UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MECÁNICA

DETECCIÓN DE DAÑO EN PUENTE MEDIANTE ALGORITMOS DE NOVELTY DETECTION

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MECÁNICO

YONJAIRO SANDOVAL HUENCHUAL

PROFESORA GUÍA:
VIVIANA MERUANE NARANJO

MIEMBROS DE LA COMISIÓN:
RUBÉN BOROSCHEK KRAUSKOPF
ENRIQUE LÓPEZ DROGUETT

Este trabajo ha sido parcialmente financiado por Proyecto FONDEF ID17I10018.

SANTIAGO DE CHILE
2020

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL MECÁNICO
POR: YONJAIRO SANDOVAL HUENCHUAL
FECHA: 2020
PROF. GUÍA: VIVIANA MERUANE NARANJO

DETECCIÓN DE DAÑO EN PUENTE MEDIANTE ALGORITMOS DE NOVELTY DETECTION

En el marco del proyecto FONDEF ID17I10018 "Gestión de Inspecciones en Puentes de Acero basada en Monitoreo y Pronóstico de Daño Mediante Integración de Sensores y Procesamiento de Imágenes" se está desarrollando un puente prototipo a escala de laboratorio al que se le introducirá daño por fatiga. El objetivo del presente trabajo de título es implementar algoritmos de detección de daño a partir del monitoreo de vibraciones en un puente atirantado. Estos algoritmos luego serán implementados en el problema del puente prototipo y serán capaces de detectar la presencia de daño en el puente, por lo que deben ser probados de antemano.

Para esto, se implementaron métodos de Novelty Detection, los cuales son mecanismos de detección de novedades en un sistema, definiendo un estado "normal" de funcionamiento. A partir de este estado se generan pruebas para clasificar dichas novedades a partir de nueva información de lectura del sistema. Existen diversos métodos de Novelty Detection, algunos basados en Redes Neuronales Convolucionales, Variational AutoEncoders, Auto-Associative Networks, entre otros. Dentro del alcance de este trabajo de título está evaluar varios de los distintos tipos de algoritmos para evaluar la existencia de una falla en el puente. Estos algoritmos consisten en una CNN, un VAE clásico y un VAE combinado con clasificación de una clase de SVM.

Para la prueba de estos algoritmos se dispuso de datos obtenidos de una barra excitada por un shaker y datos de un puente atirantado, instrumentado con sensores y excitado por factores ambientales. Los escenarios de daño fueron simulados con distintas estrategias, dependiendo del caso de estudio.

El algoritmo que obtuvo mejores resultados fue el basado en VAE, seguido por el que mezcla VAE con un SVM de una clase. Ambos algoritmos resultan tener un alto grado de exactitud en la clasificación de novedades en el caso de estudio de la barra, y con menor exactitud para el caso del puente pero con oportunidades de mejora reconocibles. La CNN queda descartada por su bajo rendimiento.

Por último, se afirma que las redes VAE diseñadas cumplen con la función de identificar si una representación del espectro de frecuencias entrante corresponde al estado normal o alterado de la estructura, con distintos grados de exactitud, proyectando su aplicación al puente prototipo y casos de puentes reales después de determinar el nivel de daño mínimo que se necesita identificar.

Especialmente a mi papá y mamá,
por darme la oportunidad de estudiar,
gracias.

Agradecimientos

No puedo empezar sin insistir en el agradecimiento a mis padres. Sé todo lo que han sacrificado para que yo y mis hermanas tengamos la mejor educación posible. Gracias a mi padre por incentivar mi pensamiento crítico y mi resilencia, por mostrarme que a pesar de la caída, lo importante es levantarse para vivir un día más. Gracias a mi madre, por ayudarme a entender y cuidar mis emociones, por señalarme siempre el camino hacia la clase de hombre que quiero ser. Gracias a mis hermanas mayores, Daniela y Leslie, que desde pequeño me han ayudado a aprender y me han cuidado en cada etapa de mi vida.

Dentro de la Universidad hice muchos amigos y amigas que han significado mucho para mi. Desde Plan Común: Pau, Chino, Juan Pablo, Gualla, Belu, Sami, Dani, Hori, Scarlett, Bruno, Dasla, Malú, Seba, Provi, Felino, Negro, Guille, Pancho, Josemi, Potin. Les llevo en el corazón, infinitas gracias por todos los momentos. Los almuerzos en la U en el piño más grande que había, las tardes de estudio en la biblioteca, las noches de carrete o juego que no se acaban. Las risas no faltaron, definitivamente fueron los mejores años de mi vida.

Gracias a mis compañeros y compañeras de mecánica. El departamento es chico, la carrera es difícil, pero ustedes ciertamente transformaron mi experiencia ahí en un lugar acogedor. Gracias a todos y todas quienes se preocuparon por mi en algún momento, que me ayudaron con algo o que me permitieron hacer lo mismo y formar parte de sus vidas. Especialmente a ustedes, Meli, Kidel, Carlos. Gracias a los cabros de UniHeat (Ale, Nacho, Cristian), nunca trabajé tan a gusto como con ustedes, el mejor equipo y los mejores shots de leche con plátano.

Al Team China, por el mejor viaje que he tenido en la vida. A ustedes de nuevo, Pauli y Chino, las otras puntas del Tridente, por todo el amor y apoyo que me han dado a lo largo de los años (me salen hasta en la sopa, les tkm). A ti, Cata, porque gracias a ti he aprendido muchísimo de mi mismo. Gracias por ser mis ojos cuando no puedo ver claramente y sostenerme cuando dudo de mi mismo. A ti Cony, por darme ánimos y buenas vibras siempre durante esta última etapa, y por contenerme y escucharme cuando necesitaba a alguien que lo hiciera.

A quienes se sumaron en el camino, todos y todas con quienes tuve la oportunidad de compartir en esta etapa, valoro y atesoro cada conversación y cada momento vivido. Gracias totales.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Antecedentes Generales

1.1.1. Monitoreo de Salud Estructural

Las construcciones civiles, como puentes y edificios, están sometidas constantemente a ciclos de carga por su uso. Esto, sumado a factores medioambientales, genera un deterioro en el tiempo sobre dichas estructuras que puede derivar en fallas catastróficas. En el marco de estas circunstancias se presenta el Monitoreo de Salud Estructural (SMH por su nombre en inglés, *Structural Health Monitoring*), como el proceso de implementación de estrategias de detección y caracterización de daños en estructuras de este tipo [1].

El problema de identificación de daño en estructuras ha sido estudiado ampliamente en la literatura y por investigadores de distintas regiones durante los últimos años. Típicamente, el sistema de estudio es intervenido con la colocación de sensores como galgas extensiométricas, sensores ópticos o acelerómetros, siendo estos últimos los que entregan la información necesaria para realizar análisis vibracional sobre la estructura, lo que permite la identificación del daño.

Múltiples algoritmos de procesamiento han sido usados en esta labor, la mayoría basándose en lectura de vibraciones. En particular, durante los últimos años las ANN (Redes Neuronales Artificiales o *Artificial Neural Networks*) han sido utilizadas en el monitoreo de salud estructural, a través de modelos que integran la teoría de análisis vibracional. Gracias a esto es posible obtener información precisa sobre la condición de componentes, ensambles y el sistema en su conjunto. La mayor parte de los métodos establecidos que usan análisis vibracional se basan en establecer una correlación entre el comportamiento de la estructura/máquina y el nivel de vibraciones [2]. En su tesis de doctorado, Rytter [3] describe un nivel de jerarquía para el problema de detección de fallas, basado en cuatro niveles:

1. *Detección.* El método indica que la estructura presenta una falla.
2. *Localización.* El método da información sobre la ubicación más probable de la falla.
3. *Evaluación.* El método da un estimado del alcance o nivel de la falla.
4. *Predicción.* El método ofrece otro tipo de información de relevancia, como una estimación de la vida residual de la estructura.

El nivel sobre el cual opere un algoritmo dependerá en gran parte del tipo de datos que se le entregue. A su vez, para llegar a los últimos niveles hará falta un algoritmo potente que sea capaz de hacer uso de varios datos (respuestas vibracionales, imágenes, etc.) y procesarlos eficientemente.

1.1.2. Análisis vibracional para detección de fallas

El análisis vibracional ha sido una herramienta muy importante en los sistemas de mantenimiento predictivo en las últimas décadas. La instalación de sensores como acelerómetros es un método no invasivo, por lo que el funcionamiento de estos no requiere un desmontaje o detenimiento del sistema sobre el cual operan.

Este estudio mediante análisis se basa en el principio teórico de que cada estructura tiene su forma propia de vibrar en condiciones normales de funcionamiento. En consecuencia, esta vibración se ve alterada al alterar dichas condiciones, por algún estímulo externo o daño causado la estructura. De esta forma, si se mantiene un registro del estudio de vibraciones de la estructura, se consigue inferir la presencia de grietas u otros fenómenos perjudiciales para el sistema.

La aplicación de métodos basados en análisis vibracional para el Monitoreo de Salud Estructural es especialmente potente para el caso de un puente. Sin embargo, los cambios en la frecuencia fundamental y en los modos vibracionales causados por daño son usualmente pequeños. Durante los últimos años se han desarrollado algoritmos que permiten detectar incluso estas variaciones en puentes fuera del laboratorio: sea por un modelo de elementos finitos de puentes existentes[4] o por la obtención de respuestas temporales en puentes reales[5][6].

1.2. Motivación

Dentro de las obras civiles más importantes de un país se encuentran los puentes; permiten conectar el territorio para flujos de habitantes, mercancías, componentes de la industria, etc. Estas estructuras se exponen a un deterioramiento continuo debido a sobrecargas, uso excesivo, envejecimiento de materiales y condiciones externas como terremotos o fuertes vientos. Por lo mismo, es importante implementar medidas de inspección adecuadas, para evitar casos como el colapso del puente Loncomilla en Noviembre del 2004 (que condujo a la muerte de 10 personas) o del puente ferroviario Pitrufquén en Agosto del 2016.

En 2010, el Ministerio de Obras Públicas presentó el informe *"Chile 2020: Obras Públicas para el Desarrollo"* [7]. En dicho informe, se anunció una intervención en términos de construcción de nuevos puentes, conservación, reparación y reposición de 542 estructuras mayores y menores a 40 m. Ahora más que nunca es necesario el desarrollo de tecnologías que permitan anticiparse a las fallas de componentes.

Esta memoria se enmarca en el proyecto FONDEF ID17I10018 "Gestión de Inspecciones en puentes de acero basada en monitoreo y pronótico de daño mediante integración de sensores y procesamiento de imágenes". Se proporcionará un sistema de inspección automático de estructuras, que sea capaz de detectar, localizar y cuantificar riesgos de falla que serían imposibles de detectar al ojo humano.

1.3. Objetivos

1.3.1. General

Implementar algoritmos de detección de daños a partir de monitoreo de vibraciones en un puente. El algoritmo debe ser capaz de identificar correctamente el estado del puente, tomando como único input el espectro de frecuencias del sistema.

1.3.2. Específicos

Para cumplir el objetivo general recién mencionado, se postulan los siguientes objetivos específicos:

- Diseñar arquitecturas coherentes basadas en redes neuronales.
- Generar una metodología para procesar los datos obtenidos, y adaptarlos para poder alimentar las redes neuronales.
- Evaluar cada arquitectura de acuerdo a los parámetros escogidos para cada una.

1.3.3. Alcances

Según lo descrito en 1.1.1, los algoritmos de identificación de fallas se pueden organizar según 4 niveles. El alcance de este proyecto consiste en la **detección** de fallas por lo que se limita al primer nivel. Esta detección debe basarse únicamente en la clasificación de la respuesta en frecuencia de la estructura.

Se construyen tres algoritmos de detección, lo que permite comparar sus rendimientos de forma tal que se identifique cuáles metodologías se adaptan mejor al problema.

Capítulo 2

Análisis Bibliográfico

2.1. Análisis Espectral

2.1.1. Transformada de Fourier

La información que se obtiene gracias a los acelerómetros es muy valiosa, pero requiere de un procesamiento para ser interpretada de mejor manera por los algoritmos. En este caso, la transformada rápida de Fourier permite determinar las señales sinusoidales que contiene la señal compleja y mostrarlas en el dominio de frecuencias.

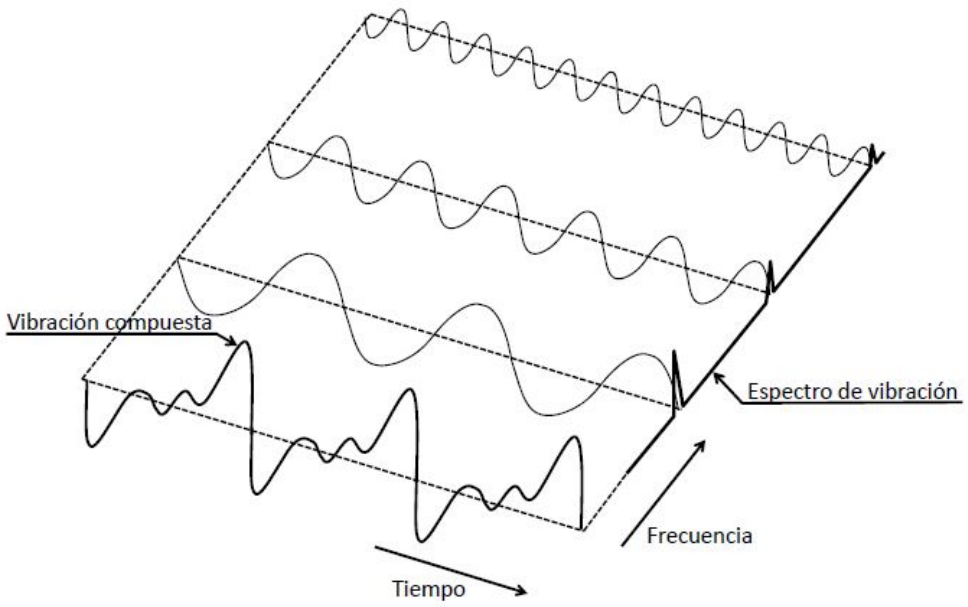

Figura 2.1: Espectro de una señal de vibración compleja [8].

La Figura 2.1 muestra una señal de vibración compuesta en el dominio del tiempo, y como esta se descompone en señales sinusoidales para mostrarse en el dominio de frecuencia. De esta manera, si g(t) es una señal periódica de periodo T, entonces posee una representación en serie de Fourier dada por:

$$g(t) = \frac{a_0}{2} + \sum_{\mathrm{i}=1}^{\infty}(a_{\mathrm{i}}\cos(\mathrm{i}w_0 t) + b_{\mathrm{i}}\cos(\mathrm{i}w_0 t)) \tag{2.1}$$

donde $w_0 = 2\pi/T$ es la frecuencia angular fundamental del sistema en rad/s [8].

Integrando término a término la serie en 2.1, se obtienen los coeficientes de Fourier según las siguientes expresiones:

$$a_{\mathrm{i}} = \frac{2}{T}\int_0^T g(t)\cos(\mathrm{i}w_0 t)\mathrm{d}t \tag{2.2}$$

$$b_{\mathrm{i}} = \frac{2}{T}\int_0^T g(t)\sin(\mathrm{i}w_0 t)\mathrm{d}t \tag{2.3}$$

Definiendo $\Delta f = 1/T$, y en consecuencia $w_0 = 2\pi\Delta f$, la serie queda como:

$$g(t) = \sum_{\mathrm{i}=-\infty}^{+\infty} G(\mathrm{i}\Delta f)\mathrm{e}^{j2\pi\mathrm{i}\Delta f t} \tag{2.4}$$

Así, es posible definir un nuevo coeficiente de Fourier que considera senos y cosenos, que viene dado por:

$$G(\mathrm{i}\Delta f) = \frac{1}{T}\int_0^T g(t)\mathrm{e}^{-j2\pi\mathrm{i}\Delta f t}\mathrm{d}t, \tag{2.5}$$

donde t: tiempo; i: entero que define la periodicidad de la señal; Δf: espaciado de frecuencias.

En la expresión 2.5, $G(\mathrm{i}\Delta f)$ corresponde al espectro de la señal de función $g(t)$, la que generalmente posee valores complejos. Es importante notar que la señal es continua, y mientras esto se cumple en los casos reales (como el estudio de las vibraciones de un puente), no es directamente aplicable dado que la adquisición de datos por medio de sensores se realiza en intervalos de tiempo, es decir, de forma discreta. Con la consideración anterior:

$$g(n\Delta t) = \frac{1}{f_s}\int_0^{f_s} G(f)\mathrm{e}^{j2\pi f n\Delta t}\mathrm{d}f \tag{2.6}$$

$$G(f) = \sum_{-\infty}^{+\infty} g(n\Delta t)\mathrm{e}^{-j2\pi f n\Delta t} \tag{2.7}$$

donde n: entero contando el número de pasos de tiempo; Δt: intervalo de muestreo; f_s: frecuencia de muestreo tal que $f_s = 1/\Delta t$.

A partir de las expresiones 2.6 y 2.7, se obtiene la Transformada Discreta de Fourier, asumiendo la hipótesis de periodicidad de un muestreo temporal de la señal:

$$g(n\Delta t) = \frac{1}{f_s}\sum_{\mathrm{i}=0}^{N_s-1} G(\mathrm{i}\Delta f)\mathrm{e}^{j2\pi n\mathrm{i}/N_s} \tag{2.8}$$

$$G(\mathrm{i}\Delta f) = \frac{1}{N_s}\sum_{n=0}^{N_s-1} g(n\Delta t)\mathrm{e}^{-j2\pi n\mathrm{i}/N_s} \tag{2.9}$$

con N_s: número de datos. Así: $T = N_s\Delta t$ y $f_s = N_s\Delta f$.

El costo computacional de la Transformada Discreta de Fourier es bastante alto: toma N_s^2 operaciones para evaluar. La Transformada Rápida de Fourier (FFT por su nombre en inglés) es un algoritmo ampliamente usado en el área de la informática que permite el cálculo de la Transformada Directa de Fourier en un número de operaciones igual a $N_s log_2 N_s$.

Es gracias a la mejora sustancial en tiempo y en exactitud, que la FFT es una herramienta fundamental en todos los sistemas de análisis y tratamiento de señales. En la actualidad es el núcleo de todos los procesadores de señal y está incluida en varias bibliotecas de análisis en Python, Matlab, etc.

2.1.2. Leakage

Para aplicar la DFT se requiere que la señal sea periodica, como se comenta en la sección anterior. Dado que los datos están sujetos a un periodo de adquisición finito, sea T, la aplicación de la transformada rápida asume que la señal es periódica con periodo T. Cuando la señal no cumple esta condición se produce un error en la hipótesis de periodicidad llamado *leakage*, ilustrado en la figura 2.2 que muestra una señal tipo coseno que es periódica en T y otra que no lo es.

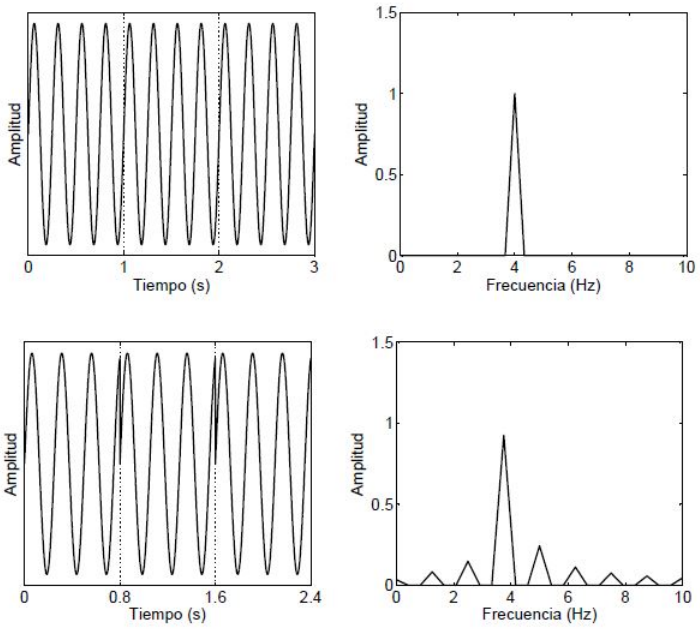

Figura 2.2: Hipótesis de periodicidad. Leakage[8]

Uno de los métodos más populares para reducir el *leakage* es multiplicar la señal por una ventana cónica. La idea es reducir las discontinuidades en los extremos de la serie gracias al truncamiento de las de dichos extremos[9], forzando la señal a ser periódica. La figura 2.3 muestra algunas de las ventanas más utilizadas, mientras que en la figura 2.4 se ilustra el efecto del *windowing* en una señal sinusoidal cuya duración no es múltiplo de su periodo T.

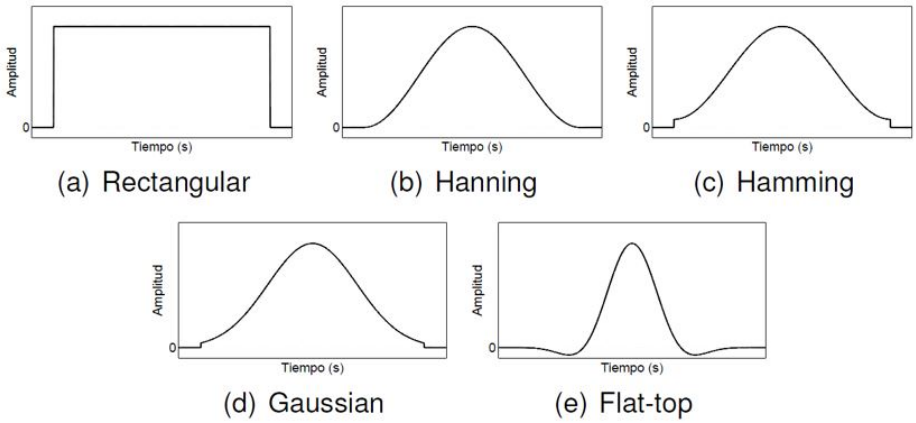

Figura 2.3: Ventanas comunes usadas para *windowing*[10].

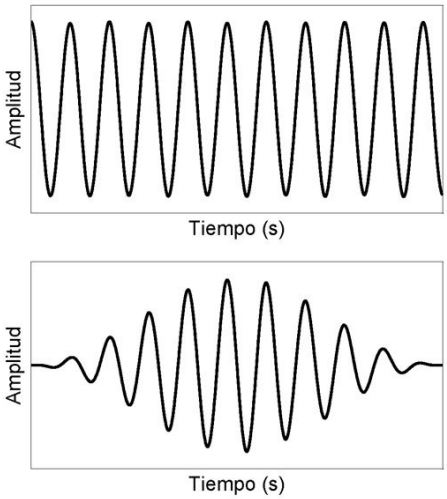

Figura 2.4: Efecto de multiplicar una ventana en una sinusoide [10].

2.2. Novelty Detection

En términos de detección de fallas (nivel 1), recientemente se ha establecido una línea de trabajo en torno a lo que se conoce como *Novelty Detection* (o *Detección de Novedades* en español), concepto que engloba métodos de distintas disciplinas. Como su nombre indica, Novelty Detection corresponde a la identificación de *novedades* en un patrón de datos analizado por una red neuronal u otros recursos computacionales entrenados para definir un estado "normal" de funcionamiento del sistema [2].

Los métodos de Novelty Detection permiten representar un rango de mediciones saludables para algún componente, usando parámetros de una condición sin daño estructural. Así, a través de nuevas lecturas de monitoreo de vibraciones, se logra detectar una falla si esta llega a ocurrir. En este punto es importante establecer una diferencia entre Detección de Anomalías (que también es ampliamente utilizado) y Detección de Novedades. Cuando se habla de detectar anomalías, se entiende que existe un conjunto de datos, dentro de los cuales hay uno o más que difieren en gran medida del resto. Mientras que en el caso de detección de novedades, solo se dispone de la data correspondiente al estado normal de funcionamiento del sistema, con lo que el objetivo es detectar *outliers* o casos aislados con los parámetros de una nueva data, en contraste a los parámetros del estado normal.

Existen numerosos métodos para desarrollar algoritmos de Novelty Detection, basados en distancias ordinarias[11], estimaciones de densidades de probabilidad[12], redes neuronales artificiales[1][2][13][14][15][16] y análisis de ondículas[17]. Dada la amplia variedad de redes neuronales que pueden utilizarse como algoritmos de detección de novedades, existen cada vez más trabajos de investigación ligados a estas sobre SHM.

2.3. Redes Neuronales Artificiales

A finales de los ochenta surgieron varios trabajos de investigación que demostraban la factibilidad del uso de Redes Neuronales Artificiales (ANN por su nombre en inglés, *Artificial Neural Networks*) en la detección de daño de estructuras[1]. Actualmente existen grupos en todo el mundo que se dedican a realizar trabajos de investigación en torno a las ANN debido a su amplia gama de aplicaciones. Para entender los algoritmos más complejos empleados en el presente trabajo es necesario revisar los conceptos básicos de este tópico.

Los modelos esenciales de redes neuronales se conocen como *feed-forward neural networks* o *Multi-layer Perceptron*, y son modelos que buscan aproximar una función f^* a través de un mapeo que aprende de los parámetros posibles. Estos modelos son típicamente compuestos por varias funciones que se 'conectan' aplicándose como una composición de funciones. Así, la cantidad de funciones conectadas define la profundidad de la red.

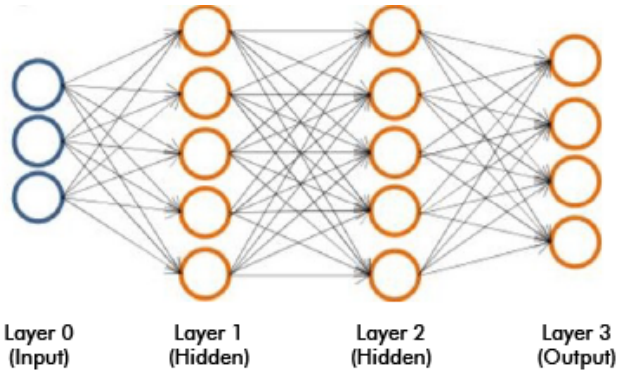

Figura 2.5: Representación esquemática de una red neuronal MLP de 2 capas ocultas.

En el contexto presentado, una red *feed-forward* se representa como una colección de elementos procesadores conectados, conocidos como nodos o neuronas, ordenados en capas. Cada nodo i está conectado a cada nodo j en la capa predecesora a través de una conexión de peso w_{ij} (ver figura 2.5). Así, la señal pasa a través de cada nodo según la sumatoria de las señales de entrada por sus respectivos pesos, generando la excitación z_i. Esta lógica define entonces una función de activación f dada por,

$$x_i = f(z_i) = f(\sum_j w_{ij} x_j). \tag{2.10}$$

Un nodo en particular, llamado *bias*, se encarga de mantener una conexión con todas las neuronas de todas las capas ocultas y output para mantener un offset constante.

2.3.1. Autoencoders (AE)

Los *autoencoders* son redes neuronales artificiales entrenados con aprendizaje profundo no supervisado y que buscan reconstruir el input original en el output[14]. Un autoencoder se compone de dos partes: un *encoder* y un *decoder*.

$$h = \sigma(W_{xh}x + b_{xh}) \tag{2.11}$$

$$z = \sigma(W_{hx}h + b_{hx}) \tag{2.12}$$

El encoder, representado por la expresión 2.11, mapea el vector de entrada x a una representación h a través de una transformación no lineal σ, donde W y b son el peso y el bias de la red neuronal, respectivamente. Por otra parte, el decoder de la expresión 2.12 mapea la representación h al input original, como una reconstrucción que usa la misma transformación no lineal σ. La diferencia entre el vector original x y la reconstrucción z se conoce como error de reconstrucción, y es la función de pérdida sobre la cual la red aprende. La figura 2.6 muestra la estructura típica de un autoencoder.

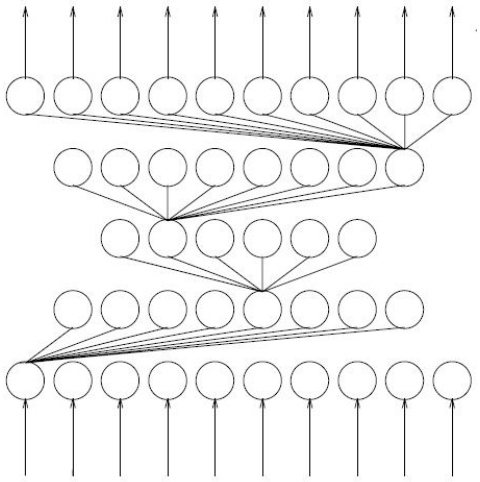

Figura 2.6: Representación esquemática de un AE[2].

La representación h es una capa oculta del input codificado de menor dimensionalidad que el conjunto de datos original. Esta representación también se conoce como *espacio latente* y corresponde a los atributos más importantes de la data procesada. La información que contiene el espacio latente es muy valiosa ya que corresponde a una reducción de la dimensionalidad de la data original, y puede ser usada como input de otro autoencoder, de forma de apilar autoencoders para formar un autoencoder profundo. Incluso puede ser usada como data de entrada en otro tipo de algoritmos para formar un algoritmo más robusto.

2.3.2. Variational Autoencoder (VAE)

La reconstrucción de probabilidad es un método que toma en cuenta la variabilidad de la distribución de variables. A su vez, un *autoencoder* es una red neuronal que se entrena bajo aprendizaje no supervisado para reconstruir datos a partir de un input.

Un *Variational Autoencoder* (VAE) es un modelo gráfico probabilístico dirigido[14] cuya función posterior es aproximada por una red neuronal, con la estructura de un *autoencoder*. En la figura 2.7 se ve de forma esquemática un modelo gráfico dirigido típico.

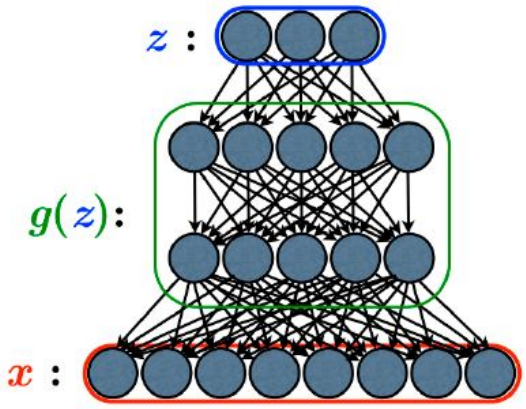

Figura 2.7: Representación esquemática de un modelo gráfico probabilístico dirigido[14].

En un VAE, la capa más alta del modelo z es tratada como una variable latente (variable que no se conoce y es inferida) donde el proceso generativo comienza, $g(z)$ corresponde al mecanismo complejo que permite la generación de los datos que forman parte de x. La función objetivo de un VAE es la minimización del valor de la verosimilitud marginal de la data.

La verosimilitud (*likelihood*) de un modelo está definida como una probabilidad condicional sobre los datos bajo el supuesto de un modelo dado. Así, la verosimilitud marginal (*marginal likelihood*) es la suma de la verosimilitud marginal de cada punto en la data de forma individual tal que $log\ p_\theta(x^{(1)}, ..., x^{(N)}) = \sum_{i=1}^{N} log\ p_\theta(x^{(i)})$, donde la verosimilitud marginal de cada punto de la data se puede reescribir como:

$$log\ p_\theta(x^{(i)}) = D_{KL}(q_\phi(z|x)||p_\theta(z)) + \mathcal{L}(\theta, \phi; x^{(i)}) \tag{2.13}$$

donde: $q_\phi(z|x)$ es la función posterior aproximada y $p_\theta(z)$ corresponde al prior de la variable latente z. El término $D_{KL}(q_\phi(z|x)||p_\theta(z))$ en la ecuación 2.13 es la divergencia de Kullback-Leibler de las aproximaciones de la posterior y el prior, y por último el término $\mathcal{L}(\theta, \phi; x^{(i)})$ es el límite inferior variacional de la verosimilitud marginal del i-ésimo punto de la data. Como la divergencia de KL es siempre mayor a cero, se tiene que:

$$log\ p_\theta(x^{(i)}) \geq \mathcal{L}(\theta, \phi; x^{(i)}) \tag{2.14}$$
$$= E_{q_\phi(z|x^{(i)})}\left[-log\ q_\phi(z|x) + log\ p_\theta(x|z)\right] \tag{2.15}$$
$$= -D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z)) + E_{q_\phi(z|x^{(i)})}\left[log\ p_\theta(x|z)\right] \tag{2.16}$$

donde $p_\theta(x|z)$ es la verosimilitud de la data x dada la variable latente z, $-D_{KL}(q_\phi(z|x^{(i)})||p_\theta(z))$ es la divergencia Kullback-Leiber entre la aproximación de la posterior y el prior de z. Finalmente, $E_{q_\phi(z|x^{(i)})}\left[log\ p_\theta(x|z)\right]$ se puede interpretar como la reconstrucción de x a través de la distribución posterior $q_\phi(z|x)$ y la verosimilitud $p_\theta(x|z)$.

La diferencia más sustancial entre un VAE y un autoencoder tradicional es la incorporación de modelos estadísticos que obligan a la red a aprender espacios latentes continuos y de gran estructuramiento. Un VAE, en lugar de comprimir la data de entrada a un código fijo en el espacio latente, la reduce a parámetros de una distribución estadística. De esta forma, se asume que la data de entrada fue generada por un proceso estadístico y no estocástico. En este contexto, la red neuronal representa la compleja relación existente entre los datos x y el espacio latente z.

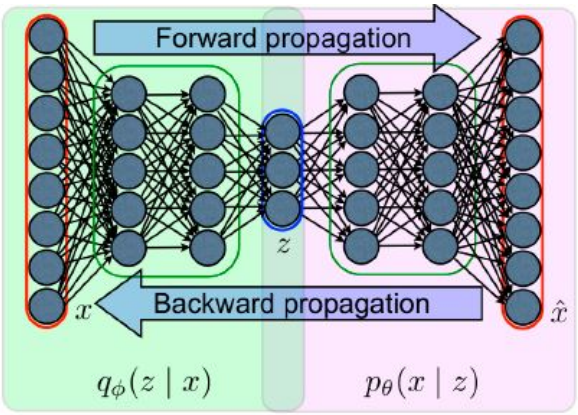

Figura 2.8: Representación esquemática de un VAE[14].

La figura 2.8 muestra un esquema explicativo de un VAE. La red aprende por *backpropagation* y elección más común sobe la distribución para la variable z es normal isotrópica. La distribución de la verosimilitud $p_\theta(x|z)$ depende de la naturaleza de los datos. Si la data es binaria, la distribución usada será Bernoulli. Si la data es continua, se usa una Normal Multivariable.

2.3.3. Convolutional Neural Networks (CNN)

Las Redes Neuronales Convolucionales (CNN por su nombre en inglés, *Convolutional Neural Networks)* corresponden a un tipo de red neuronal artificial donde las neuronas o nodos corresponden a campos receptivos multidimensionales, con lo que es capaz de trabajar con tensores de 2 o hasta 3 dimensiones. Dada su arquitectura, las CNN se han usado ampliamente en detección de objetos en imágenes, videos y clasificación de imágenes[1].

Una red neuronal común aprende patrones globales del input y los usa según sea su objetivo y su arquitectura. En contraste, las CNN aprenden patrones locales que luego pueden identificar o reconocer incluso si dicho patrón ha sido movido dentro del input, es decir, los patrones que aprenden no varían con la traslación. Por ejemplo, si una CNN aprende un patrón de una imagen en la esquina inferior derecha, luego podrá identificar este patrón si aparece en alguna otra esquina.

En la figura 2.9 se muestra la estructura de una red neuronal convolucional bidimensional de 5 capas, donde se aprecia que cada capa tiene una función específica en el tratamiento de imágenes. La capa de input solo acepta la información y la entrega a las capas convolucionales, las que se encargan de aplicar diferentes filtros a la imagen para extraer las características más importante de esta, aquellas que la definen. Estos filtros no son fijos y son entrenados por *back-propagation*.

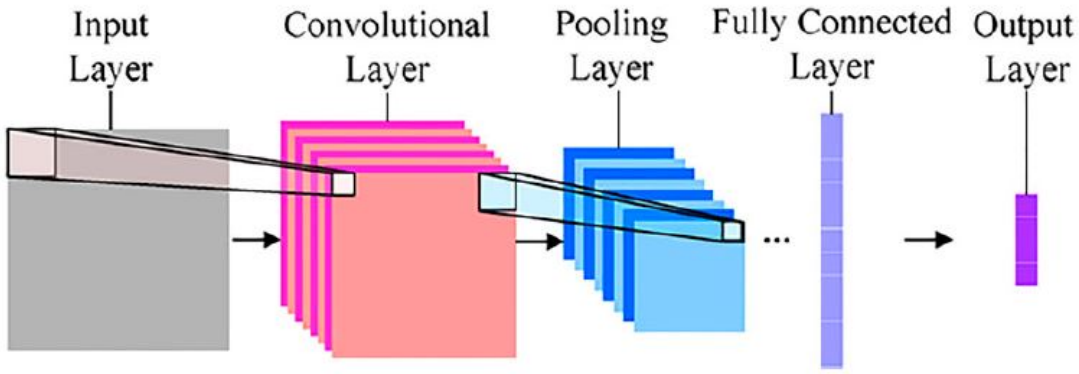

Figura 2.9: Representación esquemática de una CNN de dos dimensiones[1].

Para reducir la dimensión del output de la capa convolucional, se subsamplea este output en la capa de subsampleo o *pooling*. Finalmente, las capas correspondientes a conexión completa se comportan de manera similar a las capas ocultas de una red *feed-forward* y junto a la capa de output producen el vector de salida de la CNN.

Las CNN son usadas principalmente en el campo de visión computacional y clasificación de imágenes. Dado que la respuesta obtenida por los acelerómetros es una serie temporal, otras técnicas como regresión logística y otras redes neuronales de poca o alta profundidad pueden representar mejores soluciones si se dispone de una gran cantidad de mediciones[1].

Una CNN trabaja con tensores de distinta dimensionalidad. Una serie de tiempo puede interpretarse como un tensor de una dimensión, mientras que una imagen se puede representar como un tensor de dos dimensiones. Así, el uso de CNNs en este problema permite no solo depender de menos datos para el entrenamiento, sino que además representa una disminución del costo computacional al reordenar un tensor de una dimensión y obtener tensores de mayor dimensionalidad pero menor extensión en cada dimensión.

Convolución

Este arquetipo de red neuronal recibe su nombre por la operación matemática que emplea. La *convolución* no es más que la integral del producto de dos funciones, una de las cuales está desplazada respecto a la variable de integración[8]. Se define, entonces, como:

$$s(t) = (x * w)(t) = \int x(a)\ w(t-a)\ \mathrm{d}a \tag{2.17}$$

En la ecuación 2.17, la función x corresponde al input, mientras la función w se conoce como *kernel*. En este contexto, al output s se llama *feature map* y corresponde a la salida de la etapa convolucional de la red. Dado que el trabajo de los datos es un serie temporal obtenida por acelerómetros en tiempo discreto, conviene expresar la convolución discreta según:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{a=\infty} x(a)\ w(t-a) \tag{2.18}$$

A pesar de que la sumatoria es infinita, se puede calcular bajo el supuesto de que la función es 0 en todo su dominio, excepto en el tiempo de medición. La capa convolucional involucra una serie de filtros entrenables (como los pesos en el caso de las redes feed-forward), que se ajustan con *backpropagation*.

En estricto rigor, la operación que se da en las capas convolucionales es la función *cross-correlation*, que a su vez emplea una convolución de tal forma que:

$$f(t) \star g(t) = f(-t) * g(t) \tag{2.19}$$

Notar que a partir de la expresión 2.19, si alguna de las funciones f o g es par, la correlación cruzada es equivalente a la convolución. Finalmente, usando una imagen 2-D y un kernel que se condiga con esta imagen, la operación en las CNN se expresa como:

$$S(\mathrm{i}, j) = (X * W)(\mathrm{i}, j) = \sum_m \sum_n X(\mathrm{i} + m, j + n) W(m, n) \tag{2.20}$$

La convolución trabaja aplicando los filtros mediante la expresión anterior a cada ventana del input del tamaño del kernel. Cada uno de estos filtros transforma la imagen en un *feature map* y es replicada en todo el data set.

Maxpooling

Las capas de reducción (o *pooling layers*) son capas que cumplen la función de reducir el tamaño del output. El algoritmo típico en esta etapa es *maxpooling*, que disminuye la muestra agresivamente y opera según la ecuación 2.21.

$$y^k_{\mathrm{i},h} = max_{p,q}\, h^k_{j+p,j+q}, \tag{2.21}$$

con h^k como el *input feature map* e y^k como el *output feature map*. Las operaciones de pooling reducen la eficiencia operacional, sobre todo cuando se añaden más capas convolucionales. De todas formas, max pooling incrementa la invarianza traslacional, es decir, el valor reducido identificará el mismo máximo sin importar la posición en la que se encuentre[18].

Maxpooling consiste en extraer ventanas del *input feature map* y generar un *output feature map* con el máximo valor de cada canal. Conceptualmente, es bastante similar a la convolución, con la diferencia de que en lugar de transformar localmente con una transformación lineal entrenada, transforma con una función max.

Por lo general las CNN poseen una capa de pooling después de cada capa convolucional, lo que permite reducir el costo computacional. Después de esta secuencia de capas convolucionales y de reducción, típicamente se incluyen capas densas (totalmente conectadas) antes de entregar una predicción, tal como una red *feed-forward*.

One-Class Convolutional Neural Network

Muchos problemas de clasificación pueden abordarse como problemas binarios, en particular, el problema de detección de novedades (nivel 1) puede ser catalogado como tal. El problema de detección de fallas busca clasificar la data entrante en estados binarios: *defectuoso* o *no defectuoso*. Frente a esto, el entrenamiento de una red neuronal debe considerar que, en gran parte de los problemas, el escenario *defectuoso* no se conoce o no se encuentra disponible. De aquí nace la motivación de crear clasificadores de una clase.

La aplicación de un clasificador de una clase basado en una CNN no es trivial. Las redes convolucionales son especialmente buenas para extraer parámetros y clasificar datos cuando se conocen las clases con las que se está alimentando la red. En este contexto, una CNN logra clasificar datos más que aprender representaciones generalizadas de estos. Por lo tanto, desarrollar un clasificador de una clase basado en CNN como método de hacer *Novelty Detection* ha constituido todo un desafío.

Uno de los acercamientos que resultan más interesantes a este problema es el método usado por P. Oza y V. M. Patel[19]. La CNN de una clase que presentan los autores usa la etapa convolucional de una red pre-entrenada para entrenar un clasificador (que corresponde a las *fully connected layer* en la figura 2.9). Considerando la dificultad en la disponibilidad de datos que motiva a crear un clasificador de una sola clase (en adelante, la clase positiva), se crea una clase pseudo-negativa a partir de un sampleo de números aleatorios a partir de una distribución gaussiana. Esto permite que el clasificador diferencie la data que viene del *feature extractor* (capas convolucionales) de la data que viene del sampling de la distribución gaussiana. La figura 2.10 esquematiza este procedimiento.

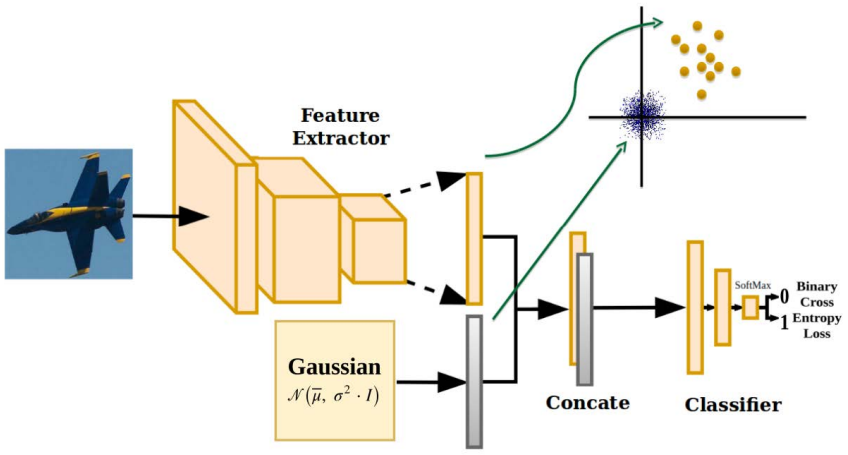

Figura 2.10: Diagrama de bloque del modelo propuesto por P. Oza y V. M. Patel. Aquí, $\bar{\mu}$ y σ son el promedio y la desv. estándar de una Gaussiana, respectivamente, e I es la matriz identidad[19].

2.4. Support Vector Machines (SVM)

Desarrolladas en los 90, las Máquinas de Soporte Vectorial (*Support Vector Machines*, SVM) han demostrado ser de mucha utilidad en problemas de clasificación de datos de distinta índole. Las SVM son una generalización de un clasificador más simple e intuitivo llamado clasificador de máximo margen (*maximum margin classifier*).

En el grafico izquierdo de la figura 2.11 se ven dos conjuntos de datos de distintas clases (C_1 y C_2). Suponiendo que son linealmente separables, existe una gran cantidad de hiperplanos que pueden servir para separar los datos. El clasificador de máximo margen entonces, como su nombre indica, es aquel clasificador que entrega el mayor margen posible entre ambos grupos, ilustrado en el gráfico derecho de la figura 2.11.

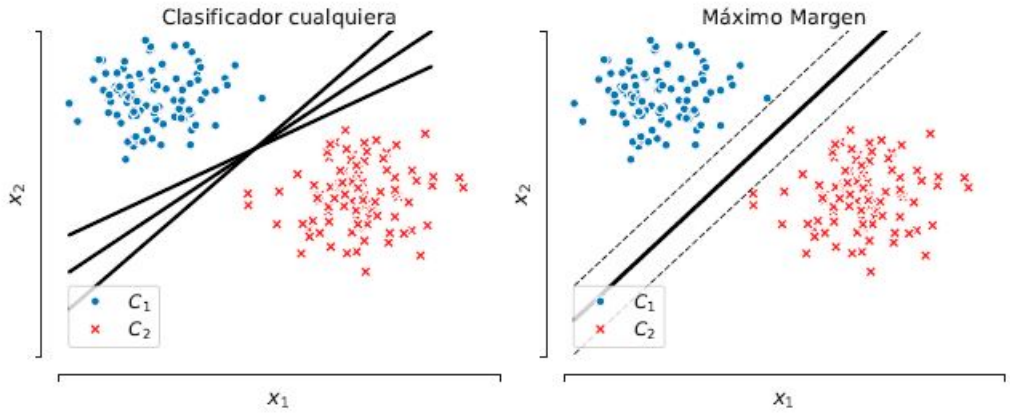

Figura 2.11: Clasificadores de datos vs modelo de máximo margen

El planteamiento anterior tiene dos debilidades bastante importantes. La primera es que en la práctica los datos no son necesariamente separables, y la segunda es que la llegada de nuevos datos puede alterar drásticamente el modelo. Existen diversos métodos para solucionar estos problemas, como el *Soft Margin* que agrega variables de holgura para clasificar mejor la mayoría de los datos, sacrificando algunos datos (que probablemente sean outliers). Los métodos de kernel solucionan de mejor forma estos problemas, introduciendo un mapeo de los datos mediante una función $\phi(x)$ (también llamada *feature map*) que agrega una dimensión al problema, en la cual las clases son separables.

La función $\phi(x)$ se define de forma particular para cada problema, proceso que puede resultar complicado. Sin embargo, muchas veces no se busca determinar la forma explícita de la función, si no más bien trabajar en base a ella, calculando $\phi(x)^T\phi(x)$ (esto es, el producto punto de los datos). En razón de lo anterior, conviene usar expresiones que expliciten la evaluación del producto punto en un espacio de alta dimensión, sin necesidad de encontrar el *feature map* explíciamente. Estas expresiones son funciones se conocen como *kernels*.

Un *kernel* es una función simétrica y definida positiva, que se puede descomponer de la forma:

$$K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle \tag{2.22}$$

En machine learning, el uso de esta herramienta se conoce como *kernel trick*. El *kernel* más utilizado es el *radial basis function (rbf) kernel*, por ser aplicable a distintos problemas de clasificación. El RBF se define como el mapeo de la similidaridad entre dos puntos x_1 y x_2 tal que:

$$K_{rbf}(x_1, x_2) = \sigma^2 \mathrm{e}xp\left(-\frac{(x_1 - x_2)^2}{2l^2}\right) \tag{2.23}$$

Las fronteras que entrega son suaves (infinitamente diferenciables) y tiene dos parámetros. El parámetro σ^2 es un parámetro de escala, y el parámetro l controla la osilación de la curva.

2.4.1. One-class SVM

One-class SVM (OCSVM) para aprendizaje no supervisado en detección de anomalías es una instancia de SVM que amplía la idea de clasificación de datos. Mientras las SVM clásicas buscan maximizar los márgenes entre distintas clases de datos buscando hiperplanos que las separen, en OCSVM los hiperplanos aprenden a distinguir la data del origen[21], es decir, la OCSVM aprende de una sola clase.

Los métodos de kernel mapean la data desde el *input feature map* en $\mathbb{R}^{\mathrm{d}}$ a un espacio de mayor dimensión $\mathbb{R}^D$ (donde D es potencialmente infinito). Como en una SVM tradicional, la data se vuelve linealmente separable por la transformación $\mathbb{R}^{\mathrm{d}} \to \mathbb{R}^D$[21].

2.5. Métricas de Evaluación

En aprendizaje supervisado y semi-supervisado, una matriz de confusión es una herramienta que permite visualizar el desempeño de un algoritmo en la clasificación de datos de testeo, cuando se conoce la clase real a la que pertenecen. Una matriz de confusión se presenta como una tabla ordenada que compara las observaciones correctas e incorrectas de cada clase. En la figura 2.12 se muestra la estructura de una matriz de confusión.

		Predicción	
		Positivos	**Negativos**
Observación	**Positivos**	Verdaderos Positivos (VP)	Falsos Negativos (FN)
	Negativos	Falsos Positivos (FP)	Verdaderos Negativos (VN)

Figura 2.12: Matriz de confusión.

A partir de los valores en la tabla se definen las métricas de confusión que permiten evaluar la clasificación hecha por la red:

- *Accuracy* o Exactitud: Corresponde a la razón de datos clasificados correctamente y el total. Es la métrica más intuitiva, pero no es un buen indicador para conjuntos cuyas clases están desbalanceadas.

$$Accuracy = \frac{VP + VN}{VP + FN + FP + VN} \tag{2.24}$$

- *Precision* o Presición: Se refiere a la dispersión del conjunto de valores positivos encontrados. Se calcula como la razón entre los verdaderos positivos y el total de predicciones de la clase positiva.

$$Precision = \frac{VP}{VP + FP} \tag{2.25}$$

- *Recall (Sensitivity)* o Sensibilidad: Corresponde a la tasa de datos positivos correctamente clasificados versus la totalidad de casos positivos que existían en realidad.

$$Recall = \frac{VP}{VP + FN} \tag{2.26}$$

- *F1 Score:* Esta métrica combina *precision* y *recall*. Toma en cuenta tanto falsos negativos como falsos positivos para la evaluación de la clase positiva.

$$F1\ Score = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \tag{2.27}$$

Capítulo 3

Metodología

La metodología de trabajo consiste en el desarrollo y evaluación de tres arquitecturas de redes neuronales que se han utilizado en la detección de anomalías en distintas estructuras, pero esta vez, construidas específicamente para resolver el problema de detección de fallas en un puente. Para cada una de estas arquitecturas se prueban distintas combinaciones de parámetros, las que se detallan más adelante en esta sección.

El primer algoritmo implementado es una red neuronal convolucional (CNN) que usa una regresión lineal como activación del output. La segunda red corresponde a un autoencoder variacional (VAE) y la última red toma los mismos parámetros que la segunda, pero incorpora un OCSVM como clasificador de los datos en el espacio latente de la VAE.

Los algoritmos son prototipados y evaluados en primera instancia haciendo uso de un data set generado de pruebas sobre una barra. Se obtienen resultados relevantes para confirmar y respaldar la metodologia antes de evaluar dichos algoritmos sobre data obtenida de un puente real. En ambos data sets se tienen respuestas temporales de aceleración, separadas en distintos casos de evaluacion. El montaje experimental, la forma de los datos y la explicación de los casos se presentan en el capítulo siguiente.

3.1. Preprocesamiento

En ambos data sets, las lecturas de los acelerómetros entregan series temporales de aceleración. Estas series temporales contienen información relevante respecto a la aparición y evolución de la falla, pero son series que abarcan una extensa ventana temporal y con muchas variables asociadas a la adquisición, como la amplitud de la señal y el ruido asociado a fuentes externas. El preprocesamiento que se detalla a continuación es un trabajo transformativo y organizativo de los datos para que sean un input coherente a las redes neuronales.

En primer lugar, como las mediciones son series temporales extensas de varios segundos de adquisición, es necesario dividir estas series en varios intervalos que puedan ser tratados de forma independiente como "mediciones" individuales. A fin de reproducir una mayor cantidad de intervalos, se permite un grado de superposición u *overlap* entre ellos. En cada dataset, tanto el valor del *overlap* como el largo de los tramos se escoge según la resolución de los espectros de Fourier y el número de datos obtenidos finalmente. La figura 3.1 muestra una representación del proceso de división por intervalos, donde la primera serie se muestra completa, y las siguientes tres son subsampleos con superposición.

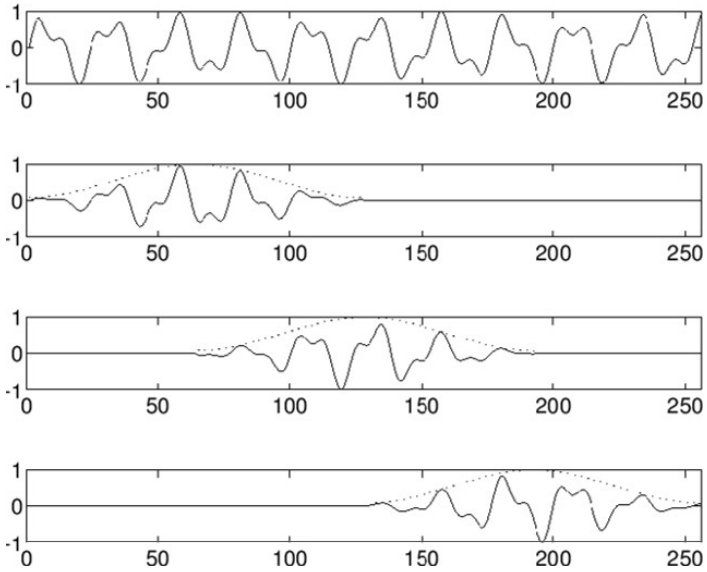

Figura 3.1: Subsampleo de intervalos con superposición[10].

En segundo lugar, cada uno de estos intervalos es normalizado dividiendo el tramo por la desviación estándar de sus datos. Dependiendo de la resolución en frecuencia buscada, se le agregan cadenas de ceros a cada intervalo para aumentar la densidad de puntos al aplicar FFT.

Luego, se aplica la transformada rápida de Fourier a cada intervalo normalizado, con el fin de obtener los espectros con los que se puede hacer el análisis en las redes. En este proceso, se escoge una ventana Hanning para el ventaneo que evita el *leakage*. Además, en el caso de los datos de la barra, se promedian estos espectros en intervalos de 5 en 5, lo que reduce la cantidad de data disponible pero soluciona el problema del ruido.

Finalmente, cada espectro se vuelve a normalizar respecto al conjunto de entrenamiento (ver sección siguiente). Es importante notar que no se incluye el vector de frecuencia como parte del data set procesado, ya que es igual para todos los espectros obtenidos y su inclusión sería redundante.

3.2. Conjuntos de Datos

En este punto es importante armar los conjuntos adecuadamente. Como se trata de un problema de detección de novedades, el conjunto de training para cada problema debe estar compuesto exclusivamente del estado normal del sistema. Luego, la ANN empleada debe usar el conjunto de validación para ampliar el concepto *normal* que ha aprendido. Bajo esta lógica, el conjunto de validación también debe constar totalmente de datos de la misma clase. Finalmente, en el conjunto de testeo se deben incluir datos de ambas clases (normal/novedad) para evaluar el aprendizaje.

La condición de entrenar con datos de una sola clase implica distintas condiciones para las redes. Se generan arbitrariamente *targets* de valor unitario (1) como etiqueta esperada, mientras que los datos de escenarios defectuosos se les asigna el valor nulo (0). Si bien estas etiquetas no son necesarias para el entrenamiento de las arquitecturas que involucran *autoencoders*, si se utilizan para evaluar la clasificación final. A fin de obtener una muestra homogénea de valores, y como requisito de convergencia de las funciones de pérdida de las redes, es necesario volver a normalizar los conjuntos. Esta normalización se realiza dividiendo cada conjunto por la diferencia entre el valor máximo y mínimo del conjunto de entrenamiento.

La estructura de estos conjuntos debe coincidir con las dimensiones definidas a la entrada de la red. Como se tienen distintas dimensiones de entrada a cada red, arbitrariamente se establece ordenar los datos en un tensor de tres dimensiones de la forma $(n_{samples},\ n_{features},\ n_{fsteps})$. $n_{samples}$ corresponde al número de muestras, $n_{features}$ corresponde al número de señales obtenidas para un mismo sample (dependiendo de la cantidad de sensores empleados) y n_{fsteps} corresponde a la cantidad de pasos en frecuencia o el largo del espectro. Así, el tensor de training se expresa como:

$$x_{train,\mathrm{i}} = \begin{pmatrix} x_{\mathrm{i},1,1} & x_{\mathrm{i},1,2} & \cdots & x_{\mathrm{i},1,p} \\ x_{\mathrm{i},2,1} & x_{\mathrm{i},2,2} & \cdots & x_{\mathrm{i},2,p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{\mathrm{i},m,1} & x_{\mathrm{i},m,2} & \cdots & x_{\mathrm{i},m,p} \end{pmatrix} \tag{3.1}$$

donde los subíndices en $X_{\mathrm{i},j,k}$ pertenecen a $n_{samples},\ n_{features},\ n_{fsteps}$ respectivamente. Por lo tanto, la expresión $x_{train,\mathrm{i}}$ corresponde a una matriz de datos obtenidos por n acelerómetros y de largo m *steps*. La colección total de entrenamiento X_{train}, que reúne todos los $x_{train,\mathrm{i}}$ viene dada por:

$$X_{train} = \begin{pmatrix} x_{train,1}, & x_{train,2} & \cdots & x_{train,n} \end{pmatrix} \tag{3.2}$$

3.3. Arquitecturas de red aplicadas

En esta sección se presentan las arquitecturas de red seleccionadas para el desarrollo de los algoritmos de detección de fallas. Cada una de estas arquitecturas implica una elección sobre una serie de parámetros fijos y variables, con el fin de determinar que configuración entrega un mejor rendimiento para cada algoritmo.

3.3.1. Metodología CNN

Se emplea un modelo de red convolucional basado en el trabajo de P. Oza y V. M. Patel[19]. Se escoge una red pre-entrenada para que sirva como *feature extractor*, para obtener vectores latentes con los cuales entrenar un clasificador, tal como ilustra la figura 2.10. Una de las ventajas de trabajar con una red pre-entrenada es poder clasificar data sets pequeños.

Una de las redes usadas en el trabajo citado es la red VGG16, bastante conocida en el campo de visión computacional y que ha sido entrenada en un data set bastante amplio, conocido como *ImageNet* (el cual posee más de 1.4 millones de imagenes y 1000 clases diferentes). Esta red fue desarrollada por Karen Simonyan y Andrew Zisserman en 2014[20]. En la figura 3.2 se presenta un esquema del trabajo típico de una base convolucional pre-entrenada, donde se toma una red convolucional y se procede eliminando el clasificador pre-entrenado para entrenar un nuevo clasificador. En este caso, los pesos de la etapa convolucional se congelan, y solo ocurre aprendizaje en el nuevo clasificador.

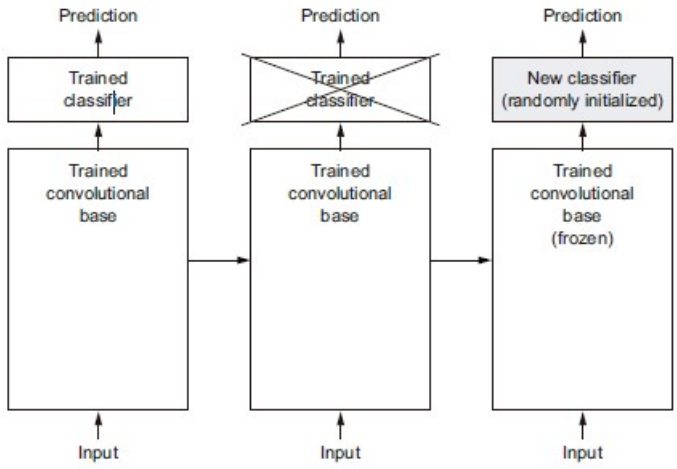

Figura 3.2: Intercambio de clasificadores conservando la base convolucional[20].

Para la arquitectura propuesta, se usan dos técnicas sobre la base convolucional: *feature extraction* y *fine tuning*, técnicas que se describen más adelante.

Arquitectura Base

La arquitectura propuesta toma la base convolucional de la red VGG16, la cual consiste en 5 bloques o etapas convolucionales. Las primeras dos tienen dos capas convolucionales y un maxpooling, las siguientes tres tienen tres capas convolucionales y un maxpooling. A esta base convolucional se le agrega una capa *Flatten* que sirve para reordenar el *feature map* en un vector latente. Para la etapa de *Fully Connected Layers* se disponen 4 capas de 512, 256, 128 y 64 nodos, con tangente hiperbólica como función de activación. Finalmente, la última capa corresponde a una capa densa de dos neuronas con función de activación *softmax*, la que entrega como *output* un vector de dos valores para cada dato procesado. Estos valores representan la probabilidad de que el dato pertenezca a la clase 0 y a la clase 1, respectivamente. La figura 3.3 esquematiza esta arquitectura.

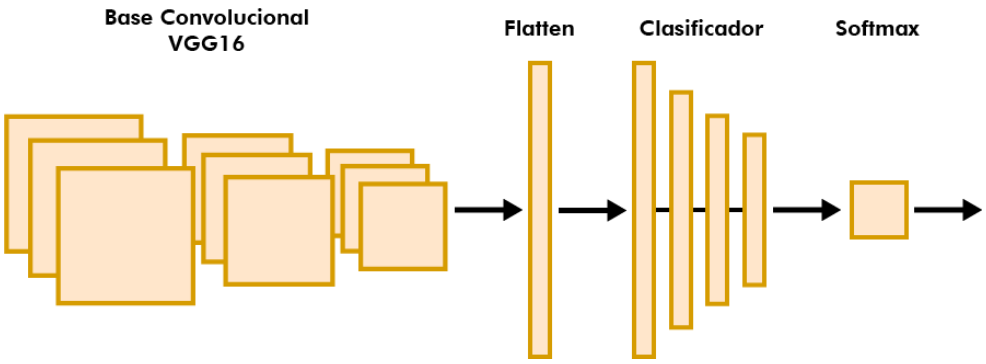

Figura 3.3: Diagrama de bloque de la arquitectura base de la CNN.

Es importante destacar además, que el tamaño de *input* que acepta la red VGG16 es de *(32,32,3)*, lo que representa una imagen de 32x32 pixeles con 3 canales (para RGB). Esto exige un número mínimo de valores para cada medición y la inclusión de una transformación *reshape* para reordenar los datos a un tensor acorde.

Otros parámetros básicos incorporados son el optimizador Adam, con un *learning rate* igual a $lr = 1\text{e} - 4$, la función *binary crossentropy* (entropía cruzada binaria) como costo o pérdida y un *batch size = 64*. Además, se utilizan condiciones de early stopping para detectar la convergencia del entrenamiento. Todos estos parámetros se determinan por ensayo y error.

Two Class Classification

Con el objetivo de evaluar la separatibilidad de los escenarios *no defectuosos* de los datos *defectuosos* en cada data set, se entrena la CNN con datos de ambas clases, lo que en la práctica es un clasificador binario o clasificador de dos clases. Los parámetros utilizados son los descritos en la arquitectura base. Durante el entrenamiento, los pesos de la etapa convolucional se mantienen congelados, es decir, no se ajustan.

One Class Classifiation

Como la motivación del trabajo es la detección de novedades, se entrena la CNN bajo el supuesto de que solo se conoce una clase: la clase positiva. En el modelo propuesto por Oza y Patel, la clase pseudo-negativa se representa por un muestreo de una matriz obtenida de una distribución gaussiana. En la práctica esto genera un clasificador binario donde la única clase que se conoce a priori es la clase positiva (por eso se llama clasificador de una clase).

Sin embargo, la metodología anterior puede no resultar sensible a un espectro de Fourier de un escenario defectuoso. Es decir, dado que los *inputs* siempre son representaciones del espectro de Fourier de la respuesta del sistema, se espera que los datos tengan una cierta estructura, tanto para el escenario defectuoso como para el no defectuoso. Lo anterior puede generar que el clasificador distinga dos clases: *representación latente de espectros de Fourier* y *muestreo gaussiano*. Si esto ocurre, el clasificador no será capaz de identificar si hay daño o no en la estructura, lo cual es el objetivo principal del presente.

El problema anterior es resuelto por los autores utilizando *instance normalization* justo antes de entrar al clasificador, lo que ayuda a estabilizar el entrenamiento[19]. Alternativamente, se proponen dos estrategias adicionales para generar datos de la clase pseudo-negativa, ambas tomando como base los datos de entrenamiento que pertenecen a la clase positiva: inducción de ruido gaussiano y reordenamiento de peaks.

La primera estrategia consiste en samplear valores de una distribución normal de parámetros $\mu = 0$ y $\sigma = 0{,}075$ y sumarlos a cada valor de los espectros de Fourier obtenidos para obtener datos pseudo-negativos (por la definición de los espectros de Fourier se usa el valor absoluto de los valores obtenidos). La segunda estrategia consiste en intercambiar los valores máximos de posición dentro del espectro, lo que corresponde al comportamiento esperado de la clase negativa (defectuosa) según la teoría de análisis vibracional, además de añadir un poco de ruido, de parámetros $\mu = 0$ y $\sigma = 0{,}005$.

Sobre cada una de estas estrategias se emplean las técnicas *feature extraction* y *fine-tuning*. Feature extraction usa la base convolucional para obtener representaciones latentes de la data, sin modificar sus pesos durante el entrenamiento (se mantienen las capas congeladas). Por otro lado, fine-tuning es una técnica complementaria a feature extraction y consiste en descongelar algunas de las últimas capas convolucionales, y entrenarlas en conjunto a las capas añadidas para obtener representaciones latentes que sean más relevantes para el problema[20]. El fine-tuning se aplicará solamente al último bloque convolucional de la base obtenida de VGG16.

Finalmente, los clasificadores se evalúan según las métricas establecidas en la sección 2.5.

3.3.2. Metodología VAE

La metodología desarrollada para la implementación de *variational autoencoders* consiste en crear espacios latentes con distintas dimensionalidades, de tal forma que la arquitectura tanto del *encoder* como del *decoder* dependen de esta dimensión latente, en términos de la cantidad de capas y el tamaño de estas. Los modelos VAE desarrollados corresponden a MLP, es decir, consisten de capas densas cuyos tamaños están determinados por la dimensión del espacio latente.

Como la data está ordenada en tensores, según lo descrito en la sección 3.2, se debe hacer un *unroll* de los datos, de forma tal que cada dato de entrenamiento, validación o prueba se ordene en un único vector que pueda pasar a través del *Multilayer Perceptron*. Este vector, entonces, corresponde a todas las mediciones de un dato concatenadas. La dimensión de dicho vector determina el tamaño de la capa de entrada o *input*, y la capa siguiente se reduce a una dimensión de 2048 neuronas. Cada capa subsecuente del encoder divide su dimensión por un factor de 4, hasta alcanzar el tamaño del espacio latente. La arquitectura del decoder es exactamente la misma pero invertida, de tal manera que cada capa amplifica el número de unidades en la capa por 4, hasta devolver la dimensión de entrada. La figura 3.4 esquematiza la arquitectura descrita.

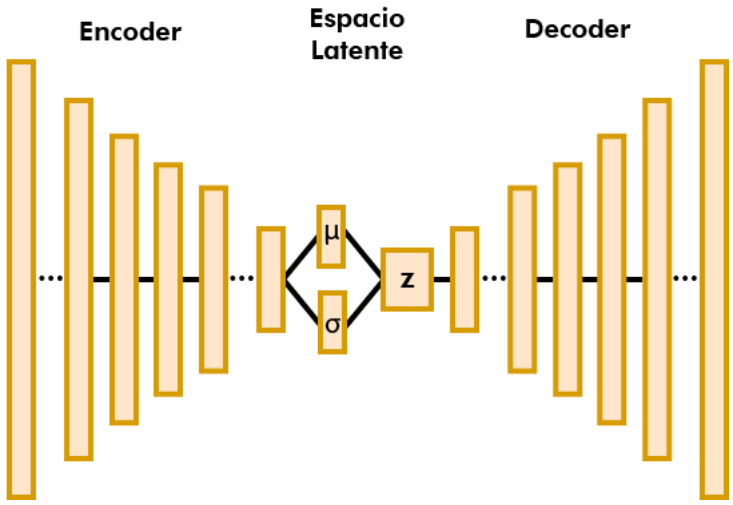

Figura 3.4: Diagrama de bloque de la arquitectura base de la VAE.

Durante el entrenamiento se utiliza el optimizador Adam con el *learning rate* por defecto $lr = 1\text{e} - 2$. La función *loss* se construye a partir de la expresión 2.16. Además se usa un *batch size* de 64 y se aplican condiciones de *early stopping* para la convergencia de la función *loss*.

Clasificación

Luego de la fase de entrenamiento, la red es utilizada para construir un clasificador. Usando el error de reconstrucción de un dato, el clasificador es capaz de predecir si la clase a la que pertenece el dato es efectivamente la clase con la que se entrenó la red o no. Para esto, es necesario definir un threshold.

En el caso de las redes prototipo, probadas en las mediciones de la barra, el threshold es definido usando el paquete de métricas del módulo SKLearn. Por otra parte, en los datos del puente se usa el conjunto de validación para establecer un threshold basado en que el error de reconstrucción distribuye como una distrubución normal. Calculando el promedio y la desviación estándar del vector de error de reconstrucción del conjunto de validación, se busca el valor límite de esta variable de forma tal que el 99.9 % de la distribución sea el margen mediante el cual los datos sean correctamente clasificados, usando la expresión 3.3.

$$Z = \frac{X - \mu}{\sigma/\sqrt{n}} \tag{3.3}$$

En la expresión anterior, Z es el valor variable de cambio tal que $Z \sim N(0, 1)$, μ y σ son el promedio y desviación estándar, respectivamente, del error de reconstrucción en el conjunto de validación, mientras que n es el número de datos en en conjunto de validación. Se escoge un valor $\tilde{Z}$ para el cual el 99.9 % de la distribución sea cubierta. Esto genera un valor $\tilde{X}$, que se escoge como el threshold para el error de recostrucción.

En la expresión 2.16, el error de reconstrucción está dado por el segundo término del lado derecho de la igualdad. En la mayoría de las aplicaciones de VAE para clasificación de datos, se aproxima este término por la función de error cuadrático medio (MSE, por su traducción en inglés). Se grafican tanto el error de reconstrucción para cada dato, separado según su etiqueta, como el threshold determinado.

3.3.3. Metodología VAE+SVM

A partir de los modelos de VAE desarrollados, se entrenan clasificadores basados en *One-Class Support Vector Machines*. Siguiendo la lógica de *novelty detection*, se usan las redes VAE entrenadas con datos del escenario *no defectuoso*, y luego se usa la parte del *encoder* para transformar la data en vectores latentes. El conjunto obtenido se usa para entrenar un OCSVM. La figura 3.5 esquematiza este proceso.

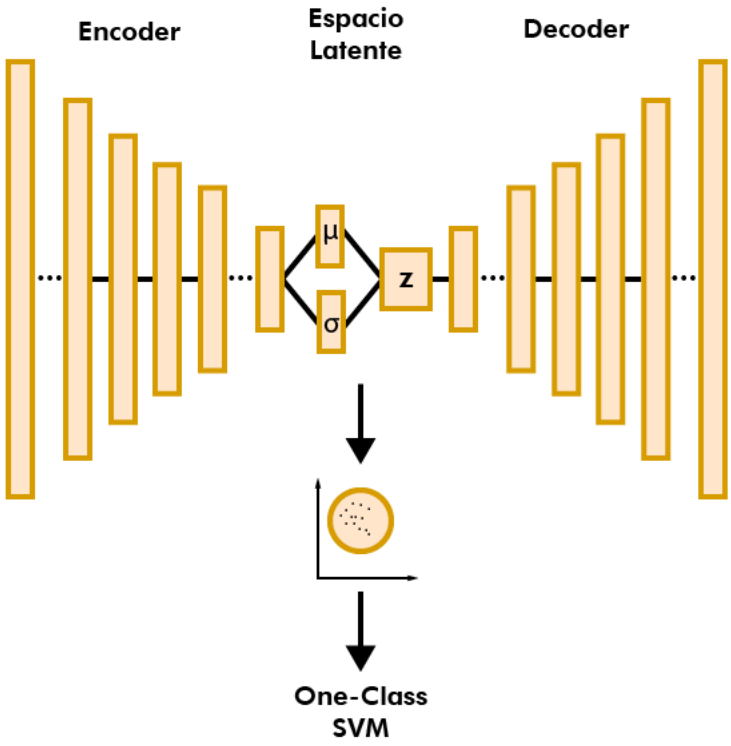

Figura 3.5: Diagrama de bloque de la arquitectura dispuesta para el método VAE+SVM.

Las representaciones latentes obtenidas de la data de entrenamiento se usan para entrenar el clasificador. Luego, cada lectura nueva que ingresa para ser clasificada pasa primero por el encoder de la VAE, y su representación latente pasa por el clasificador de una clase basado en SVM.

El kernel empleado es *radial basis function (rbf)*, definido por la expresión 2.23 Para cada clasificador desarrollado se prueban 5 valores distintos del parámetro de escala γ del clasificador. Además, mediante ensayo y error se encuentra el parámetro ν para el cual los valores de γ entregan los mejores resultados. Los valores de γ escogidos son: 0.1, 0.5, 1, 5 y 10.

Capítulo 4

Resultados

En este capítulo se presentan los resultados obtenidos al aplicarse la metodología descrita anteriormente, desde el montaje experimental y el preprocesamiento de datos hasta los resultados de las clasificaciones efectuadas por cada red neuronal, para cada configuración de parámetros.

El capítulo se divide en los dos casos de estudio: Barra y Puente. El primer caso estudio solo tiene como objetivo probar los algoritmos y realizar un bugfix general del código. A partir de la experiencia obtenida con el primer caso, se toman decisiones relevantes al uso de datos y redes en el segundo caso.

4.1. Caso Estudio 1: Barra

El primer conjunto de datos trabajados se obtuvo de parte del Laboratorio de Vibraciones Mecánicas y Rotodinámica de la Universidad de Chile. El objetivo de este conjunto de datos es probar los algoritmos y realizar un *bugfix* del código, para asegurar el correcto funcionamiento cuando se apliquen estas redes a los datos del puente.

Mediante un shaker se introducen vibraciones a la barra, las que son captadas gracias a un acelerómetro ubicado en el extremo libre, como se ve en la figura 4.1. Se realizan pruebas para cuatro barras, todas de las mismas dimensiones, pero tres de ellas tienen un *defecto* o corte transversal de 1, 2 y 5 mm de profundidad, ubicado a 250 mm del extremo libre. Así se tiene un escenario no defectuoso y tres escenarios defectuosos.

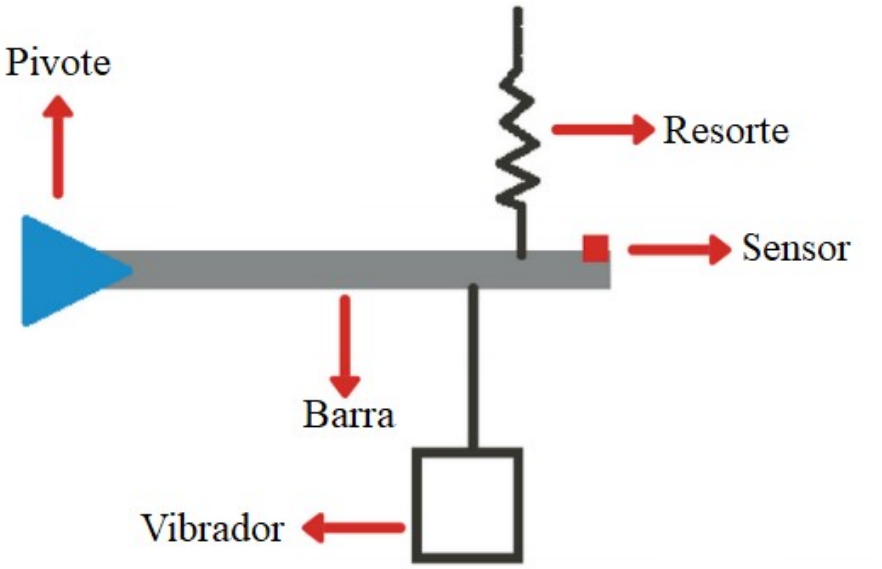

Figura 4.1: Montaje Experimental Barra

Como el montaje solo contempla un acelerómetro, se obtiene una única serie temporal en cada medición. Se realiza una medición de 10 minutos para la barra sin daño, mientras que la medición para las barras con daño fue de 5 minutos para cada una. Esta operación se repite para obtener dos mediciones de cada barra de prueba.

La frecuencia de adquisición f_s en este experimento es constante e igual a $4096Hz$. En la figura 4.2 se exponen dos de las mediciones obtenidas, la primera sin daño y la segunda con daño.

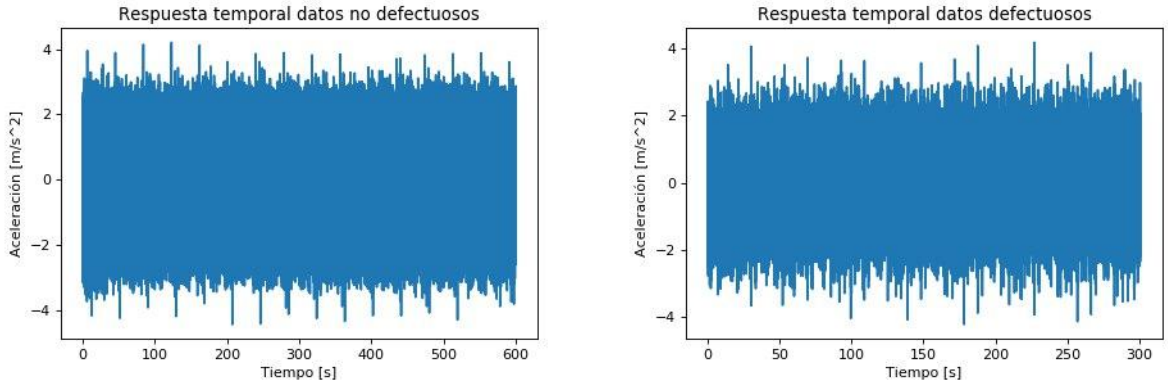

Figura 4.2: Series temporales obtenidas para las barras. A la izq. barra no defectuosa, a la der. barra defectuosa.

4.1.1. Espectros de Fourier

La data se procesa según lo discutido en la sección 3.1. Los espectros de Fourier obtenidos se ejemplifican en las figuras 4.3 para los datos de la barra.

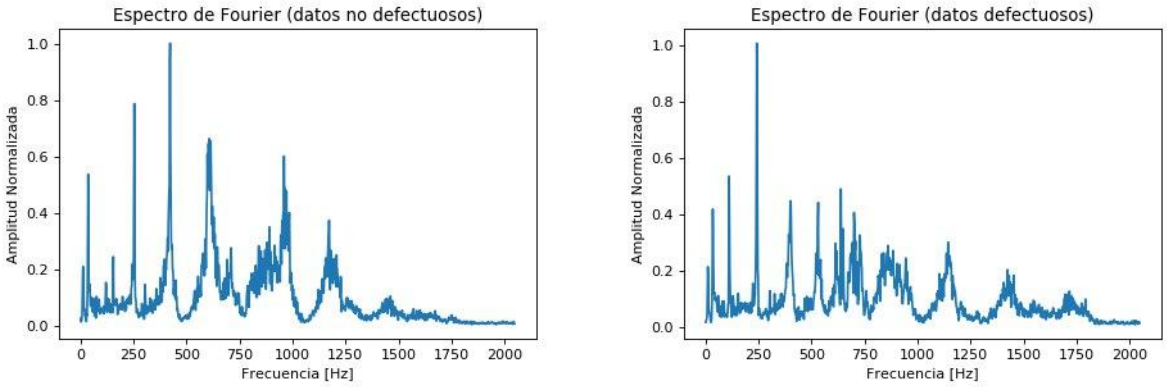

Figura 4.3: Espectros de Fourier obtenidos para el data set de la barra. A la izq. data no defectuosa, a la der. data defectuosa.

4.1.2. Resultados Red Convolucional (CNN)

Trabajo de datos

Dado que cada espectro de Fourier en este data set posee solo 1224 datos, es imposible re-ordenar estos datos para cumplir las dimensiones mínimas del input de la red VGG16 (32,32,3). Como solución se repitió cada espectro de Fourier tres veces (para llenar los 3 canales), obteniendo datos de dimensión (1224,3). Después de un re-ordenamiento, los datos ingresan a la red en tensores con dimensión (36,34,3).

Two Class Classification

Usando la arquitectura base descrita en la sección 3.3.1, se entrenó una red de clasificación binaria con 1200 datos de entrenamiento (700 datos de cada clase). El conjunto de testeo consistió en 100 datos *no defectuosos* y 100 datos *defectuosos*. Los resultados de esta clasificación se ordenan en la matriz de confusión de la tabla 4.1.

Tabla 4.1: Matriz de Confusión Clasificación Binaria Datos Barra.

	Predicción Clase Defectuosa	***Predicción Clase No Defectuosa***
Datos Defectuosos	100	0
Datos No Defectuosos	0	100

One Class Classification

De los métodos de generación de la clase pseudo-negativa se obtuvieron gráficos como los expuestos en la figura 4.4, ambos ejemplos generados a partir del mismo *sample* de escenario no defectuoso, que se muestra a la izquierda de la figura 4.3. El entrenamiento de las redes se realizó con 1400 datos de entrenamiento (700 datos del escenario no defectuoso y 700 datos pseudo-negativos generados con los métodos descritos en la sección 3.3.1).

Para el testeo se preparan 400 datos (200 de escenario no defectuoso y 200 de los escenarios defectuosos). La tabla 4.2 muestra las métricas de confusión obtenidas para la predicción descrita.

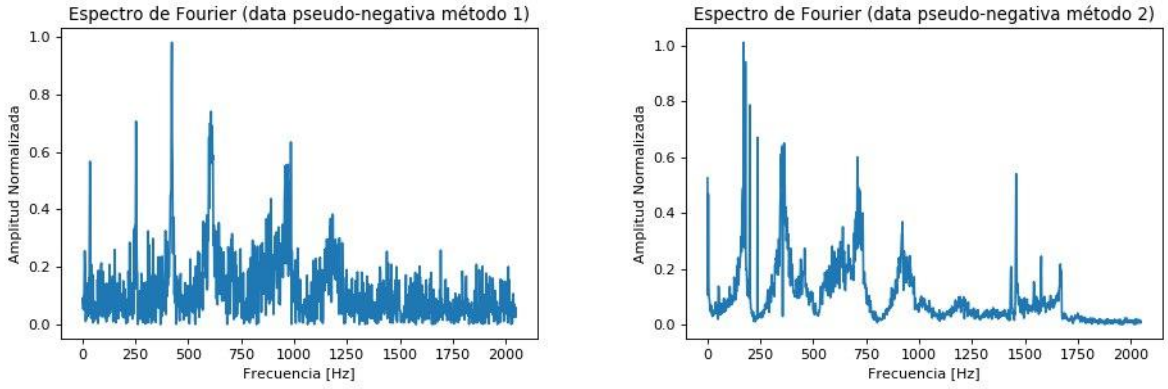

Figura 4.4: Espectros de Fourier generados como clase pseudo-negativa a partir del escenario no defectuoso en el data set Barra. A la izq. se introdujo ruido gaussiano, a la der. se cambió la posición de los 5 valores máximos.

Tabla 4.2: Evaluación de detección de daño en barra para cada método de generación de clase pseudo-negativa.

Modelo	*Accuracy*	*Precision*	*Recall*	*F1 Score*
DataGen1 Feature Ex	0.77	1	0.54	0.70
DataGen1 Fine Tuning	0.89	1	0.80	0.89
DataGen2 Feature Ex	0.87	0.89	0.84	0.87
DataGen2 Fine Tuning	0.56	1	0.12	0.69

4.1.3. Resultados Autoencoder Variacional (VAE)

Se destinaron 650 datos del escenario *no defecutoso* al entrenamiento, de los cuales 150 constituyeron el conjunto de validación. Es posible usar el *encoder* de la VAE para plotear espacios latentes, a modo de ejemplo, la figura 4.5 muestra la representación de la VAE con dimensión latente igual a 2, del dataset completo. En esta gráfica, la clase no defectuosa está representada por 0 (color azul), y los daños de 1, 2 y 5 mm de profundidad están representadas por 1, 2 y 3 (colores rojo, rosa y celeste), respectivamente.

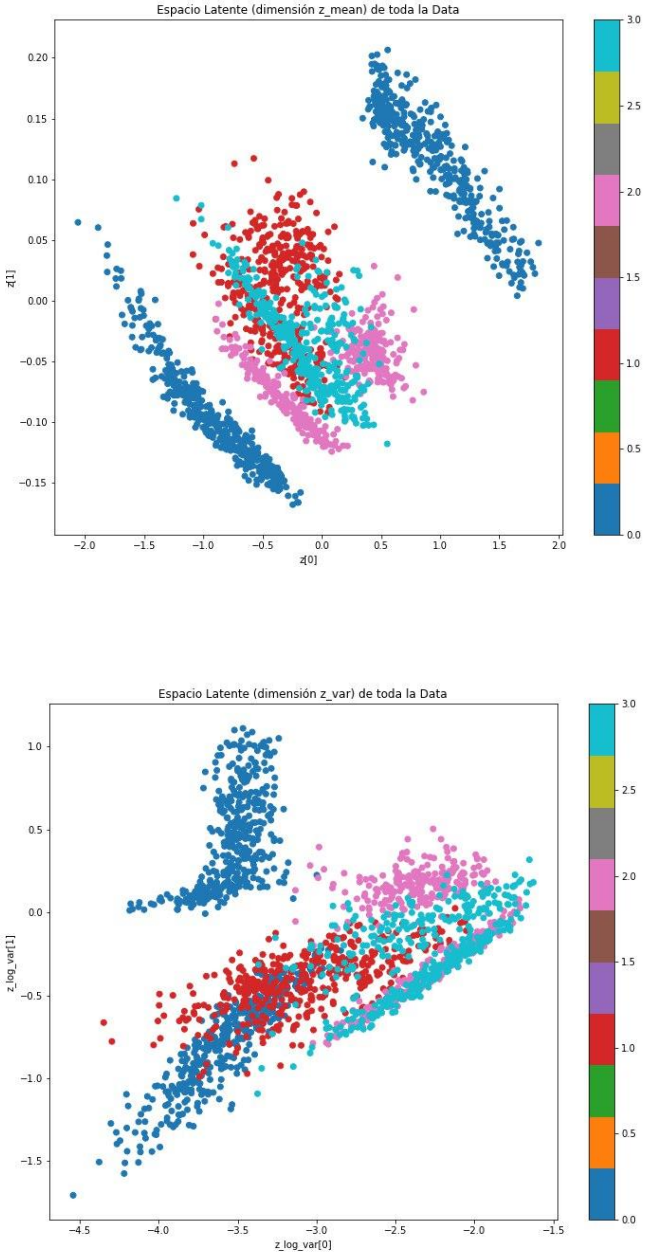

Figura 4.5: Representación del espacio latente para dim.Z=2 (data Barra). Arriba el plano del valor promedio, abajo el plano del valor desviación estándar.

Reconstrucción de datos

A continuación se presenta la reconstrucción de un dato del escenario no defectuoso (figura 4.6) y la reconstrucción de un dato no defectuoso (figura 4.7). Este ejemplo de reconstrucción se utiliza solo de forma referencial, usando como base la VAE de dimensión latente 2, ya que se encontró que la forma de reconstrucción no difiere según la dimensión latente.

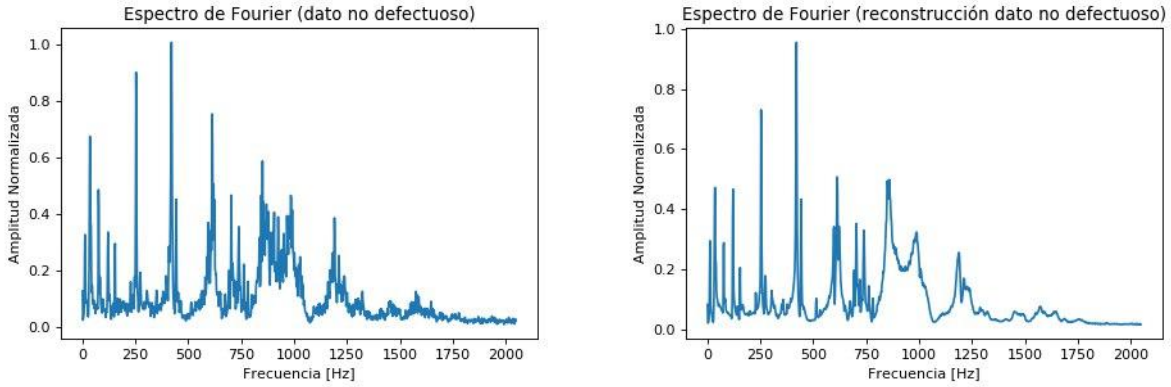

Figura 4.6: Espectros de Fourier escenario no defectuoso. A la izquiera dato de entrada a la red, a la derecha reconstrucción lograda por la VAE.

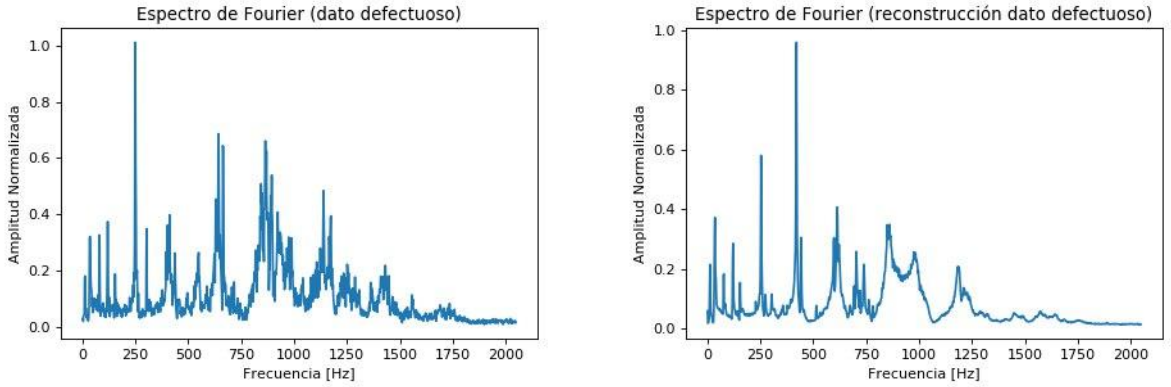

Figura 4.7: Espectros de Fourier escenario defectuoso. A la izquiera dato de entrada a la red, a la derecha reconstrucción lograda por la VAE.

Clasificación

A continuación se presentan los resultados obtenidos en la clasificación. La tabla 4.3 presenta el threshold óptimo determinado para el error de reconstrucción según las herramientas automatizadas de SKLearn. La tabla 4.4 presenta las métricas de clasificación obtenidas para cada dimensión latente y su respectivo threshold. Además, las figuras 4.8 a 4.10 muestran gráficamente esta clasificación de cada dato según su error de reconstrucción.

Tabla 4.3: Thresholds mediante herramientas SKLearn.

Dim. Latente	**Threshold (MSE)**
z=2	0.0085
z=8	0.0064
z=32	0.0082

Tabla 4.4: Métricas de Confusión para cada dimensión latente, según los thresholds obtenidos mediante herramientas SKLearn.

Modelo	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Dim. Latente 2	0.90	0.99	0.81	0.89
Dim. Latente 8	0.98	0.99	0.97	0.98
Dim. Latente 32	0.94	0.99	0.89	0.94

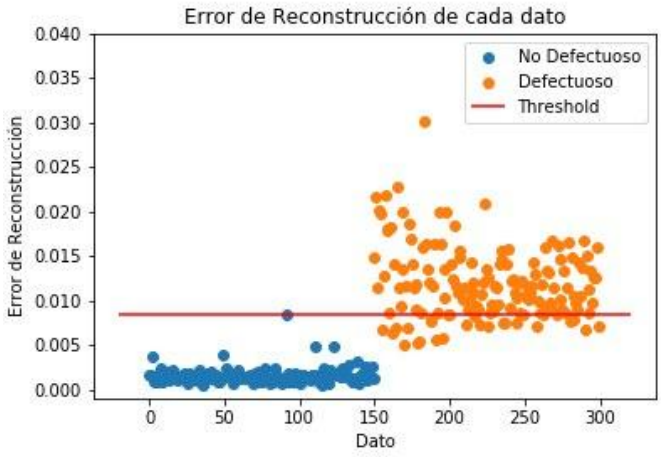

Figura 4.8: Error de reconstrucción por dato para $z = 2$, $th = 0{,}0085$

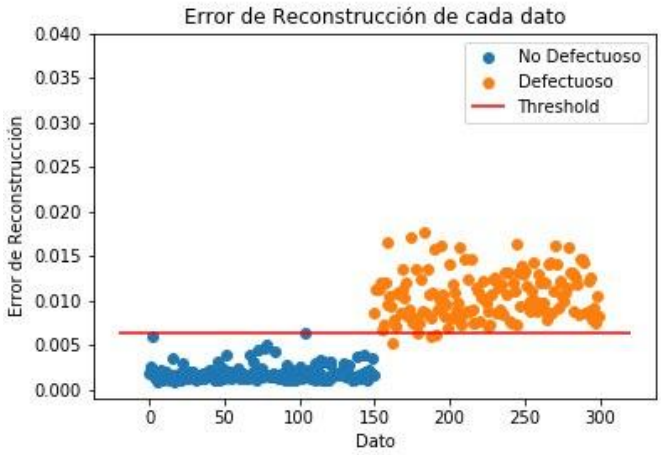

Figura 4.9: Error de reconstrucción por dato para $z = 8$, $th = 0{,}0064$

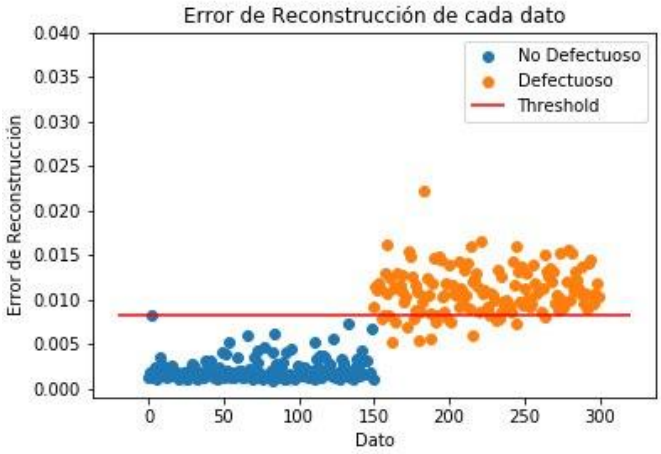

Figura 4.10: Error de reconstrucción por dato para $z = 32$, $th = 0{,}0082$

4.1.4. Resultados Autoencoder Variacional con One-Class Support Vector Machines (VAE+SVM)

La dimensión latente genera el doble valores latentes por dato (2 parámetros μ y σ x Z dimensiones). El parámetro ν se fija en el valor 0.001 después de ensayo y error. Las figuras 4.11, 4.12 y 4.13 muestran las curvas de aprendizaje para distintos valores del parámetro gamma, usando las VAEs entrenadas con dimensiones latentes 2, 8 y 32, respectivamente. Además, la tabla 4.5 resume los mejores resultados obtenidos para cada dimensión latente (considerando *accuracy* y *f1 score* como las métricas más importantes).

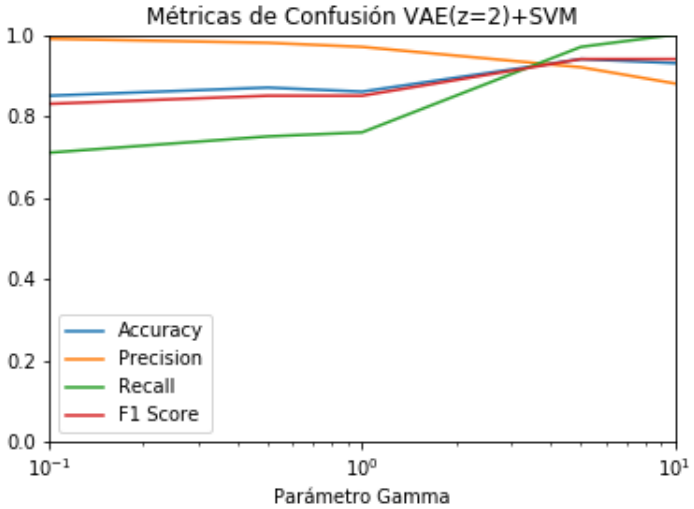

Figura 4.11: Métricas de confusión para $z = 2$ como dimensión latente, para los valores de gamma probados

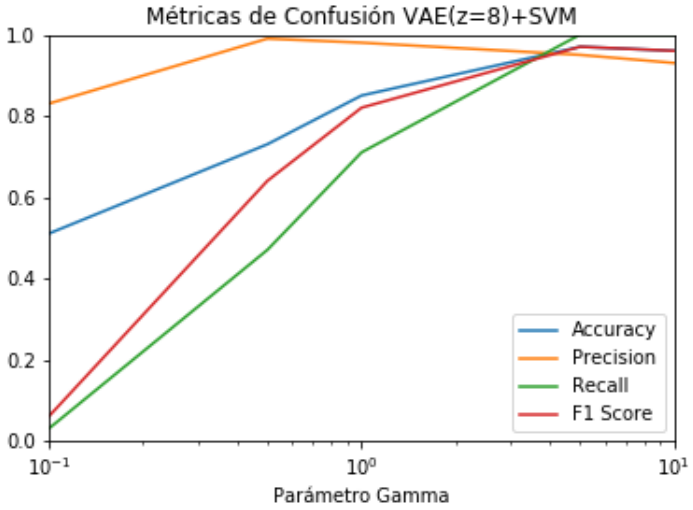

Figura 4.12: Métricas de confusión para $z = 8$ como dimensión latente, para los valores de gamma probados

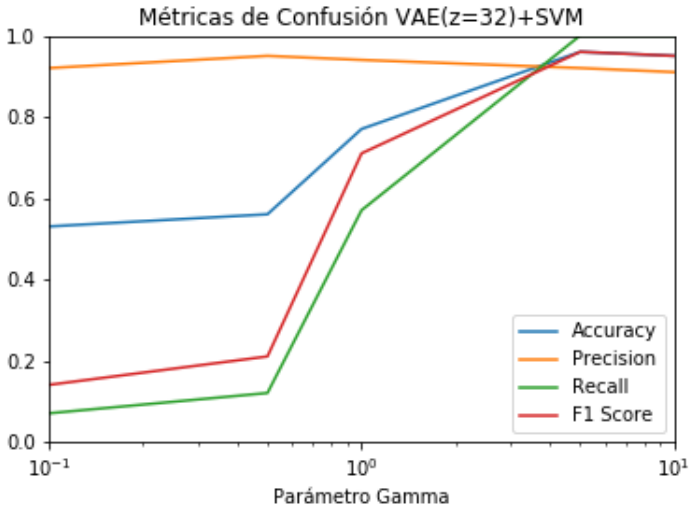

Figura 4.13: Métricas de confusión para $z = 32$ como dimensión latente, para los valores de gamma probados

Tabla 4.5: Mejores resultados de clasificación mediante OCSVM para cada VAE pre-entrenada.

Modelo	Valor de Gamma	*Accuracy*	*Precision*	*Recall*	*F1 Score*
Dim. Latente 2	$\gamma = 5$	0.94	0.92	0.97	0.94
Dim. Latente 8	$\gamma = 5$	0.97	0.95	1	0.97
Dim. Latente 32	$\gamma = 5$	0.96	0.92	1	0.96

4.2. Caso Estudio 2: Puente Atirantado

El segundo *data set* trabajado corresponde a un caso de estudio de Monitoreo de Salud Estructural desarrollado en un puente atirantado en el Estado de Nueva Gales del Sur, Australia. Dicho puente ha sido instrumentado desde Julio de 2016 con acelerómetros, galgas extensiométricas y sensores ópticos, y se ha obtenido una respuesta continua, producto de su exposición al viento, en tiempo real desde entonces.

Se tiene disponibilidad de estos datos ordenados en 4 distintos sets:

- *Constant Temperature Dataset:* Mediciones seleccionadas cuando la temperatura ambiente es de $22{,}5 \pm 1°C$. Fueron seleccionados 66 archivos desde el 1 al 22 de Noviembre de 2016.
- *Seasonal Dataset:* Mediciones de la respuesta ambiente continua por un total de 30 días, durante los meses de Enero, Julio y Octubre (10 días cada uno).
- *Single Day Dataset:* Respuesta temporal registrada durante el 2 de noviembre de 2016. La mínima de temperatura ese día fue de $12{,}1°C$ y la máxima de $30{,}4°C$.
- *Damaged-Structure Dataset:* Dividido en dos escenarios de daño. El primer caso (DC1) es una simulación donde se obtiene la respuesta del puente con un vehículo en distintas locaciones, con distancias descritas en la tabla 4.6. El vehículo aporta una masa de 2,4 toneladas y la distancia entre sus ejes es de $3{,}1m$. . El segundo caso (DC2) corresponde a la respuesta del puente bajo la carga de un bus de 13 toneladas, situado en el punto medio del puente.

Figura 4.14: Puente Atirantado sobre la Great Western Highway, AUS[22]

Tabla 4.6: Distancia entre el eje delantero del vehículo y la junta de dilatación del lado sur del puente en cada escenario de daño DC1.

Escenario de Daño	**DC1-L1**	**DC1-L2**	**DC1-L3**	**DC1-L4**	**DC1-L5**
Distancia entre el eje delantero del vehículo y la junta	6.2 m	15.5 m	24.8 m	31 m	37.2 m

Es importante notar que el escenario defectuoso DC1-L3 corresponde al vehículo en la posición central del puente, por lo que se puede adelantar que es el subcaso que tiene un mayor impacto en la respuesta temporal de vibraciones del puente. Los casos DC1-L4 y DC1-L5 vienen a ser muy similares a los casos DC1-L2 y DC1-L1, respectivamente, por simetría. La figura 4.15 muestra la ilustración esquemática del puente desde el lado, con unidades en milímetros.

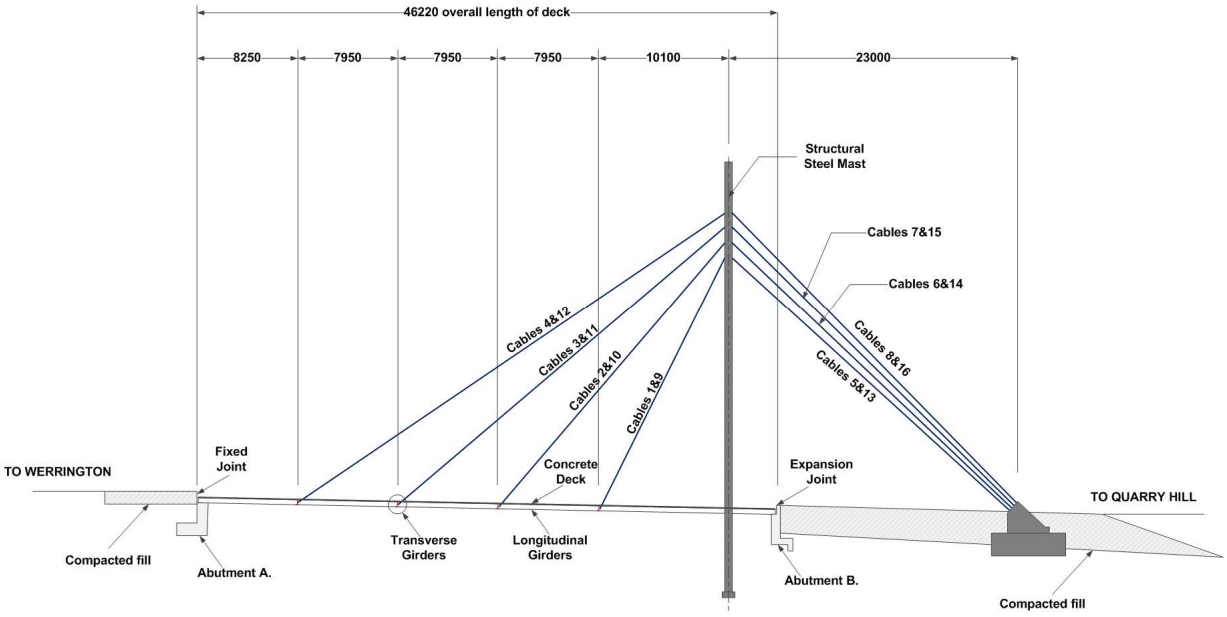

Figura 4.15: Ilustración esquemática del puente, en vista lateral (desde el oeste), con unidades en milímetros.

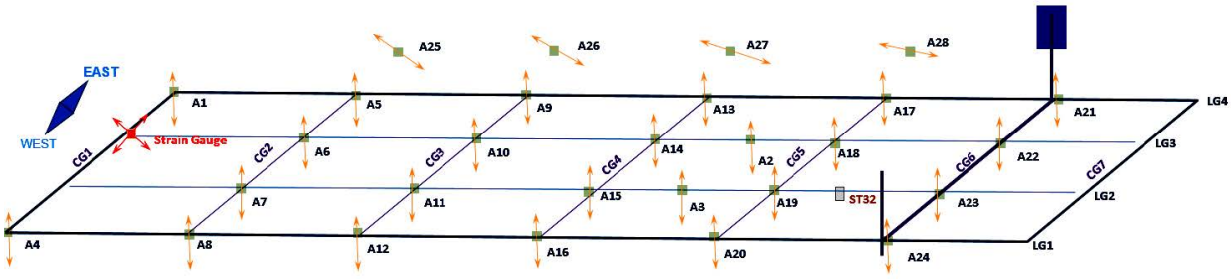

Figura 4.16: Ilustración esquemática de los acelerómetros instalados bajo la cubierta del puente.

En la figura 4.16 se presenta la grilla que se utilizó como referencia para la colocación de los acelerómetros (representados con la letra A, seguida de un número). La data disponible contiene mediciones de 28 acelerómetros, con una frecuencia de adquisición de $f_s = 600\ Hz$, pero remuestreados a 60 Hz, a fin de disminuir el peso de archivos y costo computacional involucrado en el trabajo de estos datos.

Los data sets utilizados para este caso estudio corresponden a *constant temperature*, *single day* y *damaged-structure*. Mientras que los primeros dos data sets son mediciones extensas (de hasta un día completo), las mediciones para los escenarios con falla son de apenas algunos minutos. En el caso de las mediciones DC1-L1 a DC1-L5, cada escenario registra 5 minutos, mientras que el caso DC2 registra apenas 1.5 minutos. Por otra parte, los escenarios defectuosos se realizan sin circulación de vehículos, mientras que la data de *constant temperature* y *single day* si se ve afectada por el tráfico. La figura 4.17 muestra la respuesta obtenida por el acelerómetro A2 para estos data sets, pero concatenando los archivos obtenidos en ventanas temporales mayores y haciendo un detrending para eliminar la componente constante.

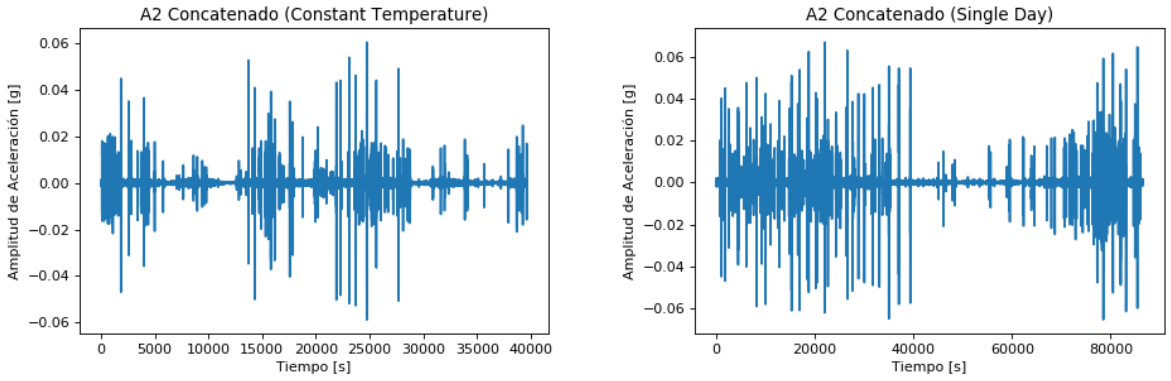

Figura 4.17: Series temporales (concatenadas) obtenidas para el puente atirantado en los datasets *constant temperature* y *single day* (acelerómetro A2).

Estas señales son concatenadas uniendo los archivos, y son posteriormente procesadas para eliminar los peaks (que se deben a la circulación de vehículos por el puente y a pequeños impactos en la localidad de cada acelerómetro). Además, se seleccionan únicamente los acelerómetros A2, A3 y A6-A20, ya que presentan menos ruido al no estar en las orillas y su eje es transversal a la cubierta del puente. Esta elección de acelerómetros se mantiene en los escenarios con daño.

La figura 4.18 muestra una de las mediciones (el mismo acelerómetro A2) usadas para la segmentación de data y posterior análisis espectral realizado, según la metodología explicada en la sección 3.1, para cada escenario. Es importante destacar que, dado que se tiene menor disponibilidad de data, el factor de *overlap* en la división de datos de los escenarios con daño es mucho mayor.

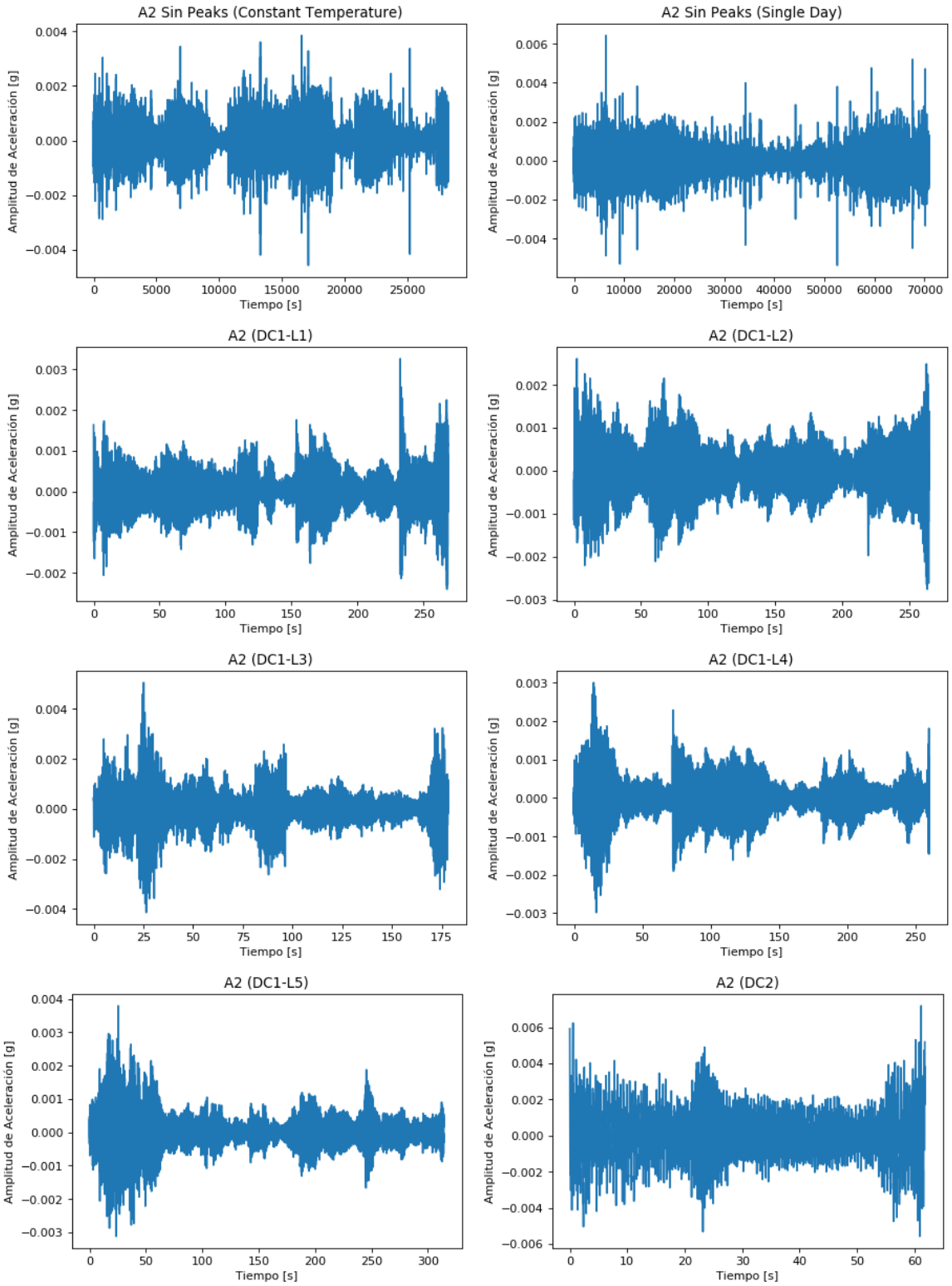

Figura 4.18: Series temporales (con eliminación de peaks) obtenidas para el puente atirantado en los datasets *constant temperature*, *single day* y *damaged-structure* (acelerómetro A2).

4.2.1. Espectros de Fourier

Del estudio realizado por Alamdari et al[22] se sabe que la frecuencia del primer modo de vibración del puente, en condiciones normales, es de 2,07Hz. Por otra parte, los escenarios de daño simulado en este trabajo presentan una variación máxima en la frecuencia del primer modo del 4,4 % (a 1,98Hz). Además, se establece que los primeros nueve modos de vibración son distinguibles en la ventana 0 - 20Hz.. A partir de esto se establecen 4 pasos en frecuencia como mínimo en la resolución para diferenciar un caso de otro, para establecer un margen suficiente entre un escenario defectuoso y uno no defectuoso.

La figura 4.19 muestra un ejemplo de los espectros de Fourier para los escenarios no defectuosos. Según lo calculado, los espectros de Fourier del caso *constant temperature* tienen un peak del primer modo de vibración entre 1,975 y 2,1Hz. Mientras que en el caso de la data *single day*, el primer modo de vibración presenta un rango de frecuencias entre 2,025 y 2,05 Hz.

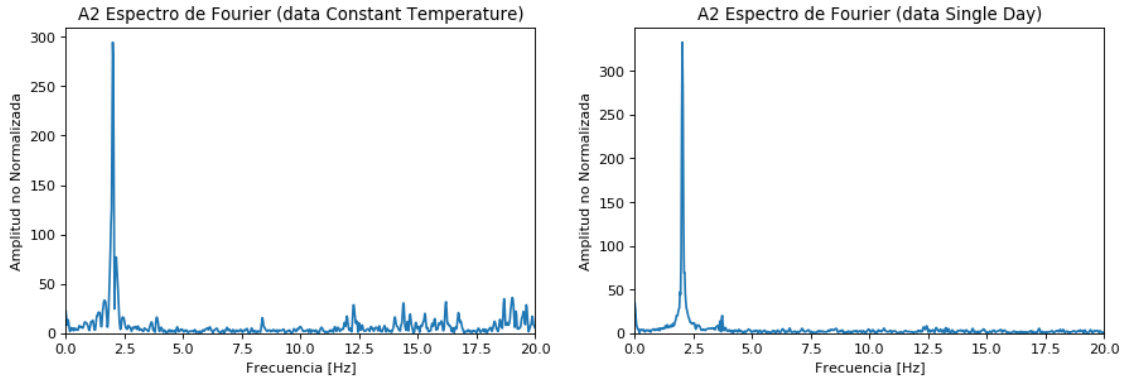

Figura 4.19: Espectros de Fourier obtenidos para el puente atirantado en los datasets *constant temperature* y *single day* (acelerómetro A2).

La figura 4.20 ejemplifica algunos espectros de Fourier obtenidos para los distintos escenarios con daño.

Como se busca una representación en frecuencia de los datos, que permita la diferenciación más clara entre un escenario y otro, se exploran dos alternativas a la presentación de los espectros de Fourier representados por las figuras 4.19 y 4.20. Estas alternativas son el logaritmo del espectro de Fourier y el espectro de potencias (espectro de Fourier al cuadrado). Las representaciones de estas alternativas, para todos los casos, se muestran en las figuras 4.21 y 4.22.

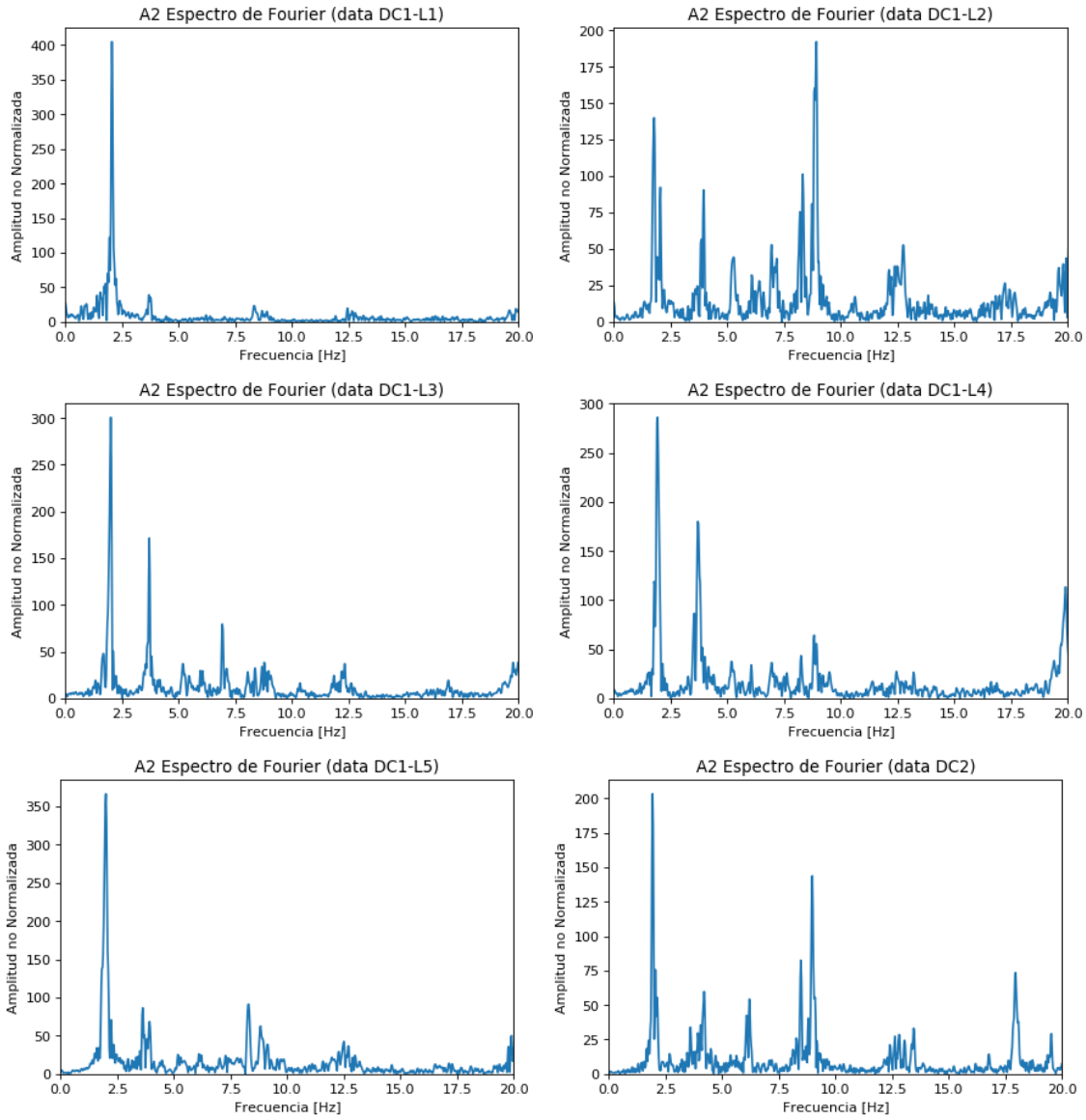

Figura 4.20: Espectros de Fourier obtenidos para el puente atirantado en los escenarios de daño simulado (acelerómetro A2).

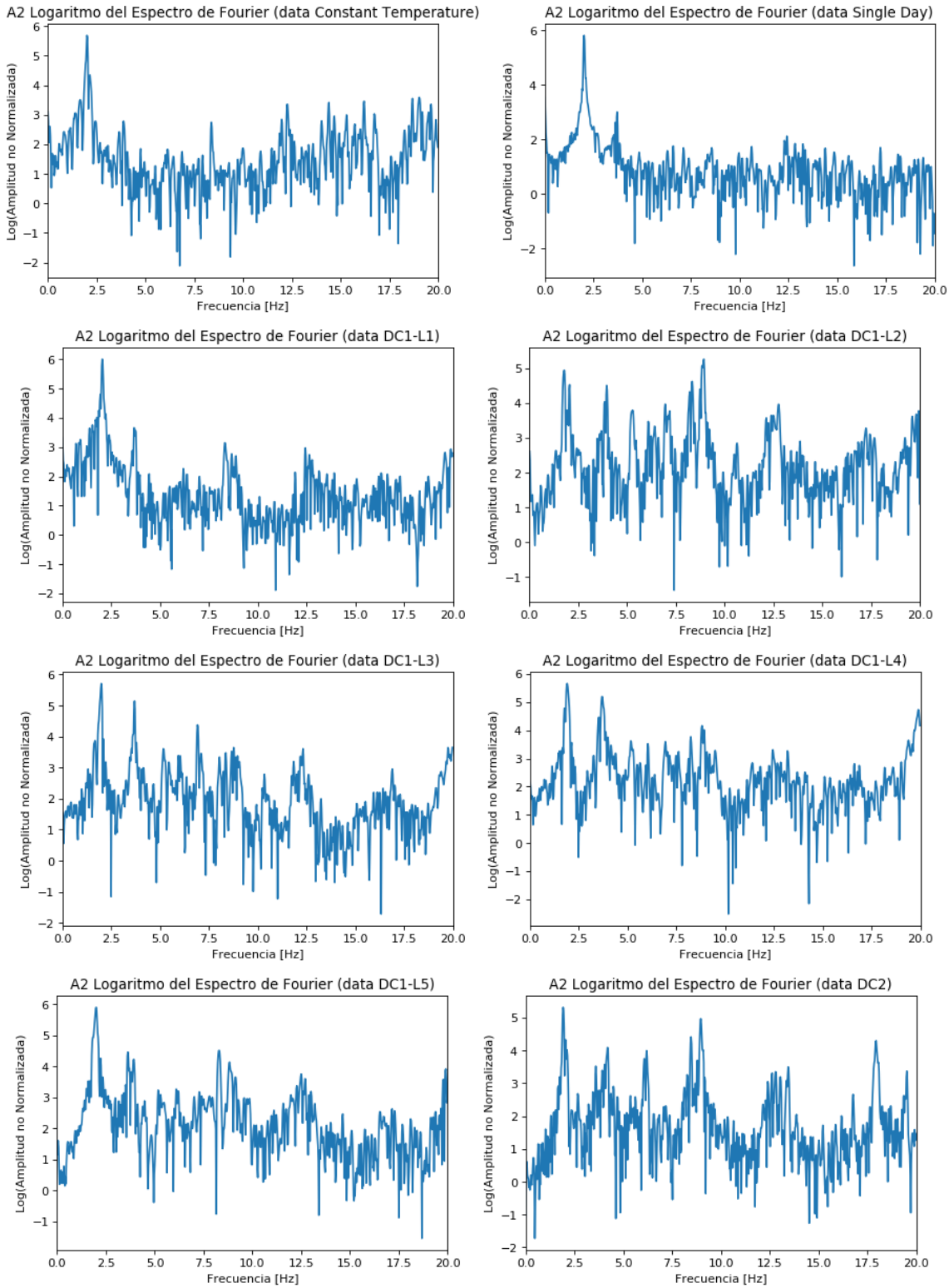

Figura 4.21: Logaritmos de los Espectros de Fourier obtenidos para el puente atirantado en cada escenario (acelerómetro A2).

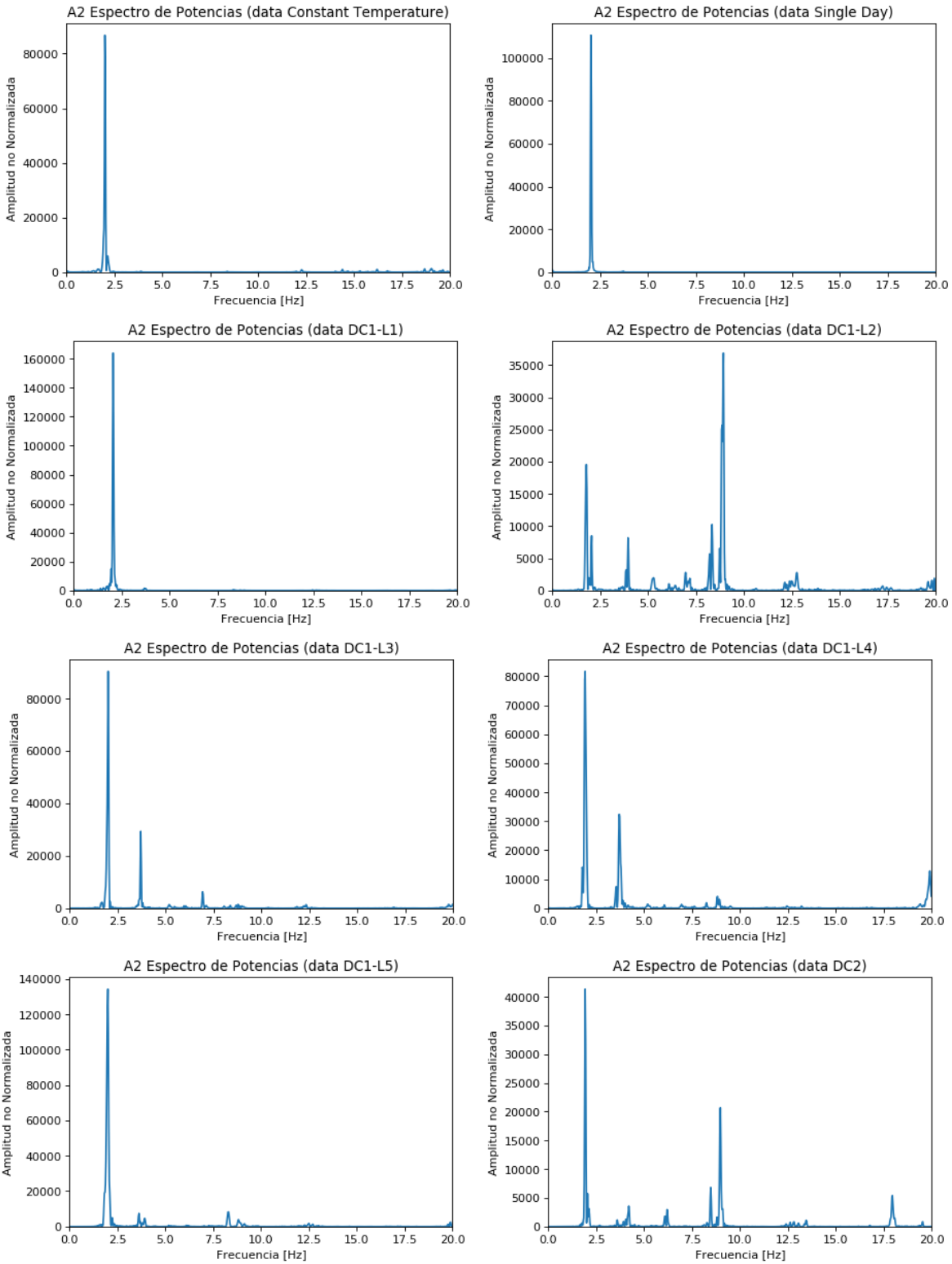

Figura 4.22: Espectros de Potencia calculados para el puente atirantado en cada escenario (acelerómetro A2).

4.2.2. Efecto de la Temperatura

Se evalúa el efecto de la temperatura en las lecturas, mediante tres VAE de dimensión latente 2 (construidas a partir de la arquitectura base, para espectro de Fourier/logaritmo de espectro de Fourier/espectro de Potencias). El entrenamiento consta de datos del data set *constant temperature* y se realizan testeos para el caso *single day*.

Las figuras 4.23, 4.24 y 4.25 muestran el error de reconstrucción para el conjunto de testeo. La horizontal *Threshold* es solo referencial, respecto al conjunto de validación según la expresión 3.3.

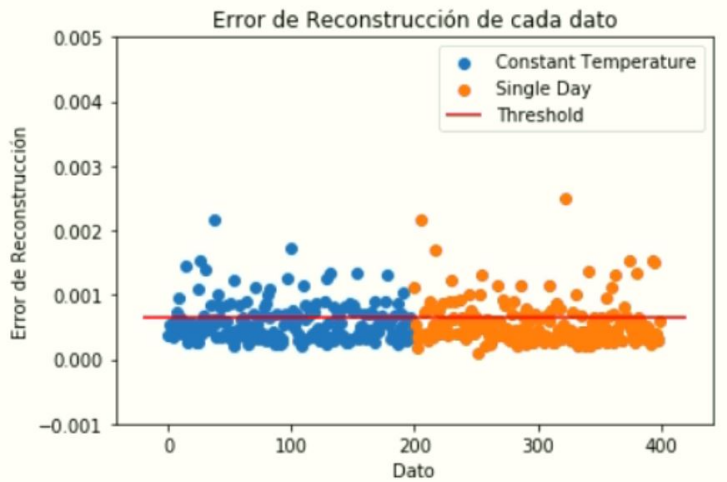

Figura 4.23: Error de reconstrucción por dato para espectros de Fourier casos *constant temperature* y *single day*.

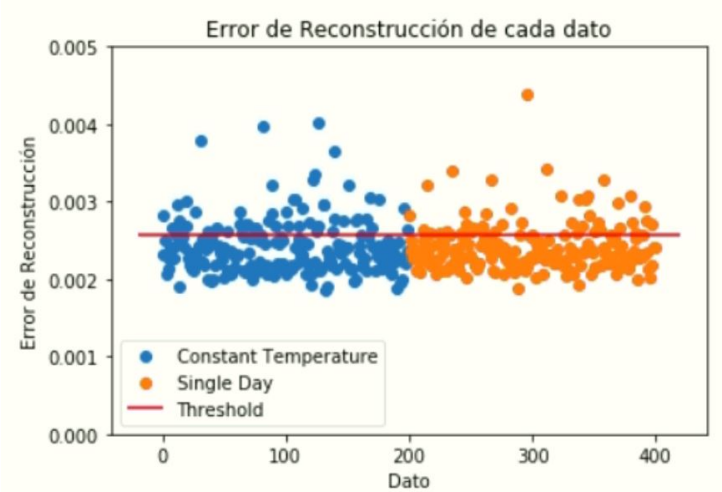

Figura 4.24: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *constant temperature* y *single day*.

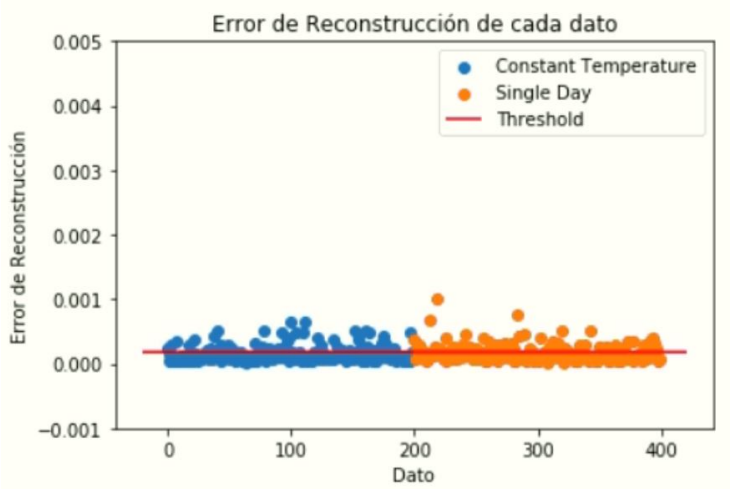

Figura 4.25: Error de reconstrucción por dato para espectros de Potencia casos *constant temperature* y *single day*.

4.2.3. Efecto de daño Damage Case 1

Se usan las VAEs presentadas en el punto anterior para testear los datos del Damage Case 1. Los resultados se resumen en las figuras 4.26, 4.27 y 4.28, y en las matrices de confusión de las tablas 4.7, 4.8 y 4.9.

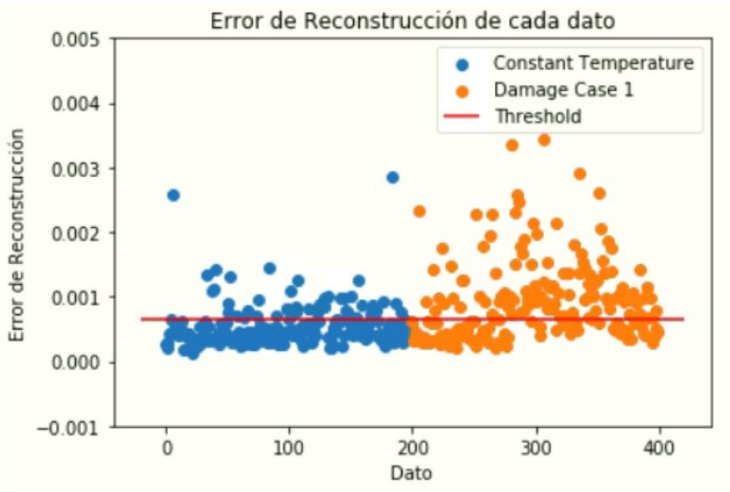

Figura 4.26: Error de reconstrucción por dato para espectros de Fourier casos *constant temperature* y *damage case 1*.

Tabla 4.7: Matriz de Confusión Clasificación Binaria. Espectros de Fourier casos *constant temperature* (data no defectuosa) y *damage case 1* (data defectuosa).

	Predicción Clase Defectuosa	***Predicción Clase No Defectuosa***
Datos Defectuosos	118	82
Datos No Defectuosos	53	147

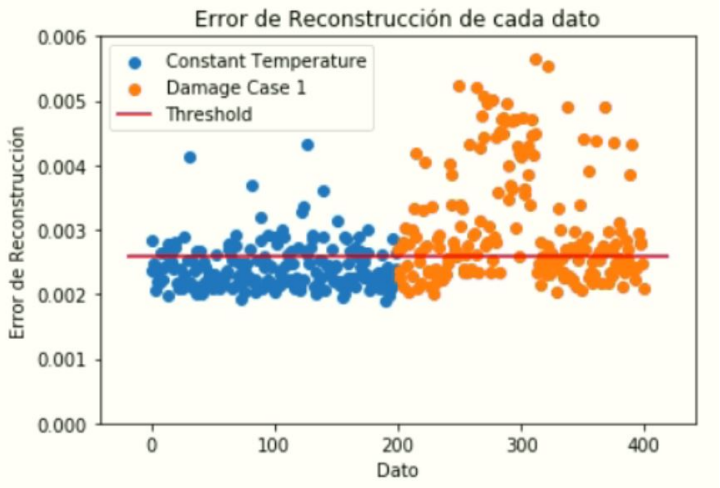

Figura 4.27: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *constant temperature* y *damage case 1*.

Tabla 4.8: Matriz de Confusión Clasificación Binaria. Logaritmos de los Espectros de Fourier casos *constant temperature* (data no defectuosa) y *damage case 1* (data defectuosa).

	Predicción Clase Defectuosa	***Predicción Clase No Defectuosa***
Datos Defectuosos	122	78
Datos No Defectuosos	53	147

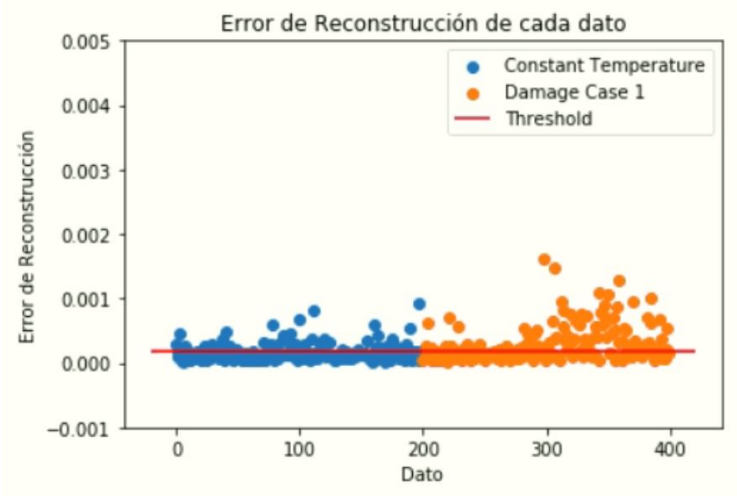

Figura 4.28: Error de reconstrucción por dato para espectros de Potencia casos *constant temperature* y *damage case 1*.

Tabla 4.9: Matriz de Confusión Clasificación Binaria. Espectros de Potencias casos *constant temperature* (data no defectuosa) y *damage case 1* (data defectuosa).

	Predicción Clase Defectuosa	***Predicción Clase No Defectuosa***
Datos Defectuosos	103	97
Datos No Defectuosos	54	146

A partir de estos resultados, se establece el procesamiento basado en el logaritmo de los espectros de Fourier como base de trabajo para el desarrollo de las siguientes redes neuronales. Por otra parte, el escenario no defectuoso corresponderá al data set *single day*, ya que contempla las variaciones de temperatura descritas al comienzo de esta sección.

4.2.4. Resultados Red Convolucional (CNN)

Trabajo de datos

Cada dato corresponde a mediciones de 800 puntos de largo en el rango de frecuencias 0 - $20Hz$, por los 17 acelerómetros seleccionados (17 logaritmos de espectros de fourier). Esto da un total de 13.600 puntos, tamaño que es imposible re-ordenar para cumplir las condiciones mínimas del input de la red VGG16 (32,32,3). Si los datos se truncan a 798 puntos, se obtiene un total de 13566 puntos por cada dato, los que se pueden reordenar en (119,38,3), y esta es la dimensión con la que los datos entran a la red.

Two Class Classification

Se disponen de un total de 448 mediciones para el escenario defectuoso. Con el fin de usar toda la data disponible, se divide este total destinando el 70 % para entrenamiento, 15 % para validación y 15 % para testeo. El número de datos para cada conjunto es balanceado con la misma cantidad de datos del escenario no defectuoso (*single day daya*). La tabla 4.10 muestra la matriz de confusión resultante del testeo.

Tabla 4.10: Matriz de Confusión Clasificación Binaria. Espectros de Potencias casos *single day* (data no defectuosa) y *damaged-structure* (data defectuosa).

	Predicción Clase Defectuosa	***Predicción Clase No Defectuosa***
Datos Defectuosos	59	8
Datos No Defectuosos	3	64

Tabla 4.11: Métricas de Confusión Clasificación Binaria. Espectros de Potencias casos *single day* (data no defectuosa) y *damaged-structure* (data defectuosa).

Modelo	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Two Class CNN	0.92	0.95	0.88	0.91

One Class Classification

La figura 4.29 muestra ejemplos de la data pseudo-negativa generada con los métodos descritos en 3.3.1. La red se entrena con 2756 datos de la clase escenario no defecutoso y 2756 datos de la clase pseudo-negativa generada, mientras que el conjunto de validación consta de 734 datos de cada clase.

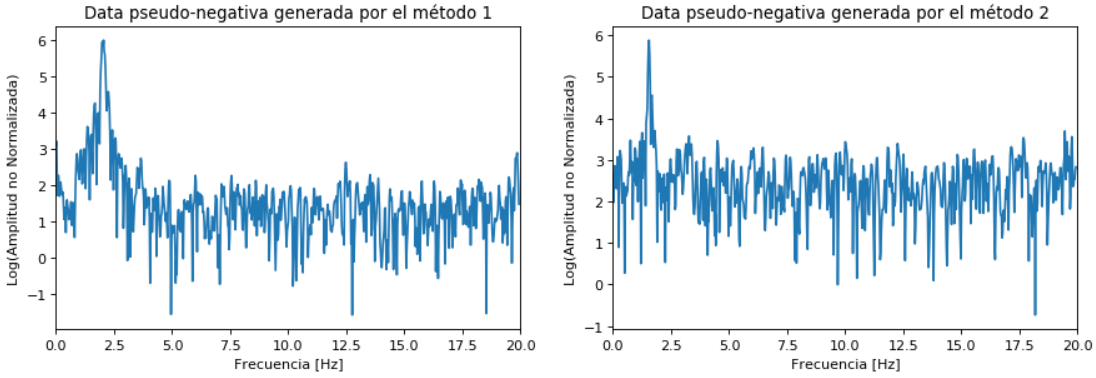

Figura 4.29: Logaritmos de Espectros de Fourier generados como clase pseudo-negativa a partir del escenario no defectuoso en el data set Puente. A la izq. se introdujo ruido gaussiano, a la der. se desplazó la posición de los peaks.

Para la fase de testeo se preparan tres grupos, los que constan de 200 datos del escenario no defectuoso y 200 datos de algún escenario defectuoso. El primer grupo de testeo incluye datos del *damage case 1*, el segundo set tiene datos de *damage case 2* y el tercero es un subconjunto de un mix de ambos casos de daño. Esta forma de construcción de los conjuntos de testeo se mantiene para todas las arquitecturas de red probadas. Las tablas 4.12, 4.13, 4.14 y 4.15 resumen los resultados obtenidos

Tabla 4.12: Métricas de Confusión Clasificación de una clase. Data pseudo-negativa generada a partir del método 1. Modelo entrenado con Feature Extraction.

	Accuracy	***Precision***	***Recall***	***F1 Score***
Test Set 1	0.46	0.39	0.14	0.21
Test Set 2	0.45	0.36	0.12	0.19
Test Set 3	0.47	0.41	0.15	0.22

Tabla 4.13: Métricas de Confusión Clasificación de una clase. Data pseudo-negativa generada a partir del método 2. Modelo entrenado con Feature Extraction.

	Accuracy	***Precision***	***Recall***	***F1 Score***
Test Set 1	0.55	0.64	0.23	0.34
Test Set 2	0.46	0.28	0.05	0.08
Test Set 3	0.52	0.56	0.17	0.25

Tabla 4.14: Métricas de Confusión Clasificación de una clase. Data pseudo-negativa generada a partir del método 1. Modelo entrenado con Fine Tuning.

	Accuracy	***Precision***	***Recall***	***F1 Score***
Test Set 1	0.47	0.36	0.08	0.13
Test Set 2	0.44	0.07	0.01	0.02
Test Set 3	0.45	0.24	0.04	0.08

Tabla 4.15: Métricas de Confusión Clasificación de una clase. Data pseudo-negativa generada a partir del método 2. Modelo entrenado con Fine Tuning.

	Accuracy	***Precision***	***Recall***	***F1 Score***
Test Set 1	0.53	0.74	0.10	0.18
Test Set 2	0.49	0.22	0.01	0.02
Test Set 3	0.52	0.65	0.07	0.12

4.2.5. Resultados Autoencoder Variacional (VAE)

El entrenamiento de las redes se lleva a cabo con 2600 datos en el conjunto de *training* y 400 en el conjunto de *cross validation*. La figura 4.30 muestra la representación de la VAE con dimensión latente igual a 2 para los datos del conjunto de testeo número 3. La clase 0 (color azul) representa los datos del escenario defectuoso, mientras que la clase 1 (color celeste) representa los datos del escenario no defectuoso.

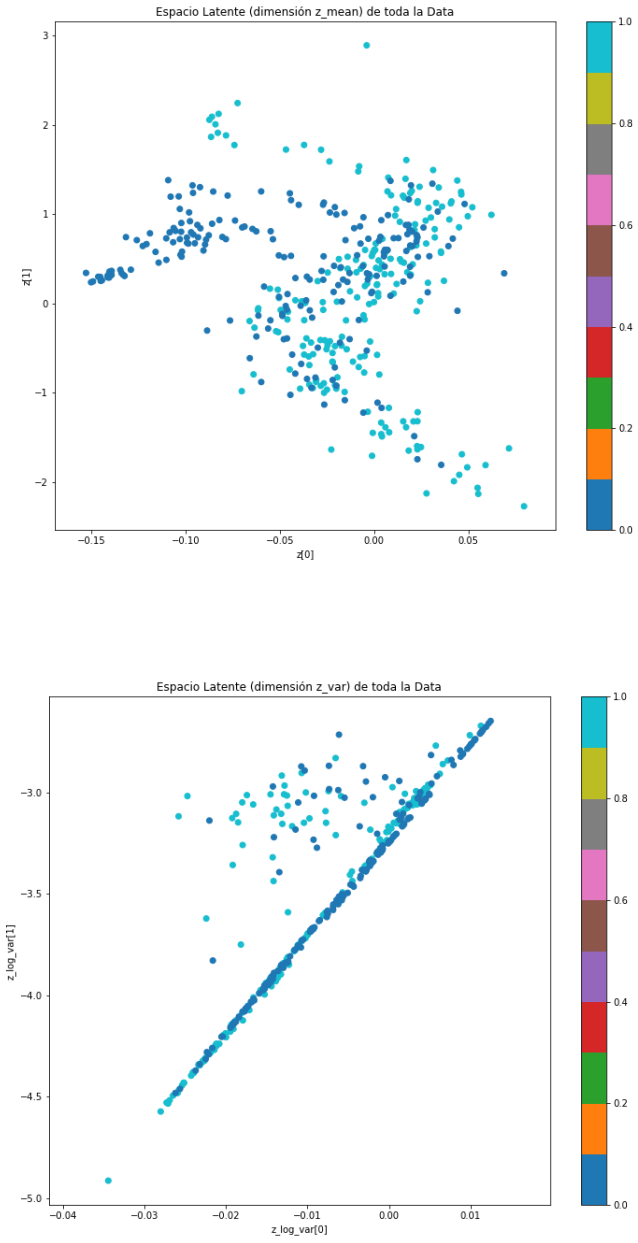

Figura 4.30: Representación del espacio latente para dim.Z=2 (data Puente). Arriba el plano del valor promedio, abajo el plano del valor desviación estándar.

Reconstrucción de datos

Se presenta la reconstrucción de un dato del escenario no defectuoso y uno del escenario defectuoso (figuras 4.31 y 4.32 respectivamente). Al igual que en el caso barra, estas figuras son ilustrativas del tipo de reconstrucción que se puede esperar para este tipo de datos, basándose en la VAE de dimensión latente 2.

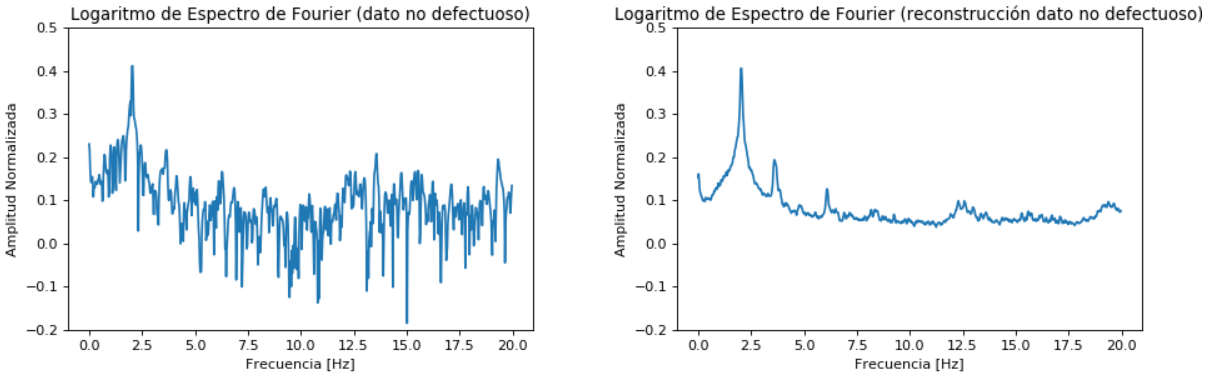

Figura 4.31: Logaritmo de Espectro de Fourier, escenario no defectuoso. A la izquiera dato de entrada a la red, a la derecha reconstrucción lograda por la VAE.

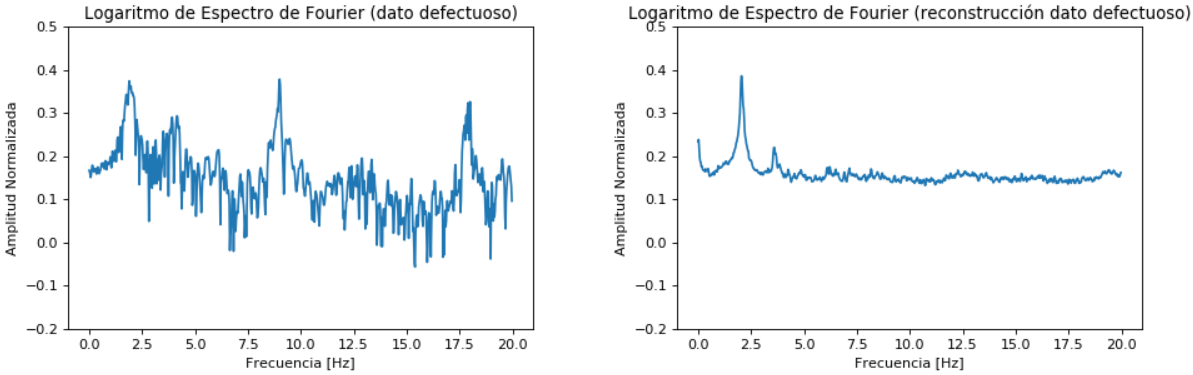

Figura 4.32: Logaritmo de Espectro de Fourier escenario defectuoso. A la izquiera dato de entrada a la red, a la derecha reconstrucción lograda por la VAE.

Clasificación

El clasificador desarrollado para cada VAE toma como base los errores de reconstrucción del conjunto de validación, según lo descrito en la sección 3.3.2. La tabla 4.16 resume el valor determinado para el threshold en cada caso y las figuras 4.33 a 4.35 muestran gráficamente este valor en contraste al error de reconstrucción de cada conjunto de validación.

Tabla 4.16: Thresholds obtenidos para cada VAE.

Dim. Latente	**Threshold (MSE)**
z=2	0.00256
z=8	0.00243
z=32	0.00249

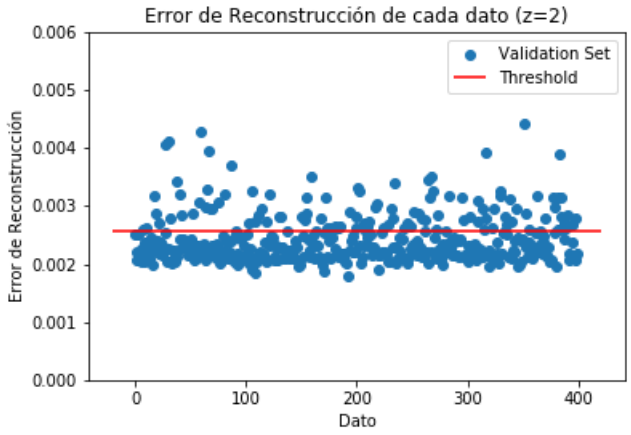

Figura 4.33: Error de reconstrucción para el conjunto de validación (Z=2).

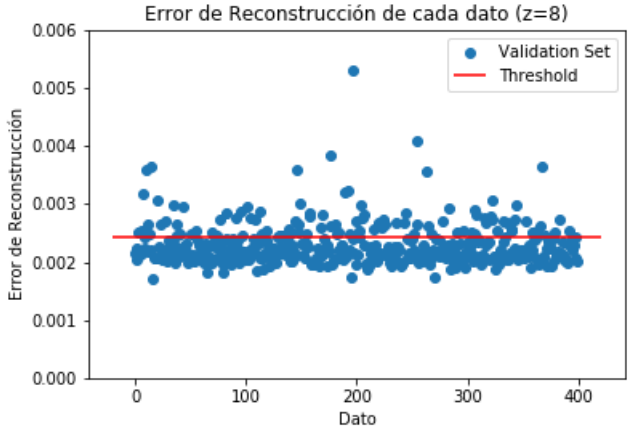

Figura 4.34: Error de reconstrucción para el conjunto de validación (Z=8).

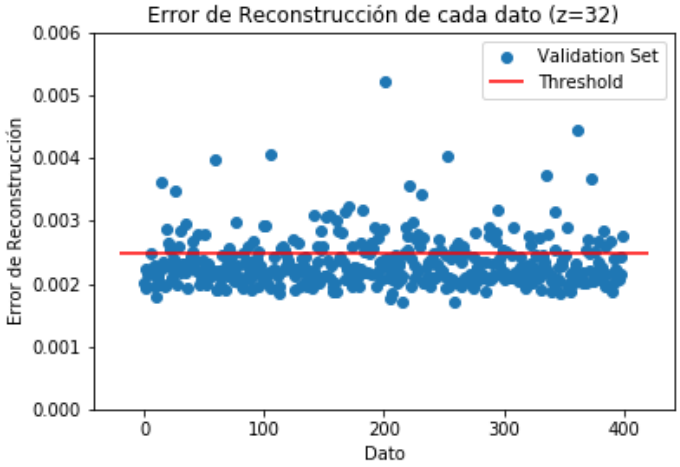

Figura 4.35: Error de reconstrucción para el conjunto de validación (Z=32).

El valor del threshold para cada VAE es constante y se usa para cada conjunto de prueba. Las figuras 4.36 a 4.44 muestran de manera gráfica esta clasificación basada en el error de reconstrucción, para cada VAE y cada conjunto de testeo construido.

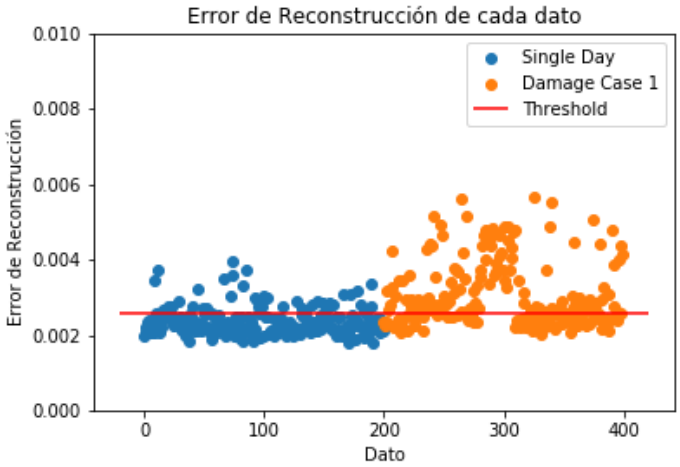

Figura 4.36: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damage case 1* (conjunto de testeo 1, z=2).

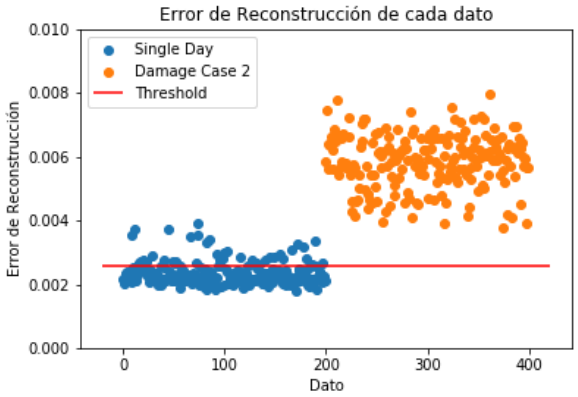

Figura 4.37: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damage case 2* (conjunto de testeo 2, z=2).

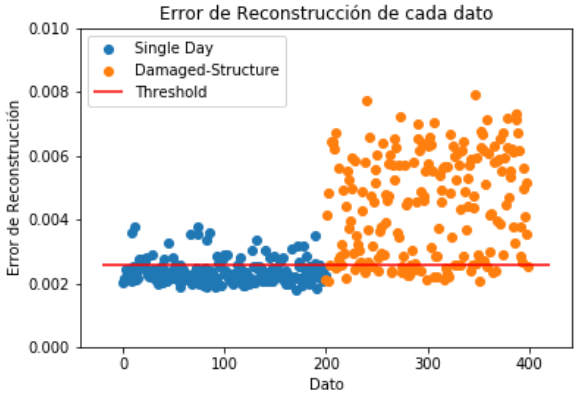

Figura 4.38: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damaged-structure* (conjunto de testeo 3, z=2).

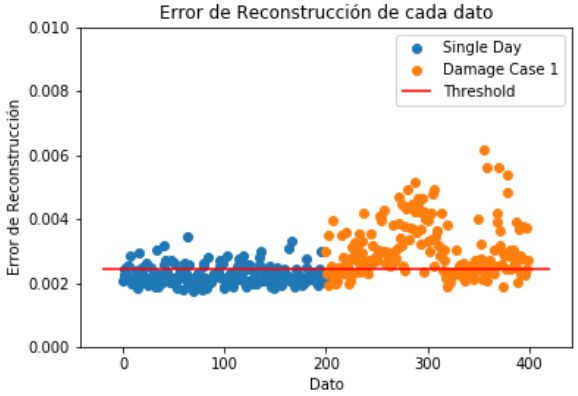

Figura 4.39: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damage case 1* (conjunto de testeo 1, z=8).

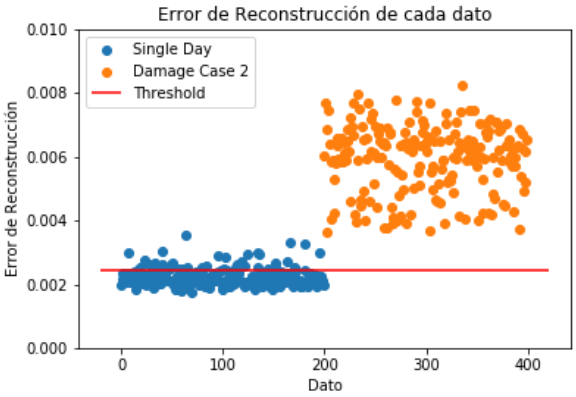

Figura 4.40: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damage case 2* (conjunto de testeo 2, z=8).

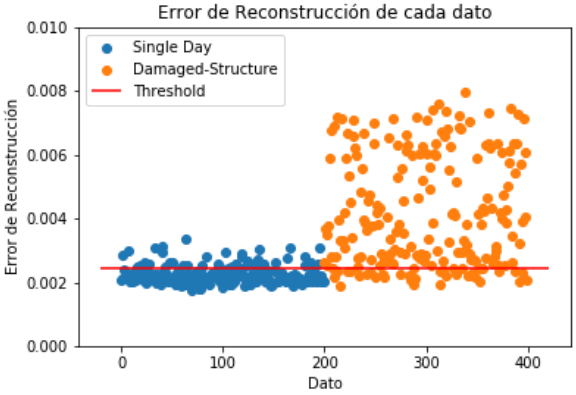

Figura 4.41: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damaged-structure* (conjunto de testeo 3, z=8).

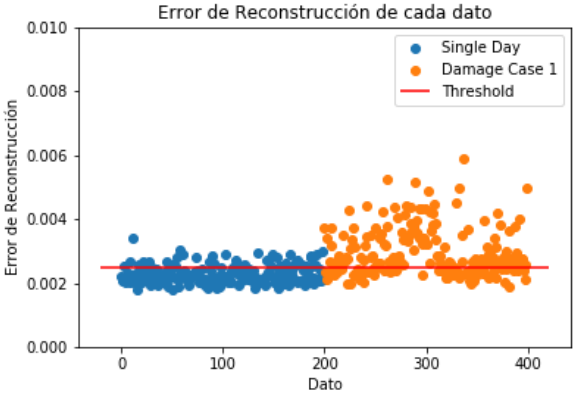

Figura 4.42: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damage case 1* (conjunto de testeo 1, z=32).

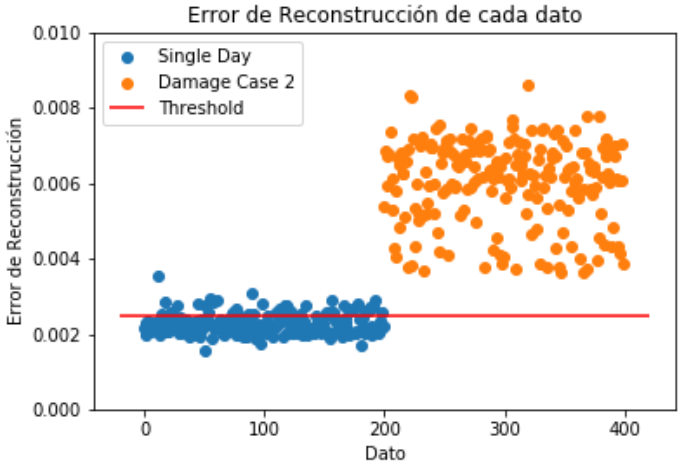

Figura 4.43: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damage case 2* (conjunto de testeo 2, z=32).

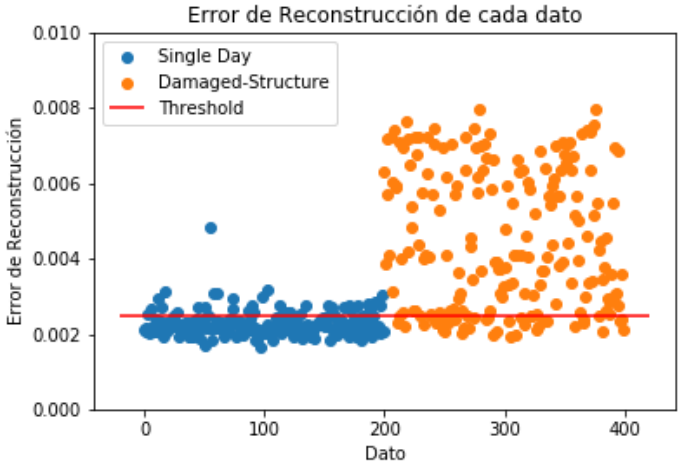

Figura 4.44: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damaged-structure* (conjunto de testeo 3, z=32).

En el caso del escenario de daño 1, existen 5 subdivisiones. Las figuras 4.45, 4.46 y 4.47 vuelven a mostrar la clasificación del conjunto de testeo 1, tomando en cuenta cada sub-escenario de daño por separado.

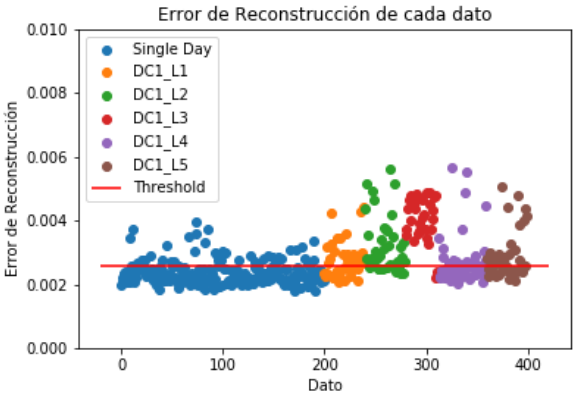

Figura 4.45: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damage case 1* (conjunto de testeo 1, z=2). Dividido por nivel de daño.

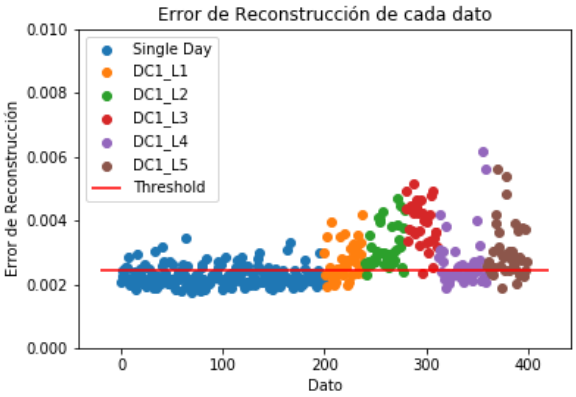

Figura 4.46: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damage case 1* (conjunto de testeo 1, z=8). Dividido por nivel de daño.

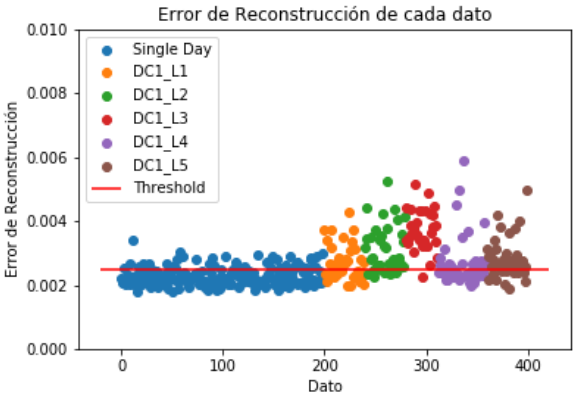

Figura 4.47: Error de reconstrucción por dato para logaritmos de espectros de Fourier casos *single day* y *damage case 1* (conjunto de testeo 1, z=32). Dividido por nivel de daño.

Las tablas 4.17, 4.18 y 4.19, presentadas a continuación, muestran las métricas de confusión obtenidas para cada VAE, ordenadas según cada conjunto de testeo.

Tabla 4.17: Métricas de Confusión para cada dimensión latente, conjunto de testeo número 1 (*damage case 1*).

Modelo	*Accuracy*	*Precision*	*Recall*	*F1 Score*
Dim. Latente 2	0.72	0.75	0.66	0.70
Dim. Latente 8	0.75	0.76	0.73	0.75
Dim. Latente 32	0.72	0.74	0.67	0.70

Tabla 4.18: Métricas de Confusión para cada dimensión latente, conjunto de testeo número 2 (*damage case 2*).

Modelo	*Accuracy*	*Precision*	*Recall*	*F1 Score*
Dim. Latente 2	0.89	0.82	1	0.90
Dim. Latente 8	0.89	0.82	1	0.90
Dim. Latente 32	0.89	0.82	1	0.90

Tabla 4.19: Métricas de Confusión para cada dimensión latente, conjunto de testeo número 3 (*damaged-structure*).

Modelo	*Accuracy*	*Precision*	*Recall*	*F1 Score*
Dim. Latente 2	0.80	0.80	0.82	0.81
Dim. Latente 8	0.81	0.80	0.83	0.81
Dim. Latente 32	0.80	0.81	0.81	0.81

4.2.6. Resultados Autoencoder Variacional con One-Class Support Vector Machines (VAE+OCSVM)

Tal como en el primer caso de estudio, el parámetro ν se mantiene en 0.001. Las figuras 4.48 a 4.56 muestran las curvas de aprendizaje de las métricas de confusión para cada valor de γ, según cada conjunto de testeo. Las tablas 4.20, 4.21 y 4.22 resumen los mejores resultados obtenidos para cada dimensión latente. Es importante notar que se escoge un único valor de γ por VAE pre-entrenada, valor que es transversal para todos los conjuntos de testeo. Finalmente, la elección de γ es principalmente en base al valor de *accuracy* y *f1 score*.

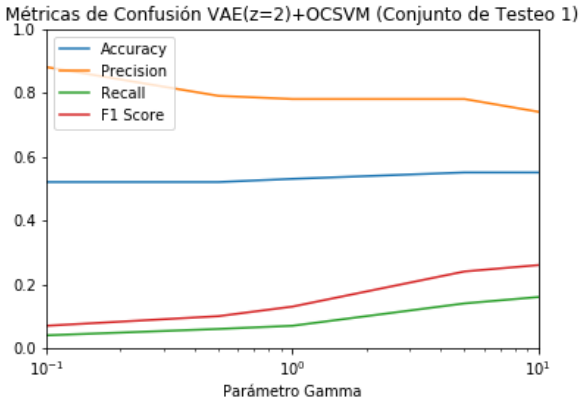

Figura 4.48: Métricas de confusión para $z = 2$ como dimensión latente, para el primer conjunto de testeo.

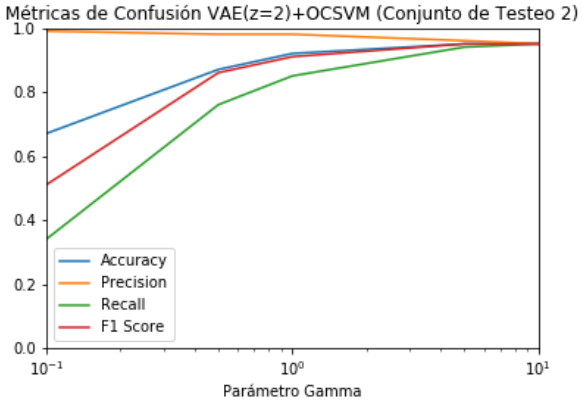

Figura 4.49: Métricas de confusión para $z = 2$ como dimensión latente, para el segundo conjunto de testeo.

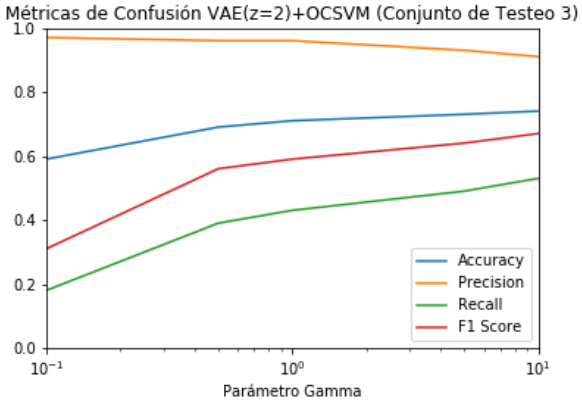

Figura 4.50: Métricas de confusión para $z = 2$ como dimensión latente, para el tercer conjunto de testeo.

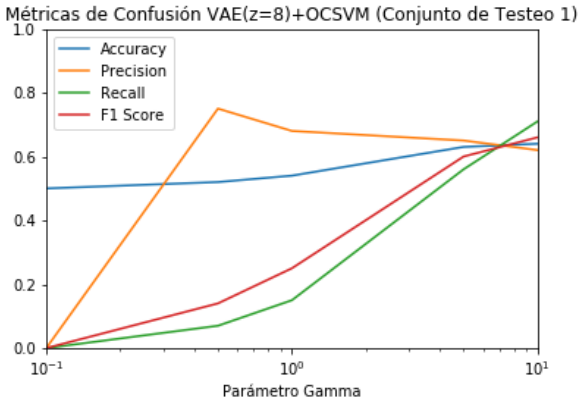

Figura 4.51: Métricas de confusión para $z = 8$ como dimensión latente, para el primer conjunto de testeo.

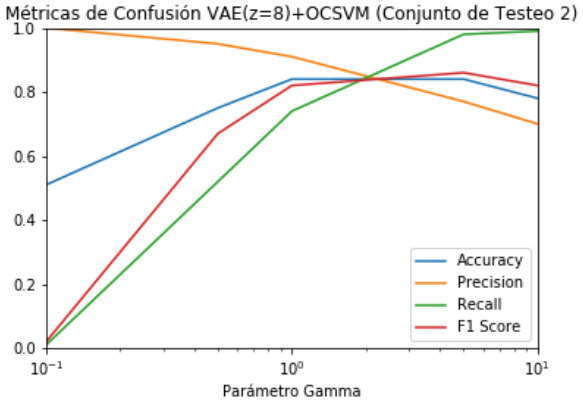

Figura 4.52: Métricas de confusión para $z = 8$ como dimensión latente, para el segundo conjunto de testeo.

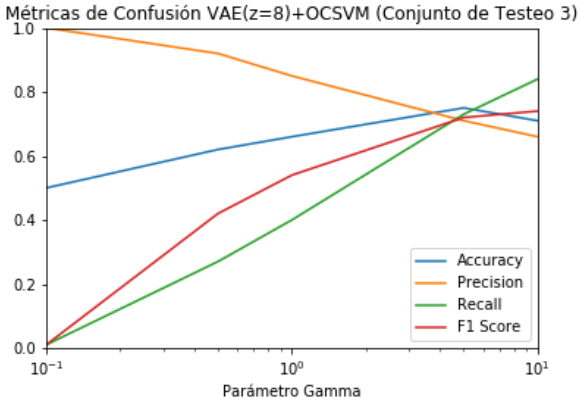

Figura 4.53: Métricas de confusión para $z = 8$ como dimensión latente, para el tercer conjunto de testeo.

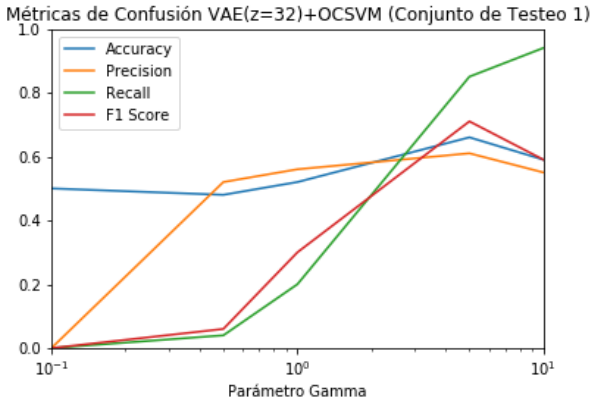

Figura 4.54: Métricas de confusión para $z = 32$ como dimensión latente, para el primer conjunto de testeo.

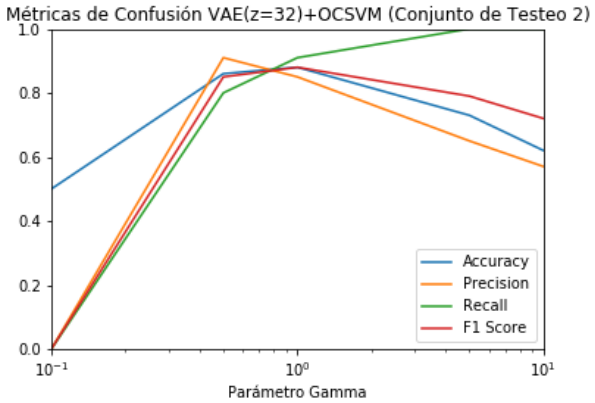

Figura 4.55: Métricas de confusión para $z = 32$ como dimensión latente, para el segundo conjunto de testeo.

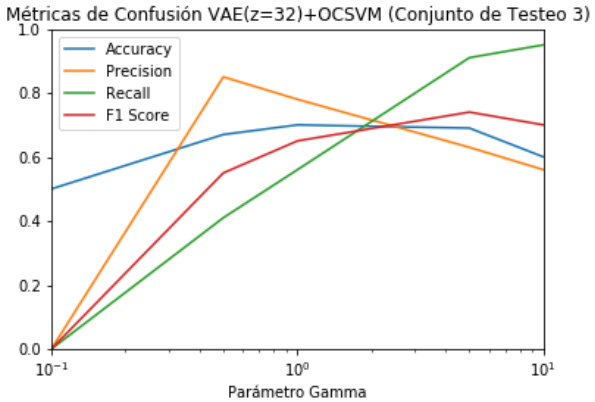

Figura 4.56: Métricas de confusión para $z = 32$ como dimensión latente, para el tercer conjunto de testeo.

Tabla 4.20: Mejores resultados de clasificación mediante OCSVM para VAE pre-entrenada con Z=2.

VAE $z = 2$	**Valor de Gamma**	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Test Set 1		0.55	0.74	0.16	0.26
Test Set 2	$\gamma = 10$	0.95	0.95	0.95	0.95
Test Set 3		0.74	0.91	0.53	0.67

Tabla 4.21: Mejores resultados de clasificación mediante OCSVM para VAE pre-entrenada con Z=8.

VAE $z = 8$	**Valor de Gamma**	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Test Set 1		0.63	0.65	0.56	0.60
Test Set 2	$\gamma = 5$	0.84	0.77	0.98	0.86
Test Set 3		0.75	0.71	0.73	0.72

Tabla 4.22: Mejores resultados de clasificación mediante OCSVM para VAE pre-entrenada con Z=32.

VAE $z = 32$	**Valor de Gamma**	***Accuracy***	***Precision***	***Recall***	***F1 Score***
Test Set 1		0.66	0.61	0.85	0.71
Test Set 2	$\gamma = 5$	0.73	0.65	1	0.79
Test Set 3		0.69	0.63	0.91	0.74

Capítulo 5

Discusión

5.1. Sobre el procesamiento de los datos

De la observación de las series temporales, mostradas en la figura 4.2 para el caso de estudio 1 y las figuras 4.17 y 4.18 para el caso de estudio 2 se distingue claramente una diferencia una diferencia en la periodicidad de las respuestas entre un caso y otro. Por una parte, las respuestas obtenidas para el caso de la barra son respuestas consistentes, en las que se reconoce una tendencia clara tanto para la data de los escenarios defectuosos como para los escenarios no defectuosos. En contraste, la respuesta temporal obtenida para el caso del puente atirantado se tienen variaciones muy grandes en la tendencia de los datos, incluso después de quitar los peaks correspondientes a la circulación de vehículos y hacer el detrending.

Lo anterior se justifica en la adquisición de datos y el contexto de cada caso. Mientras que los datos de la barra se obtuvieron en un laboratorio, con un ambiente controlado y una fuente de excitación constante (*shaker*), los datos del puente están sometidos a distintos factores ambientales. El puente está expuesto al viento y a la circulación tanto de personas como de vehículos, todos estos elementos forman la fuente de vibración ambiental, y ninguno de estos es constante.

Así es como los tramos resultantes en la segmentación del caso barra son más consistentes, y de la misma forma producen espectros de Fourier más consistentes y con una menor variabilidad respecto a los obtenidos en el caso del puente. Es importante reconocer esta diferencia ya que se presume que es la principal causa de errores en la clasificación para los datos del puente.

5.2. Sobre el efecto de la temperatura (caso Puente)

De la experiencia con el primer caso estudio se determina que la red que da mejores resultados es la VAE (más información en las secciones siguientes). Se construyen redes VAE para evaluar de forma general el comportamiento del puente y la variación en el espectro por cambios de temperatura y por presencia de daño (simulado).

En la sección 4.2.2 (figuras 4.23, 4.24 y 4.25 se muestra lado a lado el error de reconstrucción obtenido por una VAE de dimensión latente 2, para los data sets *constant temperature* y *single day*. Cada una de las figuras mencionadas muestra resultados para una presentación del espectro de frecuencias distinta (según lo comentado en 4.2.1).

Los tres autoencoders variacionales muestran que no existe una diferencia apreciable en el error de reconstrucción entre un data set y otro. Esto permite descartar el hecho de que un cambio de temperatura en el puente pueda generar una medición que sea clasificada como defectuosa, o al menos la aparición de un *outlier* no se vuelve más probable producto de la temperatura.

5.3. Sobre el efecto de daño DC1 (caso Puente)

De la comparación de las matrices de confusión mostradas en las tablas 4.7, 4.8 y 4.9, resultantes de las tres VAE mencionadas en la sección anterior, entrenadas con los mismos parámetros, se observa que la aplicación del logaritmo sobre los espectros de Fourier se traduce en un mayor número de datos correctamente clasificados para el escenario de daño.

En los espectros de Fourier obtenidos (ver figuras 4.19 y 4.20) el peak en el primer modo de vibración (cercano a $2Hz$) parece tener la mayor incidencia sobre el error de reconstrucción. La aplicación de logaritmo sobre los espectros de Fourier permite que los peaks a frecuencias mayores sean acentuados en valor, lo que entrega más herramientas a las redes neuronal para discernir si un dato corresponde a un escenario defectuoso o no defectuoso.

A pesar de que la mejoría en la exactitud de la clasificación es baja, se decide usar el logaritmo de los espectros de Fourier para el entrenamiento de las arquitecturas propuestas.

5.4. Sobre la clasificación de dos clases basada en CNN (VGG16)

La clasificación de dos clases es un tema bastante abordado, en particular se espera que una CNN sea capaz de clasificar datos con gran exactitud cuando se disponen de todas las clases durante el entrenamiento. En el caso de estudio de la barra, las clases *defectuosa* y *no defectuosa* son totalmente separables por la CNN propuesta, como se nota en la matriz de confusión de la tabla 4.1. Por otra parte, esta clasificación de dos clases no funciona con tal exactitud aplicada a los datos del caso estudio del puente (ver tabla 4.10), donde, bajo los mismos parámetros, se logran clasificar correctamente 123 de 134 datos en la etapa de testeo (92 % de exactitud).

Es en vista de la poca cantidad de data del escenario defectuoso (en el caso de estudio 2) que se justifica la elección de usar una red pre-entrenada (VGG16) en lugar de una red inicilizada de forma aleatoria.

Dado que la CNN con clasificación de dos clases posee más información para entrenar y discriminar una u otra clase, se le puede tomar como una referencia para la clasificación de una clase basada en CNN. En particular para los valores que se pueden esperar de las métricas de confusión en la evaluación de la CNN con clasificador de una clase. Esta referencia representa una aproximación a la mejor clasificación que se puede obtener con la arquitectura propuesta, pero con la consideración de que a pesar de que fue la mejor clasificación obtenida de dos clases, la función costo puede aún estar optimizada a un óptimo local.

5.5. Sobre la clasificación de una clase basada en CNN (VGG16)

En el caso de estudio de la barra, la tabla de evaluación de la clasificación de una clase (tabla 4.2) muestran que la mejor combinación de métodos corresponde a la inclusión de ruido gaussiano en la data no defectuosa como generación de data pseudo-negativa, junto a un fine tuning de los pesos de la última etapa convolucional de la VGG16. Esta decisión toma como base los valores de *accuracy* y *f1 score*.

Se nota que para todas las configuraciones probadas (excepto la generación de data con cambio de peaks y la VGG16 con pesos congelados), el valor de precisión es unitario. Esto quiere decir que cuando la data es clasificada como escenario defectuoso, lo hace con una alta confiabilidad. Como el valor de *recall* o sensibilidad es menor, se habla de que el modelo tiene problemas para detectar la clase defectuosa.

Los resultados obtenidos por la CNN con clasificador de una clase empeoran bastante cuando se aplica esta arquitectura a los datos del puente y la clases pseudo-negativas generadas para este (mostradas en la figura 4.29). Para los tres conjuntos de testeo la configuración que da mejores resultados es la data pseudo-negativa generada por intercambio de peaks junto al uso de las etapas convolucionales con pesos congelados (tabla 4.13). Este resultado es un hallazgo ya que es totalmente opuesto a lo obtenido al trabajar la data de la barra.

Los valores para todas las métricas de confusión en todos los casos son bastante bajos (tablas 4.12 a 4.15), por lo que la One-Class CNN no es un método confiable para la detección de fallas en un caso real, como el caso de estudio 2.

Como apreciación adicional, se tiene que el Test Set 1 (compuesto de daño de menor intensidad) da mejores resultados que el Test Set 2 (compuesto de daño de mayor intensidad). Esto ocurre ya que la clase pseudo-negativa generada se asemeja más a los espectros de Fourier del caso DC1. Como se buscaba poder identificar daño de la menor intensidad posible, el desplazamiento de peaks fue de 5 pasos en frecuencia como mínimo (según lo expuesto en 4.2.1), distribuidos aleatoriamente entre desplazamientos hacia la izquierda o hacia la derecha. Como la clase pseudo-negativa es más parecida a un daño de baja intensidad, es de esperar que clasifique mejor el conjunto de testeo 1.

Esto ocurre porque a pesar de que la OC-CNN propuesta es una adaptación a un clasificador de una clase, su arquitectura actúa como un clasificador binario. Es decir, la CNN distingue una forma de data de otra, pero no logra responder a la clasificación de una clase donde el problema se convierte en un valor de verdad. La pertenencia de un dato a la clase no defectuosa es un valor *si* o *no*.

Como ya se ha mencionado, una CNN es particularmente buena para aprendizaje supervisado, donde se conocen las clases del conjunto de entrenamiento. Si bien la aplicación de una CNN con clasificador de una clase es novedosa, es muy difícil de implementar, ya que se requieren datos de la clase pseudo-negativa, y como a priori no se conoce que forma tengan, el entrenamiento queda sujeto a que tan bien se es capaz de predecir el comportamiento de la clase negativa. Se sugiere desarrollar una red GAN, de tal forma que se incorpore una red neuronal que busque generar datos que se parezcan en gran medida a la data positiva, para que el clasificador logre una representación más general y completa del estado *no defectuoso* de la estructura.

5.6. Sobre la clasificación de una clase basada en VAE

Respecto a las representaciones latentes cuando la dimensión es $Z = 2$ (figura 4.5), en la data de la barra se puede apreciar que las clases defectuosa y no defectuosa son separables en la representación z_{mean}. Contrario a lo que se esperaba, la data defectuosa queda en medio de la data no defectuosa. Incluso en el plano z_{var} se puede observar un cúmulo de representaciones de la clase no defectuosa separada del resto. Esto es un buen indicador de que la VAE

aprendió a diferenciar la clase de entrenamiento del resto, lo que se confirma más adelante. En contraste, esto no se cumple a cabalidad para el caso del puente (figura 4.30), donde los cluster de la data defectuosa y no defectuosa se superponen tanto para z_{mean} y z_{var}, dejando un pequeño espacio reconocible como una diferencia real entre una representación y otra. Si bien esto da intuición sobre el entrenamiento, no es decidor, sobre todo considerando que se tienen aproximadamente 11 veces más puntos por dato en el caso del puente que en la barra, por lo que la reducción de dimensionalidad puede ser muy acentuada en el caso del puente, lo que genera malas representaciones en un espacio tan reducido.

Lo anterior se puede concluir también de la reconstrucción de los datos. Mientras que en el caso de la barra la reconstrucción es prácticamente idéntica para todas las redes desarrolladas (figuras 4.6 y 4.7), en el caso del puente la reconstrucción es menos precisa, y solo representa el skyline del espectro original (figuras 4.31 y 4.32), pero en ningún caso se logra reconstruir correctamente el *input*. De todas formas, lo que se puede reconocer como un contorno superior de la forma del espectro se reconstruye de mejor manera en la data del escenario no defectuoso (data de entrenamiento).

Según la tabla 4.4, la mejor clasificación de los datos de la barra se logra con una dimensionalidad latente $z = 8$. La clasificación en este caso es bastante buena, con un 98 % de exactitud.

En el caso del puente atirantado, el threshold determinado para cada VAE en base al conjunto de validación deja muchos outliers fuera de la clasificación dentro de este mismo conjunto, como se aprecia en las figuras 4.33, 4.34 y 4.35. Esto produce a posteriori que un número no menor de datos del escenario no defectuoso sean señalados como alertas.

De las tablas 4.17, 4.18 y 4.19 se rescata que la VAE con dimensión latente $z = 8$ otorga mejores resultados en la clasificación de los datos del *damage case 1*. A pesar de que las tres VAE clasifiquen con igual exactitud el conjunto de *damage case 2*, esto genera que la VAE con $z = 8$ también clasifique con mayor exactitud el tercer conjunto de testeo. Esta diferencia es pequeña entre una VAE y otra, por lo que no se puede descartar que la configuración de mejores métricas sea escogida por una convergencia más afortunada, en lugar de una elección óptima de parámetros.

Para el conjunto de testeo 1, constituido por datos del escenario no defectuoso (*single day*) y datos del escenario DC1, las figuras 4.36, 4.39 y 4.42 muestran valores de error de reconstrucción bastante cercanos entre ambas clases. Sin embargo, al dividir la data DC1, identificando cada prueba por separado en las figuras 4.45, 4.46 y 4.47, se logra apreciar que los sub-escenarios DC1-L2 y DC-L3 sobresalen del resto y quedan, en gran parte, por sobre el threshold. Una clasificación en torno a estos sub-escenarios obtendría mejores métricas de confusión, y podría establecer un mínimo en el nivel de daño detectable por la red (sin despreciar que todos los sub-escenarios DC1 poseen datos que son correctamente clasificados como escenarios de falla.

De la misma forma, de las figuras 4.37, 4.40 y 4.43 se puede notar que todos los datos del escenario DC2 son clasificados correctamente como clase defectuosa, ya que se ubican sobre el threshold determinado para cada VAE. Lo que empeora las métricas son los outliers de la data no defectuosa, marcados como alertas. Sin embargo, se observa que si se aumenta el valor del threshold para reducir los falsos positivos, se logra una clasificación casi perfecta de los datos DC2.

A pesar de que aumentar el valor del threshold represente un *tradeoff* entre falsos positivos y falsos negativos en algunos casos, es una opción interesante. Dependiendo del nivel de daño mínimo que se desee establecer como cota, este *tradeoff* puede mejorar considerablemente las métricas de confusión y la clasificación en general para esta arquitectura, exponiendo menos falsos positivos pero dando pie a que algunos datos con daño menor no sean detectados. En la práctica, estos daños menores pueden no tener impacto o ser detectados a tiempo cuando su nivel aumente.

5.7. Sobre la clasificación de una clase basada en OCSVM a partir de VAE.

Dados los resultados obtenidos de esta clasificación en ambos data sets, se observa que al aumentar el valor de γ o parámetro de escala del *kernel rbf* se aumenta la calidad de la clasificación hasta cierto punto, desde el cual empieza a disminuir. Se infiere a partir de la tendencia de las métricas que la OCSVM determina un espacio para la clase negativa (o clase 0) a partir de la clase positiva (clase 1 o clase de entrenamiento), con lo que al aumentar el valor de γ se amplia la caracterización de la clase negativa, con lo que se va logrando una mayor exactitud, hasta que se alcanza un peak. Pasado este valor peak, la definición de la clase negativa se amplía tanto que se empieza a clasificar data positiva como si fuera data negativa.

Con base en la exactitud y en el puntaje f1, se puede decir que el encoder de dimensión latente $z = 8$, con $\gamma = 5$ obtiene la mejor clasificación de la data de la barra, con un 97 % de *accuracy*. Coincidentemente, estos valores para la dimensión latente y γ también son los que maximizan las métricas de confusión en la clasificación de los datos del puente atirantado. Si bien $z = 2$ y $\gamma = 10$ dan una exactitud del 95 % en el conjunto de testeo 2, esta configuración no es potente para la clasificación del conjunto de testeo 1. La decisión de $z = 8$ y $\gamma = 5$ se define entonces por el conjunto de testeo 3, que involucra datos de todos los escenarios de daño simulado.

Lo anterior también puede ser discutido bajo los términos de la arquitectura anterior y el *tradeoff* entre precision y sensibilidad, estableciendo un nivel de daño objetivo sobre el cual clasificar los datos como defectuosos.

5.8. Anotaciones finales

Las tablas 5.1 y 5.2 muestran un resumen de los mejores resultados obtenidos por cada algoritmo (con los parámetros ya descritos). En el caso de la tabla 5.2, estos resultados son para el conjunto de testeo 3, que involucra datos de todos los escenarios de daño simulado.

Tabla 5.1: Comparativa de los mejores resultados de cada modelo (caso estudio 1). Se incluye clasificación de dos clases como referencia.

Modelo	***Accuracy***	***Precision***	***Recall***	***F1 Score***
TC CNN	1	1	1	1
OC CNN	0.89	1	0.80	0.89
VAE	0.98	0.99	0.97	0.98
VAE + OCSVM	0.97	0.95	1	0.97

Tabla 5.2: Comparativa de los mejores resultados de cada modelo (caso estudio 2). Se incluye clasificación de dos clases como referencia.

Modelo	***Accuracy***	***Precision***	***Recall***	***F1 Score***
TC CNN	0.92	0.95	0.88	0.91
OC CNN	0.52	0.56	0.17	0.25
VAE	0.81	0.80	0.83	0.81
VAE + OCSVM	0.75	0.71	0.73	0.72

De la tabla 5.1 se puede destacar que ningún algoritmo logró igualar estrictamente las métricas obtenidas de la clasificación de dos clases (clasificación en la que ambas clases quedan bien definidas para el clasificador). Sin embargo, los resultados obtenidos por la VAE son suficientemente cercanos como para definir este modelo como el más efectivo de las redes al hacer *novelty detection*. La naturaleza del autoencoder variacional permite entrenar clasificadores de una clase suficientemente exactos para resolver el problema.

Lo mismo ocurre en el caso del data set puente. Según la tabla 5.2, la VAE es el algoritmo más exacto para novelty detection en base a los datos disponibles.

Capítulo 6

Conclusiones

En este trabajo de investigación se presentó la metodología, desarrollo, aplicación y evaluación de 3 algoritmos distintos de detección de novedades, que toman como base las lecturas del espectro de frecuencias del sistema. Como se buscaba detectar novedades, los algoritmos fueron desarrollados y entrenados bajo el supuesto de que estas novedades son indeterminadas. El primer algoritmo consistió en una Red Neuronal Convolucional (CNN), con la que se desarrollaron clasificadores de una y dos clases. El segundo algoritmo consistió en un Autoencoder Variacional (VAE) para clasificar en torno al error de reconstrucción de los datos. Finalmente, el último algoritmo se desarrolló en base a las representaciones latentes generadas por un Autoencoder Variacional, que fueron clasificadas con una Máquina de Soporte Vectorial de una clase (OCSVM).

Los algoritmos se probaron con conjuntos de datos obtenidos mediante acelerómetros, de una barra sometida a la vibración de un shaker y un puente atirantado que se encuentra instrumentado en el Estado de Nueva Gales del Sur, Australia.

Los datos obtenidos fueron procesados con base en la teoría de análisis espectral, aplicando herramientas como la transformada de Fourier para obtener el espectro de frecuencias de las respuestas temporales registradas por los sensores.

Se determina que las redes desarrolladas son clasificadores viables para la detección de fallas, especialmente en el caso de un experimento controlado como en el caso de la barra. En el caso de estudio del puente atirantado, solo el autoencoder variacional y la técnica que lo combina con máquinas de soporte vectorial demuestran oportunidades de mejora, basada en una mejor elección del valor umbral o threshold, hacia métodos que sean aplicables a casos reales, apoyados por las métricas de confusión resultantes de la clasificación. Hasta que estas proyecciones sean estudiadas y logradas, tanto el objetivo general como los objetivos específicos no son logrados de forma satisfactoria.

Bibliografía

[1] H. KHODABANDEHLOU, G. PEKCAN AND M. S. FADALI 2017 *Struct Control Health Monit. 2019;26:e2308.* Vibration-based structural condition assessment using convolution neural networks.
https://doi.org/10.1002/stc.2308

[2] K. WORDEN, G. MANSON AND D. ALLMAN 2003 *Journal of Sound and Vibration* ***259****, 323-343.* Experimental validation of a structural health monitoring methodology: Part I. Novelty detection on a laboratory structure.

[3] A. RYTTER 1993 *Ph.D. Thesis, Department of Building Technology and Structural Engineering, University of Aalborg, Denmark.* Vibration based inspection of civil engineering structures.

[4] W. T. YEUNG AND J. W. SMITH 2004 *Engineering Structures* ***27****, 685-698.* Damage detection in bridges using neural networks for pattern recognition of vibration signatures.

[5] F. MAGALHAES, A. CUNHA AND E. CAETANO 2010 *Mechanical Systems and Signal Processing.* Vibration based structural health monitoring of an arch bridge: From automated OMA to damage detection.
doi:10.1016/j.ymssp.2011.06.011

[6] B. PEETERS, J. MAECK AND G. DE ROECK 2000 *Smart Materials and Structures* ***10****, 518-527.* Vibration-based damage detection engineering: excitation sources and temperature effects.

[7] MINISTERIO DE OBRAS PÚBLICAS, GOBIERNO DE CHILE 2010 *Chile 2020: Obras públicas para el desarrollo, 60.* Plan de Puentes.
http://www.infraestructurapublica.cl/wp-content/uploads/2017/04/MOP_CHILE_2020.pdf

[8] V. MERUANE 2017 *Apuntes para el curso ME4701. Capítulo 4: Análisis Espectral, 87.* Departamento de Ingeniería Mecánica, Universidad de Chile.

[9] https://www.overleaf.com/project/5d573055fa49c719776e348c A. GIRGIS AND F. HAM 1980. *IEEE Transactions on Aerospace and Electronic Systems, AES-16(4), 434–439.* A Quantitative Study of Pitfalls in the FFT.

[10] V. MERUANE 2019. *Clase del Módulo "Adquisición y procesamiento de señales para el diagnostico y pronóstico de fallas".* Diploma en Confiabilidad, Mantenimiento y Gestión de Activos.

[11] K. WORDEN, G. MANSON AND N. R. FIELLER 2000 *Journal of Sound and Vibration **229**, 647–667.* Damage detection using outlier analysis.

[12] C. M. BISHOP 1994 *IEE Proceedings on Vision and Image Signal Processing **141**, 217–222.* Novelty detection and neural network validation.

[13] K. WORDEN 1997 *Journal of Sound and Vibration **201**, 85-101.* Structural fault detection using a novelty measure.

[14] J. AN AND S. CHO 2015 *SNU Data Mining Center, 2015-2 Special Lecture on IE.* Variational Autoencoder based Anomaly Detection using Reconstruction Probability.

[15] W. T. YEUNG AND J. W. SMITH 2005 *Engineering Structures **27**, 685-698.* Damage detection in bridges using neural networks for pattern recognition of vibration signatures.

[16] S. ROBERTS AND L. TARASSENKO 1994 *Neural Computation **6**, 270-84.* A probabilistic resource allocating network for novelty detection.

[17] W. J. STASZEWSKI, S. G. PIERCE, K. WORDEN, W. R. PHILP, G. R. TOMLNISON and B. R. CULSHAW 1997 *Optical Engineering **36**, 1877–1888.* Wavelet signal processing for enhanced Lamb wave defect detection in composite plates using optical fibre detection.

[18] C. MODARRES, N. ASTORGA, E. L. DROGUETT, V. MERUANE 2018 *Struct Control Health Monit. 2018;25:e2230.* Convolutional neural networks for automated damage recognition and damage type identification.
`https://doi.org/10.1002/stc.2230`

[19] P. OZA AND V. M. PATEL 2018 *IEEE Signal Processing Letters **26**, 277-281.* One-Class Convolutional Neural Network
`https://doi.org/10.1109/LSP.2018.2889273`

[20] F. CHOLLET 2017 *Deep Learning with Python.* Manning Publications

[21] G. JANG AND S. CHO 2018 *Annual Conference of the Prognostics and Health Management Society. 2019.* Anomaly Detection of 2.4L Diesel Engine Using One-Class SVM with Variational Autoencoder.

[22] M. M. ALAMDARI, N. DANG, Y. WANG, B. SAMALI AND X. ZHU 2018 *Structural Health Monitoring* ***18****, 35-48.* A multi-way data analysis approach for structural health monitoring of a cable-stayed bridge.