



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

EVALUACIÓN DEL DAÑO ESTRUCTURAL EN UN PUENTE MEDIANTE REDES  
NEURONALES PROFUNDAS CONVOLUCIONALES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MECÁNICO

ORLANDO BASTIÁN CAMPOS BARRAGÁN

PROFESOR GUÍA:  
VIVIANA MERUANE NARANJO

MIEMBROS DE LA COMISIÓN:  
ENRIQUE LÓPEZ DROGUETT  
ALEJANDRO ORTIZ BERNANDIN

SANTIAGO DE CHILE  
2018

**RESUMEN DE LA MEMORIA PARA OPTAR AL  
TITULO DE: INGENIERO CIVIL MECÁNICO  
POR: Orlando Bastián Campos Barragán  
FECHA: 03/09/2018  
PROFESOR GUÍA: Viviana Meruane Naranjo**

## **EVALUACIÓN DEL DAÑO ESTRUCTURAL EN UN PUENTE MEDIANTE REDES NEURONALES PROFUNDAS CONVOLUCIONALES**

Identificar daños en una estructura permite anticiparse frente a fallas de consideración o totales, a fin de poder realizar las mantenciones necesarias. Una técnica ampliamente utilizada para encontrar la ubicación y magnitud de daño en una estructura es mediante el análisis de sus modos de vibración, ya que estos difieren si una estructura tiene o no fallas.

En la literatura se ha logrado identificar la ubicación de daño estructural del puente I-40 utilizando datos recolectados de sus modos de vibración y temperatura, modelando la estructura con elementos finitos y sus fallas como una reducción de rigidez en un elemento seleccionado. Sin embargo, el método utilizado es lento de aplicar pues requiere resolver un problema de optimización mediante un algoritmo de optimización global.

En el presente estudio se utilizan redes neuronales profundas convolucionales (RNPC), las que han demostrado su robustez respecto a otros métodos utilizados actualmente debido a su rapidez de trabajo, la confiabilidad de sus resultados y la facilidad de entrada de los datos, ya que no requieren ser previamente manipulados por el usuario.

Se identifican, localizan y cuantifican los daños estructurales del puente I-40 de Nuevo México utilizando RNPC y los datos de vibración del puente, además, se desarrolla una metodología para representar las vibraciones del puente en imágenes que puedan ser procesadas por una red neuronal profunda convolucional. Finalmente se realiza una validación de la metodología de identificación de daño propuesta, por medio de datos numéricos y experimentales.

Para procesar los datos, crear las imágenes y procesar dichas imágenes en el algoritmo de redes neuronales profundas convolucionales, se utilizará el software MATLAB.

## Tabla de Contenido

|        |   |    |
|--------|---|----|
| 1.     | Introducción .....  | 1  |
| 2.     | Antecedentes y Discusión Bibliográfica .....                | 2  |
| 2.1.   | Antecedentes y Motivación .....                             | 2  |
| 2.1.1. | Objetivo General .....                                      | 2  |
| 2.1.2. | Objetivos Específicos.....                                  | 2  |
| 2.2.   | Redes Neuronales Artificiales.....                          | 3  |
| 2.3.   | Redes Neuronales Convolucionales .....                      | 4  |
| 2.4.   | Redes Neuronales Profundas Convolucionales.....             | 7  |
| 2.5.   | Trabajos Anteriores .....                                   | 7  |
| 2.6.   | Caso de Aplicación .....                                    | 8  |
| 3.     | Metodología .....   | 11 |
| 4.     | Resultados .....  | 12 |
| 4.1.   | Modelo del Puente I-40:.....                                | 12 |
| 4.2.   | Resultados de modelo en ADINA y modos de vibración.....     | 13 |
| 4.3.   | Modelo en MATLAB .....                                      | 15 |
| 4.4.   | Escenario de Daño.....                                      | 18 |
| 4.5.   | Tratamiento de Datos .....                                  | 20 |
| 4.6.   | Creación de Imágenes .....                                  | 22 |
| 4.7.   | Programación de la Red Neuronal Profunda Convolucional..... | 25 |
| 4.8.   | Resultados RNPC .....                                       | 27 |
| 4.9.   | Resultados RNPC utilizando datos modelados.....             | 29 |
| 4.10.  | Resultados RNPC utilizando datos experimentales.....        | 32 |
| 5.     | Discusiones.....  | 37 |
| 6.     | Conclusiones.....   | 39 |
| 7.     | Bibliografía .....  | 40 |
| 8.     | Anexos .....  | 41 |
| 8.1.   | Anexos imágenes.....  | 41 |
| 8.2.   | Anexo códigos .....   | 42 |
| 8.2.1. | Función MSF y Escaladora .....                              | 42 |
| 8.2.2. | Código RNPC para importar datos de modelo .....             | 42 |
| 8.2.3. | Código RNPC para trabajar modelo .....                      | 44 |

## Índice de Tablas

|  |    |
|--|----|
| TABLA 4-1: RESUMEN DE RESULTADOS ENTRE FRECUENCIAS MODELADAS EN ADINA Y FRECUENCIAS TEÓRICAS MODELADAS POR LOS ÁLAMOS NATIONAL LABORATORY.   | 13 |
| TABLA 4-2 MODOS DE VIBRACIÓN OBTENIDOS EN PROGRAMA ADINA.....  | 14 |
| TABLA 4-3 MODOS DE VIBRACIÓN MODELADOS POR LOS ÁLAMOS NATIONAL LABORATORY (IZQUIERDA). MODOS DE VIBRACIÓN MODELADOS EN ADINA (DERECHA).....  | 16 |
| TABLA 4-4: MODOS DE VIBRACIÓN MODELADOS POR LOS ÁLAMOS NATIONAL LABORATORY (IZQUIERDA). MODOS DE VIBRACIÓN MODELADOS EN ADINA (DERECHA)..... | 17 |
| TABLA 4-5: MAC PARA MODELO FINAL Y MODELO DE LOS ALAMOS NATIONAL LABORATORY PARA DISTINTOS NIVELES DE DAÑOS. ....                            | 18 |
| TABLA 4-6: FUNCIONES DIFERENCIA UTILIZADAS EN EL TRATAMIENTO DE DATOS.....   | 20 |
| TABLA 4-7 FUNCIONES NORMALIZACIÓN UTILIZADAS EN EL TRATAMIENTO DE DATOS. .   | 21 |
| TABLA 4-8: TEST REALIZADOS.....  | 21 |
| TABLA 4-9: EJEMPLOS IMÁGENES QUE CONTIENEN DATOS DE MODOS DE VIBRACIÓN PARA DIFERENTES NIVELES DE DAÑO. ....                                 | 24 |

## Índice de Ilustraciones

|  |    |
|--|----|
| IMAGEN 2-1: ESQUEMA DE INTERACCIÓN ENTRE DOS CAPAS DE REDES NEURONALES, SE MUESTRA QUE LA INTERACCIÓN DE LA NEURONA $X_3$ ES CON TODAS LAS NEURONAS DE LA SIGUIENTE CAPA. ....   | 3  |
| IMAGEN 2-2 PROCESOS EN UNA NEURONA ARTIFICIAL .....  | 4  |
| IMAGEN 2-3 : ESQUEMA DE INTERACCIÓN ENTRE DOS CAPAS DE REDES NEURONALES, SE MUESTRA QUE LA INTERACCIÓN DE LA NEURONA $X_3$ ES SOLO CON ALGUNAS NEURONAS DE LA SIGUIENTE CAPA, ESTA SITUACIÓN OCURRE EN LAS REDES NEURONALES CONVOLUCIONALES DADO QUE EL TAMAÑO DEL KERNEL ES MENOR AL INPUT..... | 5  |
| IMAGEN 2-4: UNIDAD NEURONAL ES CAPAZ DE DETECTAR ROTACIONES Y TRASLACIONES EN IMÁGENES GRACIAS AL POOLING. ....  | 6  |
| IMAGEN 2-5: VISTA GENERAL (IZQUIERDA), SECCIÓN TRANSVERSAL DEL PUENTE (DERECHA). ....  | 8  |
| IMAGEN 2-6: VISTA DE ELEVACIÓN DE LA PORCIÓN DEL PUENTE HACIA EL ESTE, A LA DERECHA EL EMPALME A TIERRA. ....  | 8  |
| IMAGEN 2-7: SECCIÓN TRANSVERSAL DEL MODELO DEL PUENTE SIMPLIFICADO. A: PISO DE CONCRETO, B: VIGA DE PLACAS, C,D: LARGUEROS, LAS VIGAS DE PISO NO SE MUESTRAN EN LA IMAGEN. ....  | 9  |
| IMAGEN 2-8: DAÑOS INDUCIDOS EN LAS BANDAS Y BRIDAS DEL PUENTE.....   | 10 |
| IMAGEN 4-1 RESULTADOS MODELO DEL PUENTE I-40 EN EL SOFTWARE ADINA.....   | 13 |
| IMAGEN 4-2: VISTA EN PERSPECTIVA DEL PUENTE I-40, MODELO NUMÉRICO EN MATLAB. ....  | 15 |
| IMAGEN 4-3: VISTA DE PLANTA DE LA SUPERFICIE DE CONCRETO, MODELO NUMÉRICO EN MATLAB. ....  | 15 |
| IMAGEN 4-4: VISTA LATERAL SUR DE LA VIGA DE PLACA, MODELO NUMÉRICO EN MATLAB.....  | 15 |
| IMAGEN 4-5: VISTA LATERAL SUR DE LA VIGA DE PLACA, LA LÍNEA ZIGZAG INDICA LA POSICIÓN APROXIMADA DE LOS DAÑOS INDUCIDOS, MODELO NUMÉRICO EN MATLAB. ....   | 15 |
| IMAGEN 4-6:.....   | 18 |
| IMAGEN 4-7: SET DE DATOS DE MODOS DE VIBRACIÓN, CADA COLUMNA REPRESENTA LA DEFORMACIÓN DEL PUENTE EN SU VIGA NORTE Y SUR. LAS FRECUENCIAS NATURALES CORRESPONDIENTES A CADA UNO DE ESTOS SETS DE DATOS SE ENCUENTRAN EN UNA MATRIZ DIFERENTE. ....   | 19 |
| IMAGEN 4-8: ARCHIVO DE IMAGEN, LOS PÍXELES DE LA IMAGEN CORRESPONDEN A LOS VALORES DE LOS MODOS DE VIBRACIÓN TRANSFORMADOS A LA ESCALA DE GRISES CON VALORES ENTRE 0 Y 255. ....   | 22 |
| IMAGEN 4-9: ARQUITECTURA DE RNPC UTILIZADA. ....   | 26 |

|  |    |
|--|----|
| IMAGEN 4-10: DETALLE DE LOS PASOS EN LA RED NEURONAL .....   | 26 |
| IMAGEN 4-11: PORCENTAJES DE DME Y FAE OBTENIDOS DE LA RNPC.....  | 28 |
| IMAGEN 4-12: PRIMER TEST, DAÑO MODELADO LEVE, AL ELEMENTO 24 SE LE SIMULA UN DAÑO DEL 20%, EL PROGRAMA PREDICE UN DAÑO DEL 11%. LA TEMPERATURA VARIA UN 20% RESPECTO DE LA TEMPERATURA BASE, SIENDO DE -4 °C, EL PROGRAMA PREDICE UNA TEMPERATURA DE -8 °C. ....     | 29 |
| IMAGEN 4-13: SEGUNDO TEST, DAÑO MODELADO MODERADO, AL ELEMENTO 24 SE LE SIMULA UN DAÑO DEL 50%, EL PROGRAMA PREDICE UN DAÑO DEL 47%. LA TEMPERATURA VARIA UN 60% RESPECTO DE LA TEMPERATURA BASE, SIENDO DE 18 °C, EL PROGRAMA PREDICE UNA TEMPERATURA DE 11 °C..... | 30 |
| IMAGEN 4-14: TERCER TEST: DAÑO MODELADO SEVERO, AL ELEMENTO 24 SE LE SIMULA UN DAÑO DEL 99%, EL PROGRAMA PREDICE UN DAÑO DEL 97%, LA TEMPERATURA VARIA UN 80% RESPECTO DE LA TEMPERATURA BASE, SIENDO DE 29 °C, EL PROGRAMA PREDICE UNA TEMPERATURA DE 32 °C.....    | 31 |
| IMAGEN 4-15: PRIMERA COMPROBACIÓN, UTILIZANDO UN CASO DE SIN DAÑO EXPERIMENTAL LA RED NEURONAL NO ENCUENTRA DAÑOS, SE PREDICE UNA TEMPERATURA DE -14 °C. ....  | 32 |
| IMAGEN 4-16: SEGUNDA COMPROBACIÓN, UTILIZANDO UN CASO DE BAJO DAÑO EXPERIMENTAL EN EL ELEMENTO 24, SE ENCUENTRAN VARIOS DAÑOS, ADEMÁS DEL ELEMENTO 24, LA TEMPERATURA PREDICHA ES 15 °C.....   | 33 |
| IMAGEN 4-17: TERCERA COMPROBACIÓN, UTILIZANDO UN CASO DE DAÑO MODERADO EXPERIMENTAL EN EL ELEMENTO 24. LA RED RECONOCE DAÑO EN EL ELEMENTO 24 DE UN 30%, LA TEMPERATURA PREDICHA ES 0 °C.....  | 34 |
| IMAGEN 4-18: CUARTA COMPROBACIÓN, UTILIZANDO UN CASO DE DAÑO EXPERIMENTAL MODERADO EN EL ELEMENTO 24. LA RED DETECTA DIVERSOS DAÑOS EN EL PUENTE, LA TEMPERATURA PREDICHA ES 5 °C.....   | 35 |
| IMAGEN 4-19 : QUINTA COMPROBACIÓN, UTILIZANDO UN DAÑO EXPERIMENTAL SEVERO EN EL ELEMENTO 24. LA RED DETECTA UN DAÑO SEVERO EN EL ELEMENTO 24, LA TEMPERATURA PREDICHA ES 8°C. ...  | 36 |
| IMAGEN 8-1: ESQUEMA BÁSICO DEL TRABAJO CON REDES NEURONALES ARTIFICIALES.....  | 41 |

# 1. Introducción

La evaluación de daños permite detectar y caracterizar el deterioro estructural en las primeras etapas para estimar cuanto tiempo falta antes de que un mantenimiento sea necesario, dicha estimación tiene gran potencial en la longevidad y seguridad de la estructura, además de entregar beneficios económicos al reducir los mantenimientos y aumentando la confiabilidad de esta. La evaluación del daño tiene gran importancia para el área industrial, militar y también civil, puede ser aplicado a obras civiles como edificaciones, puentes, vehículos de transporte como aeronaves, helicópteros, trenes, barcos y autos, como también para aplicaciones industriales como molinos, turbinas, bombas entre otros.

Uno de los problemas principales en la evaluación de daños es determinar la presencia, ubicación y severidad de daño estructural utilizando las características dinámicas de una estructura. Actualmente se han desarrollado una serie de aplicaciones que permiten lograr este objetivo, dentro de los cuales destacan utilizar parámetros modales como los modos de vibración de la estructura y las frecuencias de resonancia en algoritmos genéticos, modelos de máxima entropía y redes neuronales artificiales, con resultados excelentes en cuanto a la cuantificación y ubicación del daño estructural, algunos de los cuales entregan resultados en tiempo real <sup>[4][5][6]</sup>.

En este trabajo se utilizan redes neuronales profundas convolucionales (RNPC), estas son una variación de las redes neuronales artificiales, algoritmo el cual tiene la capacidad de aprender pasando información a través de diferentes capas de procesamiento. Se llaman redes neuronales convolucionales porque la información que se transfiere entre capas neuronales debe pasar necesariamente por una función matemática convolución (que incidirá en una mejora de rendimiento), las redes neuronales convolucionales pueden trabajar con un amplio espectro de problemas, dentro del que destaca el análisis de imágenes, las cuales se pueden asumir como matrices en 2D, la imagen es manipulada y se puede resaltar de ella información que el usuario considere relevante, de modo que las redes neuronales detectarán patrones y los aprenderán. Se llaman redes neuronales *profundas* convolucionales a aquellas redes neuronales, que además de realizar una convolución en el traspaso de información entre cada capa de red neuronal, esta se subdivide en pequeños paquetes de información permitiendo que ciertos sectores de cada capa trabajen en distintas fases de un problema con respecto a otros sectores de las capas neuronales, dividiendo de esta forma la carga de trabajo, además de tener gran cantidad de capas de procesamiento de información. La principal ventaja de las RNPC es la robustez y rapidez de evaluación del algoritmo con respecto a los algoritmos genéticos, las RNPC, luego de ser entrenadas, pueden entregar resultados en pocos segundos, mientras que los algoritmos genéticos pueden tomar horas o días.

El objetivo de esta investigación es la evaluación de daño estructural utilizando las vibraciones del puente I-40 de Nuevo México con un algoritmo de redes neuronales profundas convolucionales. Para alcanzar este objetivo principal y a modo de objetivos específicos, será necesario transformar las vibraciones del puente en una imagen que pueda procesar los RNPC, paralelo a esto se debe modificar un algoritmo existente de RNPC para que pueda procesar las imágenes creadas y aprender de ellas, finalmente se debe automatizar los dos programas a modo de cargar los datos del puente I-40 y que los resultados sean entregados automáticamente.

## 2. Antecedentes y Discusión Bibliográfica

### 2.1. Antecedentes y Motivación

Identificar daños en una estructura permite anticiparse frente a fallas de consideración o totales, a fin de poder realizar las mantenciones necesarias. Es posible encontrar la ubicación y magnitud de daño en una estructura mediante el análisis de sus modos de vibración, ya que estos difieren si una estructura tiene o no fallas. En la literatura se ha logrado identificar la ubicación de daño estructural del puente I-40 utilizando datos recolectados de sus modos de vibración y temperatura, modelando la estructura mediante elementos finitos y sus fallas como una reducción de rigidez en un elemento seleccionado. Sin embargo, el método utilizado es lento de aplicar pues requiere resolver un problema de optimización mediante un algoritmo de optimización global. En el presente estudio se utilizan redes neuronales profundas convolucionales quienes han demostrado su robustez respecto a otros métodos utilizados actualmente debido a su rapidez de trabajo, la confiabilidad de sus resultados y la facilidad de entrada de los datos, ya que no requieren ser previamente manipulados por el usuario.

#### 2.1.1. Objetivo General

- Identificar, localizar y cuantificar los daños estructurales del puente I-40 de Nuevo México utilizando redes neuronales profundas convolucionales.

#### 2.1.2. Objetivos Específicos

- Generar un modelo representativo del puente I-40 de Nuevo México.
- Desarrollar una metodología para representar las vibraciones del puente en imágenes que puedan ser procesadas por una red neuronal profunda convolucional.
- Implementar un algoritmo basado en redes neuronales profundas convoluciones (RNPC) para la identificación de daño en el puente.
- Validación de la metodología de identificación de daño propuesta por medio de datos numéricos y experimentales.



## 2.2. Redes Neuronales Artificiales

“Redes neuronales son conjuntos de elementos de cálculos simples, usualmente adaptativos, interconectados masivamente en paralelo y con una organización jerárquica que le permite interactuar con un sistema del mismo modo que lo hace el sistema nervioso biológico<sup>[1]</sup>. Su aprendizaje adaptativo, autoorganización, tolerancia a fallos, operación en tiempo real y fácil inserción dentro de la tecnología existente, han hecho que su utilización se haya extendido en áreas como la biología, financiera, industrial, medio ambiental, militar, salud, ingenieril. Funcionan en aplicaciones que incluyen identificación de procesos, detección de fallas, modelación de dinámicas no lineales, control de sistemas no lineales y optimización de procesos.”<sup>[2]</sup>

Las redes neuronales son un modelo computacional basado en un gran conjunto de unidades neuronales simples artificiales, cada unidad neuronal trabaja de forma independiente y está conectada a varias otras, la entrada de información a una neurona se conoce como el *input layer* y la salida de información como *output layer*, las neuronas se agrupan en capas de procesos de información, donde diferentes capas de la red neuronal interactúan entre ellas incrementando o inhibiendo el estado de activación de neuronas adyacentes lo que permite obtener un resultado a una tarea a resolver. Las redes neuronales son sistemas que aprenden a resolver problemas y se forman a sí mismos en lugar de ser programados.

Usualmente las redes neuronales básicas se dividen en tres capas (pueden ser aún más), la primera capa es la entrada o *input* la segunda capa procesa los datos la cual es llamada capa escondida y la tercera capa es la salida u *output*. En cada capa se realizan diferentes procesos matemáticos, en redes neuronales usualmente se trabaja con multiplicación de matrices, lo que implica que a conectividad de un *output* de una neurona es con todos los *inputs* de neuronas de la siguiente capa tal como se muestra en la

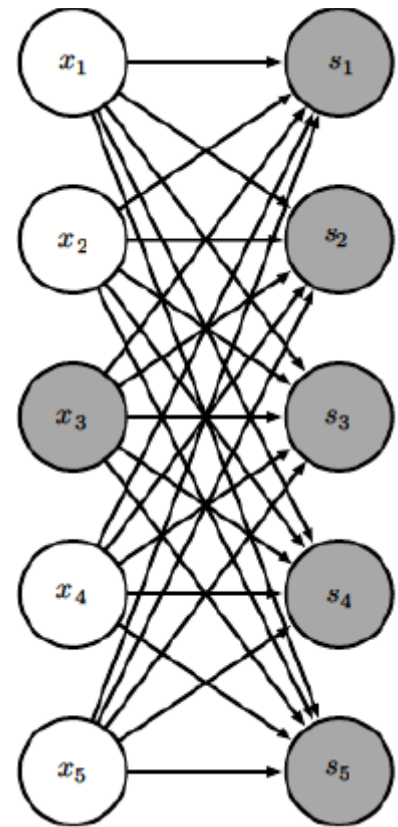


Imagen 2-1: Esquema de interacción entre dos capas de redes neuronales, se muestra que la interacción de la neurona  $x_3$  es con todas las neuronas de la siguiente capa.

Previo a utilizar una red neuronal artificial se debe desarrollar una arquitectura neuronal y seleccionar los patrones de aprendizaje los cuales permitirán al algoritmo comenzar a *aprender*, seguido de lo anterior es necesario fabricar test para comprobar la confiabilidad del algoritmo en su resolución del problema, si los test fallan se debe

iterar y volver a comenzar desarrollando una nueva arquitectura neuronal y/o seleccionando nuevos patrones de aprendizaje. Realizadas las fases nombradas el algoritmo está preparado para trabajar, la *Imagen 8-1* del Anexo muestra en forma de diagrama de flujo lo mencionado.

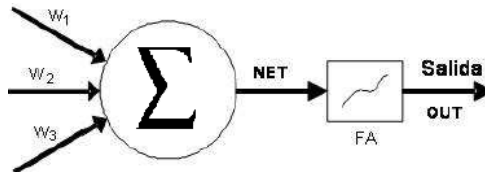
En una red neuronal existe un peso entre las redes que es un valor numérico que pondera las señales que se reciben en la entrada, el peso será la fuerza de conexión entre dos neuronas, cuando se evalúa una neurona se debe calcular el conjunto de todos los pesos que se reciben por sus entradas, término denominado NET. Cuando se diseña una red neuronal debe establecerse como van a ser los valores de activación de cada neurona por lo que debe definirse una función de activación FA con la que las neuronas procesarán las entradas y determinarán el valor del estado interno de la neurona para transmitirlo a la salida mediante una función transferencia. La función de activación y la función de transferencia son las encargadas de definir el nuevo estado de activación  $A_i$  y respuesta  $O_i$  de la neurona respectivamente, la *Imagen 2-2* es un resumen de los pasos nombrados.

La combinación de señales que recibe una neurona se calcula como:

$$NET_i(t) = \sum_{j=1}^{N-1} [W_{ij} \cdot O_j \cdot (t - 1)] \quad (2-1)$$

Donde  $W_{ij}$  representa el peso de la conexión entre una neurona emisora  $j$  y una neurona receptora  $i$ . El estado de activación se define como:

$$A_i = FA(A_i(t - 1), NET_i(t - 1)) \quad (2-2)$$



*Imagen 2-2 Procesos en una neurona artificial*

### 2.3. Redes Neuronales Convolucionales

Las redes convolucionales o redes neuronales convolucionales (CNN por sus siglas en inglés) son un tipo especializado de red neuronal artificial para el procesamiento de información que tiene una topología de cuadrículas, por ejemplo, las imágenes, que pueden ser trabajadas como una matriz de 2D de píxeles. El nombre de red neuronal convolucional indica que la red emplea la operación matemática convolución en vez de utilizar multiplicación de matrices en al menos una de sus capas. La ventaja principal de las Redes Neuronales Convolucionales con respecto a las Redes Neuronales Artificiales es que,

gracias a la operación de convolución, la red neuronal es capaz de obtener las características de los datos de información de manera automática, mientras que en las RNA estas características deben ser indicadas (programadas), lo anterior genera que un RNC sea más robusta e intuitiva frente a cambios en la información, a diferencia de una RNA.

En general las redes neuronales convolucionales van a estar construidas con una estructura que contendrá tres tipos distintos de capas, una capa convolucional, una capa de reducción o *pooling* que reduce la cantidad de parámetros al quedarse con las características más comunes y una capa clasificadora totalmente conectada, la que nos dará el resultado final de la red.

La convolución de forma general se define como:

$$s(t) = \int x(a)w(t - a)da \quad (2-3)$$

Donde la función  $x$  corresponde al *input* y la función  $w$  corresponde al *kernel*. Dado que en computación se trabaja con valores discretos, la convolución debe ser discretizada, quedando de la forma:

$$s(t) = \sum_{-\infty}^{\infty} x(a)w(t - a) \quad (2-4)$$

Sin embargo, dado que trabajamos con imágenes, la convolución debe ser definida para dos dimensiones, correspondientes a dos ejes.

$$S(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \quad (2-5)$$

Donde  $I$  corresponde a nuestro *input*, el cual sería la imagen con sus dos dimensiones y  $K$  corresponde al *kernel* que, por la naturaleza del *input*, también debe ser de dos dimensiones.

Las redes neuronales tradicionales usan multiplicaciones de matrices mediante una matriz de parámetros con un parámetro separado que indica la interacción entre cada unidad de entrada y cada unidad de salida. Lo anterior implica que cada unidad de salida interactúa con cada unidad de entrada. Las redes neuronales convolucionales tienen típicamente una interacción dispersa, esto se logra haciendo el *kernel* más pequeño que el *input* (en este caso la imagen). Cuando se procesa una imagen, se tienen millones de datos correspondientes a los píxeles, dentro de esos datos es posible detectar

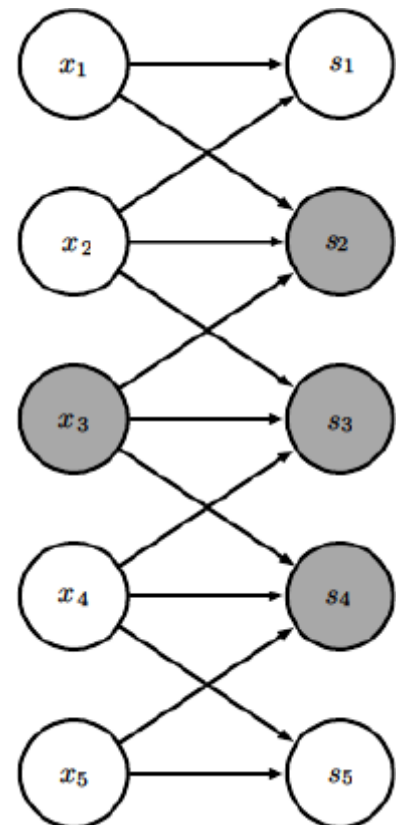


Imagen 2-3 : Esquema de interacción entre dos capas de redes neuronales, se muestra que la interacción de la neurona  $x_3$  es solo con algunas neuronas de la siguiente capa, esta situación ocurre en las redes neuronales convolucionales dado que el tamaño del *kernel* es menor al *input*.

características significativas como los bordes (por ejemplo, detectar la silueta de una persona) utilizando un *kernel* que usarían solo cientos de píxeles, esto reduce los requerimientos de memoria y hace las operaciones notablemente más rápido, en otras palabras los *outputs* de cada unidad no necesariamente interactuaron con todos los *inputs* de las unidades anteriores como se muestra en la

Imagen 2-3. Es importante remarcar que la multiplicación de matrices requiere un tiempo  $O(m \times n)$  para ser procesado, mientras que, si se utiliza un *kernel* más pequeño, se limitan el número de conexiones entre *inputs* y *outputs* requiriendo  $k \times n$  operaciones, es decir, un tiempo  $O(k \times n)$ , estas características del *kernel* nos permiten además trabajar con distintos tamaños de imágenes, propiedad que puede ser conveniente al trabajar con imágenes.

La capa de reducción o *pooling* es colocada generalmente luego de la capa convolucional. Es utilizada para reducir las dimensiones de la imagen de la entrada para la siguiente capa convolucional, por lo que existe una pérdida de información en beneficio de una menor sobrecarga de cálculos, la propiedad que nos entrega el *pooling* a grandes rasgos va a ser el poder identificar patrones en las imágenes aun cuando estos estén corridos, o rotados como se muestra en la

Imagen 2-4, por ejemplo, un algoritmo que detecte números en una imagen, si los números están corridos de la zona de búsqueda o están rotados, estos serán detectados gracias al *pooling*. Luego la capa clasificadora enviará la información a diferentes neuronas.

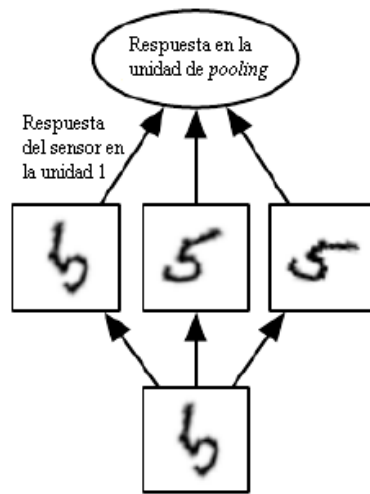


Imagen 2-4: Unidad neuronal es capaz de detectar rotaciones y traslaciones en imágenes gracias al *pooling*.

Consideremos un *kernel*  $K$  con los elementos  $K_{i,j,k,l}$  (*kernel* de 4D) que da la fuerza de conexión entre una unidad en el canal  $i$  de la salida u *output* y una unidad en el canal  $j$  de la entrada o *input*, con un desplazamiento de  $k$  filas y  $l$  columnas entre el la unidad de *output* y la unidad de *input*. Asumiendo que el input consiste en datos observados  $V$  con los elementos  $V_{i,j,k}$ . Considerando el *output* como  $Z$  con el mismo formato que  $V$ , si  $Z$  es producido mediante la convolución  $K$  a lo largo de  $V$ , entonces una red neuronal convolucional puede ser definida como:

$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n} \quad (2-6)$$

## 2.4. Redes Neuronales Profundas Convolucionales

Las **redes neuronales profundas o aprendizaje profundo (*Deep Learning*)**, se diferencia de las redes neuronales tradicionales por la cantidad de capas ocultas que la red tiene, las redes neuronales tradicionales contienen entre dos a tres capas ocultas, mientras que las redes profundas pueden llegar a tener hasta 150. La adición de nuevas capas permite a la red obtener más características para la clasificación de los datos. <sup>[3][8]</sup>

Las redes neuronales profundas convolucionales son uno de los tipos más populares de redes neuronales profundas, dada su capacidad de trabajar con imágenes y la capacidad automática de extracción de características, estas características no se entrenan previamente y se aprenden directamente de la red cuando esta se entrena con una colección de imágenes. Esta característica de extracción automatizada hace los modelos de aprendizaje profundo sean muy precisos para tareas de visión artificial, como la clasificación de objetos o patrones en ellos.

## 2.5. Trabajos Anteriores

Uno de los problemas principales en la evaluación de daños es determinar la presencia, ubicación y severidad de daño estructural utilizando las características dinámicas de una estructura. Diversos estudios se han realizado sobre el puente I-40 a fin de lograr predecir el daño en su estructura.

*Los Álamos National Laboratory* realizó una investigación en el puente I-40, enfocada en el estudio de las variaciones en los modos de vibración al inducir daño y la predicción de estas variaciones mediante un modelo. El documento *Finite Element Analysis of the I-40 Over the Rio Grande*, el cual detalla los pasos para realizar el modelo del puente y la toma de muestras de los modos de vibración, indica que solo los daños considerables en la estructura generan cambios apreciables en los modos de vibración, siendo estos detectables, a diferencia de los daños pequeños.

E. Parloo, P. Guillaume y M. Van Overmeire, estudiaron un método de localización y evaluación de daño sin utilizar un modelo de elementos finitos, basado en la sensibilidad que tienen la forma de los modos de vibración a los cambios en la masa o la rigidez de una estructura modelo. Con la sensibilidad, las diferencias en el comportamiento dinámico de la estructura en sus condiciones dañadas y sin daños puede traducirse en información de daño (ubicación y cantidad de cambios en masa o rigidez). Los resultados mostraron esta técnica es capaz de detectar daño y cuantificarlo. <sup>[9]</sup>

Scott W. Doebling y Charles R. Farrar realizaron una investigación para la identificación de daños basados en las vibraciones de los datos de tomados en el puente I-40, se utilizó una prueba significación estadística a las estimaciones de intervalo medio y de confianza de las propiedades modales y los indicadores de daños correspondientes,

utilizando como indicador de daño el cambio medido en la matriz de flexibilidad. Los resultados indicaron que el daño es detectable en todos los casos de daño, sin embargo, no es capaz de localizar el daño para el caso de máximo daño en la estructura. [10]

Meruane y Heylen realizan una investigación enfocada en la detección, localización y cuantificación de daño estructural, utilizando algoritmos genéticos, logrando satisfacer los puntos anteriores, pero utilizando un largo tiempo de procesamiento. [4]

## 2.6. Caso de Aplicación

El caso de aplicación es el puente de la interestatal 40 (I-40) sobre el Río Grande en Albuquerque, Nuevo México, referencias reales pueden ser vistas en la Imagen 2-5. Se estudia una sección del puente y no su totalidad, esta sección contempla el empalme del puente a tierra y tres pilares sobre el río, con una extensión total aproximada de 130 metros, como se muestra en la Imagen 2-6.



Imagen 2-5: Vista general (izquierda), sección transversal del puente (derecha).

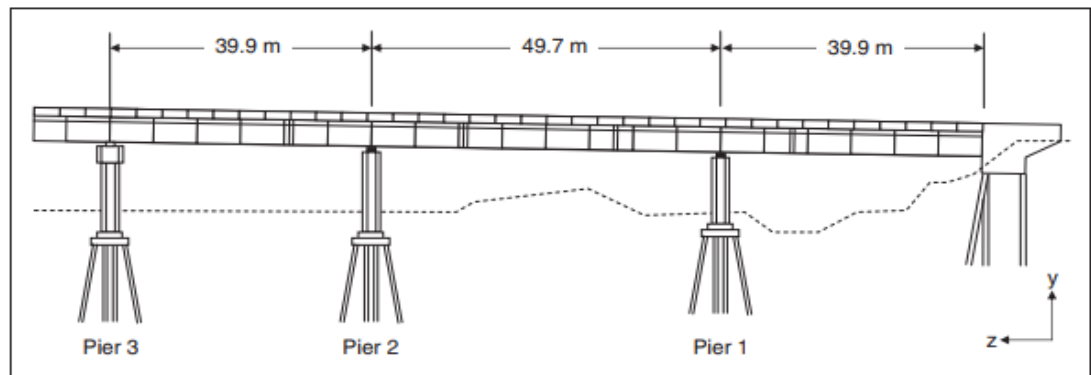


Imagen 2-6: Vista de elevación de la porción del puente hacia el este, a la derecha el empalme a tierra.

El puente i-40 se modela basándose en el modelo utilizado en *Finite Element Analysis of the I-40 Over the Rio Grande*, el documento hace una simplificación de la estructura manteniendo el comportamiento físico, el modelo simplificado se muestra en la Imagen 2-7. La totalidad de la estructura se reduce a cuatro componentes;

A- Plancha de Concreto (Concrete Slab): Corresponde a la plancha de concreto superior del puente, donde circulan los autos.

B- Vigas de Placas (Plate Girders): Vigas de acero de gran magnitud, laterales al puente, son la conexión intermediaria entre la placa de concreto superior y los pilares de concreto del puente.

C, D- Largueros (Stringers): Vigas de acero entre la plancha de concreto y la viga de piso, transfieren los esfuerzos de la plancha de piso. Son de menor tamaño que las vigas de placas.

E- Vigas de Piso. Vigas ubicadas bajo los largueros de forma perpendicular a estos, transfieren los esfuerzos de estos a las vigas de placas.

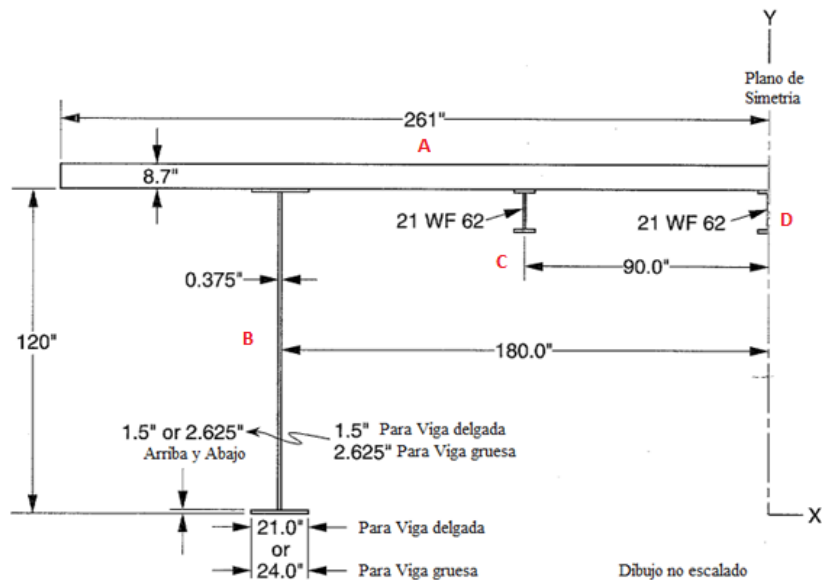
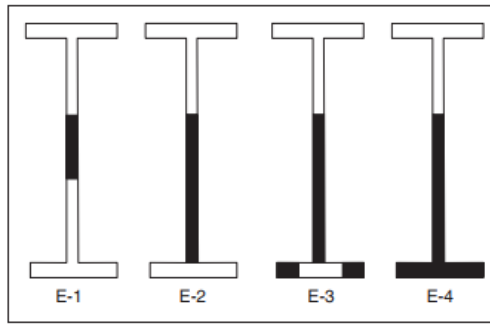


Imagen 2-7: Sección transversal del modelo del puente simplificado. A: Piso de Concreto, B: Viga de Placas, C,D: Largueros, las Vigas de Piso no se muestran en la imagen.

Imagen 2-7: Sección transversal del modelo del puente simplificado. A: Piso de Concreto, B: Viga de Placas, C,D: Largueros, las Vigas de Piso no se muestran en la imagen. El puente se hizo vibrar con un agitador, los datos fueron recolectados por acelerómetros piezoeléctricos dispuestos a lo largo del puente cuando este se hacía vibrar forzosamente, obteniendo la respuesta en frecuencia y los modos de vibración en diferentes horas del día. Se realizaron cuatro daños de diferentes magnitudes en el centro de la placa de vigas norte. Las fallas inducidas al puente intentan simular la falla por fatiga que ha sido observada en los puentes de vigas de placa. Los diferentes niveles de daño fueron introducidos haciendo cortes en la banda y en la brida de la viga, los cuales se muestran en la Imagen 2-8. El primer daño consiste en un corte de 10 mm de ancho y 0,61 m de largo, el segundo se extiende hasta la base de la banda, en el tercer daño la brida se corta a la mitad en ambos lados y en el cuarto daño se corta totalmente la brida. Tras cada corte la estructura es sujeta a un análisis modal.



*Imagen 2-8: Daños inducidos en las bandas y bridas del puente.*



### 3. Metodología

La metodología que se siguió para el desarrollo de este trabajo fue la siguiente:

- Generar un modelo del puente I-40 mediante el software de elementos finitos ADINA, luego, obtener los modos de vibración y frecuencias naturales respectivas a cada modo, los cuales, como objetivo, sea similar a los obtenidos por el modelo realizado por Los Álamos National Laboratory).
- Ajustar el modelo a los datos experimentales utilizando el software de ajuste de modelos FEMTools. Se ajusta la respuesta en frecuencia, los modos de vibración y las frecuencias naturales asociadas a cada modo de vibración.
- Exportar los resultados desde FEMTools a MATLAB, en este último, realizar un algoritmo que simule escenarios de daño del puente modelado y genere datos de respuesta en frecuencia, modos de vibración y frecuencias naturales, para luego convertirlos en imágenes para entrenar la red neuronal.
- Generar un algoritmo mediante Python para entrenar la red neuronal utilizando las imágenes del modelo, convertir los datos experimentales del puente I-40 en imágenes para corroborar si el algoritmo predice los daños en el puente.
- Se entrenarán las RNPC para que detecten patrones en las imágenes creadas a partir de los datos de vibración, los patrones encontrados permitirán encontrar, localizar y cuantificar daños en el puente I-40. Los resultados (teóricos) serán validados con los datos experimentales disponibles comentados en la sección Caso de aplicación.

## 4. Resultados

### 4.1. Modelo del Puente I-40:

Para inicios de la investigación se realiza un modelo del puente mediante elementos finitos utilizando el software ADINA. La modelación tiene como objetivo simular, de forma general, la estructura del puente I-40 de Nuevo México, por lo que elementos como remaches y pernos así también como elementos de difícil modelación no fueron considerados. El documento *Finite Element Analysis of I-40 Bridge Over the Rio Grande* <sup>[7]</sup> contiene una sección en la página 35, ver Imagen 2-7 donde se modela el puente simplificado, considerando la placa de concreto y las placas de vigas laterales como elementos planos de dos dimensiones y las demás estructuras como elementos *beam*. Las consideraciones del documento son:

- Camino de concreto del puente:
  - Modelado como un elemento *2D Shape*.
  - Material de concreto:
    - Módulo de Young: 3,6000,000 psi.
    - Ratio de Poisson: 0.2.
    - Densidad: 145 lbm/ft<sup>3</sup>
- Vigas de Placas (Plate Girders):
  - Modelado como un elemento *2D Shape*.
  - Material de acero:
    - Módulo de Young: 29,000,000 psi.
    - Ratio de Poisson: 0.3.
    - Densidad: 0.284 lbm/in<sup>3</sup>
- Largueros (Stringers):
  - Modelado como un elemento *beam*.
  - Material de acero.
  - Área transversal de viga: 21 WF 62
- Vigas de piso (Floor beams):
  - Modelado como un elemento *beam*.
  - Material de acero.
  - Área transversal de viga: 21 WF 62
- Pilares de concreto (Piers):
  - Modelado como un elemento *beam*.
  - Material de Concreto.
  - Área circular no descrita en documento, se itera en distintos modelos.

Se enfatiza que el puente simplificado fue calculado con áreas y largos efectivos que permitiesen que este se comportase similar al puente real.

## 4.2. Resultados de modelo en ADINA y modos de vibración

Finalizado el modelo en el software ADINA (ver Imagen 4-1), se procede a obtener los primeros seis modos de vibración con sus frecuencias asociadas, para esto es necesario restringir el movimiento del puente en la coordenada X y luego permitir a ADINA hacer el estudio de los modos de vibración. Se itera varias veces el área circular de los pilares y la finesa del mallado de la estructura para obtener los modos de vibración y frecuencias naturales similares a los obtenidos por el modelo de *Los Álamos National Laboratory* presentes en la Tabla 4-1.

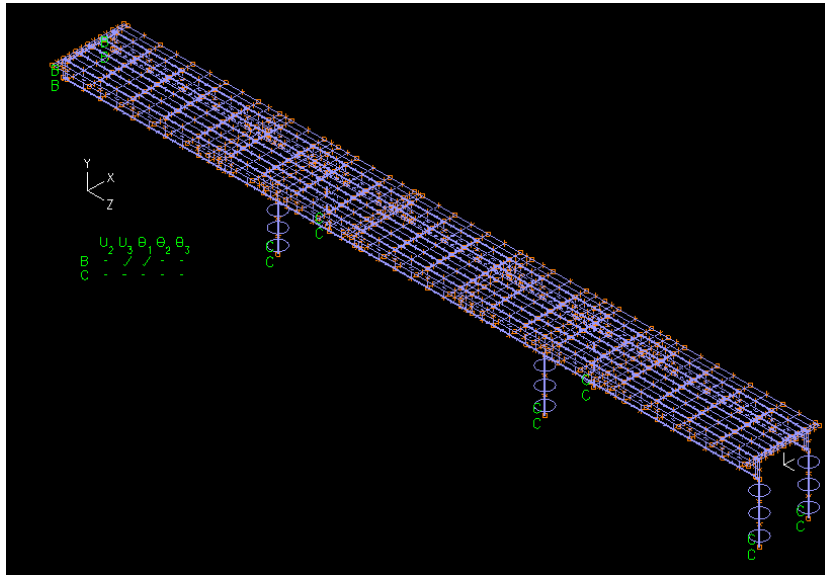


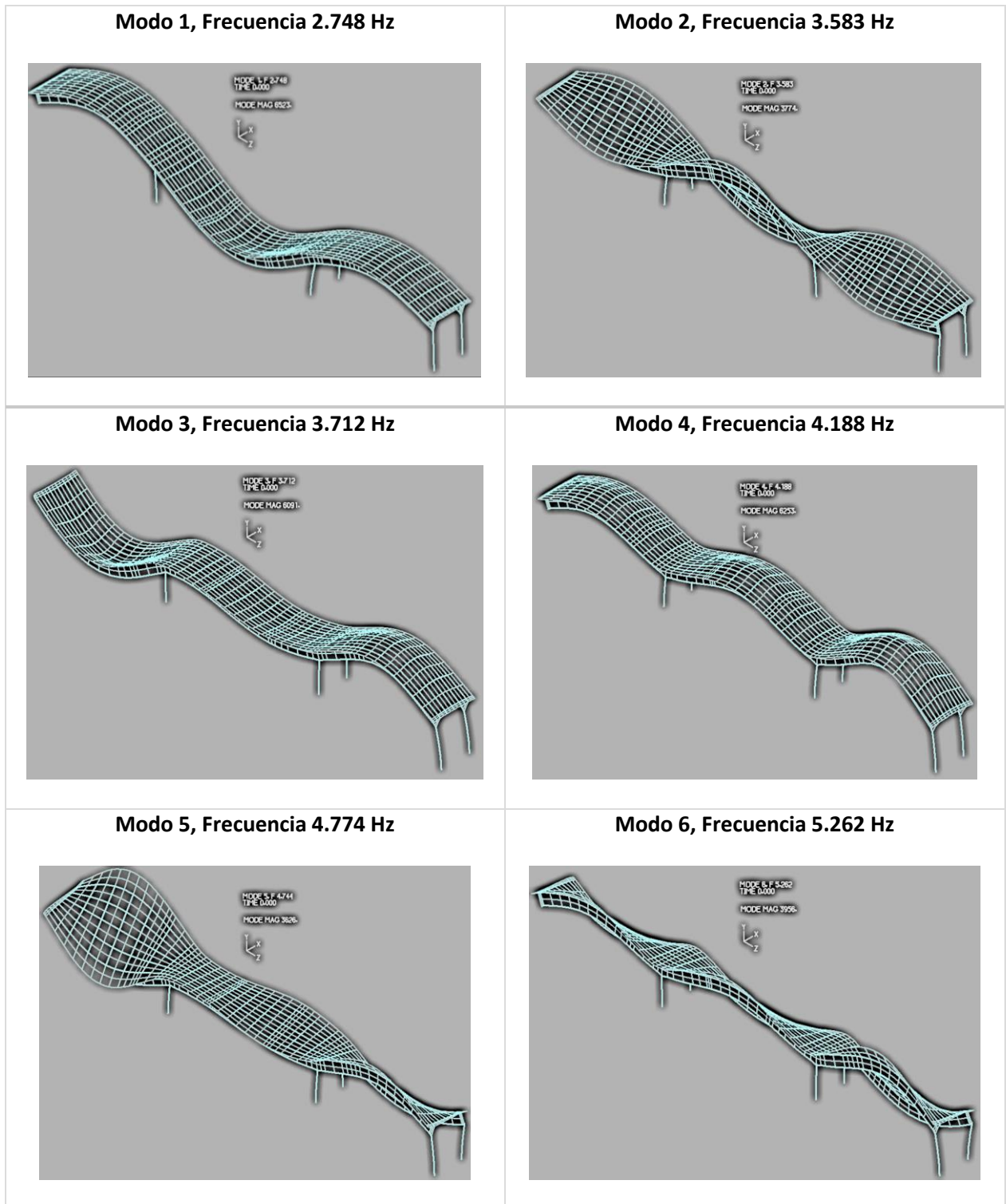
Imagen 4-1 Resultados modelo del puente I-40 en el software ADINA.

Tabla 4-1: Resumen de resultados entre frecuencias modeladas en ADINA y frecuencias teóricas modeladas por Los Álamos National Laboratory.

| Modo | Frecuencia Teórica Modelada (Hz) | Frecuencia Teórica de Referencia (Hz) | Error (%) |
|------|----------------------------------|---------------------------------------|-----------|
| 1    | 2.73                             | 2.82                                  | -3.19     |
| 2    | 3.58                             | 3.77                                  | -5.04     |
| 3    | 3.71                             | 3.81                                  | -2.62     |
| 4    | 4.18                             | 3.88                                  | +7.73     |
| 5    | 4.74                             | 5.11                                  | -7.24     |
| 6    | 5.26                             | 5.23                                  | +0.57     |

Los modos de vibración obtenidos, con su correspondiente frecuencia son los siguientes:

Tabla 4-2 Modos de vibración obtenidos en programa ADINA.



### 4.3. Modelo en MATLAB

Se continua la investigación con el modelo que contiene las frecuencias naturales similares al modelo de *Los Álamos* descritos en la Tabla 4-1. El modelo de ADINA se exporta a formato *.NASTRAN* para ser trabajado en el software FEMTools, con el fin de ajustar el modelo, para luego ser exportado a MATLAB, esta última exportación presenta problemas referidos al *toolkit* de MATLAB, por lo que se decide realizar el modelo numérico en este último programa, a fin de evitar problemas de exportaciones.

El puente es vuelto a modelar, esta vez, de forma más simple, sin añadir los largueros y las vigas de piso, tiene 48 elementos en total repartidos en las vigas de placas y 144 elementos en la superficie de concreto. El puente modelado se muestra en la Imagen 4-2, Imagen 4-3, Imagen 4-4, Imagen 4-5. Los modos de vibración se muestran en la Tabla 4-3 y Tabla 4-4.

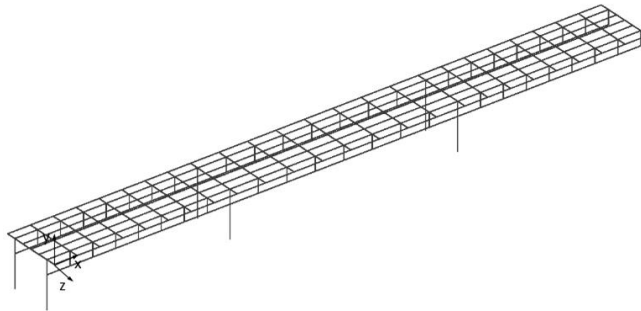


Imagen 4-2: Vista en perspectiva del puente I-40, modelo numérico en MATLAB.

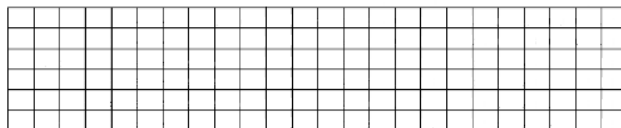


Imagen 4-3: Vista de planta de la superficie de concreto, modelo numérico en MATLAB.

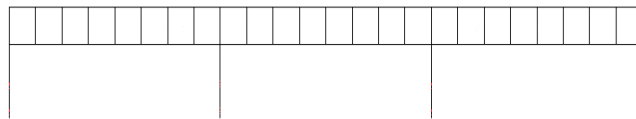


Imagen 4-4: Vista lateral sur de la viga de placa, modelo numérico en MATLAB.

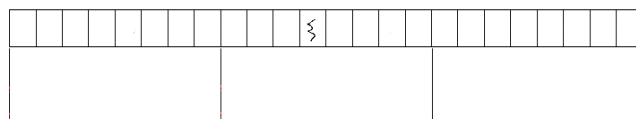


Imagen 4-5: Vista lateral sur de la viga de placa, la línea zigzag indica la posición aproximada de los daños inducidos, modelo numérico en MATLAB.

Tabla 4-3 Modos de Vibración modelados por Los Álamos National Laboratory (izquierda). Modos de Vibración modelados en ADINA (derecha).

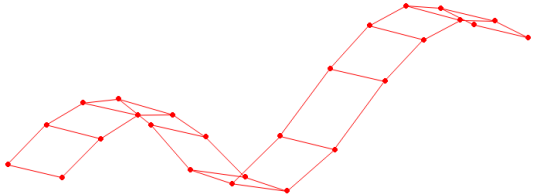
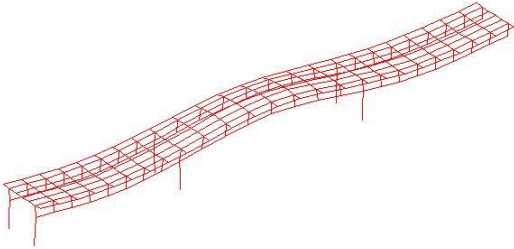
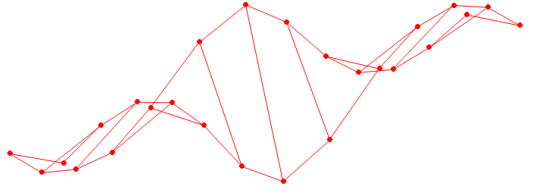
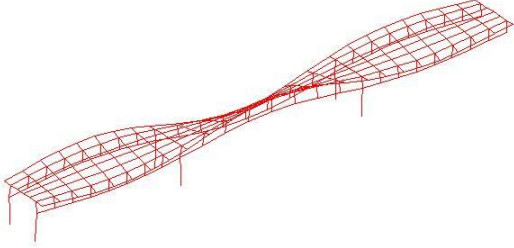
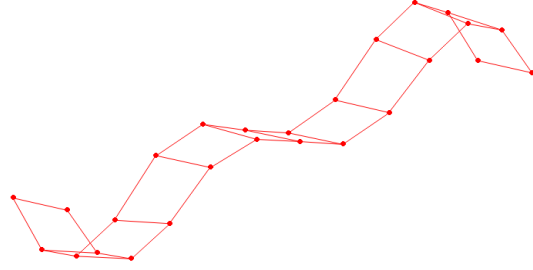
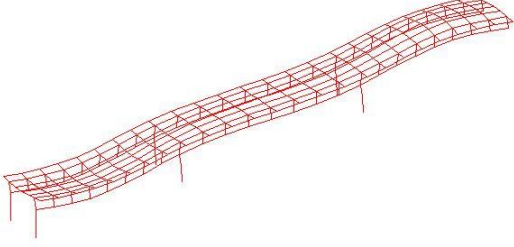
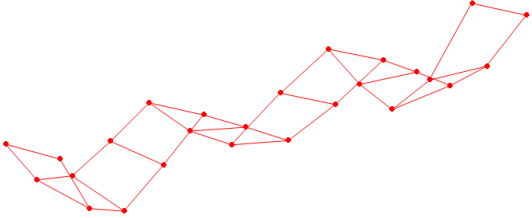
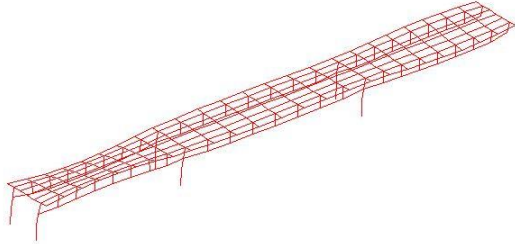
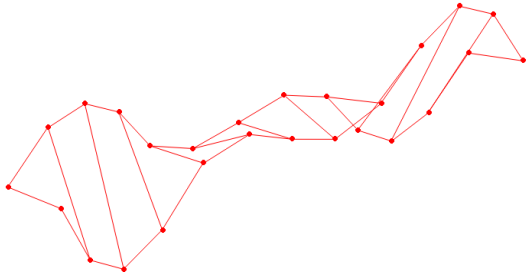
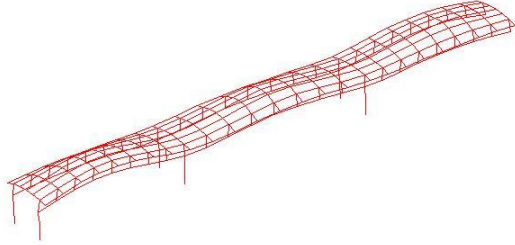
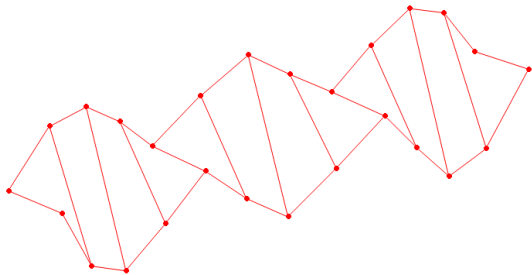
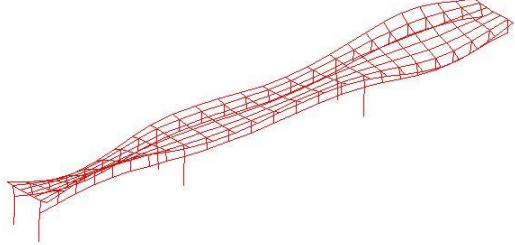
| Modelo utilizado por <i>Los Alamos National Laboratory</i> sobre el puente I-40 importado en FEMTools.  | Modelo numérico utilizado en la investigación actual, realizado en MATLAB.  |
|---|---|
| <p data-bbox="290 512 849 541">Modo de vibración modelado Los Álamos N° 1, 2.59 Hz:</p>      | <p data-bbox="889 512 1458 541">Modo de vibración modelado en MATLAB N° 1, 2.44 Hz:</p> <p data-bbox="1101 569 1247 598"><math>\omega = 2.44 \text{ Hz}</math></p>        |
| <p data-bbox="290 938 849 968">Modo de vibración modelado Los Álamos N° 2, 2.78 Hz:</p>    | <p data-bbox="889 938 1458 968">Modo de vibración modelado en MATLAB N° 2, 2.96 Hz:</p> <p data-bbox="1101 1016 1247 1045"><math>\omega = 2.96 \text{ Hz}</math></p>    |
| <p data-bbox="290 1386 849 1415">Modo de vibración modelado Los Álamos N° 3, 3.71 Hz:</p>  | <p data-bbox="889 1386 1458 1415">Modo de vibración modelado en MATLAB N° 3, 3.42 Hz:</p> <p data-bbox="1101 1476 1247 1505"><math>\omega = 3.42 \text{ Hz}</math></p>  |

Tabla 4-4: Modos de Vibración modelados por Los Álamos National Laboratory (izquierda). Modos de Vibración modelados en ADINA (derecha).

| Modelo utilizado por <i>Los Alamos National Laboratory</i> sobre el puente I-40 importado en FEMTools.  | Modelo numérico utilizado en la investigación actual, realizado en MATLAB.  |
|---|---|
| <p data-bbox="318 443 867 474">Modo de vibración modelado Los Álamos N° 4, 4.32 Hz:</p>      | <p data-bbox="899 443 1456 474">Modo de vibración modelado en MATLAB N° 4, 3.58 Hz:</p> <p data-bbox="1105 533 1252 564"><math>\omega = 3.58</math> Hz</p>        |
| <p data-bbox="318 894 867 926">Modo de vibración modelado Los Álamos N° 5, 3.96 Hz:</p>     | <p data-bbox="899 894 1456 926">Modo de vibración modelado en MATLAB N° 5, 4.01 Hz:</p> <p data-bbox="1105 989 1252 1020"><math>\omega = 4.01</math> Hz</p>     |
| <p data-bbox="318 1419 867 1451">Modo de vibración modelado Los Álamos N° 6, 4.50 Hz:</p>  | <p data-bbox="899 1419 1456 1451">Modo de vibración modelado en MATLAB N° 6, 4.04 Hz:</p> <p data-bbox="1097 1514 1243 1545"><math>\omega = 4.04</math> Hz</p>  |

Los MAC entre el modelo final y el modelo de *Los Alamos National Laboratory* para distintos niveles de daño se muestran en la Tabla 4-5.

Tabla 4-5: MAC para modelo final y Modelo de Los Alamos National Laboratory para distintos niveles de daños.

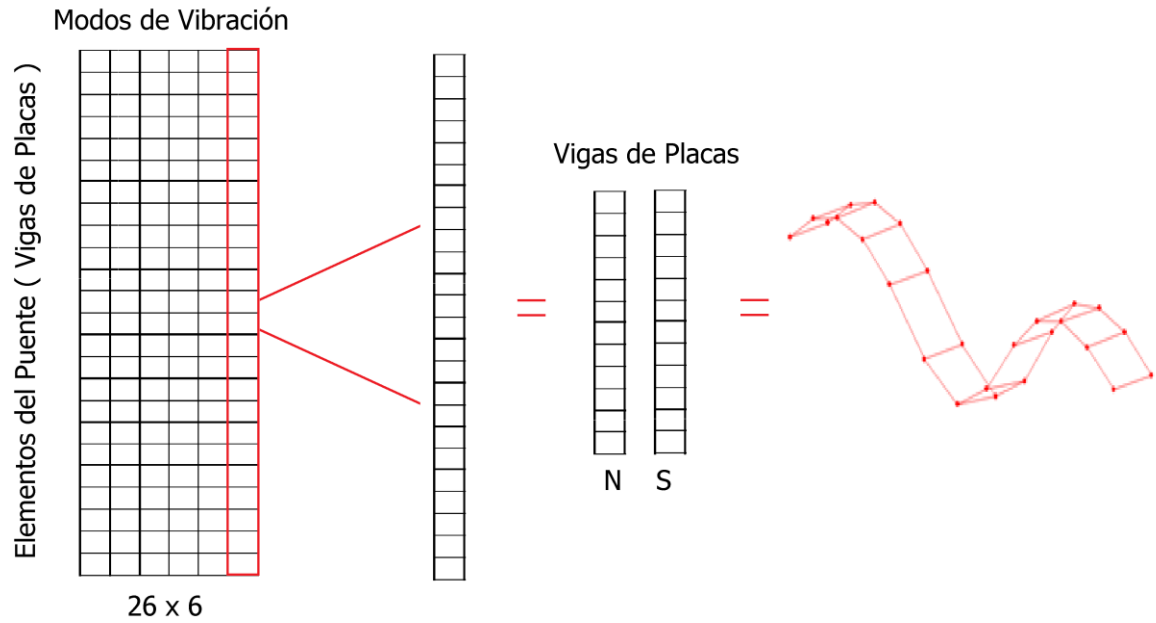
|            | <b>Daño</b> | <b>Modo 1</b> | <b>Modo 2</b> | <b>Modo 3</b> | <b>Modo 4</b> | <b>Modo 5</b> | <b>Modo 6</b> |
|------------|-------------|---------------|---------------|---------------|---------------|---------------|---------------|
| <b>MAC</b> | 0 ( 0 %)    | 0.88          | 0.91          | 0.99          | 0.96          | 0.96          | 0.96          |
|            | 1 (20%)     | 0.89          | 0.92          | 0.99          | 0.92          | 0.95          | 0.97          |
|            | 2 (40%)     | 0.99          | 0.98          | 0.99          | 0.86          | 0.93          | 0.97          |
|            | 3 (60 %)    | 0.99          | 0.97          | 0.99          | 0.85          | 0.93          | 0.97          |
|            | 4 (95 %)    | 0.92          | 0.89          | 0.99          | 0.82          | 0.91          | 0.96          |

#### 4.4. Escenario de Daño

El puente modelado cuenta con 48 elementos en las vigas de placas laterales, 24 elementos en la placa norte y 24 elementos en la placa sur, más un último elemento ficticio que simula la temperatura ambiental. El siguiente paso es crear diferentes escenarios de daño, disminuyendo la rigidez de cada elemento de las vigas de placa individualmente, además de ir variando la temperatura ambiental. El daño simulado varía de 0% a 100% en pasos de 10%, mientras que la variación de temperatura varía de 0% a 100% en pasos de 20%. Los escenarios de daños fueron calculados por un código vía MATLAB siendo 11 tipos de daños diferentes para cada elemento y 6 temperaturas diferentes para cada tipo de daño en cada elemento. Los resultados se guardan como los modos de vibración y frecuencias naturales respectivos, obteniendo un total de 3168 set de datos, para los modos de vibración y 3168 set de datos para las frecuencias naturales correspondientes a cada modo de vibración. Cada set de datos de los modos de vibración contiene una matriz de 26 x 6, donde cada columna representa a que frecuencia natural corresponde, siendo en total seis frecuencias, y las filas representan la posición que tiene el puente al deformarse, 13 para la zona norte y 13 para la zona sur, lo anterior puede entenderse como que cada set de dato, si se grafica por columnas, mostrará una imagen del puente con su respectiva deformación asociada a su modo de vibración, ver Imagen 4-7. El set de datos de las frecuencias naturales corresponde a una matriz de 6 x 1, donde cada fila corresponde al valor de la frecuencia natural de cada modo de vibración.



Debido a que las RNPC necesitan gran cantidad de datos, se decide agregar redundancia de información mediante la adición de ruido de  $\pm 5\%$  a los modos de vibración y a las frecuencias correspondientes a cada modo, se calcularon 33 ruidos de datos a los 3168 resultados, de esta manera se logró generar 107.712 set de datos, una cantidad suficiente para entrenar un RNPC.



*Imagen 4-7: Set de datos de modos de vibración, cada columna representa la deformación del puente en su viga norte y sur. Las frecuencias naturales correspondientes a cada uno de estos sets de datos se encuentran en una matriz diferente.*

## 4.5. Tratamiento de Datos

Para procesar los datos teóricos en la red neuronal, se tratan los datos previamente, aplicando una diferencia de todos los datos con un dato en común, en este caso, se utiliza un dato que no tenga daño en ningún de los 48 elementos del *Plate Girder* ni tampoco variación de temperatura. Esto se hace tanto para las matrices de modos como para las matrices de las frecuencias naturales para los 107.712 set de datos. Las funciones utilizadas para generar las diferencias se muestran en la Tabla 4-6.

La diferencia de datos se realiza debido a que el modelo numérico no coincide exactamente con los datos experimentales, cuando se entrenan las redes neuronales con los datos numéricos y luego, los resultados se prueban con los datos experimentales, la red interpreta las diferencias como daños, resultando en una falsa detección de daños. Por otro lado, aunque el modelo numérico no representa de forma exacta al modelo experimental, la variación en los modos y frecuencias naturales debido al daño si se parecen más, es por esto que se decide ocupar como entrada a la red la variación de los modos y frecuencias naturales respecto al caso sin daño.

Tabla 4-6: Funciones Diferencia utilizadas en el tratamiento de datos.

| Tipo  | Función  |
|---|--|
| A-Diferencias cuadráticas para modos de vibración | $(Valor - Valor\_Constante)^2$                                     |
| B-Diferencias absolutas para modos de vibración   | $ Valor - Valor\_Constante $                                       |
| Diferencias cuadráticas para frecuencias          | $\left(\frac{Valor\_Constante - Valor}{Valor\_Constante}\right)^2$ |

Sumado al cálculo de la diferencia, también se normalizan los sets de datos, ya que las redes neuronales trabajan de forma más eficiente con valores entre -1 y 1, trabajar con valores no normalizados puede generar error en las soluciones entregadas por la RNPC, más aún si las escalas entre dichos datos están espaciadas. Para hacer la normalización de datos se aplican las funciones de la Tabla 4-7 tanto para los sets de datos de los modos de vibración como las frecuencias naturales asociadas.

Tabla 4-7 Funciones Normalización utilizadas en el tratamiento de datos.

| Tipo   | Función   |
|--|---|
| A-Normalización con desviación estándar y media.           | $\frac{\text{valor\_matriz} - \text{Media\_Matriz}}{\text{Desviacion\_Estandar\_Matriz}}$   |
| B-Normalización unitaria (se obtienen valores entre 0 y 1) | $\frac{\text{Valor}_{\text{matriz}} - \text{Valor}_{\text{Minimo}}}{\text{Valor}_{\text{Maximo}} - \text{Valor}_{\text{Minimo}}}$ |

Para las redes neuronales se hacen diversas pruebas, estas contemplan usar set de datos:

Tabla 4-8: Test realizados

| Prueba | Tipo de test                            | Resultados  |
|--------|---|---|
| 1      | Sin diferencias y sin Normalización     | Negativos   |
| 2      | Con diferencias A y con Normalización A | Positivos para modelo, Negativos para datos experimentales. |

|   |   |   |
|---|---|---|
| 3 | Con diferencias A y con Normalización B | Positivos para modelos y experimental.                      |
| 4 | Con diferencias B y con Normalización A | Positivos para modelo, Negativos para datos experimentales. |
| 5 | Con diferencias B y con Normalización B | Positivos para modelos y experimental.                      |

#### 4.6. Creación de Imágenes

Las RNPC trabajan con imágenes, por lo que es necesario convertir los modos de vibración y sus frecuencias naturales asociadas, sin embargo, se opta por sólo convertir los modos de vibración en imágenes y, la información de las frecuencias, ingresarlas directamente al código de procesamiento. Este último paso se puede realizar dado que una RNPC, al leer una imagen en escala de grises, lee un número entre el 0 al 255, correspondiente a la escala de grises, por lo que el proceso de convertir datos en imágenes es en cierta manera, en vano, dado que la RNPC convertirá las imágenes nuevamente en datos, es por esto que es posible trabajar directamente con los datos en las redes neuronales.

Se opta por crear archivos híbridos que tuvieran información en imágenes como información directa, para esto se creó un archivo de imagen para cada set de dato de los modos de vibración utilizando la función *mat2gray* de Matlab, función encargada de tomar los valores de una matriz y convertirla a valores entre 0 y 255 para crear una imagen. los archivos tienen el siguiente patrón:

Nombre de archivo de imagen: Frec1-Frec2-Frec3-Frec4-Frec5-Frec6-\_Elemento-DañoElemento\_Elemento49-DañoElemento49-Temperatura.png

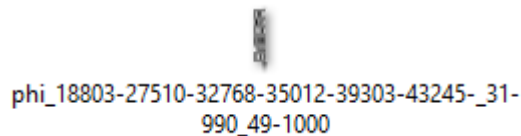


Imagen 4-8: Archivo de Imagen, los píxeles de la imagen corresponden a los valores de los modos de vibración transformados a la escala de grises con valores entre 0 y 255.

La Imagen 4-8 muestra un ejemplo de archivo de imagen, los primeros 6 datos del nombre corresponden a las *Frec.* (Frecuencias naturales), dentro de este archivo está guardado una imagen de 26 x 6 píxeles en escala de grises, la cual contiene toda la información correspondiente a los modos de vibración para cada elemento, una imagen más amplia del archivo puede ser vista en la Tabla 4-9.

Luego de ser creadas las imágenes, estas son diferenciadas entre las que corresponden a daño simulado y las que no tienen daño. Se obtuvieron 9793 imágenes sin daño y 97921 imágenes con daño. Creadas las imágenes de las simulaciones de daño de puente, es posible comenzar a entrenar las RNPC y evaluar el reconocimiento de imágenes, aunque no es posible evaluar la red neuronal con los datos experimentales, dado que no se han creado aun las imágenes de dichos datos.

Al analizar las frecuencias naturales y los modos de vibración experimentales, estos son similares a los obtenidos en el modelo, pero los datos de los modos de vibración están en diferente escala, por lo que es necesario escalarlos, específicamente al nivel de los modos del modelo, para esto se utiliza el *Modal Scale Factor* (MSF), el factor de escala entre pares de modos y la función *escaladora*, esta función utiliza la diagonal del resultado de la matriz MSF, las magnitudes para escalar la matriz de modos de vibración experimentales al nivel del modelo, dicha diagonal tiene seis valores, cada uno correspondiente a cada modo de vibración.

Una vez escalado los modos experimentales para cada caso de daño, se procede a calcular las diferencias de los modos de vibración y frecuencias de cada daño con respecto al caso experimental sin daño. Luego se normaliza utilizando las funciones normalizadoras calculadas de los modos y frecuencias modelados (ver Tabla 4-7). Al igual que los datos teóricos se prueban distintas formas el tratamiento de datos, se generan imágenes de modos de vibración y frecuencias naturales sin ningún tratamiento hasta con dos tipos de diferencias y normalizaciones.

El siguiente paso es utilizar la función *mat2gray* de MATLAB para crear las imágenes. La función *mat2gray* sólo puede convertir a imagen datos que sean mayores o iguales a cero, en ciertos casos, al intentar convertir los modos de vibración no tratados o con normalización, se tiene gran cantidad de datos negativos que, si no son tratados, se convertirían en valor equivalente a cero, por lo que se decide encontrar el valor mínimo de los casos mencionados y trasladar todos los valores de los set de datos como el valor absoluto del mínimo más el valor de los datos, esto es:

$$NV = V + |\min\{SD\}| \quad (4-1)$$

Con:

NV: Nuevo valor de dato de modo de vibración.

V: Valor de dato de modo de vibración (negativo o positivo).

SD: Set de datos de modo de vibración (total 3168x34 set de datos).

La ecuación ( 4-1 ) hace que cada set de datos de la matriz de modos de vibración, de 3168 x 34 datos, quede con el valor trasladado y mayor o igual a cero. Las frecuencias naturales también deben ser tratadas antes de convertirlas en los nombres de los archivos, dado que los valores de las frecuencias rondan entre valores de 0 y 3 Hz con números decimales. Al pasar el valor de la frecuencia con numero decimal a nombre de archivo, se puede crear confusión en la extensión de dicho archivo, como ejemplo con una frecuencia

natural, si se tiene una frecuencia de valor 2.42 Hz, al pasar esta a un archivo quedaría con nombre de foto 2.42.png, pudiendo los programas entender que la extensión del archivo es 42.png y no .png como es requerido. Para modificar lo anterior se opta por amplificar los valores de las frecuencias por un valor 10E3 o 10E6, según corresponda el caso, si existen frecuencias negativas (por el hecho de haber aplicado una normalización o una diferencia), se procede con los mismos pasos que los modos de vibración, las frecuencias naturales son trasladadas para que tengan valores mayores o iguales que cero. Los pasos anteriores se resumen en la fórmula ( 4-2 ). Finalizado esto, la creación de imágenes es exitosa, ver Tabla 4-9.

$$NF = (F + |\min\{SDF\}|) * Amplificador \quad (4-2)$$

Con:



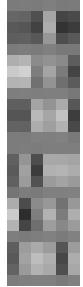


NF: Nuevo valor de dato frecuencia.

F: Valor de dato frecuencia.

SDF: Set de datos de frecuencias (total 3168x34 set de datos).

Amplificador: N° variable entre 10E3 y 10E6.

Tabla 4-9: Ejemplos imágenes que contienen datos de modos de vibración para diferentes niveles de daño.

| <b>Imágenes creadas a partir de modos de vibración experimentales</b>               |   |   |   |   |
|---|---|---|---|---|
| Sin daño<br>(D0)  | Daño inicial<br>(D1)  | Daño inicial<br>(D2)  | Daño medio<br>(D3)  | Daño extenso<br>(D4)  |
|  |  |  |  |  |

Al momento de programar la red neuronal se observa que, al importar las imágenes creadas, existe un leve error en la precisión de los datos originales con respecto a los datos

importados, un error aleatorio, esto ocurre en el proceso de trasladar los datos a imágenes y luego, estas imágenes, convertirlas en datos en la red neuronal, por lo que se decide que el paso medio de creación de imágenes no será utilizado, dado que el proceso agrega aún más ruido a los datos originales. Los datos son transportados directamente de MATLAB a la red neuronal utilizando la función de Python *scipy.io*. Si fuera el caso de necesitar utilizar más ruido en los datos, el proceso de convertir los datos en imágenes puede ser utilizado.

Para facilitar la lectura de los datos en Python, se crean cuatro matrices a ser importadas, la primera matriz contiene los modos de vibración ordenados en filas de todos los casos modelados con daño, la segunda matriz con los casos sin daños, la tercera matriz contiene todas las frecuencias naturales en filas correspondientes a los modos sin daño, la cuarta matriz las frecuencias naturales correspondientes a los modos con daños, la quinta y sexta matriz son matrices de  $N \times 4$  (con  $N$  cantidad de casos con daño o sin daño, 97921 y 9793 correspondientemente), las columnas corresponden al número del elemento con daño, el daño del elemento, el elemento 49, la temperatura del elemento 49.

#### 4.7. Programación de la Red Neuronal Profunda Convolutiva

Completada la fase de modelamiento, tratamiento de datos y exportación de datos a Python, se inicia la programación de la red neuronal profunda convolutiva, para esto se utiliza el software Python emulado bajo la plataforma de Anaconda y con una interfaz de Spyder. Para facilitar la programación de la red neuronal se utilizó el módulo de TensorFlow, el cual cuenta con librerías para desarrollar y trabajar una red neuronal. Los programas se ejecutan vía GPU utilizando una NVIDIA GTX 1080 para acelerar el cálculo de resultados.

Arquitectura de la RNPC:

- Se definen las entradas (inputs) de la red neuronal los datos correspondientes a los modos de vibración dispuestos en columnas.
- Se define la salida como un vector fila de 49 elementos, la cual indica la posición y magnitud del daño, además de su temperatura.
- Se inicializa la primera convolución:
  - Se utiliza un filtro de  $6 \times 1$  (se toma cada modo de vibración).
  - Se obtienen 256 Mapas de Características (*256 Feature Maps*).
  - Se realiza un *DropOut* al final de la convolución (opcional).
- Se inicializa la segunda convolución:
  - Se utiliza un filtro de  $2 \times 1$ .
  - Se obtienen 512 Mapas de Características (*512 Feature Maps*).
  - Se realiza un *DropOut* al final de la convolución (opcional).

- Se inicializa la *Fully Connected Layer* con 4096 unidades ocultas (neuronas).
  - Se realiza un *Flatten* de la información de las convoluciones dejando la información en forma de columnas, se concatena verticalmente a estas columnas la información de las frecuencias naturales.
  - Se ingresa la información a una *Fully Connected Layer*, utilizando una función de activación *RELU*.
  - Se utiliza un *DropOut* entre las neuronas.
  - Se utiliza un optimizador *ADAM*.
- El programa retorna la posición y cantidad de daño del elemento además de su temperatura. Además, retorna los parámetros *DME (Damage Missing Error)* y *FAE (False Alarm Error)*, el primero corresponde a una cuantificación de la falla del programa en encontrar daño donde había y el segundo corresponde a la falla del programa prediciendo daño donde no existía.

La Imagen 4-9 un resumen de la arquitectura utilizada.

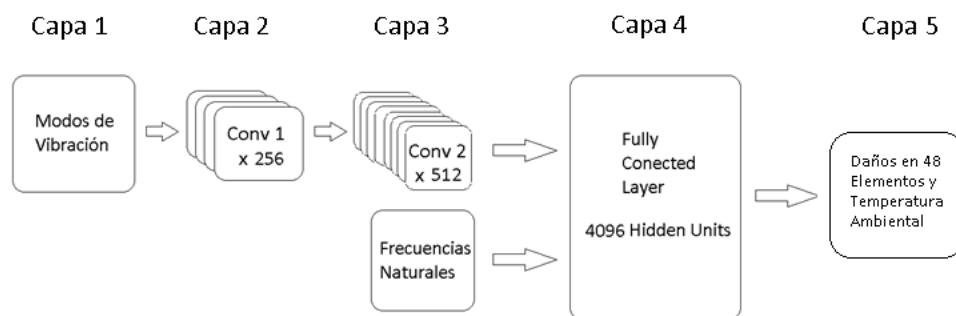


Imagen 4-9: Arquitectura de RNN utilizada.

La Imagen 4-10 muestra la arquitectura con mayor detalle, según los pasos nombrados anteriormente.

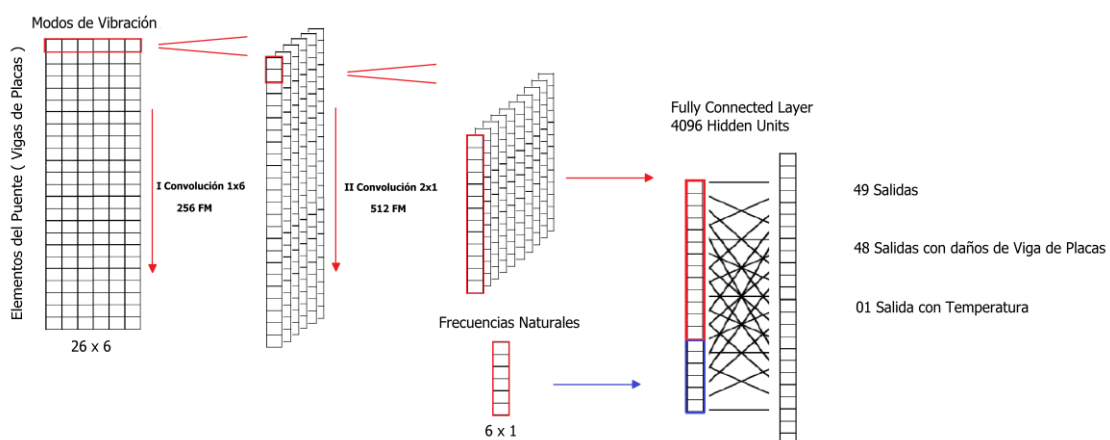


Imagen 4-10: Detalle de los pasos en la red neuronal



## 4.8. Resultados RNPC

La performance de una red neuronal es medida mediante tres indicadores, el Error de Tamaño Medio (MSE por sus siglas en inglés), Error de Falsa Alarma (FAE por sus siglas en inglés) y Error de Daño Perdido (DME por sus siglas en inglés).

El MSE es un error promedio de cuantificación, se calcula como:

$$MSE = \frac{1}{NO} \sum_i |y_i - o_i|$$

Donde  $y_i$  y  $o_i$  son las salidas estimadas y deseadas para el nodo  $i$ ,  $NO$  es el número de nodos.

El Damage Missing Error (DME) está dado por:

$$DME = \frac{1}{NT} \sum_i \epsilon_i^I, \quad 0 \leq DME \leq 1$$

Donde  $\epsilon_i^I = 0$  si el  $i$ -ésimo elemento dañado es correctamente detectado y  $\epsilon_i^I = 1$  si no.  $NT$  corresponde al número de verdaderas zonas dañadas. Si  $DME = 0$ , quiere decir que todas las zonas de daño fueron correctamente detectadas.

Se asume que un elemento es detectado como daño si el daño estimado,  $y_i$ , es mayor que un daño crítico  $\alpha_c$ . El nivel de daño crítico,  $\alpha_c$ , se define equivalente al promedio de MSE, este es el daño mínimo que la red puede evaluar.

El False Alarm Error (DME) está dado por:

$$FAE = \frac{1}{NF} \sum_i \epsilon_i^{II}, \quad 0 \leq FAE \leq 1$$

Donde  $\epsilon_i^{II} = 0$  si el  $i$ -ésimo elemento dañado está realmente dañado y  $\epsilon_i^{II} = 1$  si no.  $NF$  corresponde al número zonas de daño predichos. Si  $FAE = 0$ , quiere decir que todas las zonas de daño son efectivamente zonas de daño.

Los resultados DME (Damage Missing Error) el cual indica el porcentaje de daños que no fueron detectados en cierto rango y FAE (False Alarm Error), el cual indica, para cierto rango, el porcentaje de daño que se detectó, sin realmente haber existido dicho daño, son mostrados en la Imagen 4-11.

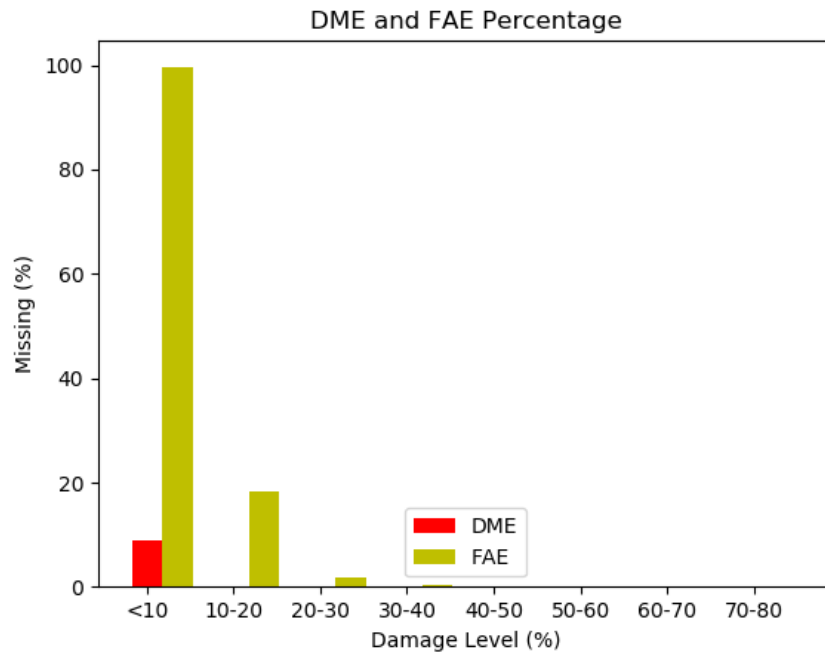
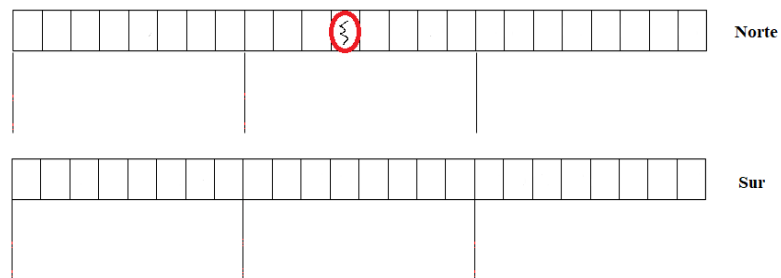
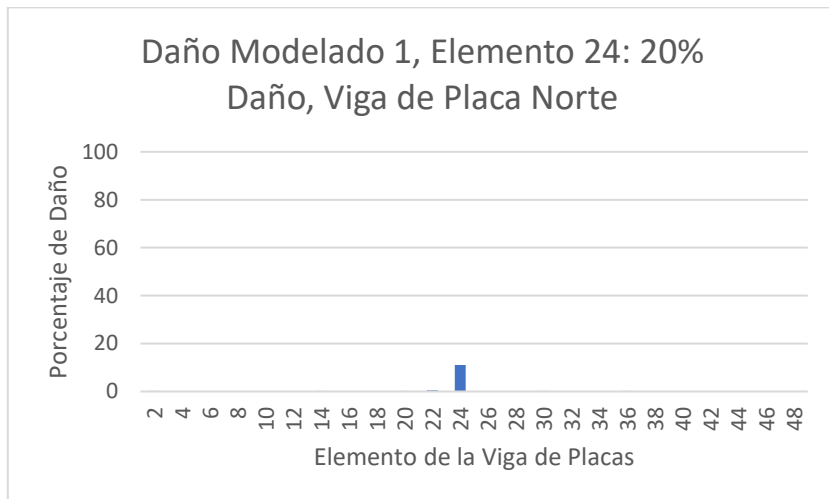
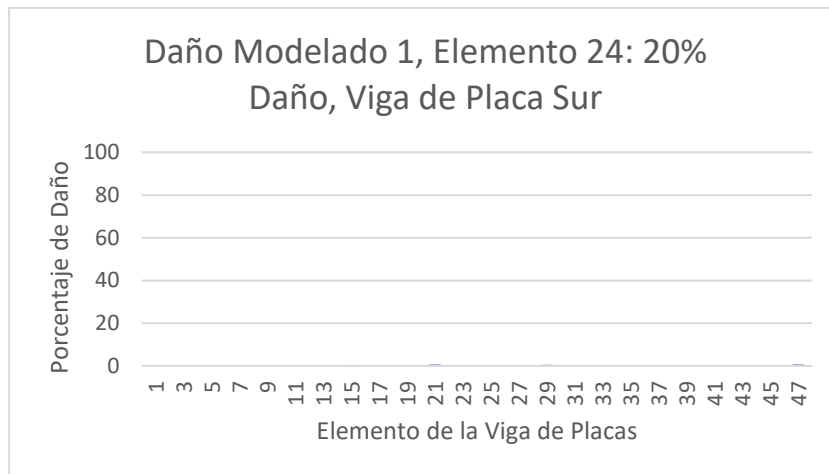


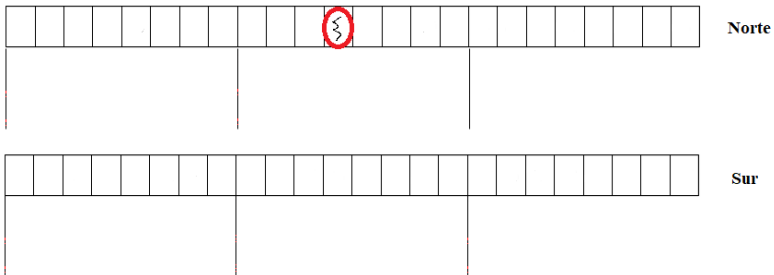
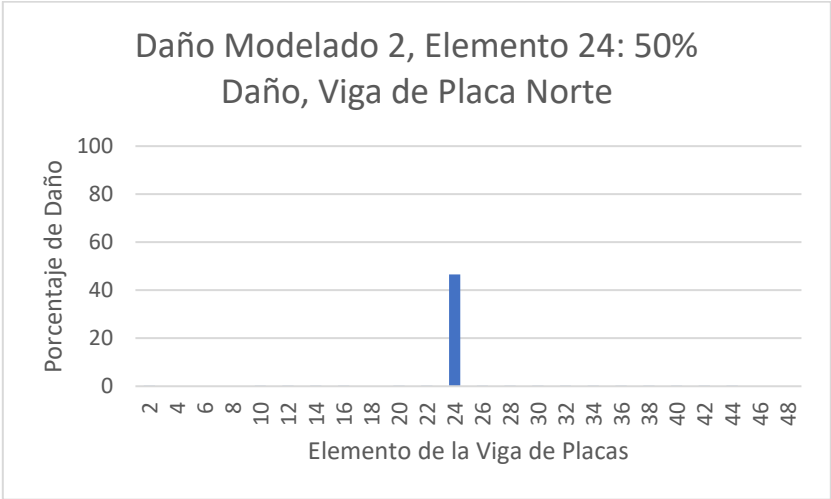
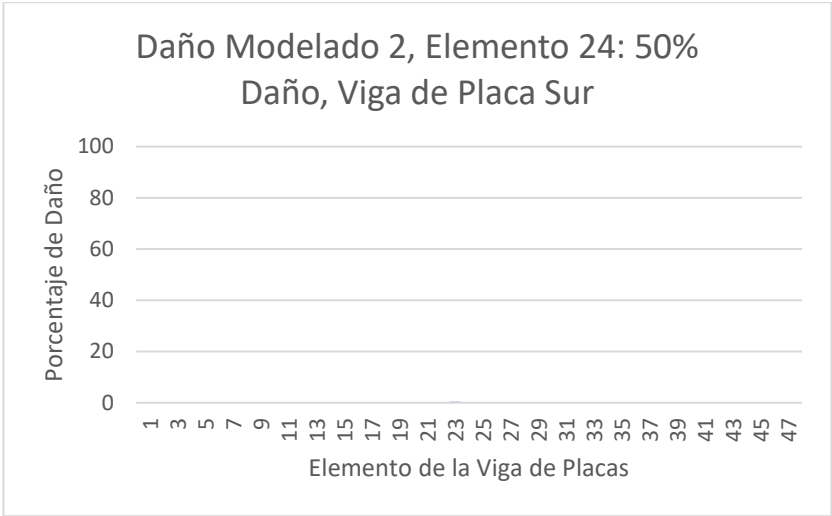
Imagen 4-11: Porcentajes de DME y FAE obtenidos de la RNPC.

A continuación, se mostrarán los resultados obtenidos luego de realizar el entrenamiento de la RNPC, luego de haber utilizado dos convoluciones, con 256 y 512 Mapas de Características, 4096 Neuronas y 200 épocas de entrenamiento.

#### 4.9. Resultados RNPC utilizando datos modelados



*Imagen 4-12: Primer test, daño modelado leve, al elemento 24 se le simula un daño del 20%, el programa predice un daño del 11%. La temperatura varía un 20% respecto de la temperatura base, siendo de -4 °C, el programa predice una temperatura de -8 °C.*



*Imagen 4-13: Segundo test, daño modelado moderado, al elemento 24 se le simula un daño del 50%, el programa predice un daño del 47%. La temperatura varía un 60% respecto de la temperatura base, siendo de 18 °C, el programa predice una temperatura de 11 °C.*

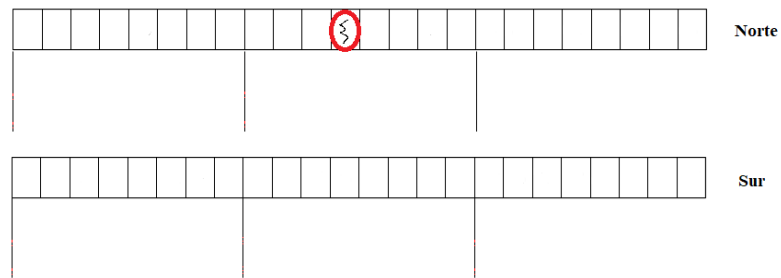
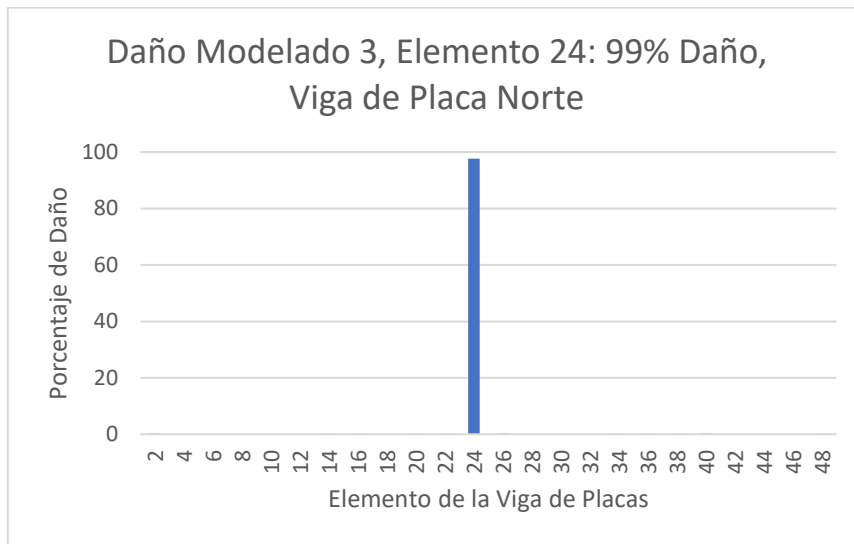
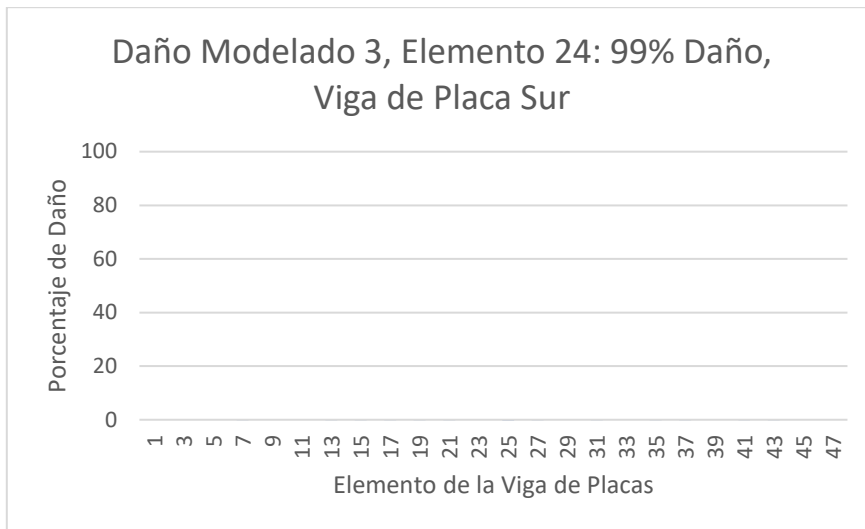


Imagen 4-14: Tercer test: daño modelado severo, al elemento 24 se le simula un daño del 99%, el programa predice un daño del 97%, la temperatura varía un 80% respecto de la temperatura base, siendo de 29 °C, el programa predice una temperatura de 32 °C.

#### 4.10. Resultados RNPC utilizando datos experimentales

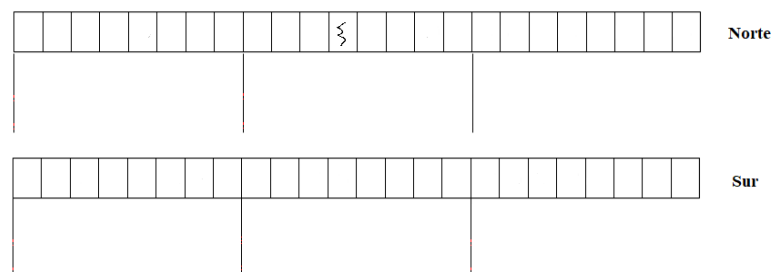
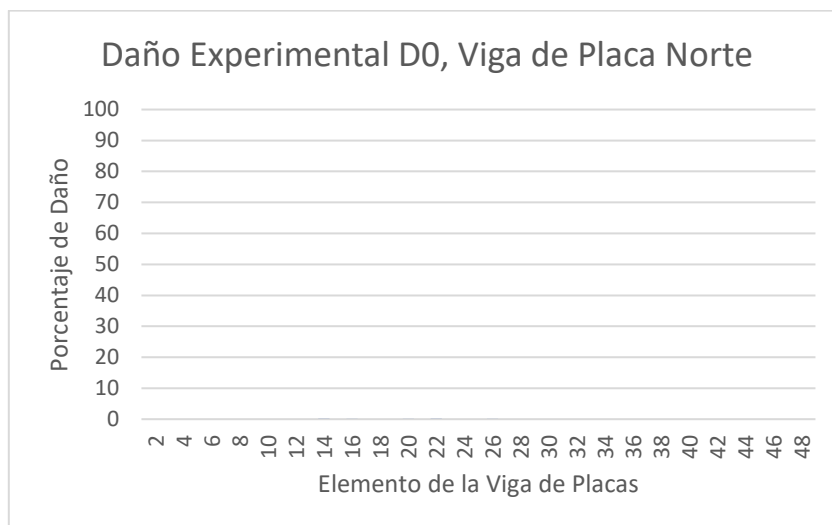
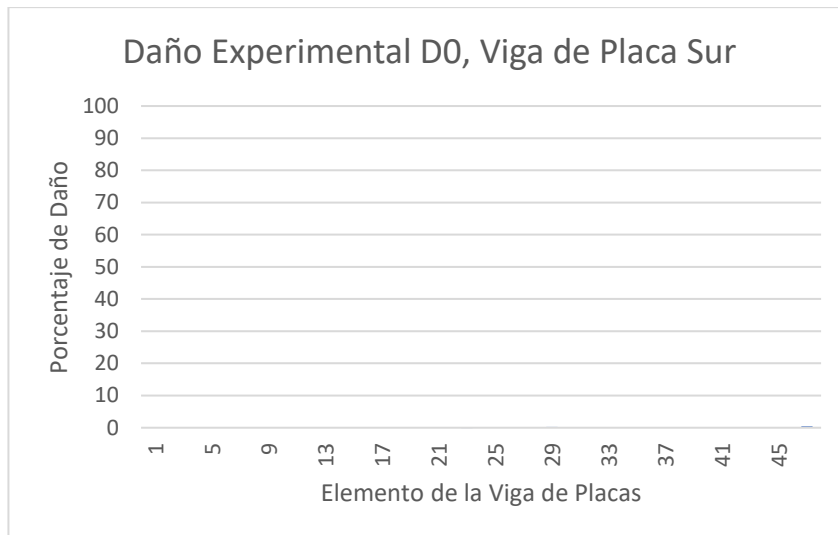


Imagen 4-15: Primera comprobación, utilizando un caso de sin daño experimental. La red neuronal no encuentra daños, se predice una temperatura de  $-14^{\circ}\text{C}$ .

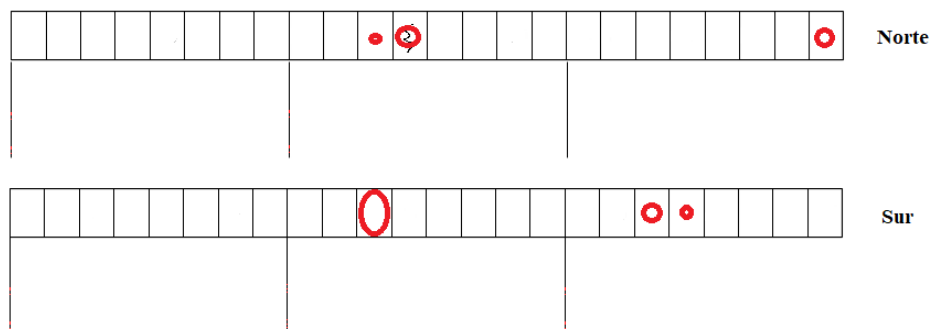
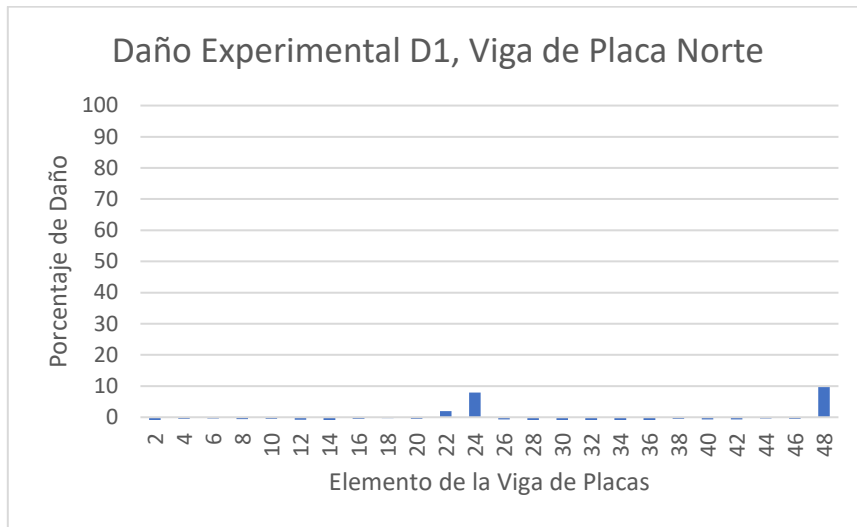
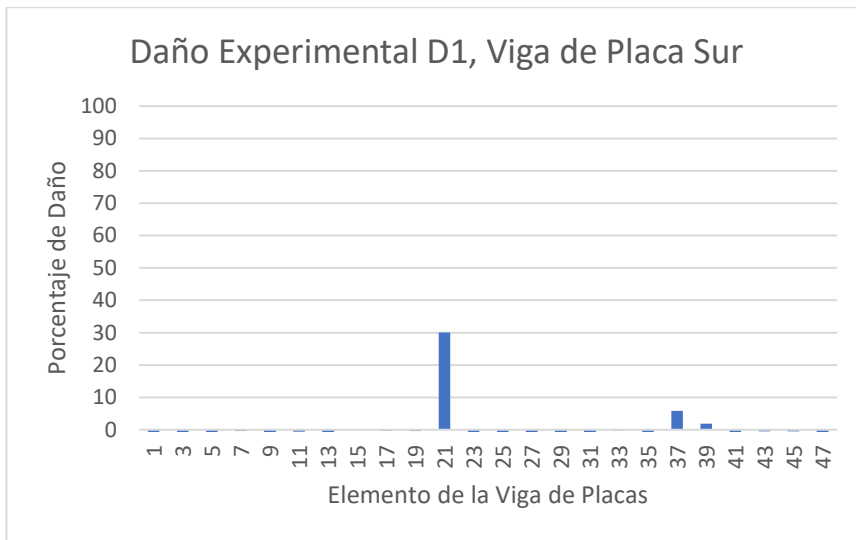


Imagen 4-16: Segunda comprobación, utilizando un caso de bajo daño experimental en el elemento 24, se encuentran varios daños, además del elemento 24, la temperatura predicha es 15 °C.

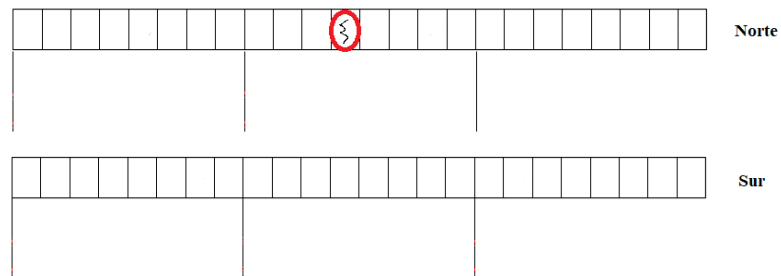
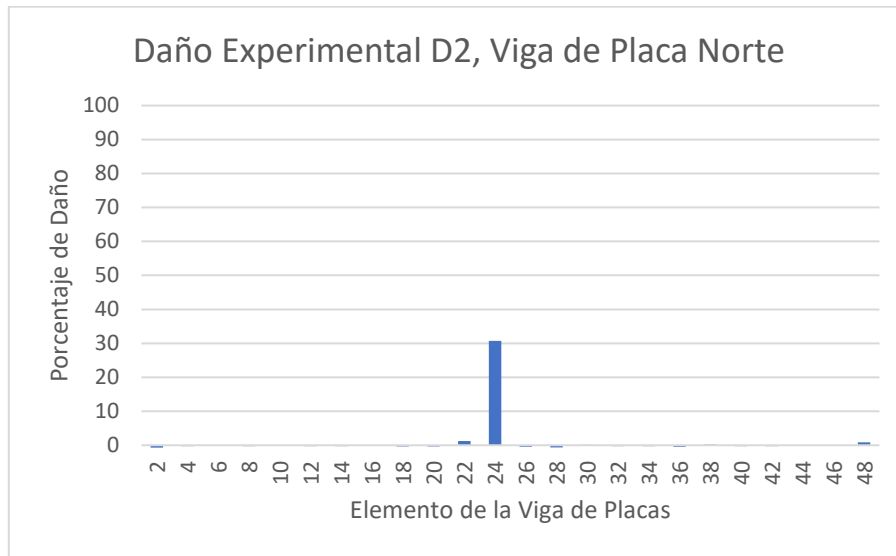
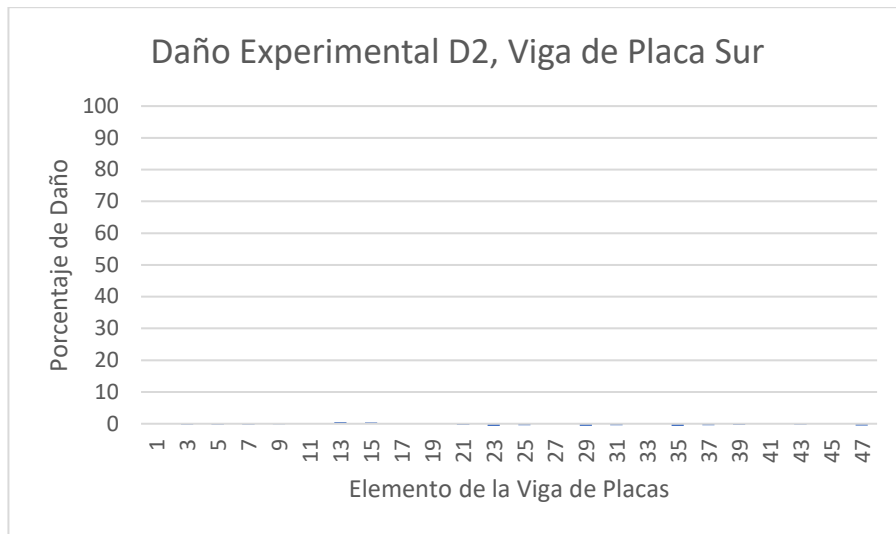


Imagen 4-17: Tercera comprobación, utilizando un caso de daño moderado experimental en el elemento 24. La red reconoce daño en el elemento 24 de un 30%, la temperatura predicha es 0 °C



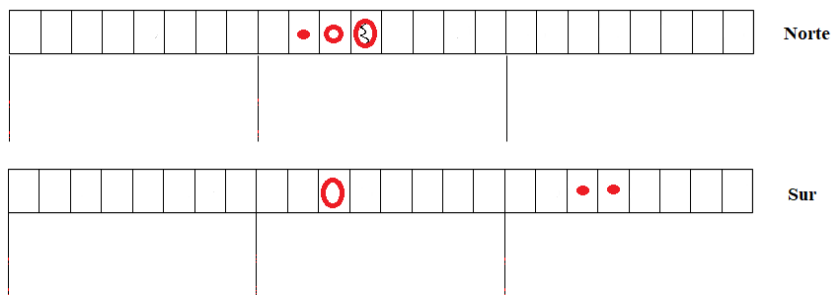
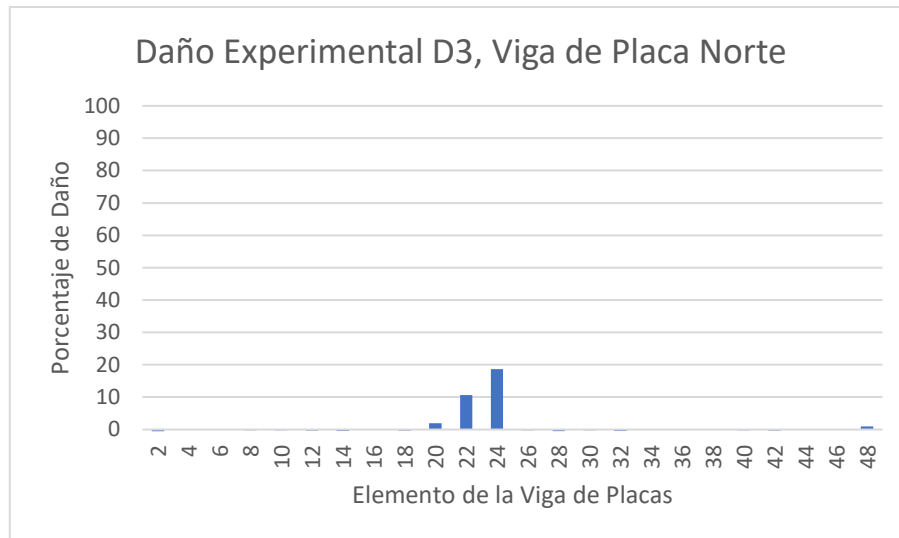
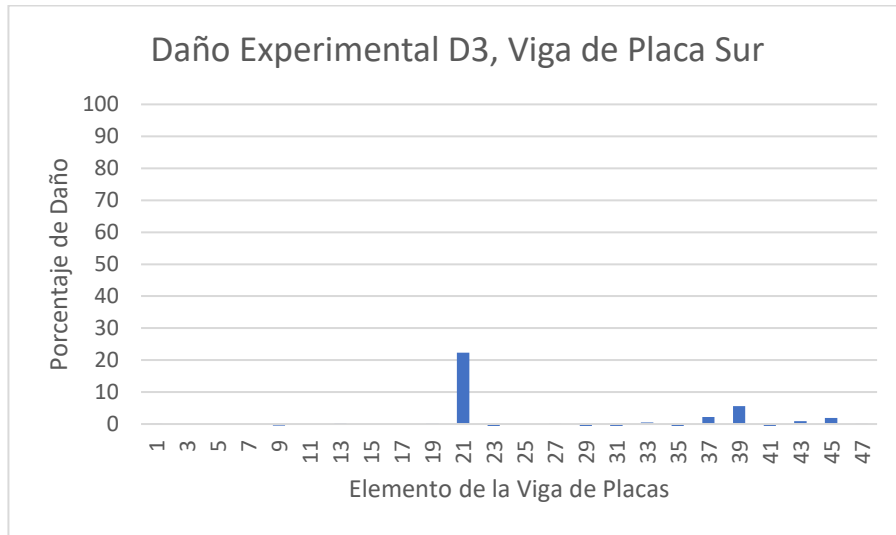


Imagen 4-18: Cuarta comprobación, utilizando un caso de daño experimental moderado en el elemento 24. La red detecta diversos daños en el puente, la temperatura predicha es 5 °C.

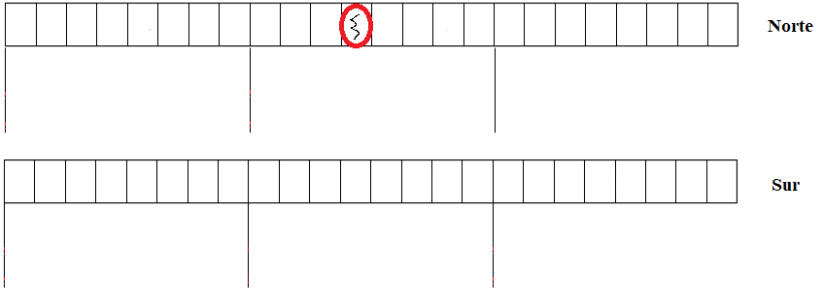
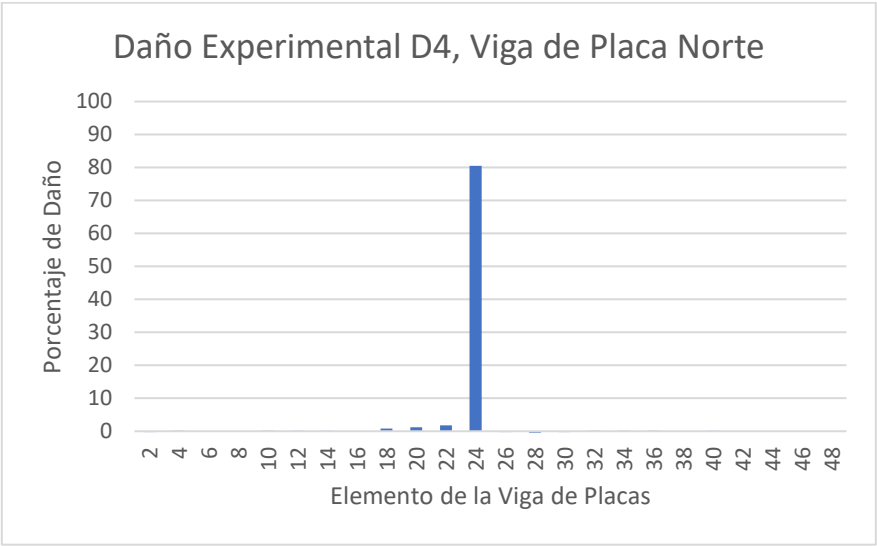
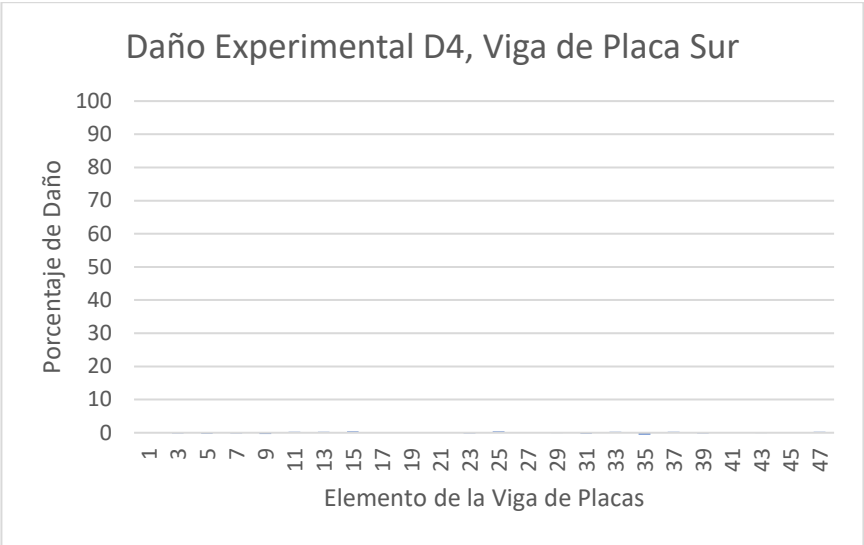


Imagen 4-19 20: Quinta comprobación, utilizando un daño experimental severo en el elemento 24. La red detecta un daño severo en el elemento 24, la temperatura predicha es 8°C.

## 5. Discusiones

Podemos dividir los resultados obtenidos en dos categorías, la fidelidad de predicción respecto al modelo y respecto a los datos experimentales. En cuanto a la primera categoría, mantiene un porcentaje elevado de certeza en daños con rangos mayores al 30%, bajo este rango, el resultado fluctúa aleatoriamente, daños bajo el 10% caen siempre en falsa alarma o no son detectados, esto se puede explicar en base a los datos trabajados en la memoria, hay que recordar que se utilizaron los modos de vibración y las frecuencias naturales. Los modos de vibración en esta investigación son trabajados como una representación gráfica de ellos, es decir, la deformación que tiene el puente. Las investigaciones previas realizadas por los Álamos National Laboratory mostraron que pequeños y medianos daños realizados al puente no generaban cambios notables en los modos de vibración, ni tampoco en las frecuencias naturales de dichos modos, es por esto, que la predicción de daño menor al 20% presenta una gran dificultad si se utilizan estos datos, por lo tanto, en futuras investigaciones puede ser interesante utilizar otros tipos de datos que sean más susceptibles y varíen más en daños.

Respecto a la fidelidad de los resultados de predicción de daños experimentales, el daño máximo (daño 4) es el único daño que logra generar una variación notable en los modos de vibración y frecuencias naturales, es por esto que la red neuronal es capaz de predecirlo, reiterando que uno de los parámetros más importantes son la forma del modo de vibración, una variación de este entregará un daño asociado. Respecto al por qué el daño 2 es correctamente detectado y los daños 1 y 3 no, la explicación se basa en el comportamiento simétrico del puente en los modos 1 y 3 en las vigas de placa norte y sur, confundiendo a la red neuronal y haciendo creer que el daño se encuentra en ambos lados del puente, como se puede ver en la Imagen 4-16 y la Imagen 4-18, no así el daño 2, que tiene un comportamiento marcado en la viga norte cuando se llega a dicho nivel de daño.

Respecto a los datos y tratamientos, son completamente necesarios, la utilización de los modos de vibración directamente (datos en bruto) con las frecuencias naturales genera gran dificultad a la red neuronal para detectar patrones de variación, esto es porque la red neuronal entrena con los datos de los modos, aprendiendo de estos, pero no entrena en la variación de estos, es por esto que se hace necesario hacer una diferencia en los datos, respecto a un dato estático (la estructura sin daño y con temperatura basal, como se hizo en esta investigación, ver sección 4.5 Tratamiento de datos). Al realizar este tratamiento de datos la red neuronal es capaz de aprender las diferencias que existen entre los modos de vibración y no el modo de vibración en sí, que es lo que se buscaba. Sin embargo, el tratamiento con diferencias resulta ser incompleto, si la red neuronal se entrena con diferencias no existe aprendizaje, esto es debido a que la diferencia de datos resulta en valores extremadamente pequeños y muy variables en escalas ( $10E-16$  hasta  $10E-3$ ), por lo que la red neuronal traduce gran cantidad de información en cero, y otra pequeña información la toma como un dato relevante para los pesos, es por esto que es necesario normalizar los datos. El paso de normalización es crucial, una mala normalización puede hacer que la red neuronal aprenda defectuosamente o que los datos que tienen valores salientes de la media no entreguen resultados satisfactorios. La normalización utilizando el promedio y la desviación estándar del total de los valores entrega resultados insuficientes, logrando predecir los daños del modelo con gran precisión pero siendo imposible encontrar daño en los datos experimentales, esto, debido a que los datos experimentales

presentan información que varían levemente de los estandarizados del modelo, al aplicar la media y la desviación estándar, ciertos valores de los modos de vibración se salen de las escalas normales, alcanzando valores 10 a 100 veces más grandes que los datos normalizados del modelo, esto resultaba en que la red neuronal es incapaz de procesar datos tan variables. La normalización unitaria, la cual hace que todos los datos queden con valores entre cero y uno, permite que la red neuronal funcione, dado que impone que todos los valores tratados del modelo y del experimento estén siempre dentro del mismo rango, la red neuronal es capaz de procesar los datos y predecir daños satisfactorios tanto para datos modelados como experimentales con un error esperable.

## 6. Conclusiones

Se generó un modelo representativo del puente I-40 de Nuevo México, los modos de vibración del modelo y sus frecuencias naturales fueron llevados a imágenes y archivos que la red neuronal pudo utilizar para entrenarse. El modelo numérico realizado en MATLAB funciona, teniendo un comportamiento físico similar al puente real, sin embargo, el puente modelado tiene un comportamiento simétrico entre las dos vigas de placas, de forma que los modos de vibración del puente pueden ser iguales si el daño ocurre en la placa norte o sur, caso que no ocurre seguido en el puente experimental. Esto genera que, al momento de evaluar los daños experimentales, se encuentren daños en la viga de placas opuestas.

Se identifica el daño, pero la localización y cuantificación puede depender en cada caso; si se trata del modelo, las predicciones son con precisión, en cuanto a los datos experimentales, existen algunos casos localizados y cuantificados con precisión y en otros existen problemas de localización o cuantificación, por lo tanto, es posible detectar los daños estructurales del puente I-40 de Nuevo México utilizando redes neuronales profundas convolucionales. Es posible identificar el daño más severo. Los daños parciales, si bien son identificables, la cantidad de daño predicha puede no ser representativa.

Es posible mejorar los resultados, haciendo más sensible la red neuronal frente a pequeños daños, para esto es necesario identificar más variables físicas del puente que se vean afectadas de forma más significativa al efectuarle daño. En cuanto al problema de daños detectados en ambos lados del puente. una posibilidad es entrenar la red neuronal para ser más robusta a problemas de simetría, para lograr esto, puede ser interesante generar escenarios de daños en diversos puntos simultáneamente en el puente, en especial, daños que se encuentren de frente en las vigas de placas, de modo que la red neuronal aprenda que el comportamiento del puente al tener daños en ambas vigas de placas (un daño frente al otro) es diferente al de un solo lado de la viga de placa, de esta manera el programa podría ser más robusto en la predicción del daño experimental disminuyendo el problema de encontrar daño simétrico.

## 7. Bibliografía

- [1] T. Kohonen. Self-organization and associative memory. Springer Verlag, New York, 1989.
- [2] M. Gestal Pose. Introducción a las Redes de Neuronas Artificiales. Depto. De Tecnologías de la Información y Comunicaciones. Universidad de Coruña.
- [3] I. Goodfellow, Y Bengio, A Courville, “Deep Learning”, Chapter 9 “Convolutional networks”.
- [4] V. Meruane, W. Heylen, “Structural damage assessment under varying temperature conditions”.
- [5] V. Meruane, J. Mahu, “Real-Time Structural Damage Assessment Using Artificial Neural Networks and Antiresonant Frequencies”.
- [6] V. Meruane, A. Ortiz-Bernandin, “Structural damage assessment using linear approximation with maximum entropy and transmissibility data”. Mechanical Systems and Signal Processing 54-55 (2015) 210-223.
- [7] Los Alamos NATIONAL LABORATORY, Finite Element Analysis of the I-40 Bridge Over the Rio Grande, P. 1 – 50.
- [8] MATLAB online. DEEP LEARNING [en línea] <https://la.mathworks.com/discovery/deep-learning.html>
- [9] E. Parloo, P. Guillaumem, M. Van Overmeire. Damage Assessment Using Mode Shapes Sensitivities. Mechanical Systems and Signal Processing .Volume 17, Issue 3, May 2003, Pages 499-518
- [10] Scott W. Doebling, Charles R. Farrar. Statistical Damage Identification Techniques Applied to the I-40 Bridge Over The Rio Grande River. Alamos National Laboratory Los Alamos, NM, 87545.

## 8. Anexos

### 8.1. Anexos imágenes

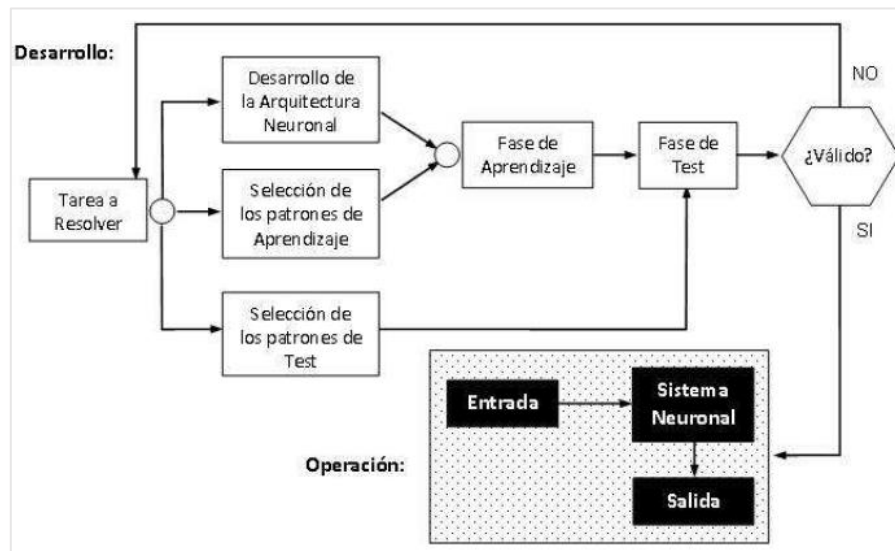


Imagen 8-1: Esquema básico del trabajo con Redes Neuronales Artificiales.

## 8.2. Anexo códigos

### 8.2.1. Función MSF y Escaladora

```
1. function [MSF] = msf(V1,V2)
2.
3. for i=1:length(V1(1,:))
4.     for j=1:length(V2(1,:))
5.         MSF(i,j)=(V1(:,i)'*V2(:,j))/(V2(:,i)'*V2(:,j));
6.     end
7. end
8.
9. function [Phied_escalado]=escalador(d0,dx)
10.     Phied_escalado=zeros(26,6);
11.     Factor=msf(d0,dx);
12.     diag_1=diag(Factor);
13.     for k=1:6
14.         Phied_escalado(:,k)=diag_1(k).*dx(:,k);
15.     end
16. end
```

### 8.2.2. Código RNPC para importar datos de modelo

```
1. import numpy as np
2. from sklearn import preprocessing
3. from utils_gen import LoadImages2,Damage
4. import os
5. import scipy.io as spio
6.
7. frec_model_con_dano = spio.loadmat(
8.     'C:/Users/Orlando/Desktop/CNN memoria/datos_modelo/phi_diff_norm_matrix_con_dano.mat',
9.     squeeze_me=True)
10. frec_model_con_dano = frec_model_con_dano['phi_diff_norm_matrix_con_dano']
11.
12. frec_model_sin_dano = spio.loadmat(
13.     'C:/Users/Orlando/Desktop/CNN memoria/datos_modelo/phi_diff_norm_matrix_sin_dano.mat',
14.     squeeze_me=True)
15. frec_model_sin_dano = frec_model_sin_dano['phi_diff_norm_matrix_sin_dano']
16.
17. w_model_con_dano = spio.loadmat(
18.     'C:/Users/Orlando/Desktop/CNN memoria/datos_modelo/w_diff_norm_matrix_con_dano.mat'
19.     , squeeze_me=True)
20. w_model_con_dano = w_model_con_dano['w_diff_norm_matrix_con_dano']
21.
22. w_model_sin_dano = spio.loadmat(
23.     'C:/Users/Orlando/Desktop/CNN memoria/datos_modelo/w_diff_norm_matrix_sin_dano.mat',
24.     squeeze_me=True)
25. w_model_sin_dano = w_model_sin_dano['w_diff_norm_matrix_sin_dano']
26.
27. target_model_con_dano = spio.loadmat(
28.     'C:/Users/Orlando/Desktop/CNN memoria/datos_modelo/registro_con_dano.mat',
29.     squeeze_me=True)
30. target_model_con_dano = np.float64(target_model_con_dano['registro_con_dano'])
31.
32. target_model_sin_dano = spio.loadmat(
33.     'C:/Users/Orlando/Desktop/CNN memoria/datos_modelo/registro_sin_dano.mat',
```



```

34.         squeeze_me=True)
35. target_model_sin_dano = np.float64(target_model_sin_dano['registro_sin_dano'])
36.
37. data0=frec_model_sin_dano
38. data1=frec_model_con_dano
39.
40. target0=np.float64(target_model_sin_dano)
41. target1=np.float64(target_model_con_dano)
42.
43. frecuencias0=w_model_sin_dano
44. frecuencias1=w_model_con_dano
45.
46. y_reg0=Damage(target0)
47. y_reg1=Damage(target1)
48.
49. y_reg=np.concatenate((y_reg0,y_reg1)) #Concatenate data
50.
51. data=np.concatenate((data0,data1)) #Concatenate labels
52.
53. omega=(np.concatenate((frecuencias0,frecuencias1)))
54.
55. #Splitting the data set into the Training set and Test set
56. from sklearn.cross_validation import train_test_split
57.
58.
59. data_train,data_test,y_reg_train,y_reg_test,omega_train,omega_test=train_test_split(
60.     data,y_reg,omega,test_size=0.2, random_state=0)
61.
62. mean_phi = data_train.mean(axis=0)
63. std_phi = data_train.std(axis=0)
64. data_train = (data_train - mean_phi) / std_phi
65. data_test = (data_test - mean_phi) / std_phi
66.
67. mean_w = omega_train.mean(axis=0)
68. std_w = omega_train.std(axis=0)
69. omega_train = (omega_train - mean_w) / std_w
70. omega_test = (omega_test - mean_w) / std_w
71.
72. #Utilizando desv estandar y media (no funciona)
73. #min_data_phi = data_train.min()
74. #max_data_phi = data_train.max()
75. #data_train = (data_train - min_data_phi) / (max_data_phi - min_data_phi)
76. #data_test = (data_test - min_data_phi) / (max_data_phi - min_data_phi)
77.
78. #min_data_w = omega_train.min()
79. #max_data_w = omega_train.max()
80. #omega_train = (omega_train - min_data_w) / (max_data_w - min_data_w)
81. #omega_test = (omega_test - min_data_w) / (max_data_w - min_data_w)
82.
83. omega_train=omega_train/40
84. omega_test=omega_test/40
85.
86. np.save(file = 'data_train', arr = data_train)
87. np.save(file = 'data_test', arr = data_test)
88.
89. np.save(file = 'y_train', arr = y_reg_train)
90. np.save(file = 'y_test', arr = y_reg_test)
91.
92. np.save(file = 'omega_train', arr= omega_train)
93. np.save(file = 'omega_test', arr=omega_test)

```

### 8.2.3. Código RNPC para trabajar modelo

```
1. import tensorflow as tf
2. import numpy as np
3. from sklearn import preprocessing
4. from utils_gen import LoadImages2
5. from Plots import BarPlot
6. import os
7. import scipy.io as spio
8.
9. def Batch(iterable, n=1):
10.     l = len(iterable)
11.     for ndx in range(0, l, n):
12.         yield iterable[ndx:min(ndx + n, l)]
13.
14. #-----Train, Test Files
15. data_train = np.load(file = 'data_train.npy')
16. data_test = np.load(file = 'data_test.npy')
17. y_reg_train = np.load(file = 'y_train.npy')
18. y_reg_test = np.load(file = 'y_test.npy')
19.
20.
21. omega_train = np.load(file = 'omega_train.npy')#mios nuevos
22. omega_test = np.load(file = 'omega_test.npy')#mios nuevos
23.
24.
25. sess = tf.InteractiveSession()
26.
27. input_x = 26 #alto de la imagen de entrada, en este caso es 6
28. input_y = 6 #ancho de la imagen de entrada, en este caso es 26
29. output_elements = 49 # Número de elementos que componen la estructura,
30.     # en este caso es 49
31.
32. input_x_w = 6 #alto de la imagen de entrada, en este caso es 6
33. input_y_w = 1 #ancho de la imagen de entrada, en este caso es 1
34.
35. #placeholder para los modos (phi)
36. x = tf.placeholder(tf.float32, shape=[None, input_x*input_y],name="x")
37. y_ = tf.placeholder(tf.float32, shape=[None, output_elements],name="y_")
38.
39. #otro placeholder para las frecuencias
40. omega = tf.placeholder(tf.float32, shape=[None, input_x_w],name="omega")
41.
42. #-----
43.
44. def weight_variable(shape):
45.     initial = tf.truncated_normal(shape, stddev=0.1)
46.     return tf.Variable(initial)
47.
48.
49. def bias_variable(shape):
50.     initial = tf.constant(0.1, shape=shape)
51.     return tf.Variable(initial)
52.
53. def conv2d(x, W):
54.     return tf.nn.conv2d(x, W, strides=[1, 1, 1, 1], padding='VALID')
55.
56.
57. hidden_units = 4096*2 #
58. image_size_x=input_x
59. image_size_y=input_y
60. x_image = tf.reshape(x, [-1,image_size_x,image_size_y,1])
```

```

61.
62. # PFirst convolution
63. filter1_x= image_size_x
64. filter1_y= 1
65. Feature_Maps1= 256
66. W_conv1 = weight_variable([filter1_x, filter1_y, 1, Feature_Maps1])
67. b_conv1 = bias_variable([Feature_Maps1])
68. h_conv1 = tf.nn.relu6(conv2d(x_image, W_conv1) + b_conv1)
69. drop1 = tf.nn.dropout(h_conv1, keep_prob=1.0)#####
70. x_conv1 = image_size_x-filter1_x+1
71. y_conv1 = image_size_y-filter1_y+1
72.
73. # Second convolution
74. filter2_x=1
75. filter2_y=2
76. Feature_Maps2 = 512
77. W_conv2 = weight_variable([filter2_x, filter2_y, Feature_Maps1, Feature_Maps2])
78. b_conv2 = bias_variable([Feature_Maps2])
79. h_conv2 = tf.nn.relu6(conv2d(drop1, W_conv2) + b_conv2)
80. #h_conv2 = tf.nn.relu6(conv2d(h_conv1, W_conv2) + b_conv2)
81. drop2 = tf.nn.dropout(h_conv2, keep_prob=1.0)#####
82. x_conv2 = x_conv1-filter2_x+1
83. y_conv2 = y_conv1-filter2_y+1
84.
85. # FullyConnected Layer
86. N_frecuencias = 6
87. W_fc1 = weight_variable([x_conv2*y_conv2 * Feature_Maps2+N_frecuencias,
88.                           hidden_units])
89.
90. b_fc1 = bias_variable([hidden_units])
91.
92. #h_pool2_flat = tf.reshape(h_conv2, [-1, x_conv2*y_conv2*Feature_Maps2])
93. h_pool2_flat = tf.reshape(drop2, [-1, x_conv2*y_conv2*Feature_Maps2])
94.
95. h_fc1 = tf.nn.relu6(tf.matmul(tf.concat([h_pool2_flat,omega],-1), W_fc1) + b_fc1)
96. keep_prob = tf.placeholder(tf.float32,name="keep_prob")
97. h_fc1_drop = tf.nn.dropout(h_fc1, keep_prob)
98.
99.
100.W_fc2 = weight_variable([hidden_units, output_elements])
101.b_fc2 = bias_variable([output_elements])
102.
103.y_conv = tf.add(tf.matmul(h_fc1_drop, W_fc2),b_fc2,name="y_conv")
104.
105.cross_entropy=tf.reduce_mean(tf.square(y_-y_conv),name='cross_entropy')
106.
107.#Metrics
108.train_step = tf.train.AdamOptimizer(1e-4).minimize(cross_entropy)
109.#train_step = tf.train.RMSPropOptimizer(1e-4).minimize(cross_entropy)
110.correct_prediction = tf.abs(y_-y_conv)
111.accuracy = 1-tf.reduce_mean(correct_prediction)
112.
113.sess.run(tf.global_variables_initializer())
114.saver = tf.train.Saver() #save the trained network
115.
116.epochs=1000
117.i=0
118.test_size=np.shape(y_reg_test)
119.bath_size=20
120.
121.n_data=86171 #training dataset size to determine the batch size
122.test1=0.0
123.test2=0.0

```

```

124.k=0#contador
125.
126.#with tf.Session() as sess:
127.
128.
129.for epoch in range(epochs):
130. for batch in Batch(range(0,n_data), 20):
131.     i=i+1
132.     if i%500 == 0:
133.
134. #agregar el reg qaca y tamos *****
135.     train_accuracy = accuracy.eval(feed_dict={
136.         x:data_train[batch], omega: omega_train[batch], y_: y_reg_train[batch],
137.         keep_prob: 1.0,} )
138.     print("step %d, training accuracy %g"%(i, train_accuracy))
139.     train_step.run(feed_dict={x: data_train[batch], y_: y_reg_train[batch],
140.         omega: omega_train[batch], keep_prob: 0.5})
141.     print("Step %g out of %g"%((epoch+1),epochs))
142.     test2=accuracy.eval(feed_dict={x: data_test, y_: y_reg_test,
143.         omega: omega_test, keep_prob: 1.0})
144.     print("test accuracy %g"%test2)
145.
146. #Missclassification
147. y=sess.run(y_conv,feed_dict={x:data_test, omega: omega_test ,keep_prob: 1.0})
148. dsim=1.0-y;
149. doutput=1.0-y_reg_test;
150. MSE=np.sum(np.abs(dsim-doutput))/(np.shape(dsim)[0]*np.shape(dsim)[1])
151. alpha=MSE
152. Dr=doutput<1.0 #Danos reales
153. alpha2=1.0-alpha
154. Dd=dsim<alpha2
155. NT=np.sum(Dr)
156. NT.astype(float)
157. NF=np.sum(Dd)
158. NF.astype(float)
159. DME0=1.0/NT*(NT-np.sum(np.multiply(Dr,Dd)))
160. FAE0=1.0/NF*np.sum(np.multiply((1-Dr),Dd))
161. print("False Negative (DME) %g"%DME0)
162. print("False Alarm (FAE) %g"%FAE0)
163.
164. if test1>=test2: #si accuracy no mejora, aumenta el contador
165.     k+=1
166. else:     #Si accuracy mejora, contador vuelve a cero
167.     k=0
168.     test1=test2
169.
170. if k==5:
171.     print('Training not improving, stopping to avoid overfitting')
172.     #Si contador llega a 5, se deja de iterar para evitar overfitting
173.     break
174.
175.#File name to save data
176.real='real1.txt'
177.calculado='calculado1.txt'
178.
179. #Save results
180.saver.save(sess, 'C:/Users/Orlando/Desktop/CNN memoria/Modelo_01/',global_step=1000)
181.np.savetxt(real, y_reg_test, fmt='%1.4e') # use exponential notation
182.np.savetxt(calculado, y, fmt='%1.4e') # use exponential notation
183.
184.constante_w=40
185.
186.#daño bajo

```

```

187.frec = spio.loadmat(
188.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/phi_784_dano_200_temp_200.mat',
189.     squeeze_me=True)
190.frec_real = frec['phi_784_dano_200_temp_200']
191.frec_real=frec_real.reshape(1,156)
192.data_real_phi=frec_real
193.wed = spio.loadmat(
194.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/w_784_dano_200_temp_200.mat',
195.     squeeze_me=True)
196.w_real = wed['w_784_dano_200_temp_200']
197.w_real=w_real.reshape(1,6)
198.
199.#data_real_phi = (data_real_phi - min_data_phi) / (max_data_phi - min_data_phi)
200.#w_real = (w_real - min_data_w) / (max_data_w - min_data_w)
201.
202.data_real_phi = (data_real_phi - mean_phi) / std_phi
203.w_real = (w_real - mean_w) / std_w
204.data_real_w=w_real/constante_w
205.resultado_DE1=sess.run(y_conv,feed_dict={x:data_real_phi, omega : data_real_w,
206.                                     keep_prob: 1.0})
207.
208.#daño medio
209.frec = spio.loadmat(
210.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/phi_1843_dano_500_temp_600.mat',
211.     squeeze_me=True)
212.frec_real = frec['phi_1843_dano_500_temp_600']
213.frec_real=frec_real.reshape(1,156)
214.data_real_phi=frec_real
215.wed = spio.loadmat(
216.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/w_1843_dano_500_temp_600.mat',
217.     squeeze_me=True)
218.w_real = wed['w_1843_dano_500_temp_600']
219.w_real=w_real.reshape(1,6)
220.
221.#data_real_phi = (data_real_phi - min_data_phi) / (max_data_phi - min_data_phi)
222.#w_real = (w_real - min_data_w) / (max_data_w - min_data_w)
223.
224.data_real_phi = (data_real_phi - mean_phi) / std_phi
225.w_real = (w_real - mean_w) / std_w
226.data_real_w=w_real/constante_w
227.resultado_DE2=sess.run(y_conv,feed_dict={x:data_real_phi, omega : data_real_w,
228.                                     keep_prob: 1.0})
229.
230.#daño alto
231.frec = spio.loadmat(
232.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/phi_2376_dano_990_temp_800.mat',
233.     squeeze_me=True)
234.frec_real = frec['phi_2376_dano_990_temp_800'] # array
235.frec_real=frec_real.reshape(1,156)
236.data_real_phi=frec_real
237.wed = spio.loadmat(
238.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/w_2376_dano_990_temp_800.mat',
239.     squeeze_me=True)
240.w_real = wed['w_2376_dano_990_temp_800'] # array
241.w_real=w_real.reshape(1,6)
242.
243.#data_real_phi = (data_real_phi - min_data_phi) / (max_data_phi - min_data_phi)
244.#w_real = (w_real - min_data_w) / (max_data_w - min_data_w)
245.
246.data_real_phi = (data_real_phi - mean_phi) / std_phi
247.w_real = (w_real - mean_w) / std_w
248.data_real_w=w_real/constante_w
249.resultado_DE3=sess.run(y_conv,feed_dict={x:data_real_phi, omega : data_real_w,

```

```

250.                                     keep_prob: 1.0})
251.
252.#dano 0
253.frec = spio.loadmat(
254.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/diff_norm_phi_escalado_d0.mat',
255.     squeeze_me=True)
256.frec_real = frec['diff_norm_phi_escalado_d0'] # array
257.frec_real=frec_real.reshape(1,156)
258.data_real_phi=frec_real
259.wed = spio.loadmat(
260.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/diff_norm_w_d0.mat',
261.     squeeze_me=True)
262.w_real = wed['diff_norm_w_d0'] # array
263.w_real=w_real.reshape(1,6)
264.
265.#data_real_phi = (data_real_phi - min_data_phi) / (max_data_phi - min_data_phi)
266.#w_real = (w_real - min_data_w) / (max_data_w - min_data_w)
267.
268.data_real_phi = (data_real_phi - mean_phi) / std_phi
269.w_real = (w_real - mean_w) / std_w
270.data_real_w=w_real/constante_w
271.resultado_D0=sess.run(y_conv,feed_dict={x:data_real_phi, omega : data_real_w,
272.                                     keep_prob: 1.0})
273.
274.#dano 1
275.frec = spio.loadmat(
276.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/diff_norm_phi_escalado_d1.mat',
277.     squeeze_me=True)
278.frec_real = frec['diff_norm_phi_escalado_d1'] # array
279.frec_real=frec_real.reshape(1,156)
280.data_real_phi=frec_real
281.wed = spio.loadmat(
282.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/diff_norm_w_d1.mat',
283.     squeeze_me=True)
284.w_real = wed['diff_norm_w_d1'] # array
285.w_real=w_real.reshape(1,6)
286.
287.#data_real_phi = (data_real_phi - min_data_phi) / (max_data_phi - min_data_phi)
288.#w_real = (w_real - min_data_w) / (max_data_w - min_data_w)
289.
290.data_real_phi = (data_real_phi - mean_phi) / std_phi
291.w_real = (w_real - mean_w) / std_w
292.data_real_w=w_real/constante_w
293.resultado_D1=sess.run(y_conv,feed_dict={x:data_real_phi, omega : data_real_w,
294.                                     keep_prob: 1.0})
295.
296.#dano 2
297.frec = spio.loadmat(
298.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/diff_norm_phi_escalado_d2.mat',
299.     squeeze_me=True)
300.frec_real = frec['diff_norm_phi_escalado_d2'] # array
301.frec_real=frec_real.reshape(1,156)
302.data_real_phi=frec_real
303.wed = spio.loadmat(
304.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/diff_norm_w_d2.mat',
305.     squeeze_me=True)
306.w_real = wed['diff_norm_w_d2'] # array
307.w_real=w_real.reshape(1,6)
308.
309.#data_real_phi = (data_real_phi - min_data_phi) / (max_data_phi - min_data_phi)
310.#w_real = (w_real - min_data_w) / (max_data_w - min_data_w)
311.
312.data_real_phi = (data_real_phi - mean_phi) / std_phi

```

```

313.w_real = (w_real - mean_w) / std_w
314.data_real_w=w_real/constante_w
315.resultado_D2=sess.run(y_conv,feed_dict={x:data_real_phi, omega : data_real_w,
316.                                     keep_prob: 1.0})
317.
318.#dano 3
319.frec = spio.loadmat(
320.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/diff_norm_phi_escalado_d3.mat',
321.     squeeze_me=True)
322.frec_real = frec['diff_norm_phi_escalado_d3'] # array
323.frec_real=frec_real.reshape(1,156)
324.data_real_phi=frec_real
325.wed = spio.loadmat(
326.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/diff_norm_w_d3.mat',
327.     squeeze_me=True)
328.w_real = wed['diff_norm_w_d3'] # array
329.w_real=w_real.reshape(1,6)
330.
331.#data_real_phi = (data_real_phi - min_data_phi) / (max_data_phi - min_data_phi)
332.#w_real = (w_real - min_data_w) / (max_data_w - min_data_w)
333.
334.data_real_phi = (data_real_phi - mean_phi) / std_phi
335.w_real = (w_real - mean_w) / std_w
336.data_real_w=w_real/constante_w
337.resultado_D3=sess.run(y_conv,feed_dict={x:data_real_phi, omega : data_real_w,
338.                                     keep_prob: 1.0})
339.
340.#dano 4
341.frec = spio.loadmat(
342.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/diff_norm_phi_escalado_d4.mat',
343.     squeeze_me=True)
344.frec_real = frec['diff_norm_phi_escalado_d4'] # array
345.frec_real=frec_real.reshape(1,156)
346.data_real_phi=frec_real
347.wed = spio.loadmat(
348.     'C:/Users/Orlando/Desktop/CNN memoria/Measured Data/diff_norm_w_d4.mat',
349.     squeeze_me=True)
350.w_real = wed['diff_norm_w_d4'] # array
351.w_real=w_real.reshape(1,6)
352.
353.#data_real_phi = (data_real_phi - min_data_phi) / (max_data_phi - min_data_phi)
354.#w_real = (w_real - min_data_w) / (max_data_w - min_data_w)
355.
356.data_real_phi = (data_real_phi - mean_phi) / std_phi
357.w_real = (w_real - mean_w) / std_w
358.data_real_w=w_real/constante_w
359.resultado_D4=sess.run(y_conv,feed_dict={x:data_real_phi, omega : data_real_w,
360.                                     keep_prob: 1.0})
361.
362.
363.##Histogramas
364.rango=np.array([0.9,0.8,0.7,0.6,0.5,0.4,0.3,0.2,0.1])
365.y_reg_test=np.loadtxt(real)
366.y=np.loadtxt(calculado)
367.dsim=1.0-y;
368.doutput=1.0-y_reg_test;
369.MSE=np.sum(np.abs(dsim-doutput))/(np.shape(dsim)[0]*np.shape(dsim)[1])
370.alpha=MSE
371.Dr=doutput<1.0 #Danos reales
372.alpha2=1.0-alpha
373.Dd=dsim<alpha2
374.NT=np.sum(Dr) #[-1] #menos la ultima
375.NT.astype(float)

```

```

376.NF=np.sum(Dd) #detectados - ultima columna
377.NF.astype(float)
378.DME0=1.0/NT*(NT-np.sum(np.multiply(Dr,Dd)))
379.FAE0=1.0/NF*np.sum(np.multiply((1-Dr),Dd))
380.DME=[]
381.FAE=[]
382.for i in range(len(rango)-1):
383.    Dr1=np.multiply((doutput>=rango[i]),(doutput<(rango[i]+0.1)))
384.    Dd1=np.multiply((dsim>=rango[i]),(dsim<(rango[i]+0.1)))
385.    NT1=np.sum(Dr1)
386.    NF1=np.sum(Dd1)
387.    DME.append(1.0/NT1*(NT1-np.sum(np.multiply(Dr1,Dd1))))
388.    FAE.append(1.0/NF1*np.sum(np.multiply((1.0-Dr),Dd1)))
389.
390.DME=np.array(DME)*100.0
391.FAE=np.array(FAE)*100.0
392.BarPlot(DME,FAE)

```