical Networks [13]

En [13], se aborda la detección e identificación de fallas en una red de 4 nodos, en lugar de un enlace punto a punto. El objetivo es detectar, identificar y localizar el soft failure (SF) en los nodos de la red.

Para la detección de fallas, se mide el BER usando un dispositivo de bajo costo denominado "Optical Testing Channel" (OTC), que emite una señal de prueba y lee los valores inyectados. Los OTC se colocan en la salida de cada nodo: OTCTX en la entrada del nodo y OTCRX en los nodos intermedios y finales. El algoritmo "Testing optIcal Switching at connection SetUp time" (TISSUE) en el controlador de la red evalúa los valores de BER y determina cuáles son aceptables, estimando los elementos de la red responsables de BER excesivo.

Las características del espectro se extraen con un OSA en cada nodo intermedio (sin necesidad en el nodo de salida). Un módulo denominado FeX encuentra puntos relevantes en el espectro para calcular características como simetría y nivel de ruido. El algoritmo SSV analiza el espectro para detectar configuraciones incorrectas, y el algoritmo FEELING se utiliza para diferenciar entre SF reales y falsos.

3.5.5. A GAN Based Soft Failure Detection and Identification Framework for Long-Haul Coherent Optical Communication Systems[14]

En este estudio, para la detección se implementó un algoritmo de Machine Learning basado en una red neuronal Generative Adversarial Network (GAN), un algoritmo no supervisado capaz de identificar estados normales y la presencia de un soft failure (SF). El entrenamiento se realizó con muestras normales del enlace. El diagrama de flujo del proceso se muestra en la figura A.2 (Anexo).

Durante la detección, el algoritmo GAN utiliza un codificador para mapear muestras normales (x) a vectores latentes (z) y luego un decodificador reconstruye las muestras a partir de (z), produciendo salidas (x'). Estas salidas se vuelven a codificar a nuevos vectores latentes (z'). La red calcula la distancia euclidiana entre (z) y (z'): una distancia grande indica la presencia de un SF. Tanto la codificación como la decodificación son realizadas por una red neuronal convolucional unidimensional, y se utiliza la PSD residual para entrenar la GAN.

Para la identificación, se utilizó un algoritmo de ML semi-supervisado basado en clasificación Gaussiana (GPC), que muestra alta precisión con pocas muestras de SF. La red neuronal tiene dos capas escondidas con 5 neuronas, usa la función de activación "Sigma" y la optimización se realiza con "Adam". Los parámetros son el ancho de banda a -3 y -6 [dB] y las pendientes (slopes).

En la evaluación, se usaron las métricas de False Positive Rate (FPR) y True Positive Rate (TPR), definida como:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{3.4}$$

La etapa de identificación logró una precisión del 97 %.

3.5.6. BER Degradation Detection and Failure Identification in Elastic Optical Networks [15]

En [15], se desarrollan algoritmos para detectar e identificar soft failures (SF) basados en cambios en el pre-FEC BER para cuatro tipos de SF: CI, FT, FS y cyclic drift. El algoritmo "BER anomaly Detection" (BANDO), una máquina de estados finitos, detecta cambios significativos en el BER y emite notificaciones. La identificación de fallos se realiza con el algoritmo "Failure Identification Algorithm" (LUCIDA), que analiza patrones en el BER y la potencia recibida (ROP) para estimar la causa del fallo. Los algoritmos están descritos en las figuras A.3 y A.4 del anexo.

3.5.7. Confidentiality-preserving machine learning algorithms for soft-failure detection in optical communication networks [18]

En [18], se propone un método para garantizar la confidencialidad en sistemas de telemetría mediante la codificación de características y se comparan cuatro algoritmos de aprendizaje automático para detectar fallos suaves basados en telemetría codificada. El enfoque de detección se basa en un algoritmo de clasificación binaria que identifica si hay un fallo o no, utilizando datos normales para el entrenamiento y estimando umbrales lineales para permitir hasta un 5 % de errores de clasificación. Se recolectan parámetros como OSNR y pre-FEC BER cada 5 segundos, y se comparan los algoritmos FA, PCA, NLPCA y SVD en modo no supervisado, eligiendo los más efectivos y describiendo su funcionamiento teórico. Las curvas ROC muestran que PCA y SVD son los más precisos, con PCA destacando en escenarios reales al trabajar con datos de telemetría en bruto.

3.5.8. Soft Failure Identification for Long-Haul Optical Communication Systems Based on One-Dimensional Convolutional Neural Network [5]

En [5], se aborda la identificación de SF utilizando una red neuronal convolucional (CNN) combinada con un receptor DSP para extraer información adicional del espectro. El estudio se diferencia por su enfoque en la capacidad de la señal PSD para identificar cuatro tipos de fallos suaves.

Se emplea una arquitectura convencional de CNN, con capas convolucionales y de agrupamiento, para extraer características relevantes. La CNN realiza una correlación entre la PSD de la señal y un kernel deslizante con parámetros entrenables. La figura A.5 (Anexo) muestra un diagrama de flujo del proceso de identificación de SF.

Los fallos suaves se dividen en dos etapas. La primera determina si el WSS es la causa del fallo. Si se identifica un efecto relacionado con filtros, la segunda etapa clasifica el fallo como FS o FT. Si no hay relación con filtros, se investiga si está asociado con un aumento del piso de ruido (IASEN). Ambas etapas permiten identificar la causa principal (filtro o no) y secundaria (FS, FT, NLI o IASEN). La arquitectura de la CNN se detalla en la figura A.6 y su funcionamiento en la sección de algoritmos de Machine Learning.

El dataset comprende 1425 muestras, con un 80 % para entrenamiento y un 20 % para pruebas. La precisión de la CNN en la identificación de fallos es muy alta. En el peor de los casos, la precisión para la primera etapa es superior al 97 %, y para la segunda etapa, alcanza el 96 %. El algoritmo puede lograr una precisión del 100 % cuando ocurre un solo fallo suave a la vez.

3.6. Demostraciones Experimentales

A continuación se muestran las demostraciones experimentales encontradas en la bibliografía. Se muestran con sus fuentes respectivas y sus figuras encontradas en el anexo para un mejor análisis.

Est.	**Largo**	**F. Mod**	**Vel.**	**Symbol Rate**	**Ancho de Banda**	**Figura**
[4]	NE	PDM-QPSK	NE	20 [GBaud]	30 [GHz]	B.1
[11]	380 [Km]	PM-QPSK	100G	30 [GBaud]	NE	B.2
[14]	1600 [Km]	DP-16-QAM	NE	35 [GBaud]	50 [GHz]	B.3
[18]	280 [Km]	PM-QPSK	100G	NE	37.5 [GHz]	B.4
[19]	Sim: 880 [Km]	DP-16-QAM	NE	35 [GBaud]	NE	B.5

Tabla 3.1: Setup(s) tipo utilizados para la simulación de soft failures.

3.7. Modelos Matemáticos para gestión de fallos

En esta sección se explorarán 3 modelos matemáticos que servirán para análisis en el proceso de detección de fallos. Se toma en cuenta la información encontrada en [29]

3.7.1. Valor medio

El valor medio de un vector con N valores se calcula como la suma de todos los valores dividida por el número total de valores. La fórmula matemática para el valor medio (también conocido como la media aritmética) es:

$$\bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{3.5}$$

Donde $\bar{x}$ representa el valor medio, x_i es el i-ésimo valor del vector, y N es el número total de valores en el vector.

Teniendo en cuenta que la variación de BER en el tiempo es un vector con una cantidad N de elementos. Se puede intuir a partir de esto que valores medios para estados normales de operación de un enlace tenderán a un valor, y valores con fallos, tenderán a otro. Sería posible elegir una métrica a partir de esto para clasificar binariamente datos normáles versus anómalos.

3.7.2. Autocorrelación

La **autocorrelación** mide la correlación entre una serie temporal y una versión desplazada de sí misma. Se define como el coeficiente de correlación entre los valores de la serie en el tiempo t y los valores en el tiempo $t + k$, donde k es el desfase.

La fórmula para el coeficiente de autocorrelación r_k para un desfase k es:

$$r_k = \frac{\sum_{i=1}^{N-k}(x_i - \overline{x})(x_{i+k} - \overline{x})}{\sum_{i=1}^{N}(x_i - \overline{x})^2} \tag{3.6}$$

donde x_i son los valores de la serie temporal, $\overline{x}$ es la media de la serie, y N es el número total de observaciones.

Se pueden obtener varios patrones a partir de esto, para una línea recta paralela (valores normales de BER, por ejemplo) al eje X, se tiene que su valor de autocorrelación es cercana a 0 y una linea ascendente (valores anormales de BER) tienen valores que se acercan al 1.

3.7.3. Area Bajo la curva

Considera un conjunto de puntos discretos (x_i, y_i) donde x_i son las coordenadas en el eje horizontal y y_i son los valores de la función en esos puntos. Estos puntos están ordenados de tal manera que x_i x_{i+1}. El área bajo la curva para una función discreta se puede aproximar sumando las áreas de los trapecios formados entre cada par de puntos sucesivos. Matemáticamente, si tienes n puntos discretos, la suma de las áreas de estos trapecios puede ser expresada como:

$$\text{Área bajo la curva} \approx \sum_{i=1}^{n-1} \frac{1}{2} \left((x_{i+1} - x_i) \cdot (y_i + y_{i+1}) \right) \tag{3.7}$$

Si se calcula el area bajo la curva para valores anómalos y valores normales de BER, se pueden obtener resultados en donde el área para valores normales es menor que para valores anómalos, por ende se puede elegir una métrica en base a un umbral de area bajo la curva para hacer un modelo de clasificación binaria entre valores anómalos y normales de operación.

3.8. Machine Learning en gestión de fallos

La mayoría de los estudios revisados utilizan algoritmos de aprendizaje automático (ML) para la detección e identificación de fallas, por lo que es crucial entender sus conceptos teóricos. El ML, un área de la inteligencia artificial, desarrolla modelos predictivos a partir de datos, permitiendo a las computadoras aprender sin programación explícita, con aplicaciones en diversos campos. Sin embargo, presenta riesgos como la introducción de ruido y predicciones incorrectas con datos de baja calidad, por lo que los datos de entrenamiento deben ser adecuados [19]. Según [17], manejar fallas en redes ópticas es crítico y los algoritmos de ML son muy efectivos para monitorear, detectar e identificar fallos.

3.8.1. Conceptos Teóricos del Machine Learning

A continuación, se revisan los conceptos clave del Aprendizaje Automático para entender mejor los algoritmos que se usarán, evitando considerarlos como una "caja negra".

Según [17] y [20], se presentan cuatro tipos de aprendizaje en ML relevantes para esta temática:

- **Aprendizaje supervisado:** Usa ejemplos de entrenamiento con respuestas correctas para generalizar y clasificar entradas. Los datos se dividen en "features" (entradas) y "labels" (salidas), pudiendo ser problemas de regresión o clasificación, como la clasificación de fallos a partir del BER.
- **Aprendizaje no-supervisado:** Identifica similitudes entre datos no etiquetados, agrupándolos según patrones comunes ("clustering"). Se utiliza para detectar fallos mediante análisis histórico de parámetros como el BER.

En [20], se describe el proceso de Machine Learning de la siguiente manera:

1. **Recolección de datos y preparación:** Es crucial recopilar datos relevantes, asegurando su precisión y consistencia. A veces se usan bases de datos existentes, otras veces se recopilan nuevos datos, siempre garantizando que sean suficientes y limpios para el algoritmo.

2. **Selección de características:** Se identifican las características más útiles para el problema, requiriendo un conocimiento profundo para seleccionar la información relevante y excluir la innecesaria.

3. **Selección del algoritmo:** Se elige el algoritmo adecuado después de estudiar su funcionamiento y arquitectura, basado en el dataset preparado.

4. **Selección de parámetros y modelos:** Los parámetros se ajustan mediante "prueba y error" para optimizar el modelo, experimentando hasta encontrar los valores que den mejores resultados.

5. **Entrenamiento:** Utiliza los recursos computacionales disponibles para entrenar el modelo con el dataset y parámetros seleccionados.

6. **Evaluación:** Se evalúa la precisión del sistema usando métricas apropiadas antes de su implementación.

El libro también destaca conceptos y herramientas clave para evaluar el rendimiento de un algoritmo de ML:

- **Overfitting:** Ocurre cuando un modelo se ajusta demasiado a los datos de entrenamiento, fallando en generalizar a nuevos datos.

- **Entrenamiento, Prueba y Set de Validación:**

    1. **Entrenamiento:** El modelo aprende de los datos de entrenamiento.
    2. **Prueba:** Evaluación del modelo en un conjunto de datos independiente.
    3. **Set de Validación:** Datos adicionales para ajustar hiperparámetros y evitar el sobreajuste.

- **Confusion Matrix:** Tabla que muestra el rendimiento del modelo en términos de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.

- **Métricas de Precisión:** Incluyen precisión, sensibilidad (recall) y puntuación F1, evaluando el rendimiento del modelo en problemas de clasificación.

- **ROC Curve:** Representa la relación entre la tasa de verdaderos positivos y la tasa de falsos positivos, útil para evaluar modelos de clasificación binaria.
- **Precisión:** Mide la cercanía de las predicciones del modelo a los valores reales, aplicable tanto en regresión como en clasificación.

3.8.2. Tipos de algoritmos de ML orientados a la gestión de fallos

En [17] se realiza una extensa revisión de 73 artículos que exploran diversas técnicas de algoritmos de ML para la gestión de fallos, esto y sumando los resultados vistos en la revisión bibliográfica de las publicaciones científicas, se puede obtener la siguiente tabla resumen.

Algoritmo	**Categoría**	**Tipo**	**Resultados**	**Función**
Neural Networks	Detección (ANN)	Experimental	Prec:98 %	Detección de anomalías BER
	Identificaciónn (1D-CNN)	Simulación	Prec: 97 %	Detección de anomalías de BER.
Support Vector Machines	Detección (Binary SVM)	Experimental	Prec: 99 %	Detección de anomalías de BER.
	Identificación	Experimental	Prec:99 %	Identificación en fallos en filtros.
Gaussian Processes	Detección	Experimental	FPR: 0.47 % FNR: 1.47 %	Monitoreo de BER y ROP.
Network Kriging	Localización	-	-	Localización de la falla del enlace.

Tabla 3.2: Casos de uso para cada algoritmo de ML estudiado.[17]

De la tabla anterior se concluye que todos los algoritmos empleados en las distintas investigaciones obtienen buenos resultados, alcanzando sobre el 90 % de precisión, sin embargo todos tienen limitaciones en base a recursos computacionales empleados y la agregación de etapas intermedias para revalidación de resultados (como SFD-I y SFD-II visto en [4])

Finalmente, las Artificial Neural Networks (ANN) [17], inspiradas en el cerebro humano, están formadas por capas de neuronas artificiales conectadas. Ajustan iterativamente los pesos durante el entrenamiento para minimizar una función de pérdida y son versátiles en tareas como clasificación, regresión y reconocimiento de patrones. Se describe que una red neuronal artificial (ANN) está compuesta por neuronas organizadas en capas. Cada neurona recibe valores de la capa anterior, calcula una suma ponderada de estos valores y aplica una transformación no lineal (como sigmoide, ReLU o tanh) a las neuronas de la capa siguiente. Con suficiente cantidad de neuronas, una ANN puede aproximar cualquier función matemática con alta precisión, convirtiéndose en un aproximador universal. La figura proporcionada ilustra gráficamente el funcionamiento típico de una ANN.

1. **Autoencoders:** Según los autores en [21], los autoencoders son una red neuronal que

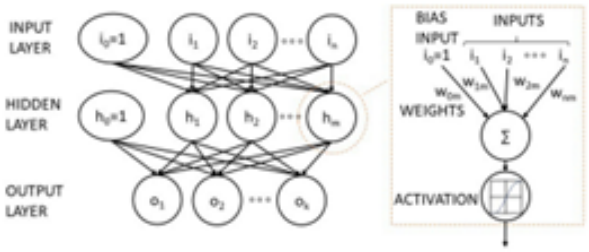


Fig. 3.5: Ejemplo de una red neuronal artificial. [17]

es entrenada para reconstruir los datos de entrada. Su propósito principal es entrenar de una manera no supervisada. En pocas palabras, este tipo de red neuronal aprende cómo se comporta un tipo de set de datos (una señal por ejemplo), la codifica y la aprende a reconstruir en base a una serie de iteraciones. La siguiente imagen representa gráficamente su uso, para un mejor entendimiento.

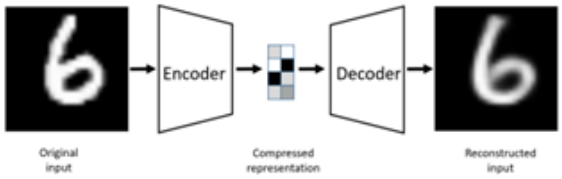


Fig. 3.6: Ejemplo de funcionamiento de un autoencoder. [21]

La parte codificadora y decodificadora de los autoencoders son redes neuronales, cuyas operaciones pueden ser lineales, no lineales y convolucionales. Existen varios tipos de autoencoders, como los dispersos, de eliminación de ruido, contractivos y variacionales, que tienen aplicaciones diferentes y pequeñas modificaciones en sus funciones. Los autoencoders son ampliamente utilizados para la detección de anomalías, una tarea no supervisada que consiste en aprender un perfil normal a partir de datos normales y luego identificar muestras que no se ajustan como anomalías. Una técnica para evaluar el rendimiento de las redes neuronales es la distancia euclidiana entre dos vectores.

En [23] se realiza un estudio comparativo de diferentes métodos de detección basados en la variación del BER en el tiempo, utilizando un algoritmo con una etapa dual de detección y otro basado solo en autoencoders. Los resultados muestran un 100 % de precisión del autoencoder para detectar fallas, tomando el TPR y el AUC como métricas de rendimiento.

2. **Redes neuronales Convolucionales (CNN):** La autora en [22], indica que la arquitectura de las redes neuronales convolucionales nace del modelo de retropropagación para el entrenamiento de redes neuronales profundas, que son capaces de distinguir unos objetos

de otros. Utilizan la convolución en al menos una de sus capas, una operación conmutativa que para datos discretos se define como:

$$s[t] = \sum x(a)w(t-a) \tag{3.8}$$

La entrada $s[t]$ es un vector o matriz multidimensional de datos, mientras que el filtro es un vector o matriz de parámetros ajustados durante el aprendizaje.

Resumen del funcionamiento y entrenamiento de una CNN:

- **Convolución:** Desliza un filtro sobre los datos de entrada, multiplicando valores con pesos del filtro para producir un mapa de características.
- **Capas convolucionales:** Aplican varios filtros al vector de entrada para extraer características en diferentes niveles de abstracción, seguidas de capas de activación no lineal, como ReLU.
- **Capas de agrupación (pooling):** Reducen la dimensionalidad de las características extraídas, preservando las más importantes (e.g., MaxPooling, AveragePooling).
- **Capas completamente conectadas:** Al final de la red, las características se aplanan y pasan a través de capas totalmente conectadas para clasificación o regresión.
- **Regularización y normalización:** Utilizan técnicas como regularización L2, eliminación (dropout), y normalización por lotes (batch normalization) para evitar el sobreajuste.
- **Funciones de pérdida y optimización:** Durante el entrenamiento, se emplea una función de pérdida para medir la discrepancia entre las predicciones del modelo y las etiquetas reales. La optimización se realiza con algoritmos como el descenso de gradiente estocástico (SGD) o sus variantes para ajustar los pesos de la red y reducir la pérdida.

Ahora, si bien los algoritmos de ML son eficientes para realizar las tareas de detección e identificación de anomalías, es necesario explicar otros algoritmos matemáticos simples que pueden ayudar de cierta forma a validar o no a estos modelos. Ya que explorar otras alternativas que utilicen menos recursos computacionales es necesario para obtener el mejor modelo.

4. Desarrollo

4.1. Introducción

Para resolver el problema en cuestión es necesario realizarlo de manera sistemática y ordenada, empleando las distintas herramientas tecnológicas que existen. Estos son: software de programación, computador, cables ethernet, fibras ópticas, amplificadores (EDFA's), filtros ópticos, atenuadores variables (VOA's), switch óptico y analizadores de espectro.

La idea es, a partir de la recolección de datos, procesamiento, entrenamiento y validación de las redes neuronales, definir un algoritmo que muestre paso a paso el proceso que se debe seguir en la detección e identificación de un soft failure.

4.2. Montaje de Setup de Laboratorio

La primera etapa en el tema de investigación se centra en la creación de las condiciones de laboratorio aptas para recrear un enlace óptico de larga distancia, y, además, que sea capaz de recrear las deficiencias en el enlace óptico (FS e IASEN).

Para esto es necesario contar con elementos que realicen la tarea de transmisión y recepción de la señal, además de EDFA's, atenuador variable, un WSS, fibras ópticas, cables ethernet, un switch de capa 2, y un computador para almacenar los datos. La elección de la longitud total del enlace se elige a partir de lo leído en [24]. Los enlaces ópticos en Chile van del orden de los kilómetros, siendo, como regla general, necesaria la colocación de amplificadores en longitudes que van desde los 80 a los 120 kilómetros, debido a los efectos de dispersión y atenuación que ofrece una fibra óptica monomodo. Típicamente, en un enlace óptico se suele colocar un WSS al final del enlace que funciona como multiplexador y demultiplexador de las señales para poder enviarlas por una sola fibra.

La siguiente figura detalla, en forma gráfica, la forma en que se debe montar el setup de laboratorio:

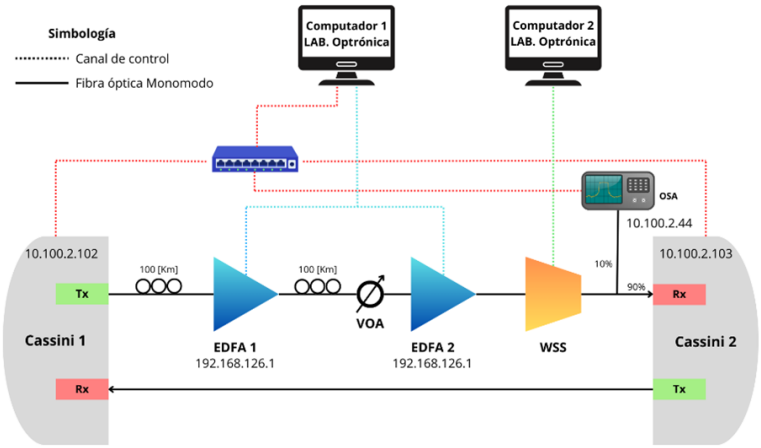


Fig. 4.1: Setup de laboratorio empleado. Autoría Propia.

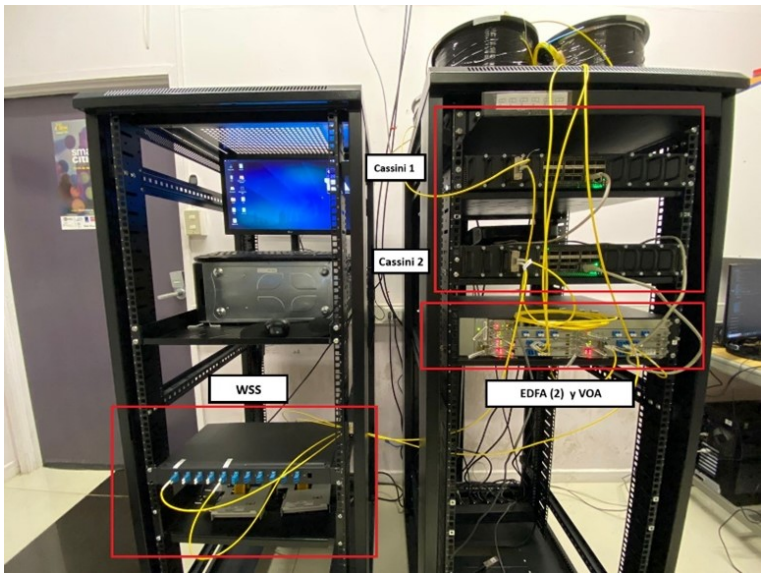


Fig. 4.2: Setup Montado en laboratorio. Autoría Propia.

La figura anterior muestra el setup montado en el laboratorio de optoelectrónoca, los elementos del enlace se montan en 2 rack. En la parte superior de la imagen se presentan dos Switch Cassini de óptica coherente, el Cassini tiene montado un DCO (transmitter) CFP2 DCO de 200G que soporta formatos de modulación como DP-QPSK, DP-8-QAM y DP-16-QAM (ver anexo C.1). En el anexo se encuentran las precauciones necesarias que se debe tener al momento de utilizar estos elementos, tales como la potencia máxima que transmiten, temperatura máxima de operación, entre otros. Es posible controlar estos Cassini realizando una conexión a la consola con un cable ethernet ya que tienen un sistema operativo, esto se puede hacer mediante un script de Python (ver anexo D.1) o mediante Putty para establecer, mediante línea de comandos, potencias de transmisión, datos de la transmisión, entre otros.

En la parte superior del rack se encuentran los 200 [Km] de fibra empleada para el montaje, cada span tiene una longitud de 100 [Km], y cada rollo de fibra tiene una longitud de 50 [Km], por lo tanto se emplean 4 rollos.

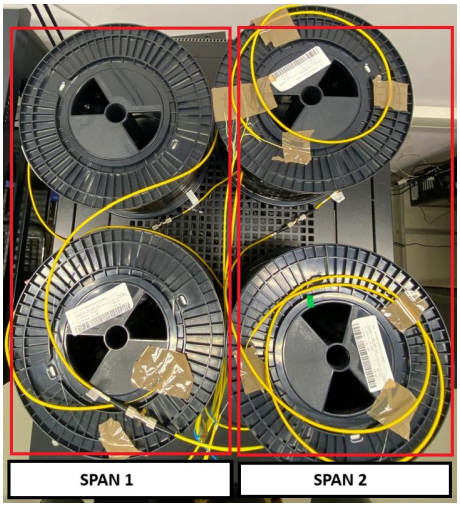
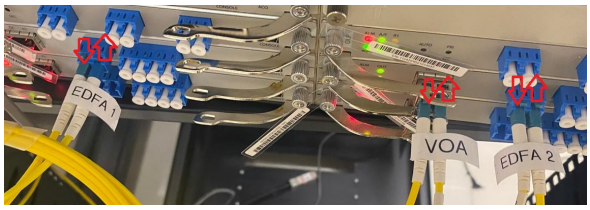


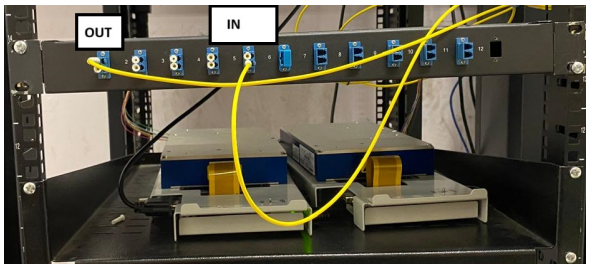
Fig. 4.3: Rollos de fibra utilizados. Autoría Propia.

Debajo de los dos switch Cassini se encuentra el Chassis 2U de FS que soporta hasta 7 módulos que se pueden conectar fácilmente, estos módulos pueden ser EDFAS, multiplexadores, etc. Notar que tiene montados 2 preamplificadores que se utilizarán en el setup de laboratorio y un atenuador variable que se utilizará para modificar el piso de ruido de la señal. Es posible controlar la ganancia de estos amplificadores y el módulo de atenuación del atenuador variable mediante la conexión de un cable ethernet a un computador, en el cual se puede manipular un software que modifica estos parámetros de atenuación y de amplificación, para un mejor entendimiento del uso de estos software, se debe seguir el manual de usuario que explica el paso a paso de la manipulación de los mismos [25].

A la izquierda del rack se encuentra el WSS, que, debido a su año de fabricación, solo tiene conexión con un computador mediante un cable de comunicación paralela bidireccional. El WSS cumplirá la función de filtro y se encargará de simular FS. Se conecta directo mediante un cable de fibra corto desde el EDFA. El manual de uso correcto de este dispositivo se encuentra en los textos [27, 28], que detalla a cabalidad el cómo configurar los canales de entrada y de salida y el ancho de banda que ocupará la señal que sale. Se realiza una configuración adecuada para que sea capaz de emular los fallos como Filter Shift por ambos lados del espectro, la siguiente figura detalla el trabajo esencial que cumplirá este elemento de la red.



(a) Conexión EDFA



(b) Conexión WSS

Fig. 4.4: Conexiones en laboratorio

Para obtener los datos necesarios, es necesario la utilización de elementos adicionales de red como switch de capa 2, cables ethernet, cables de comunicación paralela bidireccional y dos computadores, esto con la finalidad de configurar los parámetros de transmisión del transmisor y para obtener datos como el BER, OSNR, variación de la potencia, información del espectro, etc.

A continuación se muestra una tabla resumen con todos los elementos de laboratorio utilizados.

Elemento	**Cantidad**	**Pérdida por inserción**	**Datasheet**
FO largas	4 x 50 [Km].	10-10.7 [dB] c/u	-
EDFA	2	-	C.2
VOA	1	-	C.3
DCO (transmitter)	2	-	C.1
WSS	1	5 [dB]	[27, 28] - C.4
Computadores	2	-	-
OSA	1	-	[26]
Cables Ethernet	4	-	-
Cable RS232	1	-	-
Splitter 90-10	1	0.8 [dB]	C.5
FO Monomodo	5 (Aprox.)	<0.3 [dB] c/u	N/a

Tabla 4.1: Resumen de elementos utilizados en el setup de laboratorio. Fuente: Autoría propia.

Además, en la siguiente tabla se detallan los parámetros con los que deben configurar los Cassini.

Parámetro	**Valor**
Potencia Tx (Cassini 1)	0 [dBm]
Potencia Tx (Cassini 2)	-4 [dBm]
Frecuencia Operación	194 [THz]
Formato de modulación	DP-16-QAM
Cantidad de Span (s)	2
Longitud de cada Span	100 [Km]

Tabla 4.2: Configuración de transmisión y propagación. Fuente: Autoría propia

La idea teórica, es que, con la ayuda de los EDFA, sin contar las pérdidas de inserción de cada dispositivo de la red, se deba recibir un total de 0 [dBm] en el Cassini 2.

Luego, se configura la ganancia de los amplificadores para que en cada span se salga con una potencia de 0 [dBm] y lleguen al Cassini 2 con la misma potencia de transmisión.

Elemento	**Modo de funcionamiento**	**Valor (configurable)**
EDFA 1	APC	1
EDFA 2	APC	1
VOA	-	0 (inicial)

Tabla 4.3: Configuración de EDFA y VOA. Fuente: Autoría propia.

La ventaja que existe al configurar los amplificadores en el modo de funcionamiento APC (Automatic Power Control), es que la potencia se mantiene constante mientras la ganancia se modifica automáticamente para cumplir con la potencia constante que se le configura, el valor en el software [25] se setea en 1 para que así la potencia de salida de cada EDFA sea igual a 0 [dBm], es decir, si por alguna razón la señal se atenúa a través del tiempo, la ganancia se ajusta automáticamente para que la potencia de salida siga siendo la misma, pero el piso del ruido aumente, en consecuencia el BER también.

El WSS se conecta directo mediante un cable de fibra corto desde el EDFA 2, el manual de uso correcto de este dispositivo se encuentra en los textos [27, 28]. Se realiza una configuración adecuada para que sea capaz de emular los fallos como Filter Shift por ambos lados del espectro, la siguiente figura detalla el trabajo esencial que cumplirá este elemento de la red.

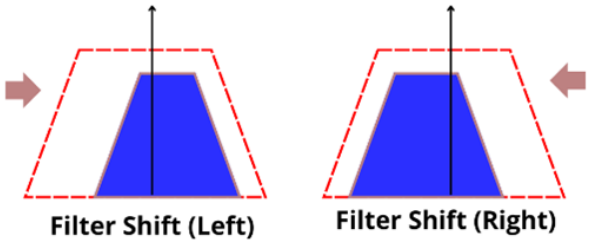


Fig. 4.5: Fallos relacionados a filtros. Fuente: Autoría propia.

Debido a falta de disponibilidad de láseres y el WSS con una resolución adecuada, fallos como el CI y FT no se considerarán en los experimentos.

Para obtener datos del espectro óptico, se conecta un OSA al enlace mediante un Splitter 90-10, 10 % de la potencia se va hacia el OSA, y el otro 90 % se va hacia finalmente hacia el Cassini 2.

Fig. 4.6: Espectro antes de llegar al Cassini.Fuente: Autoría propia.

De la figura anterior se concluye que el ancho de banda del láser es de aproximadamente 50 [GHz], lo que se ve alrededor de él es el piso de ruido generado por el EDFA. El WSS se configura de tal forma para que el ancho total de su espectro sea de 162,5 [GHz], dejando en el centro la portadora que emite el láser.

Dentro de las consideraciones importantes para el correcto uso del OSA [26], es necesario

respetar las potencias máximas que soporta la entrada (10 y 23 [dBm]), y en todo momento debe estar conectado mediante cable ethernet al computador para que obtenga los datos.

Luego, el Cassini 2 recibe la señal con una potencia de -5.19 [dBm] aproximadamente debido a la pérdida de inserción que produce el WSS. Pero esta pérdida no afecta los valores de BER y de OSNR ya que estos dependen de niveles de ruido y el peak de la señal. Se debe tomar en cuenta al final que ésta no será una potencia que se mantenga constante para tomar mediciones, debido a que se pueden añadir otros elementos de la red (como extensores de fibras opticas, por ejemplo) que añadan pérdidas de inserción.

Para extraer los datos del Cassini, así como también configurar los parámetros (tabla 4.2) se conecta con un cable ethernet a la consola, y mediante el programa "Putty" se introducen los comandos necesarios para visualizar los datos.

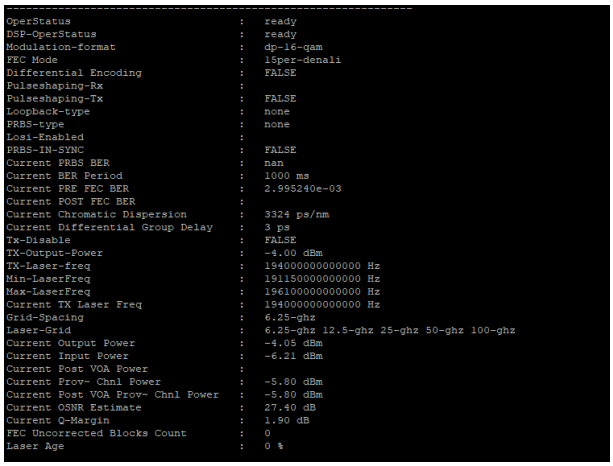


Fig. 4.7: Consola del sistema operativo del Cassini 2 (Rx). Fuente: Autoría propia.

De la figura anterior se desprende que la que la potencia recibida difiere con la entregada por el OSA anteriormente, esto se debe principalmente a que se agregó un extensor de fibra óptica que agrega pérdidas de inserción. El OSNR es de 30 [dB] y el BER aproximado es de $1{,}72 \times 10^{-3}$. Entonces, en condiciones normales de operación, el BER y el OSNR deben ser valores que giren en torno a esos números.

Ya con todos los equipos montados y operando, el siguiente paso es crear la base de datos necesaria para entrenar los algoritmos de Machine Learning.

4.3. Elaboración de Scripts para obtención de datos

La siguiente figura detalla un diagrama de flujo que explica de manera gráfica el proceso de extracción de información, preprocesamiento, creación de la base de datos y el entrenamiento de los algoritmos de Machine Learning.

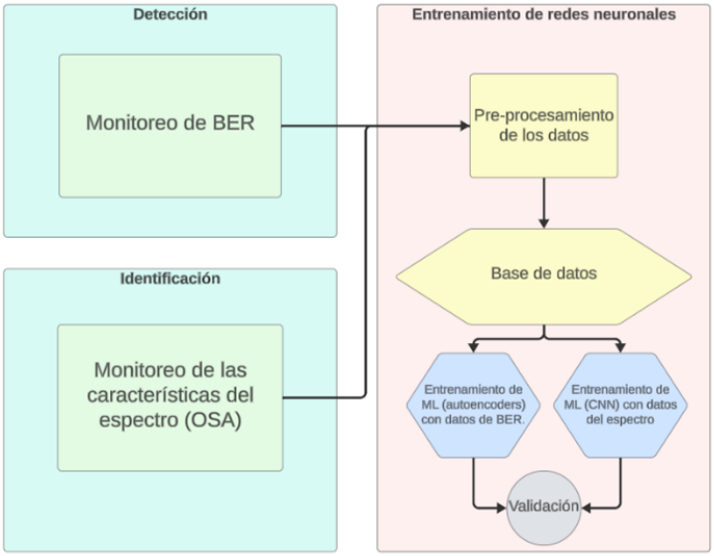


Fig. 4.8: Diagrama de flujo para el trabajo a realizar en la etapa de entrenamiento. Fuente: Autoría propia..

Esta etapa se centra en la recopilación de todos los parámetros necesarios para entrenar las redes neuronales. Según la bibliografía consultada, para lograr resultados óptimos, es fundamental disponer de conjuntos de datos de entrenamiento, prueba y validación lo suficientemente amplios. Por lo tanto es crucial recopilar tanto datos "normales" como "anómalos". Los datos normales representan condiciones óptimas de la red, sin presencia de fallas, mientras que los datos anómalos reflejan fallos progresivos en un intervalo de tiempo. Esto implica definir una ventana temporal, denominada T_w y una frecuencia de tiempo con la que se recopilan los datos, denominado frecuencia de muestreo T_m. Para esto, es necesario recopilar datos como el BER, OSNR y/o dispersión cromática, además de toda la información espectral.

Como se definió en la sección de "Montaje de Setup de Laboratorio", los datos de BER y otros parámetros, se deben extraer de un dispositivo de la red que recepcione la señal, el cual en este caso es el Cassini 2.

La información espectral se extraerá de un analizador de espectros ópticos (OSA), el cual,

al igual que el Switch Cassini, puede conectarse y proporcionar estos parámetros mediante comandos. Para automatizar el proceso de extracción de datos, se desarrolla un script de Python que hace posible la interacción con estos dispositivos y extraiga la información necesaria para su almacenamiento en una base de datos.

La plataforma Visual Basic será la utilizada para elaborar este script que realizará la tarea de controlar un analizador de espectros óptico (OSA) y un switch Cassini óptico. Para esto, se necesita estudiar a fondo el tipo de protocolo de conexión que utilizan mediante cable ethernet. El analizador de espectros utiliza un protocolo llamado "Pyvisa" y el switch Cassini un protocolo SSH. En el anexo (Anexo D.1) se adjunta el código en Python realizado para la conexión de ambos dispositivos al mismo tiempo.

Este código en Python realiza mediciones simultáneas en un dispositivo Cassini y un analizador de espectro OSA (Anritsu Optronics o Cassini) durante un período de tiempo especificado. Veamos el código paso a paso:

1. Importación de bibliotecas:
    - **paramiko:** para la comunicación SSH.
    - **re:** para el uso de expresiones regulares.
    - **time:** para la manipulación del tiempo.
    - **pandas as pd:** para el manejo de datos estructurados.
    - **datetime:** para manipular objetos de fecha y hora.
    - **os:** para operaciones del sistema operativo.
    - **pyvisa:** para la comunicación con instrumentos de medición a través de VISA (Virtual Instrument Software Architecture).
    - **numpy as np:** para operaciones matemáticas.
    - **matplotlib.pyplot as plt:** para la visualización de datos.
2. Función "Main()"
    - Define la dirección IP del OSA (ip_osa), la duración de la medición en segundos (duration), y el tiempo de muestreo (tiempo_muestra).
    - Llama a la función medicion_conexion() con los parámetros necesarios.
3. Función medicion_conexion(ip_address, ip_osa, cassini, duration, tiempo_muestra):

- Inicializa variables para almacenar los datos de las mediciones.
- Se conecta a través de SSH al dispositivo Cassini y al OSA.
- Dentro de un bucle while, realiza mediciones cada cierto tiempo hasta que se alcance la duración especificada.
- Realiza acciones como enviar comandos SSH al Cassini y al OSA para obtener datos de mediciones y almacenarlos en variables.
- Utiliza expresiones regulares para extraer datos específicos de la salida de los comandos SSH.
- Guarda los datos en archivos Excel y realiza gráficos en tiempo real de las mediciones.
- Finalmente, guarda todos los datos recopilados en un DataFrame de pandas y los exporta a un archivo Excel.

4. Al final, se ejecuta la función main() para iniciar el proceso de medición.

En resumen, este código automatiza el proceso de recopilación de datos de mediciones simultáneas en un dispositivo Cassini y un OSA, los almacena en archivos Excel y genera gráficos para visualizar los resultados en tiempo real.

La idea detrás de obtener una recopilación de mediciones simultáneas entre dos dispositivos es que se pueden relacionar fácilmente entre sí. El Cassini proporciona parámetros como el pre-FEC BER, el OSNR y la dispersión cromática, los cuales, gracias al código, son extraídos en una ventana de tiempo T_w dada y un tiempo de muestreo T_m definido, todos estos datos posteriormente son guardados en un archivo Excel. El OSA suministra la información espectral necesaria. El código captura estos dos conjuntos de datos (los parámetros del Cassini y el espectro del OSA), generando así dos archivos que están correlacionados en el tiempo y pueden ser procesados conjuntamente. Esta metodología presenta la ventaja adicional de permitir la recolección continua de datos a lo largo del tiempo, lo que es crucial para obtener la base de datos necesaria para el entrenamiento de algoritmos de Machine Learning. Esta práctica ahorra tiempo significativo, ya que permite la medición continua y simultánea con ambos dispositivos en lugar de hacerlo por separado, lo que reduce considerablemente el tiempo de trabajo.

4.4. Construcción del set de datos para entrenamiento de modelos

Luego de haber elaborado los scripts necesarios para la medición automática de parámetros del enlace, se puede dar paso la implementación de estos códigos.

4.4.1. Detección

Para esta primera etapa de medición, se debe construir un data set que muestre la variación de algún parámetro en el tiempo. Para este estudio se involucra el BER como parámetro a medir. Se define una ventana de tiempo T_w de 10 minutos, y un tiempo de muestreo de 8 segundos entre cada muestra. Esto resulta en un vector de BER (V_{BER}) que represente la variación del mismo en esa ventana elegida. Cada vector de BER tendrá una longitud de:

$$\dim(V_{\text{BER}}) = \frac{600\,[\text{S}]}{8\,[\text{S/muestra}]} = 75[\text{muestras}] \tag{4.1}$$

Este proceso debe iterarse reiterativamente para crear un data set robusto de información. Luego de la primera iteración, se concatena la siguiente como una segunda fila con 75 muestras más, y así sucesivamente hasta construir el data set necesario. A continuación se muestra una figura que detalla de manera gráfica sobre cómo están construidos estos set de datos.

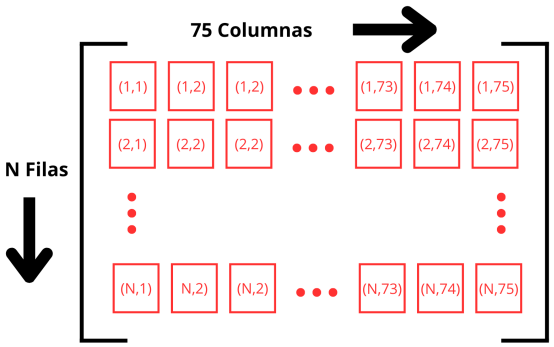


Fig. 4.9: Estructura de la base de datos para entrenamiento de algoritmo de detección. Fuente: Autoría Propia

Cada matriz (Base de datos) construida se almacenará en un archivo .CSV para luego

utilizarlo en el entrenamiento de algún modelo. Se construye una base de datos lo suficientemente extensa para que el 80 % de estas muestras sean de entrenamiento, 20 % de pruebas, y se creará además otro set para validación. En el set de entrenamiento y validación se utilizarán solo datos "normales".

Para la construcción del dataset, se debe de seguir el mismo patrón visto en la figura 3.1. Se debe repartir la base de datos entre datos "normales" de operación y datos "anómalos".

4.4.1.1. Datos Normales

Los datos normales se construyen cuando no existe ninguna condición de fallo en el enlace óptico, es decir, cuando tanto el amplificador óptico como el filtro óptico no presentan cambios. Entonces, basta con solo ejecutar los scripts vistos en la sección anterior y dejar midiendo el tiempo que se requiera necesario para construir esta base de datos.

En la bibliografía se indica que para entrenar un algoritmo no supervisado, se necesitan de pocos datos en estado normal para entrenamiento. Es por esto que luego de realizar este proceso de medición por un tiempo aproximado de dos semanas de manera continua, se logró recolectar un total de 2003 datos que, normalmente tienen esta forma:

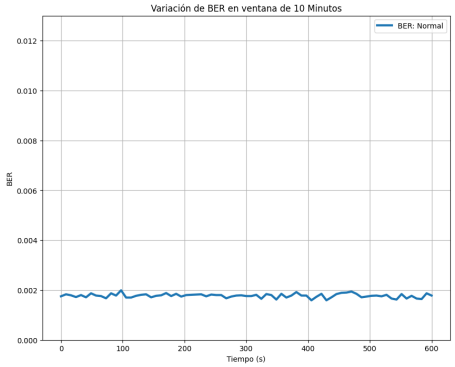
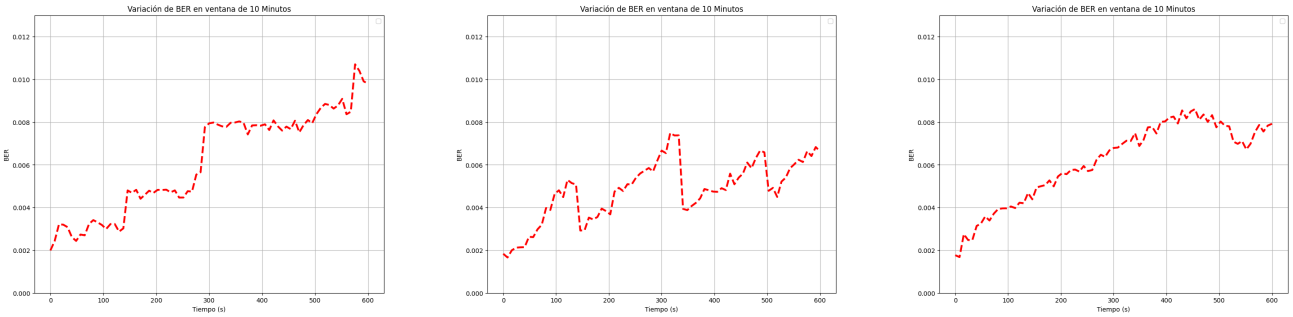


Fig. 4.10: Variación de BER en estado normal de operación. Fuente: Autoría Propia

4.4.1.2. Datos Anómalos

Para el proceso de obtención de esta base de datos, fue necesario tener que manipular los software que controlan tanto el atenuador variable y el filtro óptico. Esto se hace de manera manual y la idea es construir un set de datos representativo de estos fallos, por ende, se toman en cuenta distintos escenarios, tanto así progresivos de manera lenta, comportamientos de tipo

"oscilatorio", y otros que se comporten de manera más abrupta, la siguiente imagen representa un par de escenarios posibles de todo el data set construido:



(a) Caso 1 (b) Caso 2 (c) Caso 3

Fig. 4.11: Casos de variaciones anómalas de BER tomadas aleatoriamente. Fuente: Autoría Propia.

Para construir estos casos se deben manipular dos Software de control de dispositivos de la red, el filtro óptico y el atenuador variable montado en el rack de telecomunicaciones:

Atenuador Variable: En la figura 4.4a, se puede apreciar la conexión que se tiene tanto para el EDFA 1, el EDFA 2 y el VOA. Como el VOA está ubicado antes del EDFA 2, esta será la fuente de ruido en el setup de laboratorio.

La manipulación de este dispositivo, se hace mediante la conexión de un cable ethernet a un computador. Luego de realizar todas las configuraciones que se deben hacer para manipular el software [25], la vista principal de la consola del Chassis de FS es la siguiente:

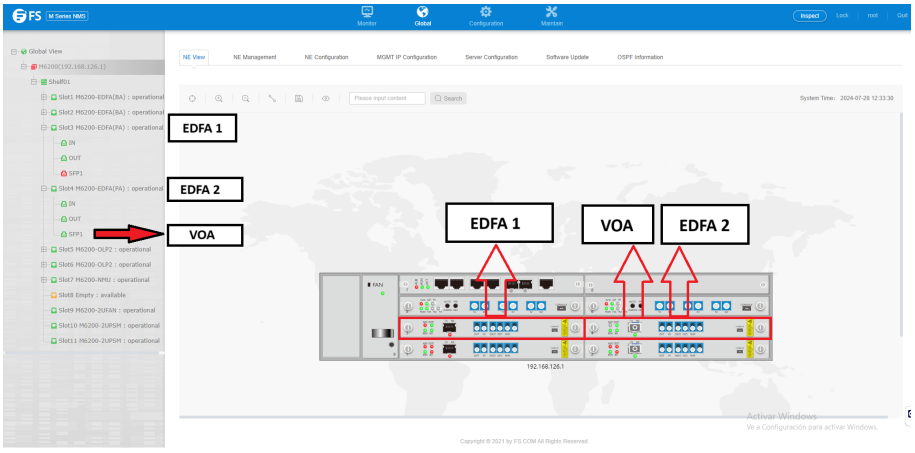


Fig. 4.12: Software para Manipulación de EDFA y VOA. Fuente: Autoría Propia.

Este Software es de fácil uso y es posible manipular la cantidad de atenuación del VOA y la ganancia de los amplificadores de la siguiente manera:

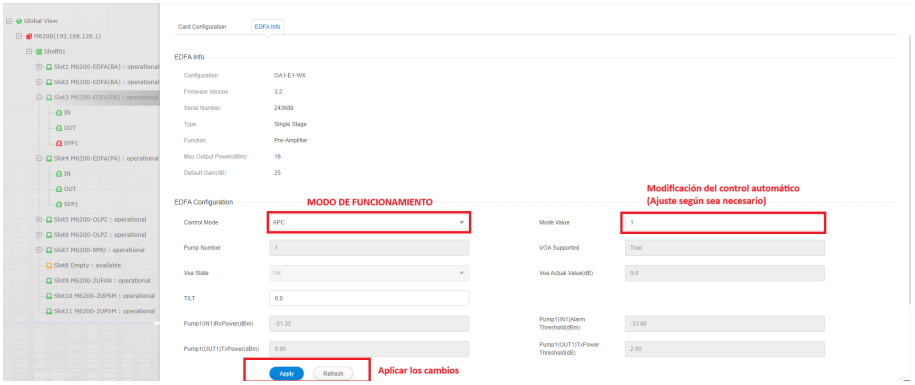


Fig. 4.13: Interfaz de EDFA (1 y 2). Fuente: Autoría propia

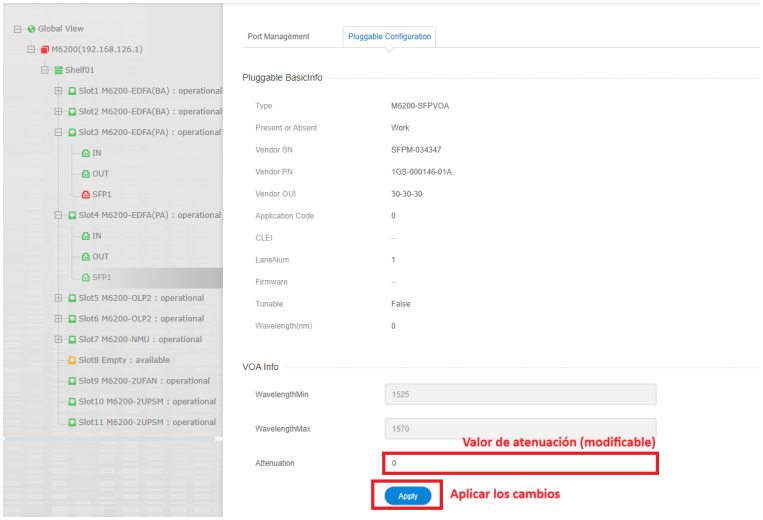


Fig. 4.14: Interfaz de VOA. Fuente: Autoría Propia

Para la recreación de SF en base a el aumento del piso de ruido, es que, paralelamente se ejecuta el código de medición que extraía el BER de los Cassini, y, en esa ventana de medición, se modificaba el modulo de atenuación (figura 4.11), de manera progresiva desde 0 hasta lo máximo que era capaz de soportar el enlace antes de la caída (normalmente 10 a 13 [dB] de atenuación). Cabe destacar que se tomaron diversos escenarios de fallo, ninguno siguiendo el mismo patrón, para que así el set de datos quede de la manera más variada posible.

En total se tomaron 100 muestras (V_{BER}) de fallos relacionados a amplificadores ópticos (IASEN).

Filtro óptico: En la figura 4.29b, se tiene la conexión IN/OUT de la fuente que viene desde el EDFA 2 y termina en el Cassini 2, debajo de la conexión se encuentra el sistema central del WSS que se conecta a un computador mediante un cable RS232. De forma parecida a la

comunicación con los Cassini, este se comunica mediante comandos en un terminal que puede ser putty u otro programa. En [27], se explica a cabalidad el funcionamiento de un WSS, sin embargo, para efectos de esta investigación, solo se utilizará como un elemento de red que corta el espectro por la izquierda, o por la derecha. El WSS permite adaptar manualmente el ancho de banda de los canales para dar paso a señales entrantes/salientes. El usuario puede definir arbitrariamente la configuración de los canales, al igual que su ancho de banda. El WSS opera desde los 191,3 [THz] hasta los 196,1 [THz], este ancho de banda se separa en 386 canales (ranuras) que tienen un ancho de 12,5 [GHz].

Canal	**Frecuencias disponibles**
0	191,3
1	191,3125
...	...
48	191,9000
...	...
96	192,5000
...	...
386	196,1000

Tabla 4.4: Resumen de ancho de banda de cada canal. Fuente: [27]

Como la frecuencia a la que opera el Cassini es de 194 [THz], y el ancho de banda de su señal es de aproximadamente 50 [GHz], los canales que se deben habilitar son el 214, 215, 216 y 217. Sin embargo, se deja un margen a cada lado del espectro para recrear los fallos de mejor manera, desde el canal 209 hasta el 220. Cada canal tiene un nivel de atenuación que se puede ajustar mediante línea de comandos.

Para explicar de mejor manera, se ejemplifica con un comando utilizado: **ura 209,1,0.0.** Esto indica que al canal 209, se habilita para que salga por el puerto 1 y que tenga un nivel de atenuación de 0 [dB], este valor de atenuación para cada canal se debe ir ajustando para ir cortando el espectro e ir recreando los SF. Si se desea cortar el espectro por el lado izquierdo, se debe atenuar desde el canal 220 hasta el 215, de forma análoga, si se quiere atenuar por el lado derecho, se atenúan los canales del 209 al 214 de manera gradual.

La siguiente figura explica de forma gráfica lo declarado anteriormente:

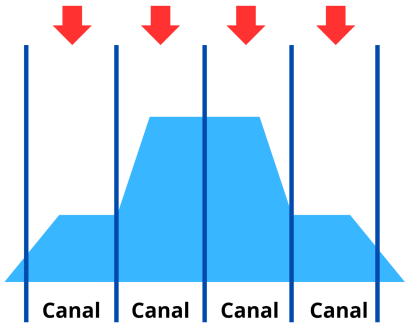


Fig. 4.15: Explicación gráfica funcionamiento WSS. Fuente: Autoría Propia

Mientras se va cortando el espectro, en respuesta se eleva el BER progresivamente de una manera similar a lo visto en la figura 4.11.

La siguiente figura muestra un ejemplo práctico utilizando los comandos para ir cortando de manera progresiva el espectro por uno de sus lados.

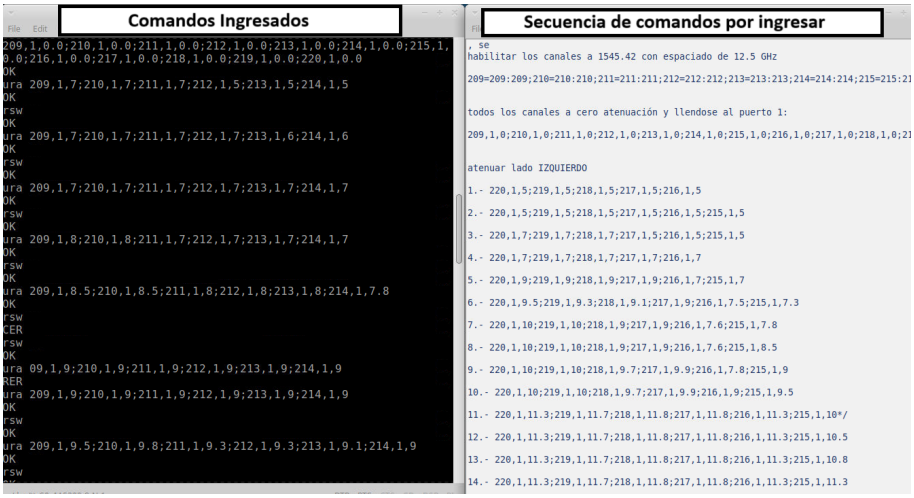


Fig. 4.16: Explicación gráfica funcionamiento WSS. Fuente: Autoría Propia

De lado izquierdo se tiene la terminal abierta en el escritorio del computador con los comandos utilizados, y en el lado derecho se encuentra un bloc de notas abierto con una secuencia de comandos a utilizar.

Cabe destacar que en cada iteración se tomaron atenuaciones distintas en cada canal para así recrear tipos de SF que tuvieran un comportamiento distinto y así, al igual que con IASEN, se construyera un set de datos variados.

En total, se tomaron 100 muestras (V_{BER}) de fallos relacionados a filtros ópticos (FS izquier-

da y FS derecha).

Comportamiento ruidoso de BER: Para constuir este set de datos, simplemente se tomaron 200 muestras de V_{BER} normales y se les añadió un ruido gaussiano con media 0, y una desviación estándar de 0.001. Además, se debe tener en cuenta que este BER debe tener un valor mínimo asociado a su formato de modulación. Que, según la figura 3.3, es de aproximadamente entre 7×10^{-4} y 1×10^{-3}.

Se puede ir ajustando esta desviación estándar para hacer más o menos ruidoso el BER.

Entonces, realizando una tabla resumen de todos los datos, se tiene lo siguiente:

Dato	**Cantidad**
Normales (Entrenamiento)	1802
Normales (Prueba y Validación)	201
Anómalo - IASEN (Validación)	100
Anómalo - FS (Validación)	100
Ruido (Validación)	201
Total	2404

Tabla 4.5: Resumen datos obtenidos para detección. Fuente: Autoría Propia

Y gráficamente los datos se ven de la siguiente manera:

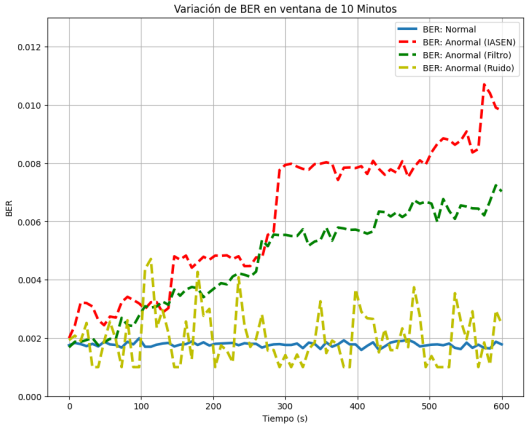


Fig. 4.17: Fallos creados en laboratorio Fuente: Autoría Propia

4.4.2. Identificación

Para esta segunda etapa, se debe construir un data set que muestre el espectro óptico. Para este estudio se involucra la densidad espectral de potencia (PSD), que es la distribución de todas las potencias en un ancho de banda. Este ancho de banda viene dado por una cantidad de puntos que un analizador de espectros puede graficar, se debe tener en cuenta que la configuración del mismo debe ser la óptima para así representar los fallos en el espectro de manera adecuada y que un modelo pueda identificarlos.

Se configura el OSA Anritsu MS9740 de la siguiente manera:

Configuración	**Valor**
Centro	1545.41 [nm]
Span	2.30 [nm]
Start Wavelength	1544.26 [nm]
Stop Wavelength	1546.56 [nm]
Log/div	10.0 [dB]
Ref Level	5.2 [dBm]
Opt Att.	On
Resolution	0.07 [nm]
VBW	1 [KHz]
Sampling Points	2001

Tabla 4.6: Configuración de Anritsu MS9740. Fuente: Autoría Propia.

Esto se puede traducir en que el espectro medido tendrá un ancho de banda que va desde los 1544.26 [nm] hasta los 1546.56 [nm], y se graficará con 2001 puntos de potencias en [dBm] a lo largo de este ancho de banda.

Entonces, al momento de extraer los datos mediante un script, lo que se guardará es un vector ($V_{espectro}$) que tendrá 2001 elementos. Si se realiza un proceso reiterativo de la extracción de los datos y se etiquetan, la matriz de la base de datos quedaría de la siguiente manera.

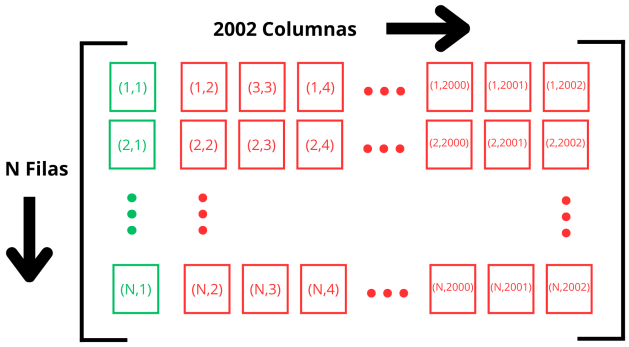


Fig. 4.18: Estructura de la base de datos para entrenamiento de algoritmo de identificación. Fuente: Autoría Propia

En donde la primera columna se indica la etiqueta (0: Normal, 1: FS Izquierda, 2: FS Derecha, 3: IASEN) y las siguientes 2001 columnas a la izquierda la distribución de la potencia en todo ese ancho de banda medido.

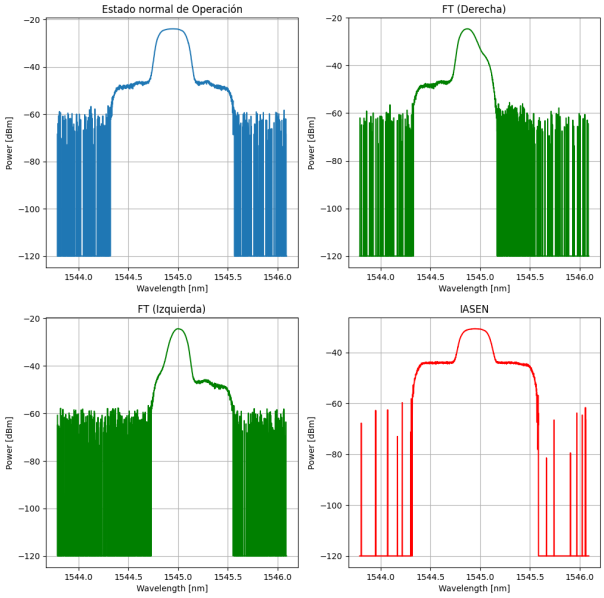


Fig. 4.19: Fallos creados en laboratorio (Espectro) Fuente: Autoría Propia

Cabe destacar que, para construir el data set, la estrategia fue la misma que la utilizada en la etapa de detección, pero esta vez tomando en cuenta también los datos del espectro óptico. El tiempo de muestreo es a libre elección ya que para identificación no es necesario tomarlo como un parámetro que mejore o empeore los rendimientos de los modelos, para identificar un fallo solo se necesita el espectro.

Se tomó en cuenta un estado normal de operación debido a que los espectros entre IASEN

y NORMAL son parecidos, solo difieren en el piso de ruido y en la potencia peak de la señal. Es por esto que se entrenará un modelo tomando todas esas referencias.

Se tomará en cuenta que un 70 % de estos datos serán para entrenamiento, un 15 % para prueba, y un 15 % para validación. Sin embargo, el set de validación tendrá en cuenta otro set de datos, para reforzar la elección del modelo.

Explicando el ultimo punto, se tomarán en cuenta distintos escenarios, como efectos ruidosos, y cambios en la potencia Rx. Todo esto se debe a que se necesita crear un modelo que generalice bien las características de estos fallos y pueda identificarlos sin importar la potencia con la que llega al receptor. Se toma en cuenta que los datos del entrenamiento de un algoritmo contienen información de una señal que tiene una potencia peak con un poco menos de -20 [dBm], entonces, se tomarán escenarios cuando esa potencia peak varía.

Entonces, realizando una tabla resumen de todos los datos, se tiene lo siguiente:

Dato	**Entrenamiento**	**Prueba**	**Validación**
Normales	3396	768	748
FS Derecha	4141	888	854
FS Izquierda	3979	821	856
IASEN	4554	966	987
Variación	-	-	3445
Total	16070	3443	6890

Tabla 4.7: Resumen datos obtenidos para Identificación. Fuente: Autoría Propia

En total, se crearon 26403 datos para entrenamiento, prueba y validación de un modelo de identificación.

4.5. Elección del modelo para detección

En esta sección, se realizará un estudio y análisis de 4 distintos modelos para detección de SF basado en la variación del BER en el tiempo. Ya con todos los datos obtenidos, se utilizarán modelos basados en el valor medio, autocorrelación, área bajo la curva y un modelo de Machine Learning basado en redes neuronales (Autoencoders).

4.5.1. Autoencoder

Cómo fue descrito en la sección de metodología, para programar esta red neuronal se utilizó la plataforma de Google Colab. Para esto se cargan 3 archivos .CSV que contienen los datos de BER en estado normal de operación, y dos anómalos relacionados con IASEN y Filtro Óptico. Luego se grafican como y se visualizan en la figura 4.17.

La arquitectura de la red neuronal es la siguiente:

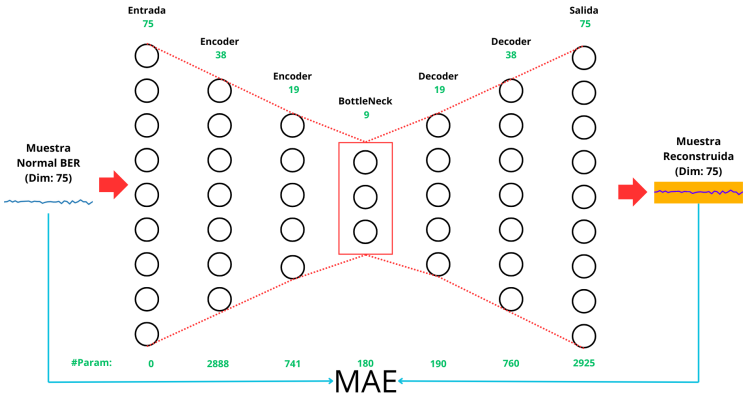


Fig. 4.20: Arquitectura Autoencoder Fuente: Autoría Propia

La capa de entrada tiene una dimensión de 75 Neuronas que se codifican para comprimirse en 38-19-9, para luego decodificarse en un proceso análogo hasta llegar a la dimensión original de salida de 75, todas las neuronas son Fully Connected (Dense) y su función de activación es ReLu, para la salida se utiliza una función Sigmoide, debido a la naturaleza binaria de la clasificación (Anómalo o no anómalo). La cantidad de parámetros totales a entrenar es de

La salida reconstruida se compara con la entrada con una metra conocida como MAE (Error Absoluto Medio), que se rige bajo la siguiente fórmula:

$$MAE = \frac{\sum_{i=1}^{75} |x_i - \hat{x}_i|}{75} \tag{4.2}$$

Luego se procede a construir el modelo de entrenamiento con los siguientes parámetros:

Luego, se grafican los resultados del entrenamiento:

Parámetro	Valor
Optimizador	Adam
Learning Rate	0.0001
Épocas de entrenamiento	1000
Batch Size	32
Shuffle	True
EarlyStopping	Sí

Tabla 4.8: Parámetros para entrenamiento de Autoencoder. Fuente: Autoría Propia.

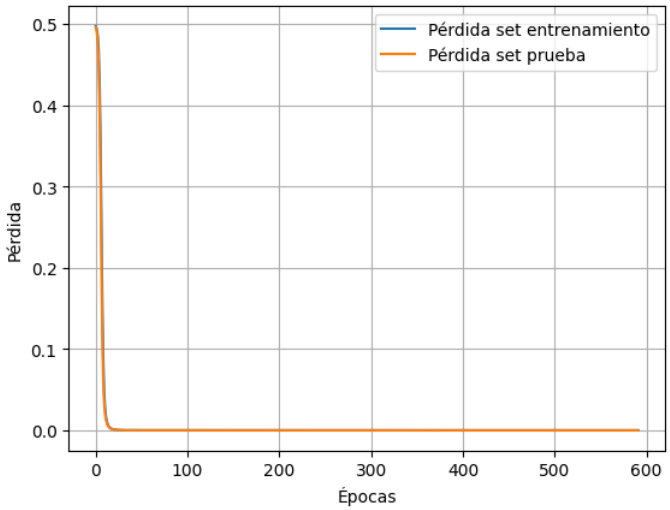


Fig. 4.21: Entrenamiento de Autoencoder. Fuente: Autoría Propia

Debido a la funcion *EarlyStopping* el entrenamiento se detiene en torno a las 500 épocas de entrenamiento, que es justo cuando el modelo obtiene los mejores resultados en torno a la diferencia de pérdidas en el set de entrenamiento versus el set de pruebas. No existe *OverFitting* en el entrenamiento.

Evaluación del modelo: ¿Qué tan bien reconstruye un V_{BER} normal v/s uno anormal?

Para esto se hace uso del set de pruebas de BER normal y el set de validación correspondiente a BER Anormal.

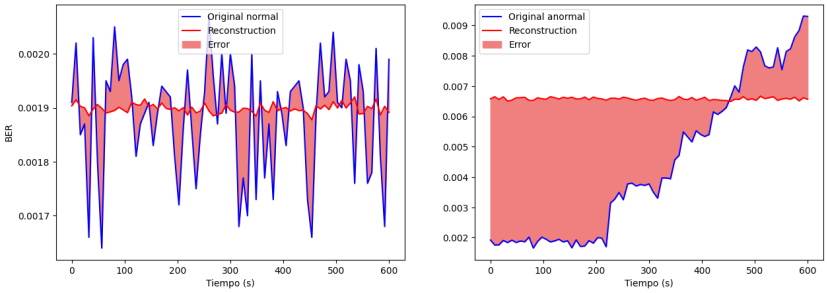


Fig. 4.22: Error de reconstrucción dato normal v/s anormal. Fuente: Autoría Propia

Preliminarmente, el modelo de autoencoder, al ser entrenado con solamente datos normales de operación, si se le ingresa un dato nuevo anormal, el error de reconstrucción será alto. Entonces, sería posible elegir una métrica para clasificar, en base a MAE, si un dato de entrada es anómalo o normal.

Para evaluar el rendimiento del modelo se utiliza la función ROC mediante la elección de distintos umbrales de clasificación que una función en Google Colab calcula:

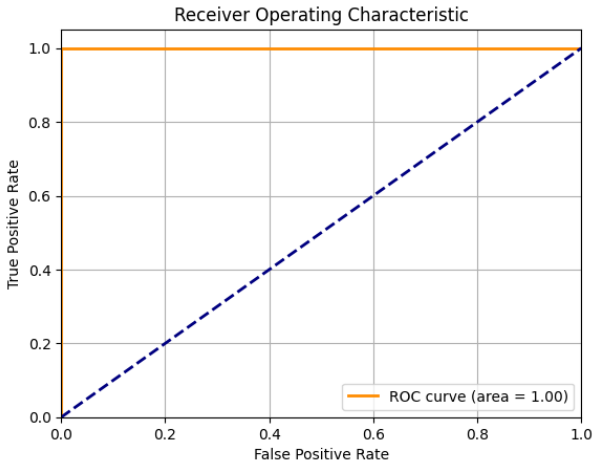


Fig. 4.23: Error de reconstrucción dato normal v/s anormal. Fuente: Autoría Propia

Cómo el area bajo la curva es cercano a 1, esto quiere decir que el modelo clasifica todos los casos como positivos y no tiene falsos positivos. Esto quiere decir que se está en presencia de un modelo perfecto.

Luego, se confecciona un histograma que contiene todos los errores de reconstrucción para todos los datos de validación, se compara con los errores y se elige el mejor umbral en base al valor máximo en la diferencia entre el TPR y el FPR (índice de Youden).

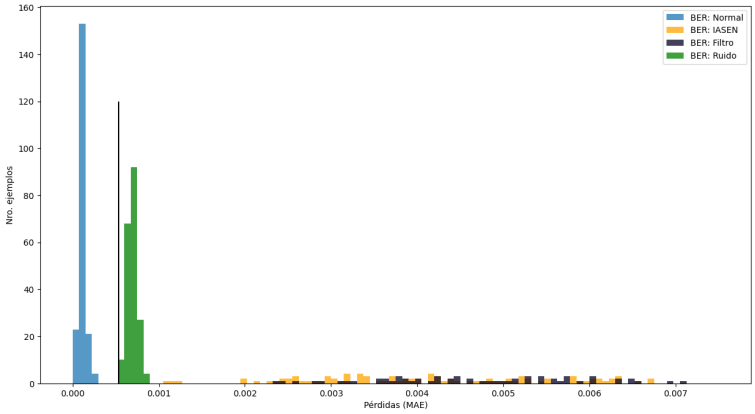


Fig. 4.24: Histograma Autoencoder. Fuente: Autoría Propia

De la figura anterior se concluye lo declarado anteriormente, los datos anómalos tienden a tener un error de reconstrucción mayor que para datos normales. En base a la curva ROC, se máximiza la diferencia entre TPR y FPR y se elige un umbral de decisión (Anómalo o Normal) el cual es de: $\mathbf{0.00053758 = 0.053\,\%}$. El error de reconstrucción, es muy bajo pero debido a la escala son valores aceptables.

Por consiguiente, debido al umbral elegido, (linea negra en figura 4.24), se calcula la especificidad (TNR) para la categoría 1: Normales y la sensibilidad (TPR) para las categórias 2,3 y 4: FS, IASEN y BER Ruidoso.

Especificidad (Cat 1: Normal)	100 %
Sensibilidad (Cat 2: IASEN)	100 %
Sensibilidad (Cat 3: FS)	100 %
Sensibilidad (Cat 4: Ber Ruidoso)	100 %

Tabla 4.9: Especificidad y sensibilidad de modelo de detección (Autoencoder). Fuente: Autoría Propia.

4.5.2. Valor medio

Este modelo se basa principalmente en calcular el valor medio de cada fila de la matriz de valores normales y anómalos. Si se grafica el histograma de los valores medios de todos los datos, al igual que en la figura 4.24 se tiene lo siguiente:

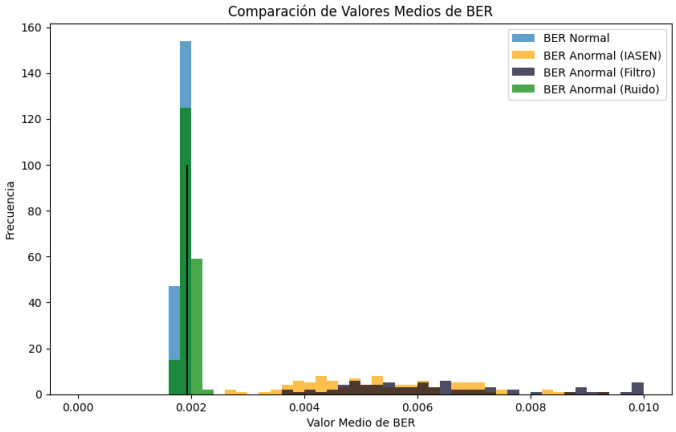


Fig. 4.25: Histograma Valor Medio. Fuente: Autoría Propia

El umbral elegido (Índice de Youden) es de: **0.001** de valor medio. Y las métricas de especificidad y sensitividad son las siguientes:

Especificidad (Cat 1: Normal)	95.0 %
Sensibilidad (Cat 2: IASEN)	100 %
Sensibilidad (Cat 3: FS)	100 %
Sensibilidad (Cat 4: Ber Ruidoso)	55.2 %

Tabla 4.10: Especificidad y sensibilidad de modelo de detección (Valor Medio). Fuente: Autoría Propia.

4.5.3. Autocorrelación

Este modelo se basa principalmente en calcular el valor de autocorrelación de cada fila de la matriz de valores normales y anómalos. Si se grafica el histograma de los valores de autocorrelación de todos los datos, al igual que en la figura 4.24 se tiene lo siguiente:

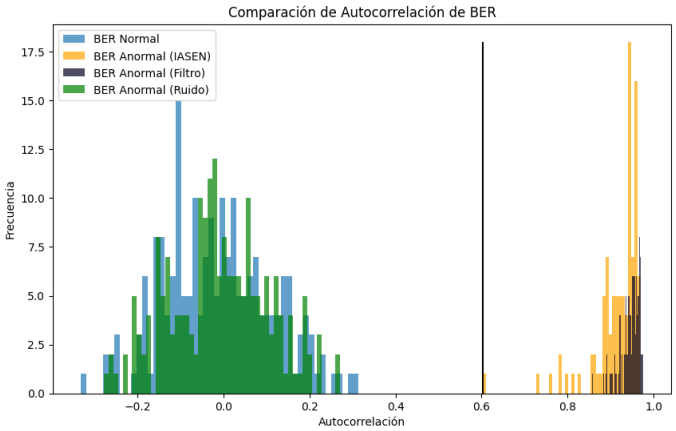


Fig. 4.26: Histograma Autocorrelación. Fuente: Autoría Propia

El umbral elegido (Índice de Youden) es de: **0.6028** de autocorrelación. Y las métricas de especificidad y sensitividad son las siguientes:

Especificidad (Cat 1: Normal)	100.0 %
Sensibilidad (Cat 2: IASEN)	100 %
Sensibilidad (Cat 3: FS)	100 %
Sensibilidad (Cat 4: Ber Ruidoso)	0 %

Tabla 4.11: Especificidad y sensibilidad de modelo de detección (Autocorrelación). Fuente: Autoría Propia.

4.5.4. Area bajo la curva

Este modelo se basa principalmente en calcular el valor del area bajo la curva de cada fila de la matriz de valores normales y anómalos. Si se grafica el histograma de los valores del área bajo la curva de todos los datos, al igual que en la figura 4.24 se tiene lo siguiente:

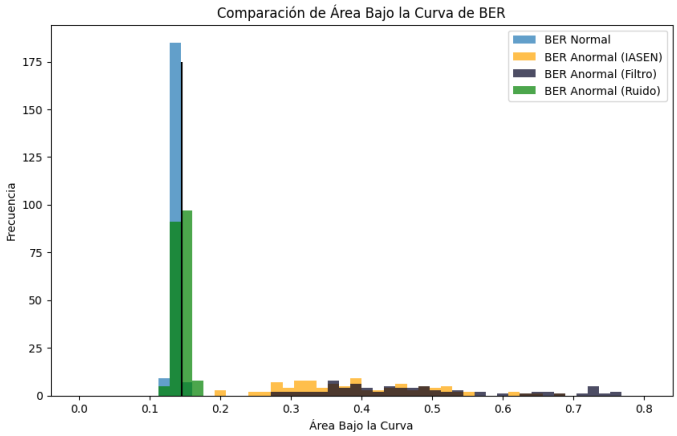


Fig. 4.27: Histograma Área bajo la curva. Fuente: Autoría Propia

El umbral elegido (Índice de Youden) es de: **0.145169** de area bajo la curva. Y las métricas de especificidad y sensitividad son las siguientes:

Especificidad (Cat 1: Normal)	100.0 %
Sensibilidad (Cat 2: IASEN)	100 %
Sensibilidad (Cat 3: FS)	100 %
Sensibilidad (Cat 4: Ber Ruidoso)	48.8 %

Tabla 4.12: Especificidad y sensibilidad de modelo de detección (Area Bajo la curva). Fuente: Autoría Propia.

4.5.5. Discusión

A continuación se muestra la curva ROC de los cuatro modelos:

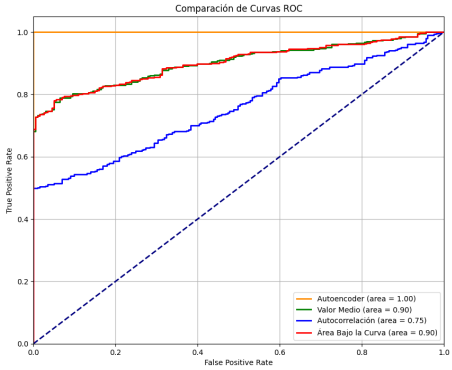
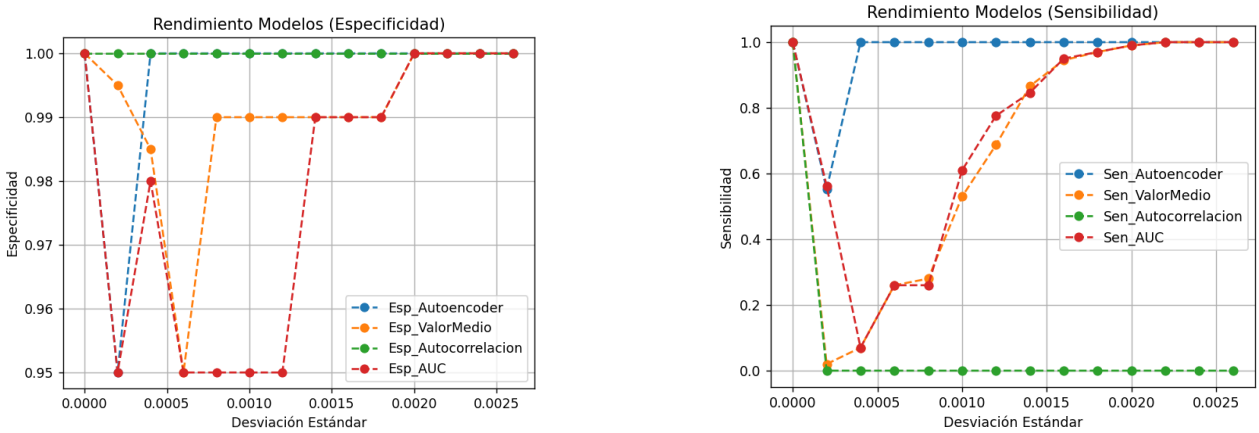


Fig. 4.28: Histograma Autocorrelación. Fuente: Autoría Propia

Para evaluar de mejor forma el modelo, y gracias a que el efecto ruidoso del BER es una simulación con una desviación estándar de 0.001 y una media cero, podemos tomar como parámetro la desviación estándar y variarla desde 0 hasta algún valor cuando todos los modelos alcancen su mejor rendimiento:



(a) D. Estandar v/s Esp.

(b) D. Estandar v/s Sen.

Fig. 4.29: Rendimiento de los modelos según desviación estándar (ruido)

En conclusión, los 4 modelos son buenos para detectar fallos correspondientes a FS e IASEN, sin embargo, para detectar anomalías en base a BER ruidoso, el que tiene una mejor sensibilidad es un autoencoder, debido a que alcanza un 100 % de precisión antes que los demás modelos. Es decir, mientras menos ruidoso sea la variación del BER en el tiempo, el modelo con mejores resultados para detectar es el Autoencoder.

4.6. Elección del modelo para identificación

En esta sección, se realizará un estudio y análisis de 4 distintos modelos para identificación de SF basado en la variación del BER en el tiempo. Ya con todos los datos obtenidos, se estudiarán modelos basados en redes neuronales convolucionales en una dimensión, K-Nearest Neighbors, SVM y Random Forest.

4.6.1. Redes Neuronales Convolucionales en 1D

Como fue descrito en la sección de metodología, para programar esta red neuronal se utilizó la plataforma de Google Colab. Para esto se carga 1 archivo .CSV que contiene, en cada fila, la información del espectro óptico en el ancho de banda medido y su respectiva etiqueta: 0 para normal, 1 para FS derecha, 2 para FS izquierda y 3 para IASEN. (Ver figura 4.19).

La arquitectura de la red nueuronal convolucional es la siguiente:

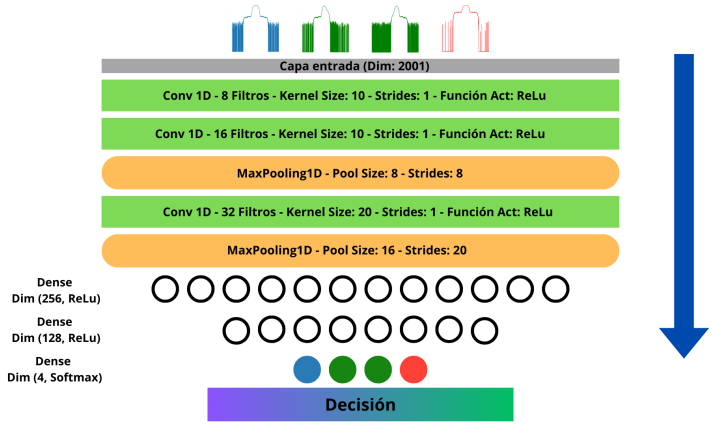


Fig. 4.30: Histograma Autocorrelación. Fuente: Autoría Propia

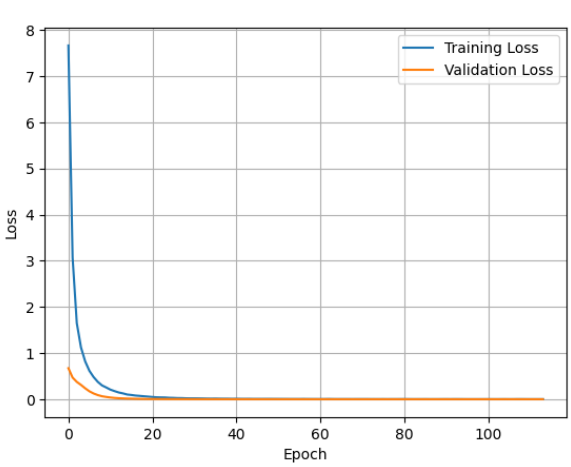
Cabe destacar que luego de cada capa "Dense" se encuentra un "Dropout" de 0.5 para evitar el *OverFitting* en el entrenamiento. La cantidad de total de parámetros a entrenar es de 135436.

Luego de procesar los datos y transformar las etiquetas a formato *One-Hot*, se procede al entrenamiento de la red neuronal.

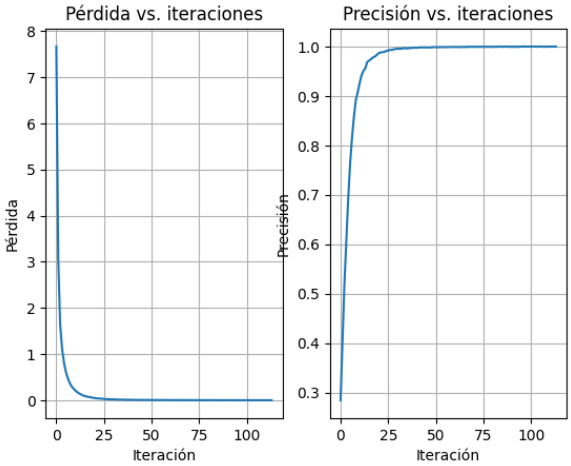
Parámetro	**Valor**
Optimizador	Adam
Learning Rate	0.00001
Épocas de entrenamiento	200
Batch Size	64
Verbose	2
EarlyStopping	Sí

Tabla 4.13: Parámetros para entrenamiento de 1D-CNN. Fuente: Autoría Propia.

Luego, se grafican los resultados del entrenamiento:



(a) Épocas v/s precisión.



(b) Épocas V/s Pérdida.

Fig. 4.31: Entrenamiento 1D-CNN. Fuente: Autoría Propia.

El modelo, en el entrenamiento, alcanza un **100 % de precisión** en el entrenamiento con los datos de validación.

Luego, graficamos la matriz de confusión con los datos del set de prueba para verificar si realmente el modelo alcanza el 100 % de precisión.

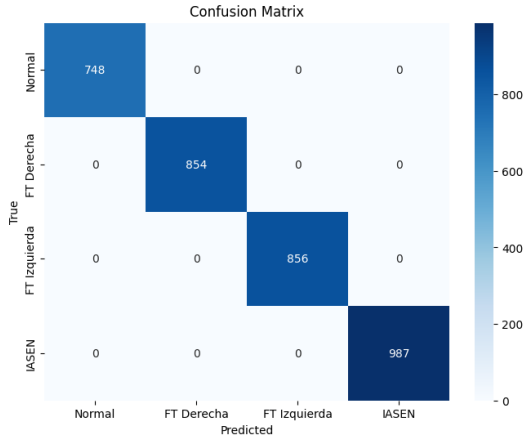


Fig. 4.32: Matriz de Confusión, validacion 1D-CNN. Fuente: Autoría Propia

El modelo alcanza un **100 % de precisión** con los datos de validación.

4.6.2. K-Nearest Neighbors

De forma análoga que para el entrenamiento de la red 1D-CNN, se implementa un algorimto de aprendizaje supervisado llamado K-Nearest Neighbors (KNN). La implementación de este algoritmo en Google Colab es simple, y se utilizan los datos de entrenamiento de los espectros con sus respectivas etiquetas para clasificarlos.

Se utiliza la métrica de precisión para saber cuántos de la muestra total de datos sabe clasificar bien y la matriz de confusión para ver gráficamente su rendimiento (con datos de la validación).

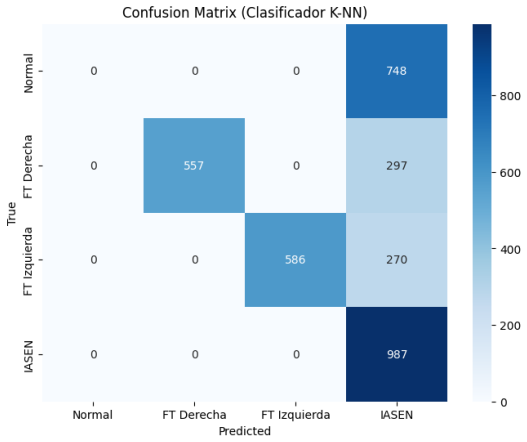


Fig. 4.33: Matriz de Confusión, validacion KNN. Fuente: Autoría Propia

La precisión del modelo para clasificar los distintos SF alcanza un **61.6 %**

4.6.3. Random Forest

Se implementa un algorimto de aprendizaje supervisado llamado Random Forest. La implementación de este algoritmo en Google Colab, al igual que con KNN es simple, y se utilizan los datos de entrenamiento de los espectros con sus respectivas etiquetas para clasificarlos.

Se utiliza la métrica de precisión para saber cuántos de la muestra total de datos sabe clasificar bien y la matriz de confusión para ver gráficamente su rendimiento (con datos de la validación).

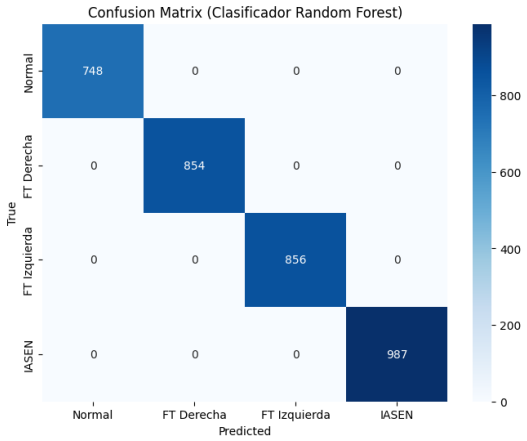


Fig. 4.34: Matriz de Confusión, validacion Random Forest. Fuente: Autoría Propia

La precisión del modelo para clasificar los distintos SF alcanza un **100 %**

4.6.4. SVM

Finalmente, se implementa otro algorimto de aprendizaje supervisado llamado Support Vector Machine (SVM). La implementación de este algoritmo en Google Colab, al igual que con Random Forest y KNN es simple, se utilizan los datos de entrenamiento de los espectros con sus respectivas etiquetas para clasificarlos.

Se visuliza la matriz de confusión resultante de este modelo con los datos de validación.

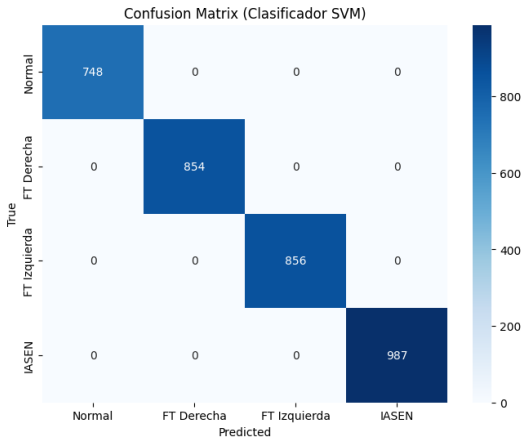


Fig. 4.35: Matriz de Confusión, validacion SVM. Fuente: Autoría Propia

La precisión del modelo para clasificar los distintos SF alcanza un **100 %**

4.6.5. Discusión

Aunque los algoritmos Random Forest y SVM muestran un 100 % de precisión al identificar un SF con los mismos datos de validación que se usaron para su construcción, es necesario volver a validar estos tres modelos: 1D-CNN, Random Forest y SVM. El objetivo de esta re-validación es determinar cuál de estos modelos puede identificar con mayor precisión cuando los datos de entrada son diferentes a los datos con los que se entrenaron originalmente.

La razón principal de esta re-validación es que, si se quiere implementar un modelo de identificación en sistemas reales, se necesita evaluar cuál de los tres algoritmos puede adaptarse mejor a las variaciones que ocurren en la vida real. Estas variaciones incluyen cambios en la potencia Rx debido a factores como la retirada de algún componente (lo que provoca variaciones en la pérdida de inserción), modificaciones en la ganancia de amplificación, y aumentos o disminuciones en la potencia de transmisión.

Siguiendo esta lógica, se puede tabular la precisión de estos tres modelos y graficarla en función de la potencia Rx. Para esto, se puede simular los datos agregando ruido gaussiano a los datos de validación, con una desviación estándar de 0 y una media variable. La media actuará como un "variador" de la potencia peak de la señal Rx. Si se varía la media entre -20 y 20, se obtendrá una variación en la potencia peak que fluctuará -20 [dBm] y +20 [dBm] de de los datos con los que se entrenaron los modelos, ya que la potencia original es aproximadamente -5 [dBm]. De esta manera, se obtendrá un nuevo conjunto de validación con datos muy diferentes a los utilizados para entrenar el modelo. En resumen, el objetivo de esta tarea es observar cómo responden los diferentes modelos a distintos escenarios de operación, simulando entornos reales.

Entonces, utilizando la plataforma Google Colab, luego de simular los datos y crear un nuevo dataset con estas variaciones en potencia Rx, se tienen los siguientes resultados:

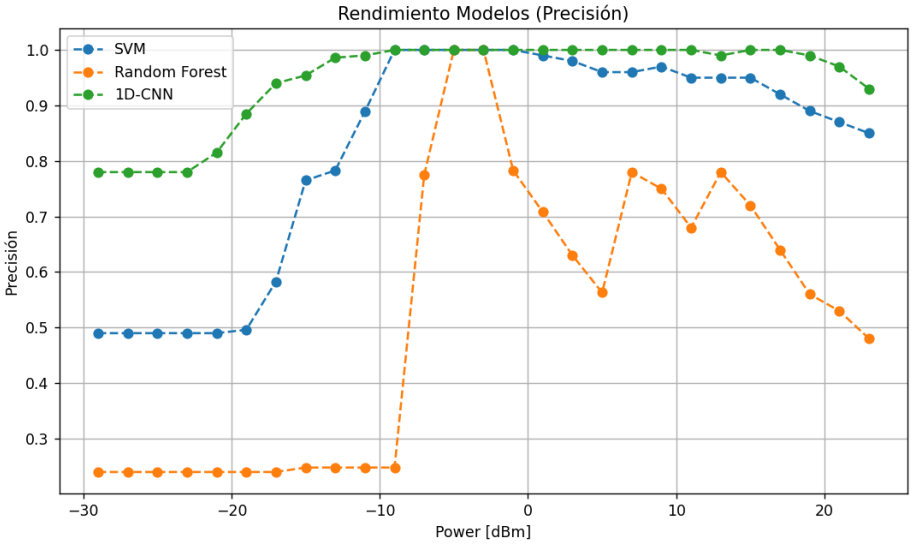


Fig. 4.36: Comparación de precisión de los tres modelos frente a distintos escenarios. Fuente: Autoría Propia

De la figura anterior se concluye que, a distintos escenarios variando la potencia Rx, los modelos que mejor generalizan las características, debido a su nivel de precisión son los algoritmos SVM y 1D-CNN, manteniendo este último, mejores resultados. Es posible observar tambien que en un determinado valor de potencia todos los modelos convergen al 100 % de precisión debido a que estos fueron los datos con los que fueron entrenados.

4.7. Algoritmo Final

Finalmente, luego de construir los modelos, entrenarlos, y compararlos con diferentes técnicas existentes, es posible elegir el algoritmo final que cumpla con la tarea de detectar y posteriormente identificar un Soft Failure.

Los criterios de elección fueron en base a rendimiento (precisión y sensibilidad), siendo la comparación entre distintos modelos el que proporciona la decisión final del algoritmo a implementar.

Por ende, el algoritmo final a implementar está representado en el siguiente diagrama de flujo:

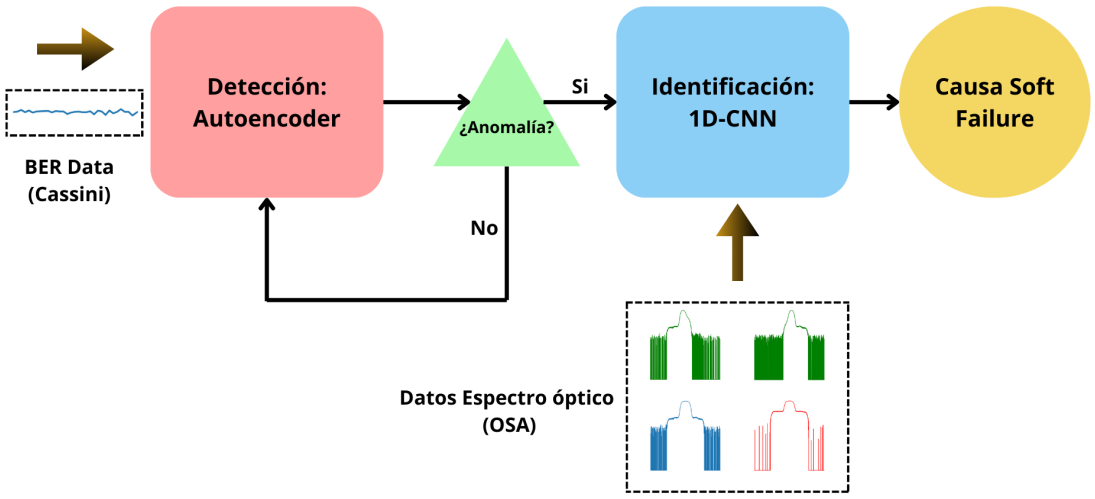


Fig. 4.37: Diagrama de flujo algoritmo final. Fuente: Autoría Propia

En donde el umbral de decisión de detección, es decir, el error de reconstrucción del autoencoder que se elige es de **0.053 %**

5. Conclusiones

5.1. Conclusión

En el desarrollo de esta memoria de título, se realizó un estudio profundo sobre los Fallos Suaves (Soft Failures), abarcando su naturaleza, causas y los efectos de no tratarlos a tiempo. Actualmente, los fallos progresivos en una red óptica pueden disminuir su calidad y afectar a millones de personas que dependen de estos servicios.

Es fundamental no solo detectar e identificar un Fallo Suave, sino también eliminar el factor humano en este proceso y reducir los tiempos de restauración del enlace. Esta es la motivación detrás de este trabajo de investigación: encontrar el mejor modelo que pueda realizar esta tarea en el menor tiempo posible.

Después de revisar la bibliografía disponible, se logró definir un algoritmo para la detección de Fallos Suaves basado en la variación de un parámetro de la señal óptica conocido como Bit Error Rate (BER). Este parámetro sigue un patrón de comportamiento según el estado del enlace, lo que facilitó su comprensión y posterior implementación en un código de programación. Se evaluaron cuatro modelos diferentes, y el modelo de Autoencoder de Machine Learning mostró el mejor rendimiento en cuanto a precisión, por lo que fue elegido para realizar el primer paso del algoritmo completo.

Posteriormente, se seleccionó el modelo de identificación que, utilizando la información del espectro óptico, obtuvo los mejores resultados en términos de precisión. Se estudiaron cuatro modelos diferentes, evaluándolos en distintos escenarios para observar su respuesta a entornos cambiantes. El modelo elegido en base a estos resultados, fue la red neuronal convolucional de una dimensión.

Una de las limitaciones de este trabajo fue la capacidad de recrear los Fallos Suaves. Aunque se definieron cuatro tipos de fallos en la revisión bibliográfica, solo se pudieron crear dos experimentalmente, debido principalmente a las limitaciones del equipo disponible en el laboratorio.

En esta investigación, los resultados muestran que durante la etapa de detección, el modelo basado en autoencoder fue capaz de identificar con precisión el 100 % de los datos anómalos en

comparación con los datos normales. Después de esta detección inicial, el modelo fue probado nuevamente al introducir ruido Gaussiano en los datos de BER normales. En esta revalidación, el autoencoder demostró ser más sensible y eficiente que otros modelos matemáticos, manejando mejor los datos ruidosos. En la etapa de identificación, una red neuronal convolucional unidimensional logró una precisión del 100 % al clasificar los fallos suaves (Soft Failures). Este modelo superó a otros enfoques de Machine Learning en diferentes escenarios donde se ajustaron algunos parámetros del enlace, mostrando una alta resiliencia a distintos escenarios de operación, justificando su posible uso en entornos reales.

Finalmente, tras realizar todos los estudios comparativos y evaluar el rendimiento en función de la precisión y sensibilidad de los algoritmos para su capacidad de detección e identificación, se definió un algoritmo final que detalla gráficamente el proceso que debería seguir un software potencial para realizar esta tarea.

5.2. Trabajo a Futuro

El trabajo futuro se basa principalmente en las limitaciones encontradas durante el desarrollo de esta memoria de título. Es necesario realizar un estudio que considere los efectos no lineales (BER ruidoso) en un enlace para su identificación, además de incluir los otros dos fallos (*Channel interference* y *Filter Tightening*) para obtener un modelo robusto que contemple estos nuevos escenarios.

La continuación de esta memoria se centra en proponer un nuevo paso al algoritmo final: la localización de un Fallo Suave (Soft Failure). A través de la gestión de la red, es posible monitorear todos sus elementos, obtener información de estos y determinar cuál de los amplificadores ópticos presenta la falla o cuál filtro está degradando la señal.

Bibliografía

[1] «El uso de internet a Nivel Mundial – Datos estadísticos» R. Fernández. Statista, 2024 , [En Línea] Disponible en: https://es.statista.com/temas/9795/el-uso-de-internet-en-el-mundo/#topicOverview

[2] A. Puerres, D. Salazar, M. Jiménez «Estudio y Simulación de los Efectos No Lineales Modulación Cruzada de Fase (XPM) y Mezcla de Cuatro Ondas (FWM) en una Fibra Óptica Monomodo.» Escuela politécnica nacional, Ecuador, 2016.

[3] E. Carrizo «Con Google y una Inversión Inicial de US 55 millones: gobierno anuncia Proyecto de Cable Humboldt, la fibra óptica submarina que conecta con Oceanía".». Artículo de "La Tercera", Chile, 2024.

[4] L. Shu, Z. Yu, Z. Wan, J. Zhang, S. Hu, K. Xu «Dual-Stage Soft Failure Detection and Identification for Low-Margin Elastic Optical Network by Exploiting Digital Spectrum Information». Journal of Lightwave Technology, Vol. 38, NO.9. Mayo 2020

[5] H. Lun, M. Fu, X, Liu, Y. Wu, L. Yi, W. Hu, Q. «Soft Failure Identification for Long-Haul Optical Communication Systems Based on One-Dimensional Convolutional Neural Network».Journal of Lighthwave Technology, Vol 38, NO. 1. Junio 2020.

[6] Juan. «¿Qué son las potencias ópticas TX/RX y sus funciones.» FS, página official, Julio, 2021. [En línea] Disponible en: https://community.fs.com/es/article/entender-la-potencia-optica-tx-rx-del-transceiver-sfp-html.html

[7] F. Ramos «Monitorización de OSNR en redes ópticas.» Revista Conectrónica, Agosto, 2009. [En línea] Disponible en: https://www.conectronica.com/fibra-optica/redes-opticas/monitorizacion-de-osnr-en-redes-opticas

[8] H. Orellana, F. Valdés. «Red de transporte óptico coherente.» Presentación en Power Point Ingeniería de Desarrollo, Ingeniería Civil en Telecomunicaciones, Noviembre 2023, Departamento de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de Concepción.

[9] J. Martínez «La dispersion cromática» Blog de Prored, Noviembre, 2018. [En línea] Disponible en: https://www.prored.es/la-dispersion-cromatica/

[10] P. Urrutia «El efecto de dispersion en las Fibras Opticas» Revista Electroindustria, Diciembre, 2012. [En línea] Disponible en: https://www.emb.cl/electroindustria/articulo.mvc?xid=1967&ni=el-efecto-de-dispersion-en-las-fibras-opticas

[11] S. Shahkarami, F. Musumeci, F. Cugini, M. Tornatore «Machine-Learning-Based Soft-Failure Detection and Identification in Optical Networks» Politécnico di Milano, Pisa, Italia, 2018.

[12] L. Velasco, B. Shariati, A. Vela, J. Comellas, M. Ruiz «Learning from the Optical Spectrum: Soft- Failure Identification and Localization [Invited]» Optical Communications Group, Universidad Politécnica de Cataluña, Barcelona, España, 2018.

[13] L. Velasco, B. Shariati, A. Vela, J. Comellas, M. Ruiz «Soft Failure Localization in Elastic Optical Networks» Optical Communications Group, Universidad Politécnica de Cataluña, Barcelona, España, 2018.

[14] H. Lun, M. Fu, X, Liu, Y. Wu, L. Yi, W. Hu, Q. Zhuge «A GAN Based Soft Failure Detection and Identification Framework for Long-Haul Coherent Optical Communication Systems» Journal of Lightwave Technology, Vol 41, NO. 8, Abril 2023.

[15] A. Vela, M. Ruiz, F. Fresi, N. Sambo, F. Cugini, G. Meloni, L. Poti, L. Velasco, P. Castoldi «BER Degradation Detection and Failure Identification in Elastic Optical Networks» Journal of Lightwave Technology, Vol 35, NO. 21, Noviembre, 2017.

[16] F. Ramos «Caracterización de dispositivos fotónicos empleando fuentes de ruido ASE» Revista Conectrónica, Agosto, 2018. [En ínea] Disponible en: https://www.conectronica.com/fibra-optica/instrumentos-para-fibra-optica/caracterizacion-de-dispositivos-fotonicos-empleando-fuentes-de-ruido-ase#:~:text=El%20ruido%20de%20emisiÃşn%20espontÃąnea,amplificadores%20Ãşpticos%20(normalmente%20EDFAs).

[17] F. Musumeci, C. Rottondi, G. Corani, S. Shahkarami, F. Cugini, M. Tornatore «A Tutorial on Machine Learning for Failure Management in Optical Networks» Journal of Lightwave Technology, Vol. 37, NO. 16, Agosto, 2019.

[18] M. Silva, A. Sgambelluri, A. Pacini, F. Paolucci, A. Green, D. Mascarenas, L. Valcarenghi «Confidentiality-preserving machine learning algorithms for soft-failure detection in optical communication networks» Journal of Optical Communications and Networking, VOL. 15, No. 8, Agosto, 2023.

[19] J. Orozco «Machine Learning y su importancia en la actualidad» Artículo de Ipade Business School, Junio, 2023. [En línea] Disponible en: https://www.ipade.mx/newsmedia/tendencias/machine-learning-y-su-importancia-en-la-actualidad/

[20] S. Marsland «CMachine Learning: An Algorithmic Perspective, second edition» Boca Raton, Florida, Estados Unidos, CRC Press, 2015.

[21] D. Bank, N. Koenigstein, R. Giryes «Autoencoders» Tel Aviv University, Abril, 2021.

[22] N. Cascado «Redes Neuronales Convolucionales y aplicaciones» Trabajo de fin de grado, Grado en ingeniería Matemática, curso 2021-2022, Facultad de Matemáticas, Universidad Complutense de Madrid.

[23] K. Sun et al. «Digital Residual Spectrum-Based Generalized Soft Failure Detection and Identification in Optical Networks» in IEEE Transactions on Communications, vol. 71, no. 1, pp. 324-338, Jan. 2023.

[24] Regulación y Mercados «DISEÑO TÉCNICO DE LA TRONCAL NACIONAL DE INFRAESTRUCTURA DE TELECOMUNICACIONES (TNIT) DE FIBRA ÓPTICA REQUERIDO PARA LAS NECESIDADES DE LA INDUSTRIA 4.0» Santiago, Marzo, 2017.

[25] FS «M Series NMS Network Management User Manual» Año desconocido. [En línea] Disponible en: https://resource.fs.com/mall/doc/202402061747548mvvzr.pdf

[26] Anritsu Corporation «MS9740 Optical Spectrum Analyzer Operation Manual» Fourth Edition, September, 2021, Japón.

[27] S. Manríquez «Laboratorio de sistemas y redes ópticas flexibles basado en WSS» Informe Proyecto de Título de Ingeniero Civil Electrónico. Escuela de Ingeniería Eléctrica. Facultad de Ingeniería, Pontificia Universidad Católica de Valparaíso, Valparaíso, 2017.

[28] Finisar «Quick Start Guide For Evaluation Kit for DWP» Septiembre, 2011.

[29] S. C. Gupta, V. K. Kapoor «Fundamentals of Mathematical Statistics», 11th ed. Sultan Chand and Sons, 2014.

[30] Keiser, Geoffrey K. «Optical Fiber Communications.» 4th ed., McGraw-Hill, 2011.

[31] Z. Dong, F. N. Khan, Q. Sui, K. Zhong, C. Lu and A. P. T. Lau «Optical Performance Monitoring: A Review of Current and Future Technologies» in Journal of Lightwave Technology, vol. 34, no. 2, pp. 525-543, 15 Jan.15, 2016, doi: 10.1109/JLT.2015.2480798.

A. ANEXO: Algoritmos de modelos encontrados en Bibliografía

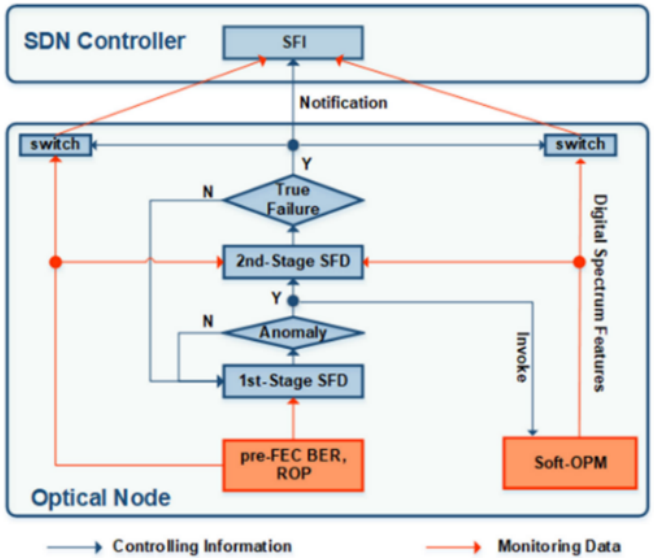


Fig. A.1: Diagrama de flujo para la propuesta de algoritmo de SFD y SFI. [4]

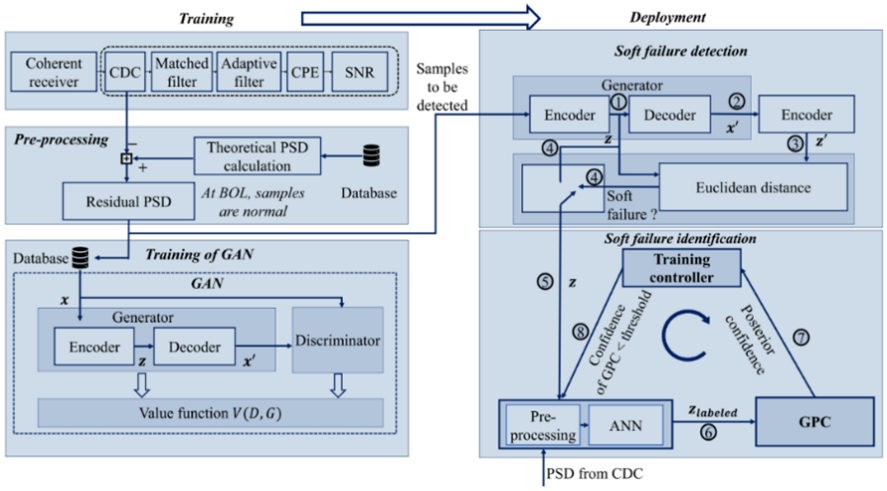


Fig. A.2: Diagrama de flujo para detección e identificación de SF. [14]

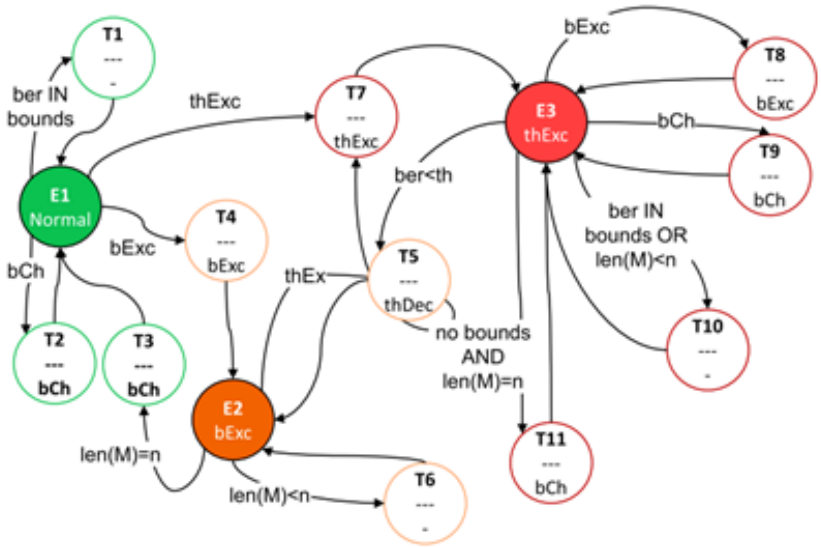


Fig. A.3: Máquina de estados finitos (BANDO). [15]

INPUT: *notif*, α, β, δ
OUTPUT: <*class, prob, timeToMaxBER*>

1: *connId* ← *notif*.conn_id
2: *lastBer* ← *notif*.getLast().ber
3: *threshold* ← *notif.threshold*
4: storeNotif(*connId*, *notif*)
5: **if** (*lastBer* / *threshold*) < δ **then return** <*'no'*,-,->
6: *M*.D={<*t,ber,*P_{Rx}>} ← getData(*connId*)
7: *M*.N={<*t,type,data*>} ← getNotif(*connId*)
8: P_H ← computeFeatureProbs(*M*, α) (Table II)
9: P_Q ← computeFailureProbs(P_H, β)
10: *failureClass* ← arg max$_{(q \in Q)}$(P_Q)
11: *tmax* ← -
12: **if** P_H(BERTrend) > 0 **then**
13: *tmax* ← computeTMax (*M*)
14: **return** <*failureClass*, P_Q(*failureClass*), *tmax*>

Fig. A.4: Algoritmo LUCIDA. [15]

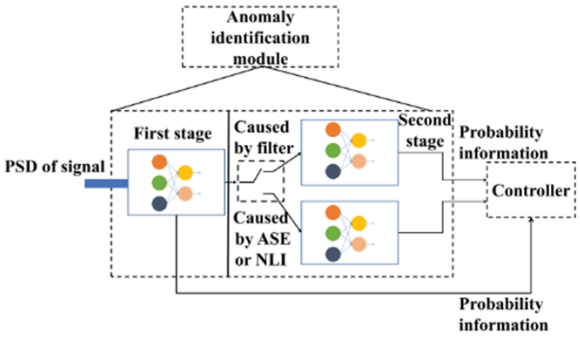


Fig. A.5: Algoritmo de identificación [5]

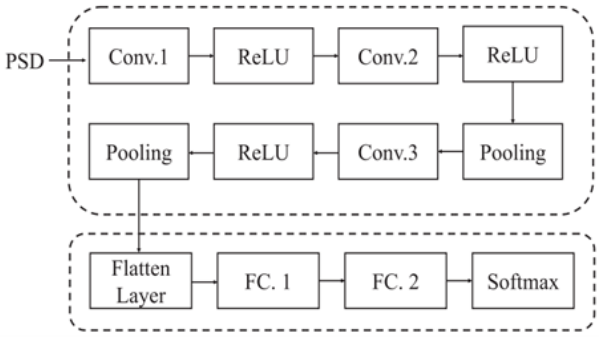


Fig. A.6: Diagrama de la arquitectura de un algoritmo ML CNN. [5]

B. ANEXO: Setup Experimental - Bibliografía

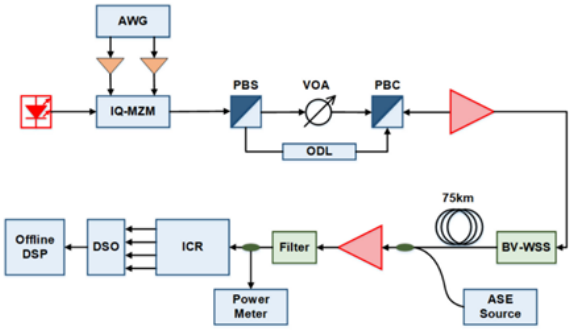


Fig. B.1: Setup óptico propuesto para los experimentos. [4]

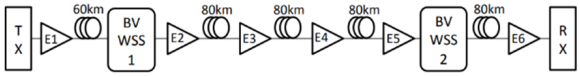


Fig. B.2: Esquema de enlace óptico propuesto. [5]

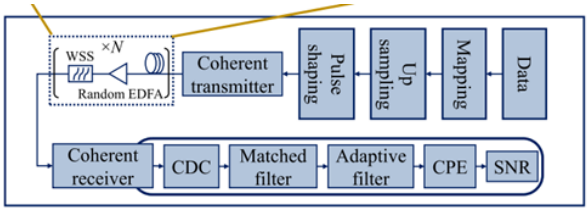


Fig. B.3: Setup montado para realizar pruebas de algoritmos de detección e identificación. [14]

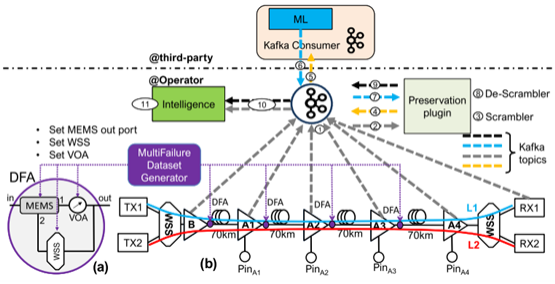


Fig. B.4: Setup montado para la detección se soft failures. [18]

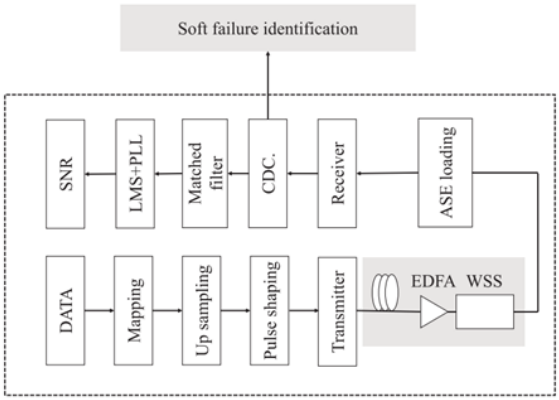


Fig. B.5: Setup experimental para identificación de soft failures. [18]

C. ANEXO: Especificaciones Técnicas

100/200G CFP2 DCO supported applications

	Condition	100G Metro Access	100G DP-QPSK Regional/LH	200G DP-8QAM Regional	200G DP-16QAM Metro
Baud rate	SDFEC, non diff	~28 Gbaud	~31 GBaud	~43 GBaud	~31 GBaud
Typical OSNR 10^{-15} post FEC	SDFEC, non diff	14 dB	12 dB	18 dB	21 dB
CD compensation	SDFEC, non diff	>2,000ps/nm	> 40,000 ps/nm	> 20,000 ps/nm	> 20,000 ps/nm
Rx sensitivity	>35dB OSNR	-30dBm	-35dBm	-29dBm	-27dBm
FEC (NCG)		Staircase (9.4 dB)	Staircase (9.4 dB)	N/A	N/A
	15% FEC overhead	-	SD FEC (10.6 dB)	SD FEC (10.9 dB)	SD FEC (11.0 dB)
Part numbers		TRB100DAA-01/TRB100EAA-01		TRB200DAA-01/TRB200EAA-01	
Availability (volumes)		April 2020 (GA)		April 2020 (GA)	

6.25GHz Tuning steps
ITU-T Fixed and Flex Grid
Long reach or High capacity
Up to 128 channels C band

Fig. C.1: Datasheet Transceiver DCO.

Specifications

Tipo de amplificador	Preamplificador	Longitud de onda de funcionamiento	1529nm~1561nm
Ganancia óptica	25dB±5dB	Ganancia plana	1.5dB (típica)
Potencia de entrada	-32dBm~-4dBm	Potencia de salida saturada	≤16dBm
Figura de ruido	5.5dB (típica)	Conector óptico	LC/UPC
Modo de funcionamiento	AGC/APC	Gestión	SNMP v2, WEB
Temperatura de funcionamiento	-10 a 50°C (14 a 122°F)	Temperatura de almacenamiento	-20 a 80°C (-4 a 176°F)
Consumo de energía	≤15W	Carcasa	Módulo enchufable (ocupa 1 ranura en el chasis administrado M6200)

Fig. C.2: Especificaciones técnicas de operación de EDFA.

Especificaciones

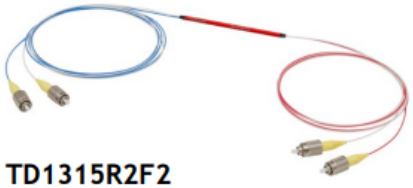
Tipo de atenuación	Oscura o brillante	Rango de longitud de onda	1520~1570nm
Atenuación	0~20dB	Pérdida de inserción (con PD)	2.5dB
Pérdida de dependencia de longitud de onda	0-10dB ≤0.6dB 10-20dB ≤1.5dB	Precisión de monitoreo de potencia	±1.0dB
Tiempo de respuesta de atenuación	300ms	Pérdida de retorno	≥40dB
Temperatura de funcionamiento	-5 a 70°C (23 a 158°F)	Temperatura de almacenamiento	-40 a 85°C (-40 a 185°F)

Fig. C.3: Especificaciones técnicas de operación de SFP VOA.

Fig. C.4: Especificaciones Técnicas de WSS.

THORLABS

Dual-Window Fiber Optic Coupler 1310/1550 nm, 90:10 Ratio



TD1315R2F2

Description

Thorlabs' 2x2 TD1315R2F2 1310/1550 nm dual-window fiber coupler offers a ±40 nm bandwidth at each wavelength. It is designed to minimize excess loss between input and output ports.

Specifications

TD1315R2F2	
Coupling Ratio[a]	90:10
Coupling Ratio Tolerance	±3.0%
Center Wavelength	1310/1550 nm
Bandwidth[a]	±40 nm
Excess Loss[a]	≤0.15 dB (Typ.)
Insertion Loss[a]	≤0.8 dB / ≤11.7 dB
Polarization-Dependent Loss (PDL)[a]	≤0.2 dB
Optical Return Loss (ORL) / Directivity[a]	≥60 dB
Max Power Level[b]	1 W (with Connectors or Bare Fiber) 5 W (Spliced)
Fiber Type[c]	SMF-28
Port Configuration	2x2
Fiber Lead Length and Tolerance	0.8 m +0.075 m/-0.0 m
Connectors	2.0 mm Narrow Key FC/PC
Package Size	Ø0.12" x 2.76" (Ø3.2 mm x 70.0 mm)
Jacket	Ø900 µm Hytrel® Loose Tube
Pigtail Tensile Load	10 N
Operating Temperature Range	-40 to 85 °C
Storage Temperature Range	-40 to 85 °C

a. All values are specified at room temperature at each center wavelength without connectors and measured through the white input port as indicated below.
b. Specifies the total maximum power allowed through the component. Coupler performance and reliability under high-power conditions must be determined within the user's setup. See Usage Tips for safety and handling information.
c. Corning SMF-28 fiber type will be specified on the documentation that ships with the coupler.

Fig. C.5: Especificaciones Técnicas Splitter.

D. ANEXO: Código

```
1
2  import paramiko
3  import re
4  import time
5  import pandas as pd
6  from datetime import datetime
7  import os
8  import pyvisa # type: ignore
9  import numpy as np # type: ignore
10 import datetime
11 import matplotlib.pyplot as plt # type: ignore
12
13
14 #Funcion para obtener datos de OSA ANRITSU OPTRONICA Y CASSINI AL MISMO
       TIEMPO
15 #Como los obtiene al mismo tiempo, datos estan correlacionados entre ellos
16
17 #funcion principal
18 def main():
19     print("\n")
20     # Información de conexión
21     #cual cassini ocuparas?
22     cassini = 1 # 1 (arriba) 2 (abajo)
23
24
25     ip_osa = '10.100.2.44'
26
27     #RECOMENDACION, ELEGIR DOS NUMEROS DE DURATION Y TIEMPO_MUESTRA QUE SEAN
           DIVISIBLES ENTRE ELLOS
28     duration = 20  # Duración en segundos
29     tiempo_muestra = 5 #tiempo de muestreo (tiempo entre muestras)
30
31     if cassini == 1:
32         ip_address = "10.100.2.102"
33         medicion_conexion(ip_address, ip_osa, cassini,duration,tiempo_muestra
               ) #llamamos a la funcion que hace las mediciones
34     elif cassini == 2:
35         ip_address = "10.100.2.103"
```

```
36          medicion_conexion(ip_address, ip_osa, cassini,duration,tiempo_muestra
                ) #llamamos a la funcion que hace las mediciones
37      else:
38          print("Debes seleccionar un cassini (1 o 2) para poder continuar")
39
40  #realiza mediciones cada (tiempo_muestra) segundos
41  def medicion_conexion(ip_address,ip_osa,cassini,duration,tiempo_muestra):
42      # Crear figura y ejes para graficar
43      fig, ax = plt.subplots()
44
45
46      username = "ocnos"
47      password = "ocnos"
48      command = "show coherent-module 3"
49
50      cantidad_muestras = duration/tiempo_muestra  #OJO SIEMPRE ELEGIR DOS
            NUMEROS QUE SEAN DIVISIBLES ENTRE ELLOS
51      direccion_carpeta = "C:/Users/optronica/Desktop/conexion cassini/Muestras
            "
52
53      # Variables para almacenar datos
54      modulation_formats = []
55      pre_fec_bers = []
56      input_powers = []
57      osnr_estimates = []
58      timestamps = []
59      n_muestra = []
60      output_power = []
61      chromatic_disp = []
62      diff_group_delay = []
63
64
65      # Contador para el nombre del archivo Excel
66      file_counter = 1
67      contador = 0 #contador para archivos de excel de los datos del OSA
68
69      # Crear una instancia de SSHClient
70      ssh = paramiko.SSHClient()
71
72      rm = pyvisa.ResourceManager()
73
74      # Configurar la política para aceptar claves del servidor automáticamente
```

```
75      ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
76
77      # Variable para verificar si las sesiones de SSH (cassini) y del OSA está
            n abiertas o no
78      ssh_session_open = False
79      osa_session_open = False
80
81      try:
82          # Verificar si la sesión SSH ya está abierta
83          if not ssh_session_open:
84              # Conectar al dispositivo
85              ssh.connect(ip_address, username=username, password=password)
86              print("Sesion␣cassini:␣" + str(cassini) + "␣Iniciada␣con␣éxito!")
87
88              # Abrir un canal SSH
89              channel = ssh.invoke_shell()
90
91              # Esperar a que aparezca el prompt y enviar el comando 'enable'
92              channel.send("enable\n")
93              time.sleep(0.3)  # Espera para asegurarse de que el prompt haya
                    aparecido
94
95              # Marcar la sesión como abierta
96              ssh_session_open = True
97
98          #verificar si la sesion del OSA está abierta o no
99          if not osa_session_open:
100
101             obj = rm.open_resource('TCPIP0::' + ip_osa + '::inst0::INSTR')
102
103             # Solicitar identificación
104             obj.write('*IDN?')
105             print("Conexión␣con␣OSA␣(optronica)␣realizada␣con␣éxito:" + obj.
                    read()) #muesra marca, modelo y numero de serie del OSA
106
107             osa_session_open = True
108
109         start_time = time.time()
110         print("Iniciando␣medición␣-␣Duración:␣" + str(duration) + "[s]␣-␣
                Numero␣de␣muestras␣a␣tomar:␣" + str(cantidad_muestras))
111         print("\n")
112
```

```
113         #bucle while realiza las mediciones de manera reiterativa cada cierto
                tiempo
114         while (time.time() - start_time) < duration:
115             obj.write('SSI')
116             while int(obj.query('*OPC?')) != 1:
117                 print('...../n')
118                 time.sleep(0.1)
119 
120             # Leer datos de la medida
121             c = list(map(float, obj.query('DCA?').split(',')))
122             Signal_StatWavelength = c[0] #donde comienza el espectro medido
123             Signal_StopWavelength = c[1] #donde termina
124             Signal_Nt = int(c[2]) #cantidad de puntos utilizados para
                    graficar (lo entrega el OSA)
125 
126             # Enviar el comando deseado
127             channel.send(command + "\n")
128             time.sleep(0.2)  # Espera para dar tiempo de procesar y recibir
                    la salida
129 
130             # Inicializar una lista para almacenar la salida completa
131             full_output = []
132 
133             # Definir una función para manejar la salida paginada
134             def receive_output():
135                 output = channel.recv(4096).decode("utf-8")
136                 full_output.append(output)
137                 return output
138 
139             # Leer la salida hasta que ya no haya más páginas
140             while "--More--" in receive_output():
141                 channel.send("␣")  # Enviar un espacio para obtener más datos
142                 time.sleep(0.2)
143 
144             # Concatenar la salida completa en una sola cadena
145             output_string = "".join(full_output)
146 
147 
148             # Guardar datos en un archivo excel
149             timeH = datetime.datetime.now().strftime('%H')
150             timeM = datetime.datetime.now().strftime('%M')
151             timeS = datetime.datetime.now().strftime('%S')
```

```
152             time_junto = timeH + timeM + timeS
153
154             # Utilizar expresiones regulares para extraer los valores específicos
155             match_modulation_format = re.search(r"Modulation-format\s*:\s*([^\n]+)", output_string)
156             match_pre_fec_ber = re.search(r"Current␣PRE␣FEC␣BER\s*:\s*([^\n]+)", output_string)
157             match_input_power = re.search(r"Current␣Input␣Power\s*:\s*([\-\d\.]+␣\w+)", output_string)
158             match_osnr_estimate = re.search(r"Current␣OSNR␣Estimate\s*:\s*([^\n]+)", output_string)
159             match_output_power = re.search(r"Current␣Output␣Power\s*:\s*([^\n]+)", output_string)
160             match_chrom_disp= re.search(r"Current␣Chromatic␣Dispersion\s*:\s*([^\n]+)", output_string)
161             match_group_delay= re.search(r"Current␣Differential␣Group␣Delay\s*:\s*([^\n]+)", output_string)
162
163             Signal_Power_dBm = obj.query_binary_values('DBA?', datatype='d', is_big_endian=False) #decodificamos los bits que envia el OSA en formato double
164             Signal_Wavelength = np.linspace(Signal_StatWavelength, Signal_StopWavelength, Signal_Nt) #graficamos los puntos en longitud de onda
165
166             # creamos una matriz que tenga "Nt" filas y 5 columnas (dependiendo de la cantidad de variables recogidas)
167             df_osa = pd.DataFrame({
168                 'Potencia␣[dBm]': Signal_Power_dBm,
169                 'Wavelength␣(um)': Signal_Wavelength,
170                 'Start␣(um)' : Signal_StatWavelength,
171                 'Stop␣(um)' : Signal_StopWavelength,
172                 'N␣Points' : Signal_Nt
173                 })
174
175             filename = 'C:/Users/optronica/Desktop/conexion␣cassini/Muestras/' + time_junto + "_"+ str(contador+1) +'.xlsx'
176             df_osa.to_excel(filename, index=False)
177
178             # Grafico (solo visual), que va mostrando la forma del espectro mientras se tomen las mediciones
```

```
179             ax.clear()
180             ax.plot(Signal_Wavelength, Signal_Power_dBm)
181             ax.set_xlabel('Longitud de onda')
182             ax.set_ylabel('Potencia')
183             ax.set_title(f'Gráfico de Potencia vs Longitud de onda - Medición
                    {contador+1}')
184             plt.pause(0.1)
185 
186             if all(match is not None for match in [match_modulation_format,
                    match_pre_fec_ber, match_input_power, match_osnr_estimate,
                    match_output_power,match_chrom_disp,match_group_delay]):
187                 n_muestra.append(contador)
188                 modulation_formats.append(match_modulation_format.group(1).
                        strip())
189                 pre_fec_bers.append(match_pre_fec_ber.group(1).strip())
190                 input_powers.append(match_input_power.group(1).strip())
191                 osnr_estimates.append(match_osnr_estimate.group(1).strip())
192                 output_power.append(match_output_power.group(1).strip())
193                 chromatic_disp.append(match_chrom_disp.group(1).strip())
194                 diff_group_delay.append(match_group_delay.group(1).strip())
195                 timestamps.append(timeH+":"+timeM+":"+ timeS)
196 
197             tiempo_restante = duration - (tiempo_muestra*contador)
198             print("Tiempo restante de medición: " + str(tiempo_restante) + "[
                    s] - Muestras restantes: " + str(cantidad_muestras))
199             contador +=1
200             cantidad_muestras -= 1
201 
202             time.sleep(tiempo_muestra-1)  # se le suma +1 al tiempo entre
                    iteracion debido al tiempo de respuesta de los cassini
203 
204 
205 
206     finally:
207         # No cerrar la conexión SSH si la sesión está abierta
208         if ssh_session_open:
209             ssh.close()
210             obj.close()
211 
212     # Crear un DataFrame de pandas con los datos
213     df = pd.DataFrame({
214         "Time": timestamps,
```

```
215         "N": n_muestra,
216         "Modulation Format": modulation_formats,
217         "PRE FEC BER": pre_fec_bers,
218         "Input Power [dBm]": input_powers,
219         "OSNR Estimate [dB]": osnr_estimates,
220         "Chromatic Disp [ps/nm]" : chromatic_disp,
221         "Diff Group Delay [ps]": diff_group_delay,
222         "Output Power [dBm]": output_power
223     })
224 
225     # Nombre del archivo Excel con el contador
226     excel_filename = f"datos_coherent_module{file_counter}.xlsx"
227     excel_filepath = os.path.join(direccion_carpeta, excel_filename)
228 
229     # Verificar si el archivo ya existe y aumentar el contador si es
        necesario
230     while os.path.exists(excel_filepath):
231         file_counter += 1
232         excel_filename = f"datos_coherent_module{file_counter}.xlsx"
233         excel_filepath = os.path.join(direccion_carpeta, excel_filename)
234 
235     # Guardar el DataFrame en un archivo Excel
236     df.to_excel(excel_filepath, index=False)
237     print("\n")
238     print("Medición terminada con éxito!")
239     print("\n")
240 
241 
242 main() #ejecutamos la funcion main para colocar el codigo de esta funcion
    arriba
```

Listing D.1: Código conexión OSA-Cassini