CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL INSTITUTO POLITÉCNICO NACIONAL

UNIDAD ZACATENCO

DEPARTAMENTO DE CONTROL AUTOMÁTICO

Redes neuronales convolucionales para el modelado de sistemas no lineales con aplicación al monitoreo de daño estructural.

Trabajo que presenta

Mario Antonio Lopez Pacheco

para obtener el Grado de

Doctor en Ciencias

en la Especialidad de

Control Automático

Director de la Tesis:

Dr. Wen Yu Liu

Ciudad de México Septiembre 2021

Al Consejo de Ciencia y Tecnología (CONACyT) y al Centro de Investigación y Estudios Avanzados (CINVESTAV), por el apoyo económico para la realización de mis estudios de posgrado.

A mis padres Edith Pacheco Ramírez y Mario López Ortega por todo el cariño, amor y apoyo que me han brindado toda mi vida.

A mi hermana, tías y tíos que me han apoyado durante toda mi educación.

A mis amigos por su apoyo y amistad.

A mi director de tesis el Dr. Wen Yu Liu por brindarme su apoyo, tiempo y guía para la realización de este trabajo.

A todos, Gracias.

Mario Antonio Lopez Pacheco

Resumen

En este trabajo de tesis se proponen tres tipos de arquitectura de una red neuronal convolucional (CNN) para la identificación de sistemas no lineales. Cada una de ellas, presenta ventajas respecto a otros métodos al igual que entre ellas, por ejemplo atenuar el ruido de medición, trabajar únicamente con datos de entrada o cuando los datos están incompletos. Las tres arquitecturas propuestas son: un red neuronal convolucional en el dominio del tiempo, una red neuronal convolucional en el dominio de la frecuencia y una red neuronal convolucional con valores complejos. Estas nuevas propuestas de arquitecturas además de las ventajas ya mencionadas, evitan realizar un procesamiento de los datos. Los métodos propuestos son aplicados sobre conjuntos de datos de prueba comparándolos con otros métodos existentes y entre sí, además de la aplicación en la detección de daño en estructuras de edificios.

Abstract

In this thesis work, three types of architecture of a convolutional neural network (CNN) are proposed for the identification of non-linear systems. Each one of them has advantages over other methods as well as between them, for example reducing the measurement noise, working only with input data, or when the data is incomplete. The three architectures proposals are a convolutional network in the time domain, a convolutional neural network in the frequency domain, and a convolutional neural network with complex values. These new architectural proposals, in addition to the advantages already mentioned, avoid data processing. The proposed methods are applied on benchmark sets comparing them with other existing methods and with each other, in addition to the application in damage detection of building structures.

Índice general

Índice de Figuras

Índice de Tablas

Símbolos y abreviaturas

CNN	Red neuronal convolucional
$CNNt$	Red neuronal convolucional en el dominio del tiempo con valores reales
DFT	Transformada discreta de Fourier
$CNNf$	Red neuronal convolucional en el dominio de la frecuencia
$CNNfc$	Red neuronal convolucional en el dominio de la frecuencia como clasificador
$CVCNN$	Red neuronal convolucional en el dominio del tiempo con valores complejos
AI	Inteligencia artificial
SVM	Máquina de soporte vectorial
RBM	Máquinas de Botzmann restringidas
$LSTM$	Red de memoria a largo plazo
$NARX$	Modelo exógeno autoregresivo no lineal
$ReLU$	Unidad lineal rectificada
$DCNN$	Red neuronal convolucional profunda
BP	Retro propagación
MSE	Error cuadrático medio
MLP	Red neuronal multicapa
SHM	Monitoreo de salud estructural

$MDOF$	Múltiples grados de libertad
$BIBO$	Entrada acotada, salida acotada
SCT	Secretaría de comunicaciones y transportes
$RMSE$	Error cuadrático medio
PCA	Análisis de componentes principales
RBF	Funciones de base radial
SAR	radar de apertura sintética

Capítulo 1

Introducción

La identificación de sistemas es el proceso de estimar modelos de sistemas dinámicos de acuerdo datos obervados de este [1]. Los modelos matemáticos con los cuales se realiza este proceso van desde una ecuación lineal hasta modelos más complicados como lo son las redes neuronales. Una de las propiedades importantes de las redes neuronales es su naturaleza adaptativa, lo que quiere decir que obtienen su conocimiento de su entorno mediante un proceso adaptativo llamado aprendizaje [2]. Se ha demostrado que las redes neuronales pueden ser utilizada tanto para la identificación de sistemas como para el control de ellos [3].

El aprendizaje profundo (deep learning) es un conjunto de algoritmos dentro del aprendizaje automático (machine learning), el cual utiliza un aprendizaje multinivel para representar datos o características de manera jerárquica [4]. Las dos principales áreas de aplicación del aprendizaje profundo son la clasificación y la regresión. La red neuronal convolucional (CNN), es un caso particular de los algoritmos de aprendizaje profundo y consiste del empleo de capas convolucionales y capas de submuestreo alternadas [5] seguidas de una capa completamente conectada las cuales serán definidas mas adelante. Sin embargo, gracias a las propiedades de la convolución, en el procesamiento de imágenes se han empleado para resolver problemas de clasificación de imágenes u otro tipo de señales, siendo que también pueden llegarse a ocupar para la identificación y control de sistemas.

En este trabajo se presenta el uso de redes neuronales convolucionales para la modelación

de sistemas dinámicos no lineales aplicando tres arquitecturas de red diferentes: Descrita en el dominio del tiempo con valores reales; descrita en el dominio de la frecuencia transformando la convolución en productos directos y operando sobre los componentes principales de frecuencia del sistema y por último, descrita en el dominio del tiempo pero con valores complejos. Se presenta una comparación de estas arquitecturas así como su aplicación dentro de la detección de daños en estructuras de edificios, para lo cual se estima un parámetro del modelo de Bouc Wen, el cual es utilizado en un análisis de energía para determinar si la estructura presenta daño. Además, se utiliza la red neuronal convolucional en el dominio de la frecuencia como un clasificador para localizar el daño en la estructura.

Se prueba además la existencia de una entrada óptima para los filtros convolucionales aleatorios, los cuales consisten en vectores de números aleatorios que se convolucionan con los datos de entrada en las capas convolucionales [6]. Se presenta un análisis de su respuesta ante este tipo de entradas.

1.1. Motivación

La gran variedad de algoritmos de redes neuronales tiene una vasta cantidad de aplicaciones, clasificación de imágenes, regresión de datos, reconocimiento de voz, etc. [7]. En particular, en identificación, los resultados que se han obtenido son muy buenos y difieren gracias a las propuestas de muchas arquitecturas de redes neuronales (redes neuronales multicapas, [8] redes neuronales de base radial [9]) y algoritmos para el aprendizaje (retro propagación, mínimos cuadrados, aprendizaje por refuerzo) [10]. En el caso ideal o en simulaciones donde se tiene un conocimiento total del sistema, los datos no contienen alteraciones a causa de diferentes fuentes de ruido, como pueden ser los sensores. Pero en sistemas reales, existen muchas fuentes de ruido que pueden afectar la parte de identificación y en particular el control del sistema. Existen métodos para reducir estos efectos como son los filtros, pero para poder hacer un correcto uso de ellos es necesario tener un poco de conocimiento del sistema.

Las redes neuronales convolucionales se han utilizado para la tarea de clasificación, en la

cual se han obtenido muy buenos resultados y se han mejorado con el paso de los años gracias a los componentes computacionales que de igual manera van mejorándose. Recientemente, mas resultados de la aplicación de las redes neuronales en la identificación de sistemas se han presentado [11, 12, 13], y el propósito siempre es desarrollar algoritmos más eficientes para cualquier tarea que se proponga como objetivo, tanto en precisión como en velocidad.

1.2. Objetivos

- Proponer una estructura de red neuronal convolucional profunda en el dominio del tiempo ($CNNt$) para la identificación de sistemas.
- Diseñar el algoritmo de aprendizaje de la $CNNt$.
- Probar el comportamiento de la $CNNt$ utilizando conjuntos de datos de prueba.
- Proponer una estructura de red neuronal convolucional profunda en el dominio de la frecuencia ($CNNf$) para la identificación de sistemas.
- Diseñar un algoritmo de aprendizaje de la $CNNf$.
- Probar el comportamiento de la $CNNf$ utilizando conjuntos de datos de prueba.
- Aplicar las CNNs propuestas en la detección de daños en estructuras de edificios.
- Diseñar un clasificador basado en una CNN para la localización de daño.
- Implementar el clasificador basado en una CNN para la localización de daño en estructuras.
- Proponer una estructura de red neuronal convolucional profunda en el dominio del tiempo con valores complejos ($CVCNN$) para la identificación de sistemas.
- Diseñar un algoritmo de aprendizaje de la $CVCNN$.

- Probar el comportamiento de la $CVCNN$ utilizando conjuntos de datos de prueba.
- Realizar simulaciones numéricas agregando perturbaciones a los datos de prueba.
- Comparar entre sí resultados de las redes neuronales propuestas y con otros métodos ya publicados.

1.3. Estructura

El trabajo de tesis está estructurado en 7 capítulos. El Capítulo 1 contiene la introdcción del trabajo realizado. El Capítulo 2 consiste en el marco teórico del aprendizaje profundo para la identificación de sistemas, que incluye una breve historia de la evolución de algoritmos de aprendizaje profundo y propiedades y arquitecturas de las redes neuronales convolucionales ya existentes. El Capítulo 3 trata la identificación de sistemas no lineales utilizando redes neuronales convolucionales en el dominio del tiempo, la arquitectura profunda propuesta y su aplicación en la detección de daños. En el Capítulo 4 se trata nuevamente la identificación de sistemas utilizando la CNN en el dominio de la frecuencia, además de su aplicación y mejoras en la detección de daños. Un teorema sobre la respuesta óptima de los filtros de acuerdo al tipo de señal de entrada se presenta en este capítulo. En este mismo capítulo se incluye la implementación de un clasificador para localizar daños en estructuras basado en información del sistema, las diferencias principales con la CNN utilizado para la identificación de sistemas en la parte de aprendizaje son mencionadas. En el Capítulo 5, la identificación de sistemas no lineales se aborda utilizando una CNN con valores complejos, la cual permite identificar sistemas cuando existen perturbaciones en los datos. Finalmente, en los Capítulos 6 y 7 las conclusiones y el trabajo a futuro son presentados.

1.4. Contribuciones

En este trabajo de tesis se trata con el problema de la identificación de sistemas no lineales mediante redes neuronales convolucionales. Esto se logra proponiendo arquitecturas

para la red neuronal convolucional, esto es, definir la cantidad de unidades que se encuentran en cada capa de la red y la cantidad de capas que contendrá, además del tipo de operaciones que se llevan dentro de ella, ya sea las funciones de activación o en este caso si se trabajará en el dominio del tiempo o de la frecuencia, o si se utilizarán valores reales o complejos para los *hiper-parámetros*.

Una de las propuestas consiste utilizar una arquitectura de red neuronal convolucional profunda en el dominio de la frecuencia, la cual se plantea reduzca el tiempo de computo requerido y reduzca el efecto de ruido en los datos. Una vez construida la red, el entrenamiento se realiza utilizando el algoritmo de retro propagación. La aplicación de esta nueva propuesta es en la detección de daño en estructuras de edificios, con lo que la CNN identificará un parámetro del modelo de Bouc Wen el cual es necesario para determinar los ciclos de histéresis y la energía del sistema, así con base en estas dos elementos se determinará si existe o no daño en la estructura del sistema. El modelo de Bouc Wen utiliza una gran cantidad de parámetros del sistema los cuales en la mayoría de veces son difíciles de determinar y se opta por métodos adaptables para determinarlos. Con esto queremos evitar la parte adaptable. En este ámbito de la detección de daños, se propone un algoritmo de localización de daño utilizando la CNN como un clasificador utilizando ademas de los datos obtenidos de los sensores, la señal de salida de la CNN, con el cual el desempeño del clasificador mejora que cuando no se introduce esta señal.

Un teorema sobre los efectos de utilizar filtros aleatorios en las capas convolucionales es propuesto. Este teorema trata sobre la existencia de una entrada óptima con la cual los filtros tendrán una mayor respuesta.

La propuesta de utilizar valores complejos en los *hiper-parámetros* conlleva ventajas en el modelado de sistemas, la arquitectura propuesta, como los resultados lo indica, es una mejor alternativa para el modelado de sistemas cuando existen perturbaciones en ellos, ya sean ruidos de medición o pérdida de datos.

Publicaciones

De este trabajo se han realizado varias publicaciones, las cuales se enlistan a continuación:
Artículos en revistas especializadas:

- Yu, W., & Pacheco, M. (2019). Impact of random weights on nonlinear system identification using convolutional neural networks. Information Sciences, 477, 1-14.
- Morales-Valdez, J. & Lopez-Pacheco, M. & Yu, W. (2020) . Automated damage location in Building structure by using Vibration Data and Convolutional Neural Network. Structural Control and Health Monitoring Editorial Office
- Lopez-Pacheco, M., Morales-Valdez, J., & Yu, W. (2020). Frequency domain CNN and dissipated energy approach for damage detection in building structures. Soft Computing, 24(20), 15821-15840.
- Lopez-Pacheco, M. & Yu, W. Complex Valued Convolutional Neural Networks for Nonlinear System Modeling. Neural Processing Letters 2021. (manuscrito en revisión)

Artículos presentados en congresos internacionales y nacionales:

- Morales-Valdez, J. & Lopez-Pacheco, M. & Yu, W. (2019). Damage detection of building structure based on vibration data and hysteretic model. 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE)
- Morales-Valdez, J. & Lopez-Pacheco, M. & Yu, W. (2019). Detección de daños en edificios basada en datos de aceleración y CNN . Congreso Nacional de Control Automático 2019
- M. Lopez and W. Yu (2017). Nonlinear system modeling using convolutional neural networks,"2017 14th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE), 2017, pp. 1-5.

Capítulo 2

Aprendizaje profundo y redes neuronales para la identificación de sistemas dinámicos no lineales

2.1. Redes neuronales para el modelado de sistemas

Las redes neuronales son una abstracción de los sistemas nerviosos biológicos, constituidas por un conjunto de unidades llamadas neuronas conectados unas con otras. El primer modelo fue propuesto por McCulloch y Pitts en 1943 [14], era un modelo binario, donde cada neurona tenía un umbral fijo. De aquí surgen dos tipos de modelos: los inspirados en la biología y los artificiales.

Los modelos de tipo biológico comprenden las redes que tratan de simular los sistemas neuronales, además de ciertas funciones como la auditiva o visual. En el cerebro humano se calcula que existen alrededor de cien millones de neuronas y cada una de ellas con alrededor de mil sinapsis.

La segunda categoría de redes neuronales son las artificiales [15]. Este tipo de red siguen basándose en las redes neuronales biológicas, aunque poseen otras funcionalidades y estructuras de conexión.

Las neuronas y las conexiones entre ellas constituyen la clave para el procesamiento de información. Una neurona consta de tres partes: cuerpo, dendritas y axón, Figura 2.1.

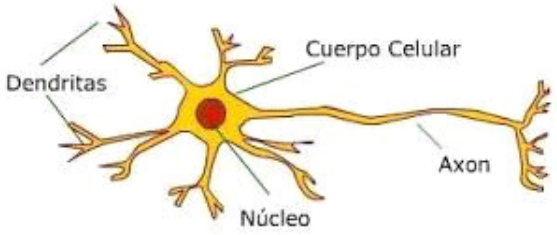


Figura 2.1: Neurona Biológica.

Las relaciones que entablan dos neuronas no son del todo conocidas ya que dependen del tipo particular de neurona que se esté tratando [16]. En general, el proceso de comunicación inicia cuando la neurona envía su información a otras neuronas a través de su axón y se transmite por diferencias de potencial. Este proceso se modela como una regla de propagación $u(\cdot)$. La neurona adyacente recoge las señales por su sinapsis sumando todas las influencias positivas o negativas. Si las influencias positivas dominan, la neurona manda una señal positiva a otras neuronas.

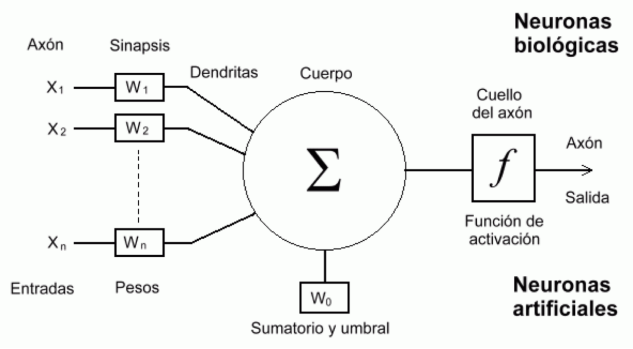


Figura 2.2: Neurona artificial basada en la biología.

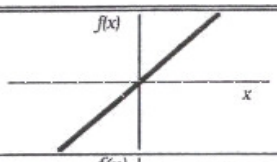
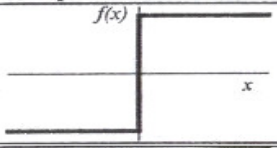
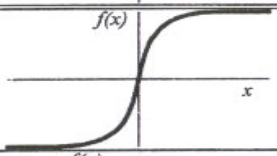
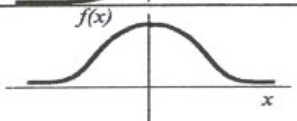
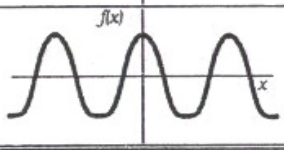
Una neurona artificial se presenta en la Figura 2.2, se indican los componentes que pueden compararse con las neuronas biológicas. Se tiene un conjunto de n entradas con sus

correspondientes pesos sinápticos w. Una función de suma Σ, una función de activación f y una salida y. La neurona puede adaptarse a su entorno modificando el valor de los pesos sinápticos. En este modelo la salida está dada por:

$$y = f(\sum_{i=1}^{n} w_i x_i) = f(u(x)) \tag{2.1}$$

La función de activación $f(\cdot)$ se elige de acuerdo a la tarea a realizar. Las funciones de activación más comunes se muestran en la Tabla 2.1.

Tabla 2.1: Funciones de activación [17].

	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	{-1, +1} {0, +1}	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	[0, +1] [-1, +1]	
Gaussiana	$y = Ae^{-Bx^2}$	[0,+1]	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	[-1,+1]	

2.1.1. Redes neuronales multicapa

Un caso particular de las redes neuronales es cuando las neuronas se organizan por capas [18], como se observa en la Figura 2.3, para este ejemplo se tiene una red neuronal con 2 capas de nodos (neuronas) y dos capas de pesos, comúnmente cuando se habla de capas, se refiere a las capas de pesos y no de neuronas.

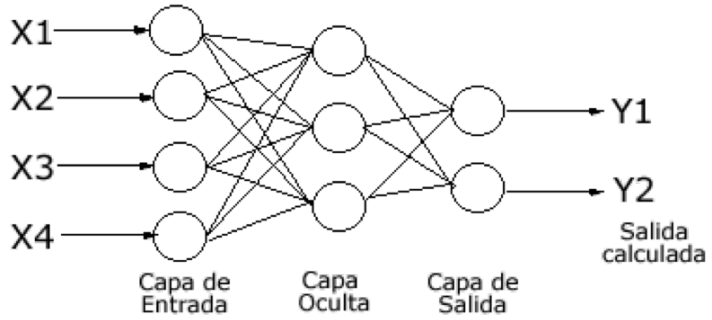


Figura 2.3: Red Neuronal Multicapa.

Si consideramos w_{ji} como el peso sináptico entre la neurona de entrada i y la neurona oculta j. La señal de entrada que recibe una neurona de la capa oculta está dada por la suma ponderada de las entradas de la red con su respectivo peso sináptico w_{ji}:

$$u(x_i) = \sum_{i=1}^{N} w_{ji} x_i + \theta_j \tag{2.2}$$

donde θ_j es el umbral o sesgo de la neurona. La señal de salida de esta neurona en la capa oculta está dada por:

$$y_j = f(u(x_i)) \tag{2.3}$$

donde $f(\cdot)$ es la función de activación [15]. Lo mismo ocurre para obtener la salida de la red, obteniendo la suma ponderada de las salidas de las neuronas en la capa oculta y aplicando sobre la suma una función de activación.

Hay dos etapas en una red neuronal: aprendizaje y generalización. En la etapa de aprendizaje o entrenamiento, como se menciona en [19], se utiliza un conjunto de datos o patrones de entrenamiento para calcular los pesos sinápticos de la red, iniciados en algún valor aleatorio, los cuales se van modificando de manera iterativa utilizando los datos de prueba, de manera que, una función de costo elegida se minimice, en su mayoría se escoge el error entre la salida de la red y la salida del sistema. En la etapa de generalización se busca que la red pueda pueda seguir realizando su tarea de manera autónoma sin la necesidad de que actualice sus parámetros, se pretende que se utilicen datos que no fueron utilizados durante la etapa de aprendizaje para validar que la red funcione correctamente y no solamente haya memorizado el conjunto de entrenamiento.

En el caso de modelado de sistemas dinámicos no lineales, se ha demostrado que una red multicapa con una única capa oculta puede aproximar a cualquier función dándole la suficiente cantidad de neuronas [20].

Para el aprendizaje de redes neuronales existen tres paradigmas populares [21].

El primero de ellos es el aprendizaje supervisado, en este tenemos un conjunto de datos de entrada y salida del sistema (maestro), y lo que se quiere es encontrar los valores de los pesos de la red neuronal (esclavo) a partir de este conjunto de datos. La función de costo utilizada comúnmente es el error cuadrático medio entre la salida de la red y la salida del sistema. Esta función de costo se minimiza a través del algoritmo de entrenamiento propuesto. Algunas tareas que entran dentro de este paradigma es el reconocimiento de patrones y la regresión. Puede verse como un aprendizaje donde tenemos un maestro que provee realimentación continua.

El segundo paradigma corresponde al aprendizaje no supervisado. En este caso se tienen algunos datos de entrada y una función de costo a minimizar. Está función varía dependiendo de la tarea a realizar. La estimación de distribuciones estadísticas se encuentran dentro de este paradigma.

El tercer paradigma corresponde al aprendizaje por refuerzo. Generalmente no existe datos de entrada y estos se generan con agentes que interaccionan con el entorno. En cada instante, el agente realiza una acción y el entorno produce una señal y costo instantáneo.

El objetivo es encontrar alguna política que minimice el costo a largo plazo. El entorno es desconocido y puede modelarse como un proceso de decisión de Markov [22] .

2.1.2. Identificación de sistemas dinámicos no lineales

El punto de inicio en el análisis de un sistema es su representación por un modelo matemático, el cual usualmente es un operador entre entradas y salidas del sistema, estos datos generalmente se presentan como series de tiempo. Estas relaciones entrada-salida, a menudo se pueden representar como una operación de convolución, o como un conjunto de ecuaciones diferenciales [23].

Un modelo es una representación simplificada de un sistema. La mayoría de los modelos matemáticos utilizados son lineales, debido a su facilidad para ser manipulados comparados con los sistemas no lineales, y pueden representar en forma precisa el comportamiento de sistemas reales en muchos casos prácticos [24]. Un modelo es útil para realizar simulaciones ya que representa la dinámica del sistema y para el diseño de controladores.

En la mayoría de las técnicas de control es necesario conocer un modelo del sistema para aplicarlas. El modelado del sistema se puede realizar a través de diferentes métodos, la aplicación de las leyes de la física que rigen el comportamiento del sistema es una de las más utilizadas [25]. Sin embargo, la aplicación de este método requiere de un conocimiento suficiente del sistema, de manera que cada uno de los parámetros de este puedan ser determinados correctamente, lo que en la mayoría de los casos es algo difícil de conseguir.

Las redes neuronales son modelos de caja negra, ya que sólo es necesario conocer los datos de entrada y salida. Estos métodos de modelado basados en datos todavía tienen problemas, el principal es que no existe un método definido para determinar los *hiper-parámetros* de la red neuronal, es decir, el número de capas ocultas, el número de nodos o neuronas en cada una de las capas [26]. En el caso de las redes neuronales convolucionales, estos *hiper-parámetros* corresponderían a la cantidad de capas convolucionales y de submuestreo, a la cantidad de filtros convolucionales y su tamaño. El teorema de aproximación universal [20] asegura que un modelo neuronal puede aproximar virtualmente a cualquier sistema continuo usando

cierto número de nodos usando una sola capa. Sin embargo, aumentar el número de nodos en esta capa conduce a otro problema, el sobreajuste de los datos [27]. Como alternativa, en lugar de sólo aumentar la cantidad de nodos en una sola capa oculta, se recomienda agregar más capas ocultas con una menor cantidad de nodos por capa, esto se conoce como redes neuronales profundas. En los últimos años, los modelos de esta nueva área, el aprendizaje profundo, se han desempeñado en muchas áreas mejor que los modelos existentes [28], tanto teórica como prácticamente [29].

La identificación de sistemas es una herramienta que permite representar el comportamiento de sistemas con base en datos experimentales obtenidos del sistema. Esto es un proceso iterativo, y el éxito de la identificación depende de la calidad de las señales de entrada y de la identificabilidad del sistema, la cual podemos definir como la posibilidad de conocer los valores reales de los parámetros del modelo a partir de un número infinito de observaciones [30].

2.1.3. Aprendizaje profundo

La Inteligencia Artificial (AI) es un campo en auge con muchas áreas de aplicación. Lo que busca es resolver tareas repetitivas, identificar imágenes, realizar diagnósticos médicos, entre otras tareas que para el ser humano sean tareas intuitivas de realizar, pero difíciles de explicar. Uno de los primeros algoritmos fue presentado por IBM, *Deep Blue* [31]. Este consiste en un algoritmo para jugar ajedrez, el cual venció al campeón mundial Garry Kasparov en 1997. Estos algoritmos se basan en conocimientos previos dados por las personas que los diseñaron. Una computadora por su parte no puede aprender conocimientos como lo hace un ser humano, estos conocimientos deben ser formalizados, y en general no lo son, ya que cada persona ve las situaciones desde su propio punto de vista. Dado este problema, la AI busca la forma de adquirir sus propios conocimientos a través de patrones en los datos en bruto. Esta capacidad de la AI se le conoce como aprendizaje automático (machine learning).

Uno de los algoritmos más sencillos del aprendizaje automático es la regresión logística [32] y su desempeño se basa en la distribución de los datos de entrada. Otro algoritmo del

aprendizaje automático es el *Naive Bayes*, generalmente utilizado como un clasificador [33]. Estos algoritmos se basan en aprender características de los datos de entrada para poder realizar su tarea. Sin embargo, saber que características se necesitan aprender sigue siendo un reto para estos algoritmos.

Dentro del aprendizaje automático se encuentran los denominados algoritmos de aprendizaje profundo (deep learning). Estos intentan resolver el problema de representación de los datos, creando representaciones más simples que representen conceptos aún más complejos que los iniciales [34]. Las dos ideas base del aprendizaje profundo son: el poder aprender de la mejor representación posible de los datos y que al tener una estructura profunda se pueda aprender más características de los datos.

Las principales áreas de aplicación de los algoritmos de aprendizaje profundo son la clasificación [35] y predicción de series de tiempo [36]. A continuación se muestran los principales algoritmos del área del aprendizaje automático.

Máquinas de soporte vectorial

El algoritmo denominado máquina de soporte vectorial (SVM) es uno de los principales algoritmos dentro del aprendizaje automático, el cual tiene mejor precision con bajo costo computacional en tareas de regresión lineal o logística. Se utiliza en su mayoría para clasificación de objetos.

El objetivo de este algoritmo es encontrar un hiperplano en un espacio N-dimensional que clasifique de manera clara a los datos. Separar dos conjuntos de datos se puede hacer mediante muchos hiperplanos, el algoritmo de SVM encuentra el hiperplano que tenga el margen más grande (la distancia entre conjuntos). Los datos que se encuentra más cerca del hiperplano son los llamados vectores de soporte y son los que influyen en la posición y orientación del hiperplano, Figura 2.4.

Considere la clasificación lineal de los datos $(x_1, y_1), \cdots, (x_n, y_n)$, donde y_i solo tienen valores 1 y -1, es decir solo pertenece a dos clases. Se desea encontrar un hiperplano de la forma $f(x) = w^T x + b$, de tal manera que $y_i f(x_i) > 0$ indique que la clasificación es correcta. w son los pesos, b el sesgo. Cualquier hiperplano puede ser escrito como el conjunto de puntos

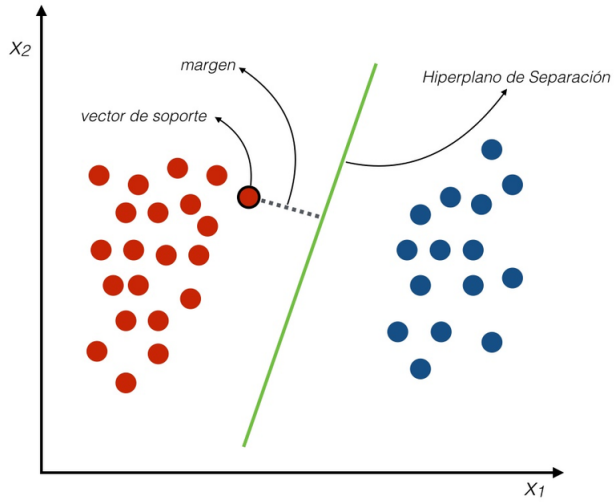


Figura 2.4: Separación lineal usando SVM.

x que satisfacen $w^T x + b = 0$.

Los parámetros $\frac{b}{||w||}$ determinaran el desplazamiento del hiperplano del origen. Si los datos se pueden separar linealmente, podemos definir dos hiperplanos mas $w^T x + b = 1$ y $w^T x + b = -1$ y la distancia entre estos dos es $\frac{2}{||w||}$ el cual es el margen del hiperplano central. Por lo tanto, para obtener el margen se necesita minimizar $||w||$, esto se convierte en un problema de optimización.

Si x_i es de la clase 1, se tiene:

$$(w^T x_i + b) \geq 1, y_i = 1 \tag{2.4}$$

Si x_i es de la clase 2, se tiene:

$$(w^T x(i) + b) \leq -1, y_i = -1 \tag{2.5}$$

Combinando las expresiones (2.4) y (2.5), se tiene que:

$$\min_{(w,b)} ||w|| \tag{2.6}$$

$$\text{sujeto a:} y_i(w^T x_i + b) \geq 1, \quad i = 1, 2, \cdots, m \tag{2.7}$$

Obteniendo este valor mínimo de w se tiene el máximo margen posible para los datos, algunas modificaciones se han realiazdo para incluir datos clasificados erróneamente [37]. Dentro de las aplicaciones de estos algoritmos se encuentra la categorización de textos [38], en clasificación de imágenes médicas [39, 40], para la predicción de series de tiempo de datos financieros [41], entre otras muchas aplicaciones.

Máquinas de Boltzmann restringidas

Las máquinas restringidas de Boltzmann (RBM) [42] definen una distribución de probabilidad p sobre un vector de datos x

$$p(x) = \sum_{h} \frac{e^{-E(x,h)}}{\sum_{u,g} e^{-E(u,g)}} \tag{2.8}$$

El parámetro h corresponde a características no observadas. Esta distribución de probabilidad se define por una función de energía E. Usando esta energía, la probabilidad condicional $p(h|v)$ y $p(v|h)$ tienen la forma:

$$p(v_i = y|h) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-a_i-\sigma_i \sum_j w_{ij} h_j)^2}{2\sigma_i^2}} \tag{2.9}$$

$$p(h_j = 1|v) = sigmoide(b_j + \sum_{i} \frac{v_i}{\sigma_i} w_{ij}) \tag{2.10}$$

donde a_i y b_j son los sesgos asociados a la entrada v_i y las variables h_j. w_{ij} representa los pesos entre estas dos variables. La arquitectura de este algoritmo se presenta en la Figura 2.5.

Si se requiere trabajar con mas de una capa, se genera una estructura en cascada, conectando la capas una después de otra como se observa en la Figura 2.6. En este caso, la probabilidad $p(h_1|v)$ es la entrada para poder calcular $p(h_2|h_1)$ y en el sentido contrario ocurre lo mismo. Solo la primera capa tiene valores reales, todas las demás son capas binarias. Trabajos recientes de aplicación de este tipo de algoritmos son presentados en [43, 44, 45].

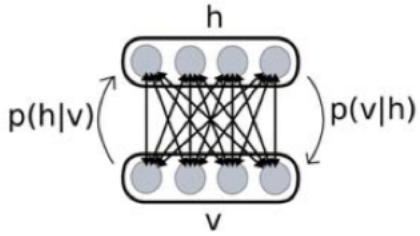


Figura 2.5: Arquitectura de una máquina restringida de Boltzmann.

Red de memoria a largo plazo

Las redes de memoria a largo plazo ($LSTM$) fueron introducidas en 1997 [47] para solucionar problemas presentes en las redes neuronales recurrentes, utilizando células de memoria. Las $LSTM$ utilizan el concepto de compuertas, con las cuales definen el comportamiento de cada célula. La entrada pasa a través de varias compuertas, cada compuerta tiene una función específica: escritura (compuerta de entrada), lectura (compuerta de salida) o reinicialización (compuerta de olvido). La activación de la célula se hace de manera similar a una red recurrente. Las ecuaciones de actualización de una $LSTM$ son:

$$i_t = \sigma_i(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{2.11}$$

$$f_t = \sigma_f(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2.12}$$

$$c_t = f_t c_{t-1} + i_t \sigma_c(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{2.13}$$

$$o_t = \sigma_o(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{2.14}$$

$$h_t = o_t \sigma_h(c_t) \tag{2.15}$$

donde i, f, o, c son las compuertas de entrada, olvido, salida y activación, respectivamente. σ es la función de activación para cada compuerta, x_t es la entrada a la célula en el instante t, W son los pesos y b los sesgos [48]. En la Figura 2.7 se observa la estructura de una célula de $LSTM$.

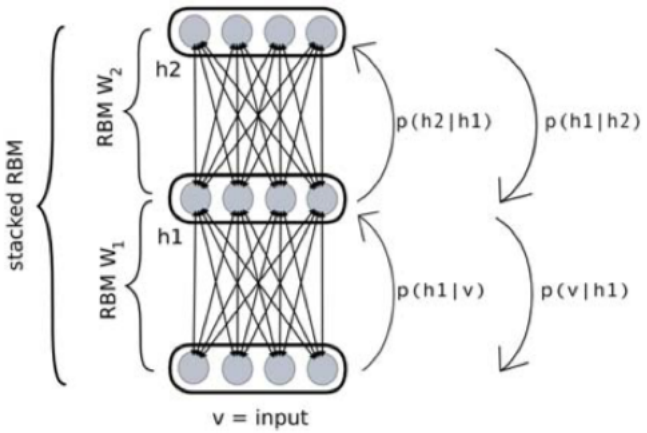


Figura 2.6: Arquitectura profunda de una máquina restringida de Boltzmann [46].

Para construir una arquitectura profunda varias de células, éstas se deben apilar una después de la otra,. La Figura 2.8 muestra 3 células conectadas entre sí. Algunas de sus aplicaciones están mencionadas por distintos autores [50, 51, 52]

2.2. Redes Neuronales convolucionales

Uno de los modelos de aprendizaje profundo más populares son las redes neuronales convolucionales CNN [53], que se utilizan principalmente en tareas que involucran imágenes.

Presentadas por LeCun *et al* [26] a principios de la década de los noventa, las redes convolucionales son un ejemplo de una arquitectura de red neuronal especializada, la cual incluye conocimiento sobre la invarianza de formas bidimensionales utilizando patrones de conexión local y con restricciones en los pesos [54]. Están inspiradas en la organización de la corteza visual de los animales que parte de los estudios de Hubel y Wiesel [55], en la cual la respuesta de cada neurona se expresa matemáticamente como una convolución. Cada neurona en la corteza visual responde a estímulos externos en una cierta región espacial llamada campo receptivo. Los campos receptivos se sobrepone con sus adyacentes formando

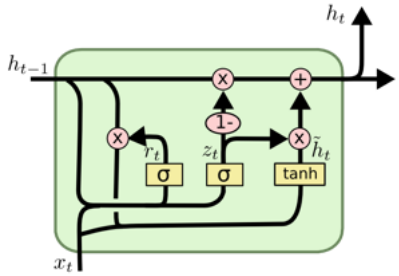


Figura 2.7: Arquitectura de una célula de memoria de largo plazo [49].

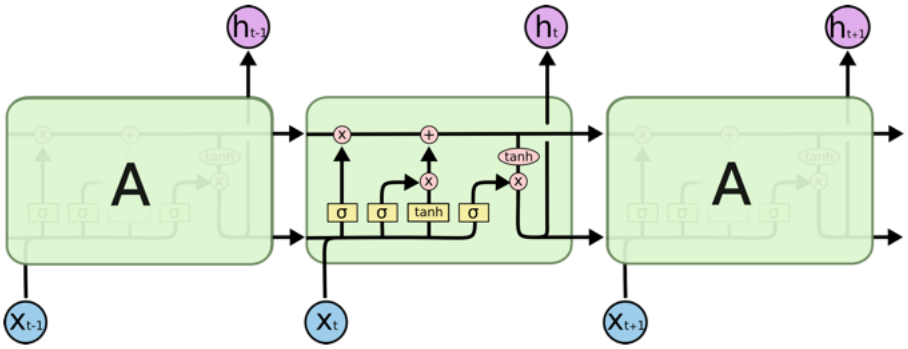


Figura 2.8: Arquitectura de una red de memoria de largo plazo [49].

una retícula del campo de visión [56].

Las redes neuronales convolucionales presentan una arquitectura multicapa, donde cada capa está constituida por un número determinado de convoluciones con su respectiva función de activación no lineal, las dos principales funciones utilizadas en este tipo de red son la $ReLU$ o *tanh* [57]. Se han presentado muchas variaciones en la arquitectura de la CNN como es el caso presentado en [58], donde presenta una CNN en el dominio de la frecuencia.

Las CNN se utilizan para el procesamiento y la clasificación de imágenes [59, 60, 61], que es el área donde se han presentado el mayor rendimiento y resultados, principalmente para imágenes médicas [62, 63]. También existen aplicaciones en otras áreas como la predicción

de series de tiempo [64, 65, 66, 67]. En [68], la serie de tiempo se forma en el modelo autorregulado, mejorando algo el rendimiento. Para el modelado de sistemas tenemos [69] donde presenta un modelo basado en redes neuronales para la estimación de casos de COVID-19 en los siguientes meses de su análisis. En [70], la identificación del sistema no lineal se transfiere al modelado de series de tiempo utilizando el modelo $NARX$. En [71], CNN es entrenada para modelar incertidumbres en el sistema dinámico. Estas incertidumbres se superan con la CNN debido a las propiedades del peso de las acciones y la conectividad dispersa.

2.2.1. Propiedades

La principal característica de este tipo de algoritmo de aprendizaje profundo es la operación de convolución, la cual se ocupa en las primeras capas de la red y su principal uso es la obtención de propiedades de los datos de entrada. Tomando de referencia la corteza visual, tenemos dos tipos de células [72]: las células simples tiene una respuesta máxima a patrones específicos dentro de su campo receptivo; y las células complejas que son localmente invariantes a la posición exacta de patrones en su campo receptivo debido a que presentan campos receptivos de mayor tamaño. De aquí obtenemos dos propiedades más de la red neuronal convolucional.

Convolución

Una convolución es un operador matemático que trabaja con dos argumentos (funciones), f y g, y las transforma en una tercera. La convolución en tiempo continuo se define como la integral del producto de dos funciones, una de ellas desplazada una distancia t [73]. Sean las funciones f y g, su convolución está denotada como:

$$f \otimes g = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2.16}$$

El intervalo de integración depende del dominio sobre el cual están definidas las funciones. Para el caso discreto, lo comúnmente utilizado con las redes neuronales convolucionales, se tiene que la integral se intercambia por la suma, dando como resultado:

$$f \otimes g = \sum_{-\infty}^{\infty} f(k)g(n-k) \tag{2.17}$$

donde una de las funciones está desplazada una distancia n, en ambos casos $\otimes$ representa la operación de convolución. Las principales propiedades de la convolución son:

- **Conmutatividad**

$$f \otimes g = g \otimes f \tag{2.18}$$

- **Asociatividad**

$$f \otimes (g \otimes h) = (f \otimes g) \otimes h \tag{2.19}$$

- **Distributividad**

$$f \otimes (g + h) = f \otimes g + f \otimes h \tag{2.20}$$

- **Asociatividad multiplicado por un escalar**

$$a(f \otimes g) = (af) \otimes g = f \otimes (ag) \tag{2.21}$$

- **Regla de la derivación**

$$D(f \otimes g) = Df \otimes g + f \otimes Dg \tag{2.22}$$

  Donde Df denota la derivada de f, y en el caso discreto el operador de diferencia.

- **Convolución con delta de Dirac**

$$\begin{gathered} f(t \otimes \delta(t) = f(t) \\ f(t) \otimes \delta(t - t_0) = f(t - t_0) \\ f(t - t_1) \otimes \delta(t - t_0) = f(t - t_0 - t_1) \end{gathered} \tag{2.23}$$

Conectividad dispersa

Las redes neuronales convolucionales aprovechan las correlaciones espacialmente locales reforzando los patrones de conectividad local entre neuronas de capas adyacentes. Refiriéndose a la Figura 2.9 para una neurona o unidad de la capa convolucional $m+1$, sus entradas son un subconjunto de las neuronas de la capa m, en este caso una neurona de la capa $m+1$ tiene un campo receptivo de 3. Cada neurona del subconjunto correspondiente en la capa m tiene a su vez un campo receptivo de 3, los cuales son adyacentes.

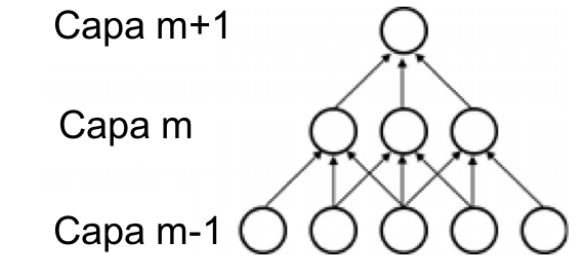


Figura 2.9: Conectividad dispersa.

Dado que cada neurona tiene un campo receptivo de tamaño fijo, no toman en cuenta las variaciones fuera de este, produciendo un tipo de filtrado con su campo receptivo. De este hecho, es que los *hiper-parámetros* de las capas convolucionales son conocidos como filtros convolucionales [74].

Pesos compartidos

La arquitectura de las redes neuronales convolucionales asegura que los filtros produzcan la respuesta más fuerte a un patrón de entrada localmente espacial [75]. En este tipo de red neuronal, cada filtro se utiliza a lo largo de todo el campo visual de la capa. Estas unidades comparten la misma parametrización, es decir pesos y sesgos, formando un mapa de características.

En la Fig 2.10 se muestran en la capa m tres unidades que pertenecen al mismo mapa de

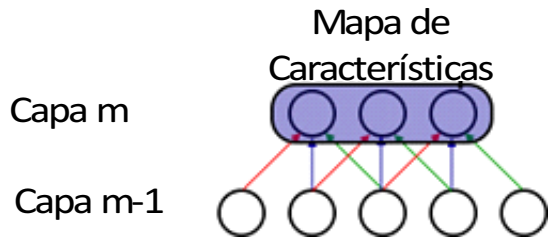


Figura 2.10: Mapa de características.

características. Las líneas del mismo color indican que el mismo peso sináptico es utilizado. Utilizar estas unidades a lo largo del campo de visión permite identificar características sin importar su posición en él. Además, el empleo de pesos compartidos mejora la eficiencia de aprendizaje reduciendo el número de parámetros libres para ser aprendidos [76]. También reduce la brecha entre el error en la etapa de entrenamiento y en la etapa de generalización [77] y la cantidad de memoria requerida para hacer funcionar la red neuronal. Una capa Convolucional completa está constituida por una gran cantidad de mapas de características, los cuales detectan diferentes propiedades de manera local a lo largo de todo el campo de visión.

2.2.2. Arquitectura

La CNN consiste de capas de convolución y de submuestreo en cascada conectadas a una o varias capas completamente conectadas al final de la estructura.

Capa convolucional

Es el elemento principal de las redes neuronales convolucionales. Los *hiper-parámetros* de esta capa consisten en un conjunto de filtros convolucionales $k^{(\ell)}$, los cuales tiene un campo receptivo muy pequeño. En el paso hacia delante de aprendizaje, cada filtro se convoluciona con todo el campo de visión $y^{(\ell-1)}$, produciendo un mapa de características $y^{(\ell)}$. Cada ele-

mento del mapa de características puede interpretarse como la salida de una neurona que extrae información de una pequeña región de la entrada y comparte parámetros con neuronas que se encuentran en el mismo mapa. En general, el $j-$ésimo mapa de características de una capa está dado por [76]:

$$y_j^{(\ell)} = f\left(\sum_{i \in M_j} y_i^{(\ell-1)} k_{ji}^{(\ell)} + b_j\right) \tag{2.24}$$

donde M_j representa el conjunto de entradas seleccionadas y el superíndice ℓ representa la capa actual. La variable representa cada elemento de los filtros convolucionales en esta capa el mapa. Cada mapeo de salida tiene un sesgo diferente. La convolución proporciona facilidad para trabajar con entradas de tamaño variable.

La convolución transpuesta puede verse como el paso hacia atrás correspondiente a la convolución, también conocida como deconvolución [78].

Capa de submuestreo

Una parte importante de las redes neuronales son las capas de submuestreo, son utilizadas después de las capas convolucionales. La función principal de estas capas es reducir el tamaño de la entrada, aunque realizar esta reducción de tamaño conlleva perdidas de información, pero también trae beneficios a la red como la reducción de carga de cálculo en capas posteriores, además de reducir el sobreajuste de la red. Se dice que una red está sobre ajustada cuando tiene un desempeño muy bueno en la etapa de entrenamiento, pero en la etapa de generalización no, debido a que el modelo solo memoriza los datos de prueba y no puede generalizar las reglas para predecir datos futuros [27].

La salida de esta capa tiene la forma general:

$$y_j^{\ell} = f(\beta_j down(y_j^{(\ell-1)}) + b_j) \tag{2.25}$$

donde $down(\cdot)$ representa una función de submuestreo. La operación más común para realizar esta reducción es *max-pool* [79], que divide la entrada en bloques, y de cada uno de estos bloques conserva el elemento de mayor valor. En la Figura 2.11 se muestra la operación de

max-pool reduciendo de una matriz de 4×4 a una matriz de 2×2 con los valores máximos en cada subregión creada en la matriz de entrada.

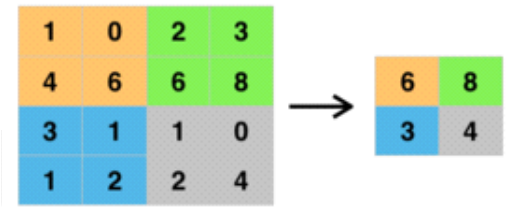


Figura 2.11: Operación *max-pool*.

Lo más común es utilizar filtros de dimensión 2×2 con un paso de submuestreo de 2 unidades.

Algunas otras técnicas utilizadas en la capa de submuestreo se muestran a continuación:

- **Agrupamiento estocástico.** Es un método inspirado en deserción. En lugar de elegir el máximo valor dentro de cada región de agrupamiento, el agrupamiento estocástico elige al azar la activación de acuerdo a una distribución multinomial, la cual asegura que las activaciones no maximales del mapa de características sean posibles de utilizar.

- **Agrupamiento** L_p. Está inspirado en el proceso de agrupamiento biológico en las células complejas [80]. Teóricamente el agrupamiento L_p da una mejor generalización que el *max-pool*. Puede representarse como:

$$y_{i,j,k} = \left[\sum_{(m,n) \in \mathcal{R}_{i,j}} (a_{m,n,k})^p \right]^{\frac{1}{p}} \tag{2.26}$$

  donde $y_{i,jk}$ es la salida del operador en la posición (i, j) en el k-ésimo mapa de características y $a_{m,n,k}$ es el valor de la característica en la posición (m, n) dentro de la región de agrupación $\mathcal{R}_{i,j}$ del k-ésimo mapa de características.

- **Agrupamiento multi-escala sin orden.** Es utilizado para mejorar la invarianza de las redes neuronales convolucionales sin perder su habilidad discriminativa [81]. Se extraen características profundas de activación para la imagen total y para los parches locales a diferentes escalas. La activación de la imagen completa se realiza como en otras redes convolucionales.

La combinación entre capas convolucionales y de submuestreo, está inspirado en el trabajo de Hubel y Wiesel sobre la noción de dos tipos de células en la corteza visual [82].

Capa completamente conectada

Al final de las combinaciones de las capas convolucionales y de submuestreo, se utilizan capas completamente conectadas donde cada elemento corresponde a una neurona y el número total de neuronas en estas capas corresponderá al número de clases que se desean predecir. La última capa por lo general es utilizada para tareas de clasificación. El operador softmax es el más usado. Otro método utilizado es el SVM, que combinado con las características de la red neuronal convolucional puede resolver diferentes tareas de clasificación [83].

Para el caso de modelado de sistemas, solo se requiere una neurona en la capa de salida y no es necesario que se incluya una función de activación en esta última capa, dado que las funciones de activación, como la tanh o $ReLU$, están acotadas y no podrían representar señales fuera de su imagen. Si se desea utilizar una función de activación en la última capa es conveniente normalizar la señal a modelar.

Funciones de activación

Algunas de las funciones de activación mas usadas en CNN se describen a continuación.

- *Sigmoide*

  Definida como:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.27}$$

Actualmente esta función ya no es muy utilizada, debido a los problemas que causa, como lo son la saturación de la salida y la desaparición de los gradientes en la retro propagación, provocando un mal entrenamiento. Otra razón es que no está centrada en cero produciendo una dinámica de zigzag en el aprendizaje.

- *Tangente hiperbólica*

  Definida como:

$$f(x) = \tanh x \tag{2.28}$$

  Esta función da resultados mejores que la función sigmoide porque está centrada en cero. Aunque sigue presentando el mismo problemas de la saturación de la salida.

- *ReLU*

  Unidad lineal rectificada ($ReLU$), se define como:

$$f(x) = \max(0, x) \tag{2.29}$$

  Es muy utilizada en redes profundas ya que acelera el aprendizaje por un factor de 6 [83] comparado con las funciones anteriores y evita el problema del desvanecimiento del gradiente cuando hay muchas capas. La parte negativa de esta función, es que durante el proceso de aprendizaje muchas neuronas pueden dejar de funcionar. Esto se da cuando el gradiente mueve los pesos de tal manera que la neurona no se activa, entonces el gradiente será siempre cero para esa neurona y no se actualizará.

2.3. Algoritmos de aprendizaje profundo para el modelado de sistemas no lineales

Consideremos un sistema discreto desconocido descrito por la ecuación:

$$\bar{x}(k+1) = f\left[\bar{x}\left(k\right), u\left(k\right)\right], \quad y(k) = g\left[\bar{x}\left(k\right)\right] \tag{2.30}$$

donde $u(k)$ es el vector de entrada, $\bar{x}(k)$ es el vector de estados internos del sistema y $y(k)$ es el vector de salida. f y g son funciones no lineales suaves, $f, g \in C^{\infty}$.

Denotemos ahora los conjuntos de entrada y salida

$$Y(k) = [y(k), y(k+1), \cdots y(k+n-1)]$$
$$U(k) = [u(k), u(k+1), \cdots u(k+n-2)]$$

Si $\frac{\partial Y}{\partial \bar{x}}$ es no singular en $\bar{x} = 0$ y $U = 0$, esto nos lleva al siguiente modelo exógeno autoregresivo no lineal ($NARX$):

$$y(k) = \Upsilon[\varpi(k)] \tag{2.31}$$

donde $\Upsilon(\cdot)$ representa la ecuación no lineal en diferencias correspondiente a la dinámica del sistema, con:

$$\varpi(k) = [y(k-1), \cdots y(k-m_y), u(k), \cdots u(k-m_u)]^T \tag{2.32}$$

$\varpi(k) = [\varpi_1 \cdots \varpi_l]^T$, $l = m_y + m_u + 1$, $k = 1 \cdots N$, N es el total de datos para este sistema en particular.

Para la identificación del sistema no lineal (2.31), dos tipos de modelos pueden ser utilizados:

- Modelo paralelo.

  Está definido como

$$\begin{aligned} &\hat{y}(k) = NN[u(k), \cdots, u(k-n_u)] \\ &\text{ó} \\ &\hat{y}(k) = NN[\hat{y}(k-1), \cdots, \hat{y}(k-n_y), \\ &u(k), \cdots, u(k-n_u)] \end{aligned} \tag{2.33}$$

  donde $\hat{y}(k)$ es la salida del modelo, $NN[\cdot]$ es la estructura del modelo a utilizar. n_y y n_u son los órdenes de regresión para la salida y la entrada respectivamente. Esta representación es conocida como modelo paralelo [84]. La principal característica es que la salida real del modelo, $y(k-1)$, en instantes anteriores no se incluye como entrada, en cambio la salida de la red es la que puede o no incluirse.

- Modelo serie-paralelo.

  Se representa como:

$$\begin{aligned}\hat{y}(k) = NN[&y(k-1), \cdots, y(k-n_y),\\ &u(k), \cdots, u(k-n_u)]\end{aligned} \tag{2.34}$$

  En este caso la regresión de datos es idéntica que el modelo $NARX$ (2.32), con órdenes de regresión distintos $n_u \neq m_u$, $n_y \neq m_y$. Este tipo de modelo es el más utilizado, ya que se cuenta tanto con datos de entrada como de salida de los sistemas a modelar.

Uno de los problemas que tiene el modelo paralelo es que no puede modelar sistemas no lineales complejos tales como sistemas con ruido en los datos, o cuando hay pérdida de ellos, esto es debido que no contamos con suficiente información utilizando únicamente los datos de entrada del sistema. Al no conocer la información de salida, aún cuando el sistema no presente alteraciones, los modelos tienden a diverger [85], este tipo de modelos es utilizado para realizar predicciones, las cuales en general no son acertadas.

En ambos modelos de sistema, se utiliza la variable $\hat{\varpi}$ para denotar la entrada a las redes neuronales utilizadas, por ejemplo para el modelo serie-paralelo tenemos

$$\hat{\varpi}(k) = [y(k-1), \cdots, y(k-n_y), u(k), \cdots, u(k-n_u)] \tag{2.35}$$

Este modelos presentados será utilizados en las secciones de simulaciones para cada arquitectura propuesta en este trabajo.

En general la aplicación de estos algoritmos se realiza en sistemas dinámicos no lineales discretos determinísticos, debido a que se trabaja con datos de sensores los cuales tienen un tiempo de muestreo preestablecido generando únicamente valores discretos del sistema, los sistemas en tiempo discreto son representados por medio de ecuaciones en diferencias, que en la mayoría de ocasiones se obtienen a partir de una aproximación de ecuaciones diferenciales que modelen al sistema. Son sistemas determinísticos debido a que en un futuro se busca emplear técnicas de control para modificar su comportamiento, de manera que necesitamos que el sistema genere la misma señal ante el mismo estímulo. Por último, los algoritmos

como las redes neuronales son empleadas principalmente en sistemas en lazo abierto, ya que lo que se busca con estos algoritmos es poder aprender el comportamiento del sistema para posteriormente ser utilizado para otros fines como lo son el control.

Capítulo 3

CNN para la identificación de sistemas: Dominio del tiempo

En este capítulo se describe la aplicación de las redes neuronales convolucionales para el modelado de sistemas dinámicos no lineales utilizando la operación de convolución en el dominio del tiempo con valores reales. Se presentan simulaciones numéricas con diferentes combinaciones de *hiper-parámetros*, cantidad de capas y cantidad de filtros convolucionales por capa, de tal manera que se determine la mejor estructura de la red para cada situación. En este caso, la longitud de los filtros se mantendrá fija. Lo que se busca es encontrar una estructura de red lo suficientemente pequeña para satisfacer los criterios del modelado.

Además, se presenta un estudio de aplicación de la CNN en la identificación del cambio de la posición de la estructura respecto a la velocidad, el cual es utilizado para detectar fallas mediante un método de energía. Este método requiere tanto de la velocidad como el el cambio en la posición debido a la histéresis, este último siendo difícil de determinar dado que se requiere conocimiento de los parámetros físicos de la estructura, los cuales la mayoría de veces no son conocidos.

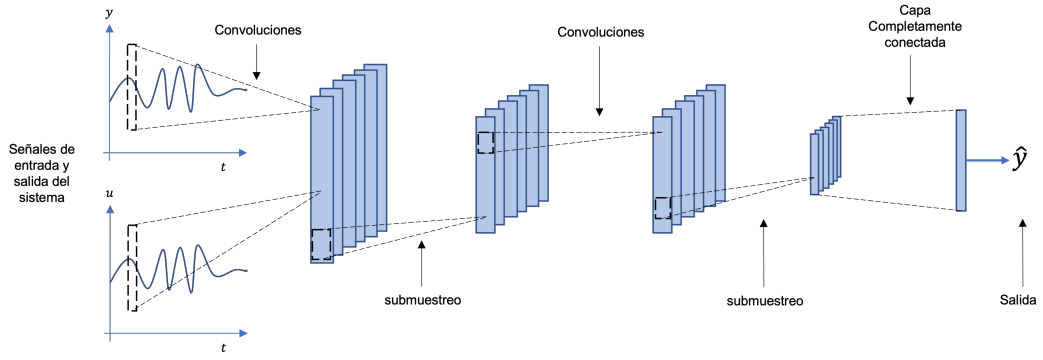


Figura 3.1: Arquitectura de un Red Neuronal Convolucional para el modelado de sistemas.

La arquitectura o estructura general de una red convolucional se muestra en la Figura 3.1 utilizada para la identificación de sistemas. Cuenta con dos capas de convolución y dos capas de submuestreo intercaladas, seguidas de dos capas completamente conectadas con una única salida.

3.1. Arquitectura en el dominio del tiempo

Considere la arquitectura de la red neuronal convolucional en el dominio del tiempo con valores reales ($CNNt$) también denominada como red neuronal convolucional profunda ($DCNN$) propuesta en la Figura 3.2 construida mediante bloques para identificar el sistema mostrado en la expresión (2.31). Consta de $\frac{n}{2}$ bloques convolucionales conectados en cascada y al final conectados a una capa completamente conectada. Cada Bloque convolucional consta de dos capas, una capa convolucional con h filtros convolucionales $K_h^{(\ell)}$, seguidos de una función de activación $ReLU$; y una capa de submuestreo que utiliza la operación *max-pool* con una reducción $s_{\ell+1}$, siendo ℓ el indicador de la capa actual. La salida de la capa completamente conectada es la salida de la $CNNt$ y cuenta con pesos sinápticos W. Para Figura

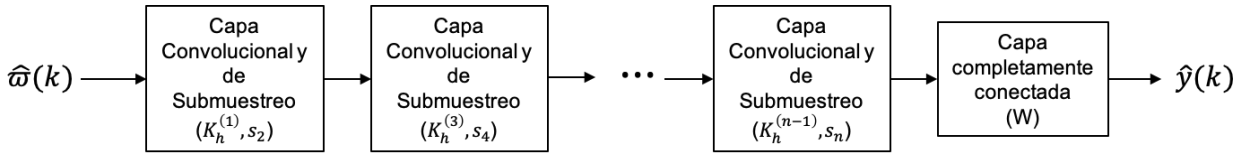


Figura 3.2: Arquitectura profunda de la CNN para el modelado de sistemas no lineales.

3.1.1. Bloque convolucional

Cada bloque convolucional consta de dos capas, una convolucional y otra de submuestreo. La primera capa realiza la operación de convolución entre la salida del bloque anterior y los filtros convolucionales de esta capa, para el bloque inicial, la convolución de los filtros se realiza con el vector definido en (2.35). Matemáticamente se expresa como

$$\chi_h^{(\ell)} = K_h^{(\ell)} \otimes y_h^{(\ell-1)} \tag{3.1}$$

$K_h^{(\ell)}$ son los h-filtros en la capa actual ℓ con dimension $K_h^{(\ell)} \in \Re^{f_l}$, $y_h^{(\ell-1)}$ es la salida del bloque anterior y χ es uno de los h mapas de características generado a partir de realizar la convolución, y su longitud está definido por $p_\ell = (p_{\ell-1} - f_\ell + 2P)/S \ + 1$. La operación de convolución se realiza con un paso $S = 1$ y un relleno de ceros $P = 1$ donde $p_{\ell-1}$ es la dimensión del vector de entrada $y_h^{(\ell-1)}$ y la dimensión de los filtros $f_\ell = 3$.

Una vez obtenida los mapas de características, se procede a aplicar la función de activación $ReLU$. La salida de la capa de convolución, $y_h^{(\ell)}$ se define como:

$$y_h^{(\ell)} = max(0, \chi_h^{(\ell)}) \tag{3.2}$$

La dimensión de estos nuevos vectores siguen siendo la misma que la de los mapas de características, ya que la operación $ReLU$ se aplica individualmente a cada elemento de los mapas de características.

Una vez aplicada la función de activación, una capa de submuestreo es utilizada. La finalidad de esta capa es reducir el tamaño de los datos para así poder evitar problemas de sobre ajuste de la red. Se propone utilizar la operación de *max-pool*, con la cual solo la

respuesta mas grande de los filtros es la que se seguirá propagando a través de la CNN. La salida de esta capa se define como:

$$y_h^{(\ell)} = max\text{-}pool\left(y_h^{(\ell-1)}, s_\ell\right) \tag{3.3}$$

s_ℓ representa la cantidad de elementos a agruparse dentro del vector para que se elija el de valor más alto. La dimensión del vector de salida esta definida como $p_\ell = \frac{p_{\ell-1}}{s_\ell}$.

3.1.2. Capa completamente conectada

Una vez terminados los bloques convolucionales, la estructura se termina con una capa completamente conectada, para ello, las h-salidas del último bloque convolucional, se apilan en el vector ϑ

$$\vartheta = \left[y_1^{(n)T}\ y_2^{(n)T}\ \cdots ; y_h^{(n)T}\right]^T \tag{3.4}$$

la dimension de este vector corresponde a multiplicar las salidas por la longitud de cada una, i.e. $\vartheta \in \Re^{p_{n+1}}$ donde $p_{n+1} = p_n * h$.

Una vez creado este nuevo vector, la salida de la $CNNt$ se expresa mediante la siguiente ecuación

$$\hat{y}(k) = W^{\mathrm{T}}\vartheta \tag{3.5}$$

donde W es un vector de pesos sinápticos de dimension $W \in \Re^{p_{n+1}}$, lo cual produce una salida escalar. Como se puede observar en esta última capa, ninguna función de activación es aplicada, dado que para el modelado de sistema la salida puede encontrarse fuera del codominio de las funciones activación.

3.1.3. Entrenamiento de la $CNNt$

Entrenamiento empleando retro propagación

El aprendizaje o entrenamiento de los *hiper-parámetros* de la $CNNt$ se realiza mediante el algoritmo de retro propagación (BP) como se observa en la Figura 3.3.

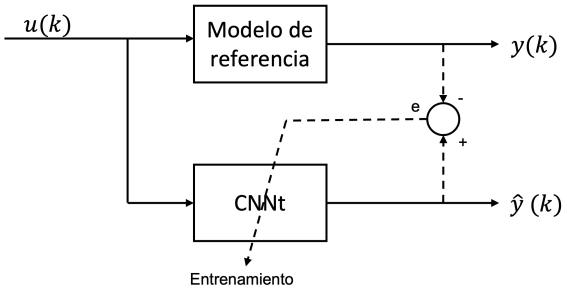


Figura 3.3: Proceso de entrenamiento.

Este algoritmo calcula el gradiente de una función de costo propagado hacia atrás a través de la $CNNt$ respecto a cada *hiper-parámetro* a actualizar, para la $CNNt$ únicamente se actualizan son los filtros convolucionales y los pesos sinápticos de la capa de salida, el cual se utiliza para actualizarlos. Lo primero es definir una función de costo la cual es una medida del desempeño de la $CNNt$, en el caso de modelado e identificación de sistemas, la mas utilizada es la del error medio cuadrático, definida como:

$$J(k) = \frac{1}{2}e^2(k) = \hat{y}(k) - y(k) \tag{3.6}$$

Para calcular el gradiente se utiliza la regla de la cadena, y los *hiper-parámetros* se actualizan según la regla delta. El algoritmo empieza por la capa de salida actualizando los pesos sinápticos W de la siguiente manera:

$$w_i(k+1) = w_i(k) - \eta^{(\ell)} \frac{\partial J}{\partial w_i} \tag{3.7}$$

donde w_i es cada uno de los elementos del vector W, $\eta^{(\ell)}$ es la tasa de aprendizaje y esta se define para cada capa, para garantizar que estos pesos sinápticos están acotados, la convergencia para este algoritmo se presenta en [86] en el caso de tasas de entrenamiento constantes y pequeñas. La derivada parcial de la función de costo respecto del sináptico queda como:

$$\frac{\partial J}{\partial w_i} = \frac{\partial J}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial w_i} = e\vartheta_i \tag{3.8}$$

donde ϑ_i es el elemento del vector ϑ correspondiente a al peso w_i. Finalmente, la actualización de lo pesos sinápticos de la capa de salida es:

$$w_i(k+1) = w_i(k) - \eta^{(\ell)} e \vartheta_i \tag{3.9}$$

El gradiente hacia los bloques convolucionales se calcula, utilizando la regla de la cadena, como:

$$\frac{\partial J}{\partial \vartheta} = \frac{\partial J}{\partial e}\frac{\partial e}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial \vartheta} = eW \tag{3.10}$$

El gradiente a través de los bloques convolucionales pasa primero por la capa de submuestreo. En esta capa, la operación inversa del *max-pool* es realizada y el gradiente se obtiene mediante:

$$\frac{\partial J}{\partial y^{(\ell-1)}} = up\left(\frac{\partial J}{\partial y^{(\ell)}}\right) \tag{3.11}$$

donde $up(\cdot)$ es un operador que aumente la dimensión de su argumento para igualarlo al de la entrada de la capa de submuestreo actual, y colocando el gradiente en las posiciones en las cuales *max-pool* encontró el valor máximo en la etapa hacia adelante, colocando las demás posiciones en cero.

Para la capa convolucional, el gradiente de la función de costo respecto a cada peso es calculado como sigue:

$$\frac{\partial J}{\partial K_h^{(\ell)}} = y_h^{(\ell-1)} \otimes rot180(\delta_h^{(\ell)}) \tag{3.12}$$

donde

$$\delta_{h,i}^{(\ell)} = \frac{\partial J}{\partial y_{h,i}^{(\ell)}} f^{'}(x_{h,i}^{(\ell)}) \tag{3.13}$$

$f^{'}(\cdot)$ es la derivada de la función $ReLU$. Esta derivada está definida a continuación:

$$f^{'}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otro caso} \end{cases}$$

Para poder actualizar los *hiper-parámetros* de las capas convolucionales, la regla delta es utilizada, por lo tanto, se tiene:

$$K_h^{(\ell)}(k+1) = K_h^{(\ell)}(k) - \eta\left(y_h^{(\ell-1)} \otimes rot180(\delta_h^{(\ell)})\right) \tag{3.14}$$

Finalmente, la retro propagación del gradiente a través de la capa convolucional hacia el bloque anterior está dada por:

$$\frac{\partial J}{\partial y_h^{(\ell-1)}} = \delta_h^{(\ell)} \odot rot180(K_h^{(\ell)}) \tag{3.15}$$

el operador $rot180(\cdot)$ es equivalente a utilizar los valores del argumento de abajo hacia arriba, como se se pusiera boca abajo el argumento. Considere un vector x definido como $x = [x_1 \;\; x_2 \;\; x_3 \;\; x_4]^T$. Aplicando el operador $rot180(\cdot)$ al vector x se tiene que $rot180(x) = [x_4 \;\; x_3 \;\; x_2 \;\; x_1]^T$.

Entrenamiento por mínimos cuadrados

El entrenamiento en esta sección se realiza mediante la pseudo inversa generealizada de Moore-Penrose [87]. Con este método de mínimos cuadrados solamente es necesario actualizar los pesos de la capa de salida al final de la etapa de entrenamiento, se ilustra el proceso en la Figura 3.4. De este algoritmo se tiene que, los pesos sinápticos óptimos de la capa de salida

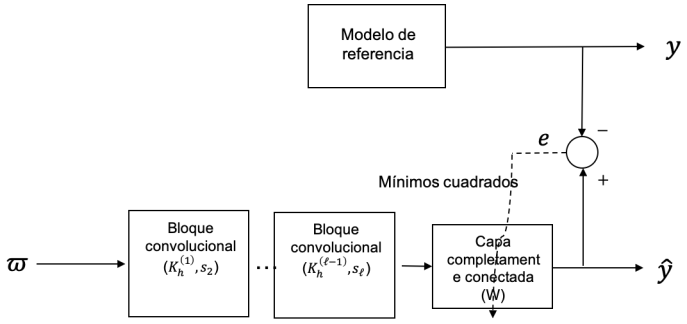


Figura 3.4: Modelo de aprendizaje por mínimos cuadrados.

W^* se obtiene mediante la minimización de la función de costo:

$$J = \sum_k \|y(k) - \hat{y}(k)\|^2 \tag{3.16}$$

y están dados por :

$$W_5^* = Y\Psi^T(\Psi\Psi^T)^{-1} = YXi^+ \tag{3.17}$$

donde Y es vector de las salidas deseadas durante las k_{train} iteraciones de la etapa de entrenamiento:

$$Y = \begin{bmatrix} y(1) & y(2) & \cdots & y(k_{train}) \end{bmatrix} \tag{3.18}$$

y Ξ^+ es la pseudoinversa de Moore-Penrose de la matriz Ξ, definida como:

$$\Xi^+ = \Xi^T(\Xi\Xi^T)^{-1} \tag{3.19}$$

donde

$$\Xi = \begin{bmatrix} \vartheta(1) & \vartheta(2) & \cdots & \vartheta(k_{train}) \end{bmatrix} \tag{3.20}$$

La existencia y unicidad de la matriz Ξ^+ se describen en [88]. Este procedimiento combinado con la inicialización de los *hiper-parámetros* de la red, filtros de las capas convolucionales y pesos sinápticos de la capa de salida, se conoce como algoritmos aleatorios [89].

3.2. Simulaciones CNN en el dominio del tiempo con valores reales.

Lo que se busca con estas simulaciones es encontrar de manera experimental los *hiper-parámetros* libres de la $CNNt$ para satisfacer un criterio de modelado. Se utilizan dos conjuntos de datos de prueba para realizar estas comparaciones, para cada uno de ellos se realizan simulaciones con 3 cantidades diferentes de bloques así como 3 cantidades diferentes de filtros por capa convolucional. En total 9 simulaciones son realizadas por cada conjunto de datos. Par esta sección solo se ocupa el modelo serie-paralelo (2.34), que es el mas utilizado en la literatura. Todas las simulaciones fueron realizadas en MATLAB, en el mismo equipo de cómputo con las siguientes especificaciones: procesador Intel Core i7 de cuatro núcleos a 2,6 GHZ con 16 GB de RAM.

3.2.1. Sistema Wiener-Hammerstein

El sistema Wiener-Hammerstein es una estructura orientada de bloques [90]. Consta de dos funciones lineales invariantes en el tiempo conectadas entre sí por una función no lineal.

Las no linealidades hacen que la identificación del sistema no lineal sea problemático. El conjunto de datos consta de 18800 parejas entrada/salida.

Para las simulaciones se utilizan 13300 parejas para el entrenamiento y las restantes para la generalización. Los valores numéricos para el orden de regresión del vector de entrada a la $CNNt$ definido en (2.35) son $n_y = 16$ y $n_u = 15$. La inicialización de los valores numéricos de los pesos sinápticos de la capa de salida se realiza de manera aleatoria en el rango de $[-1, 1]$, igualmente los filtros convolucionales se inicializan en este mismo intervalo.

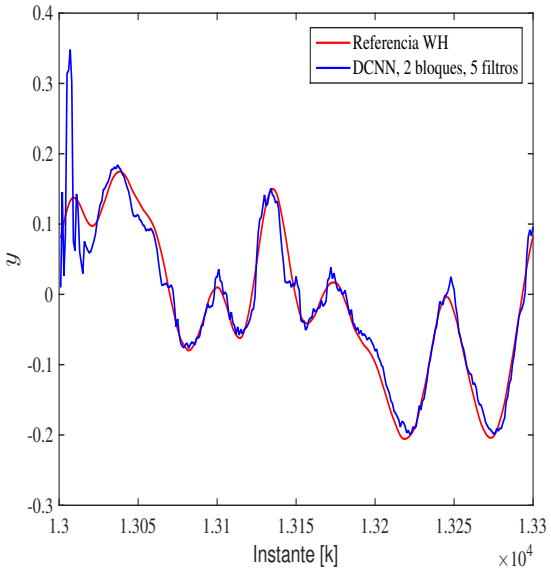
El número de filtros convolucionales para las capas convolucionales son $h = 5, 10, 20$, y el primer conjunto de simulaciones corresponden a una $CNNt$ con 2 bloques convolucionales, i.e. $n = 4$. En cada capa de submuestreo se realiza una operación *max-pool* con $s_2 = s_4 = 2$.

La Figura 3.5a muestra el resultado en la etapa de generalización utilizando los *hiper-parámetros* indicados anteriormente, con $h = 5$. La forma de comparar los resultados de cada $CNNt$ es utilizando el error medio cuadrático (MSE). Para este caso se obtuvo un valor de $5{,}2275 \times 10^{-4}$ con un tiempo de cómputo aproximado de $42s$, el cual es un resultado bastante aceptable. Se prosigue a cambiar el valor de h.

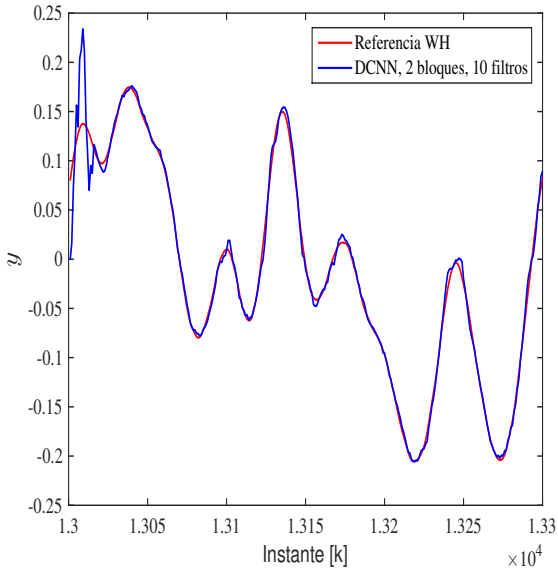
La siguiente simulación corresponde a utilizar 10 filtros por cada capa, esto aumenta el número de *hiper-parámetros* en la red, causando que el tiempo de cómputo llegue a los 49 segundos aproximadamente. En la Figura 3.5b se puede observar que el seguimiento mejoró a comparación de la simulación anterior, obteniendo un MSE de $9{,}825 \times 10^{-5}$.

Finalmente, se realiza las simulaciones ahora con 20 filtros por capa, $h = 20$. Los resultados obtenidos se pueden observar en la Figura 3.5c. El MSE en esta simulación corresponde a $2{,}873 \times 10^{-5}$ con un tiempo de cómputo aproximado de $65s$.

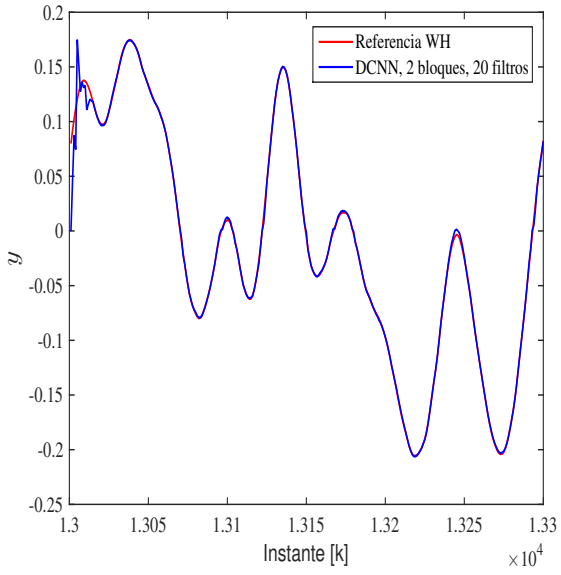
A continuación se presentan los resultados utilizando 3 bloques convolucionales, $n = 6$, similarmente, se realizan 3 simulaciones con esta arquitectura, utilizando 10,20 y 40 filtros en las capas convolucionales. La razón por la cual estos valores son mayores que en las simulaciones anteriores es que para valores inferiores a 10, la $CNNt$ no realizaba un buen modelado del sistema Wiener-Hammerstein. Las Figuras 3.6a, 3.6b, 3.6c muestran los resultados obtenidos para 10, 20 y 40 filtros respectivamente. Los MSE, en el mismo orden, son $2{,}3049 \times 10^{-4}$, $8{,}133 \times 10^{-5}$ y $1{,}0418 \times 10^{-5}$, mientras que los tiempos de cómputo aproximados son: $103s$,



(a) 2 bloques 5 filtros.



(b) 2 bloques 10 filtros.



(c) 2 bloques 20 filtros.

Figura 3.5: Resultados de simulación, sistema Wiener-Hammerstein con dos bloques convolucionales.

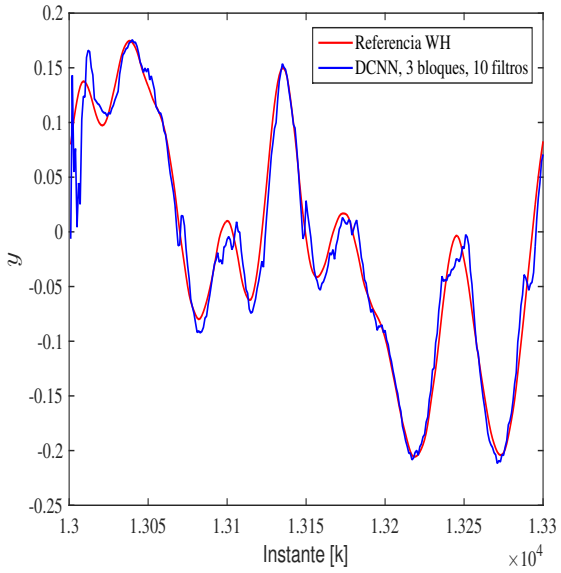
$122s$ y 139.

Se puede observar que utilizar 10 filtros con 3 bloques convolucionales genera un resultado similar a utilizar dos bloques con 5 filtros, pero el tiempo requerido para este caso aumenta considerablemente a 103,1241s. Caso similar ocurre para 3 bloques, 20 filtros y 2 bloques y 10 filtros.

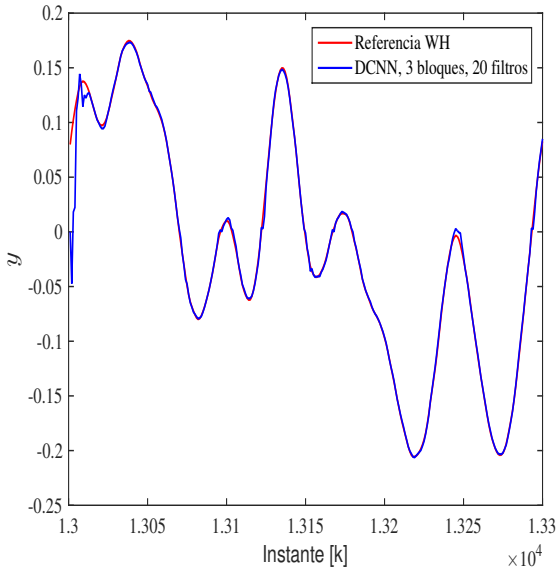
Finalmente se realizan las simulaciones utilizando 4 bloques convolucionales, $n = 8$, y utilizando $h = 40, 60, 100$. Ocurre lo mismo que en las 3 simulaciones anteriores, para valores inferiores de 40 filtros, la $CNNt$ no puede realizar el seguimiento del sistema Wiener-Hammerstein, por eso se opta por aumentar la cantidad de filtros en cada capa. El MSE para cada caso, en orden 40,60, 100 filtros son 0,0016, $2{,}8714 \times 10^{-4}$ y $1{,}1343 \times 10^{-4}$, mientras que los tiempos de computo son de $152s$, $181s$ y $207s$. Los resultados de simulación pueden observarse en las Figuras 3.7a, 3.7b. Se muestran únicamente los resultados para 60 y 100 filtros.

En la Figura 3.7c se muestra los errores de seguimiento de cuatro combinaciones distintas de bloques convolucionales y de filtros para el sistema Wiener-Hammerstein, en esta imagen se puede notar que todas las señales empiezan con valores altos de error debido a que la entrada de la $CNNt$ tiene muchos ceros, como van llegando datos las señales empiezan a acercarse a cero. Las combinaciones de 2 bloques con 20 filtros y 3 bloques con 40 filtros son las que se mantienen mas cerca del cero comparado con las otras dos señales restantes., no es tan fácilhacer una comparación con estas señales, por lo que se decide a utilizar otras formas de medir el desempeño de los algoritmos.

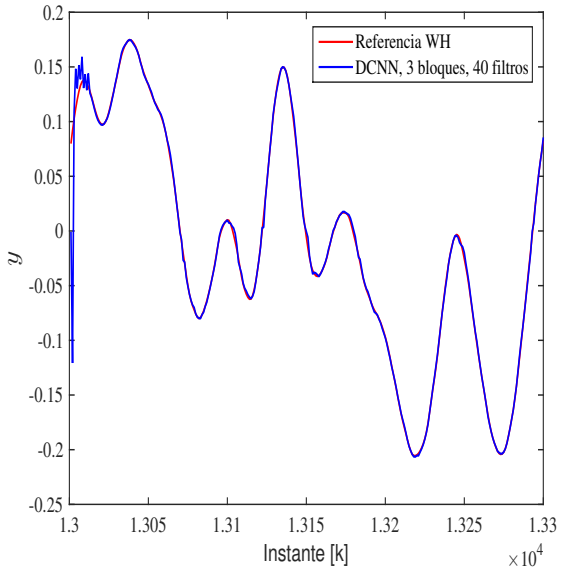
De las 9 pruebas realizadas para el modelado del sistema Wiener-Hammerstein utilizando $CNNt$, se puede concluir que a mayor números de bloques convolucionales, es necesario un mayor número de filtros en cada capa, esto permite mantener el modelado dentro de cierto rango de precisión pero aumenta considerablemente el tiempo de cómputo. La comparación de los errores medios cuadráticos así como del tiempo de cómputo para las 9 simulaciones presentadas para este sistema se muestran en Tabla 3.1. En esta tabla se puede apreciar el error medio cuadrático es muy similares en todas las pruebas.



(a) 3 bloques y 10 filtros.

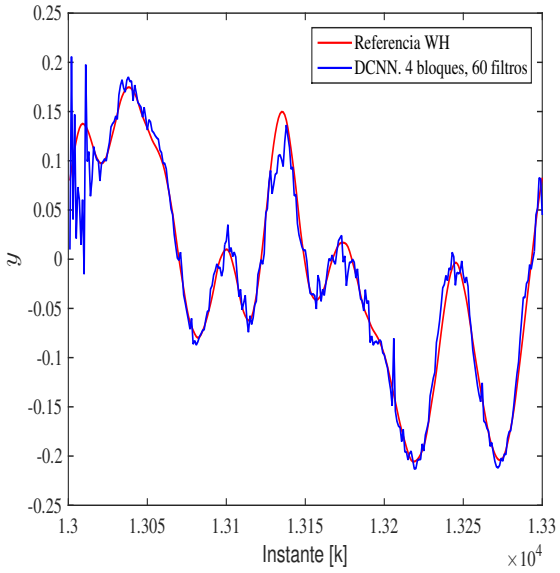


(b) 3 bloques y 20 filtros.

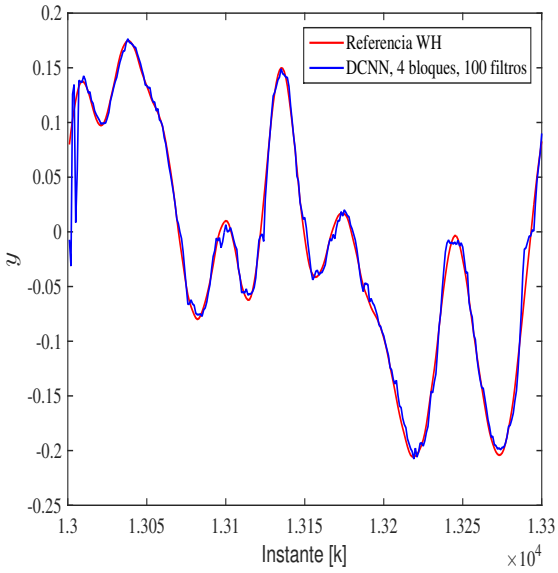


(c) 3 bloques y 40 filtros.

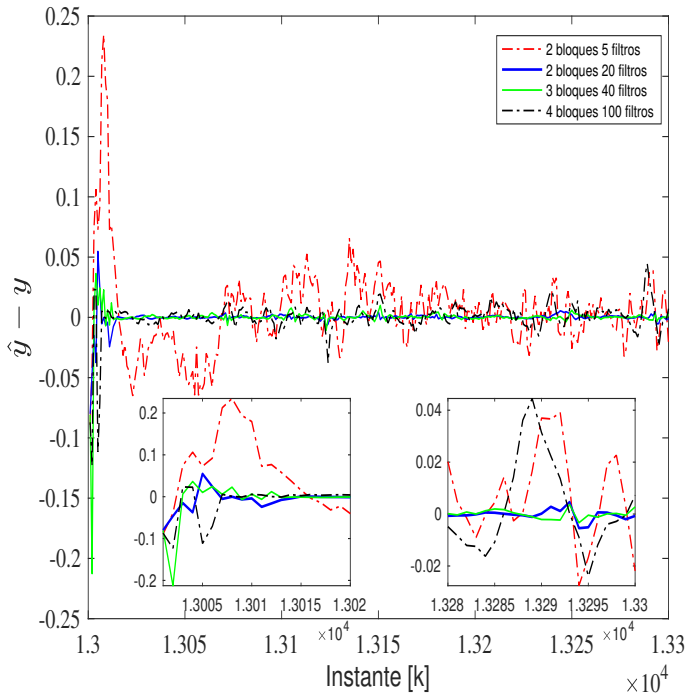
Figura 3.6: Resultados de simulación, sistema Wiener-Hammerstein con tres bloques convolucionales.



(a) 4 bloques y 60 filtros.



(b) 4 bloques y 100 filtros.



(c) Error de seguimiento para el sistema Wiener-Hammerstein.

Figura 3.7: Resultados de simulación, sistema Wiener-Hammerstein con cuatro bloques convolucionales.

Tabla 3.1: comparación de resultados para el sistema Wiener-Hammerstein.

No. filtros	MSE	Tiempo $[s]$
	2 bloques	
5	5.2275×10^{-4}	42.2451
10	9.825×10^{-5}	49.1234
20	2.873×10^{-5}	65.1974
	3 bloques	
10	2.3049×10^{-4}	103.1241
20	8.133×10^{-5}	122.4124
40	1.0418×10^{-5}	139.7942
	4 bloques	
40	0.0016	152.8452
60	2.8714×10^{-4}	181.5168
100	1.1343×10^{-4}	207.3515

3.2.2. Sistema con histéresis

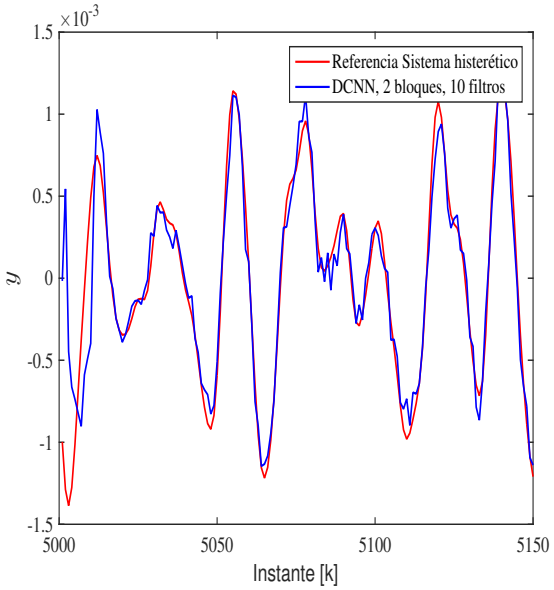
Los datos de prueba son obtenido de [91], el cual consta de un sistema con una no linealidad dinámica. En total se tiene 8192 ejemplares entrada/ salida a una frecuencia de muestreo de 750 Hz. Se utilizan 5000 datos para el entrenamiento y el restante para la generalización, el orden de regresión para este sistema es $n_y = 16$y $n_u = 15$. Los *hiper-parámetros* de la red se inicializan en el intervalo $[-1, 1]$. Al igual que con el sistema anterior, 6 simulaciones diferentes son realizadas, cambiando la cantidad de bloques convolucionales,$n = 4, 6, 8$, y la cantidad de filtros en cada capa cambiará de acuerdo a la cantidad de bloques convolucionales.

El primer conjunto de simulaciones corresponde a dos bloques convolucionales, $n = 4$, utilizando $h = 5, 10, 20$, de los cuales, utilizar 20 filtros por capa genera mejor resultado entre este conjunto. Utilizar únicamente 5 filtros generan un MSE de $1{,}0372 \times 10^{-7}$ con un tiempo de cómputo de $25s$. La Figura 3.8a muestra los resultados de simulación para el caso de 2 bloques y 10 filtros, como se puede apreciar, el seguimiento aún no es adecuado, presenta un MSE de $3{,}8802 \times 10^{-8}$ con un tiempo de $29s$ aproximadamente. El orden del error es de esa magnitud debido a la magnitud de la señal se encuentra en el rango de 10^{-3}.

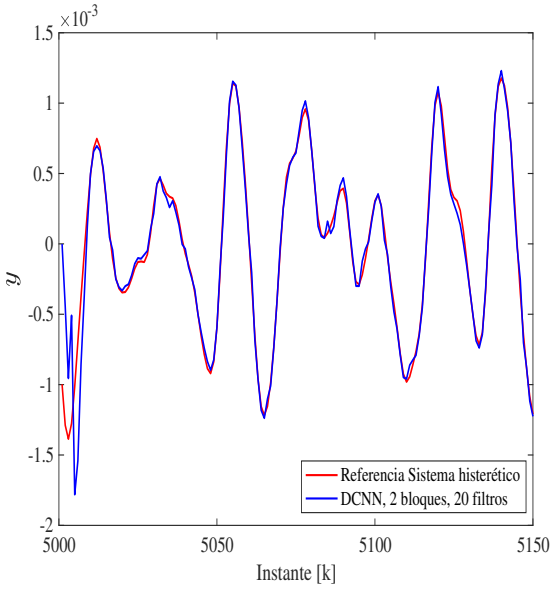
Aumentando la cantidad de filtros en las capas convolucionales a 20, mejora el desempeño de la CNNt, obteniendo un MSE de $1{,}4935 \times 10^{-8}$ con un tiempo de cómputo de $42s$. Se obtienen resultados parecidos que en el sistema anterior, aumentando la cantidad de filtros se disminuye el error de aproximación. Los resultados se pueden observar en la Figura 3.8b.

El siguiente grupo de simulaciones corresponde a utilizar 3 bloques convolucionales con 10, 20 y 40 filtros por capa. La primer simulación corresponde a utilizar 10 filtros, este el número más pequeño de filtros para que la $CNNt$ funcione de manera adecuada, ya que, con menos filtros la red no generaliza la función. Se requiere de $40s$ de entrenamiento para obtener un MSE de $9{,}4496 \times 10^{-8}$. Mientras que, para 20 filtros, se requiere de $50s$ para obtener un MSE de $2{,}0789 \times 10^{-8}$. Por último, para 40 filtros se obtiene un MSE de $1{,}0407 \times 10^{-8}$ con un tiempo de $60s$. Los resultados para 20 y 40 filtros se pueden observar en Figuras 3.8c, 3.8d.

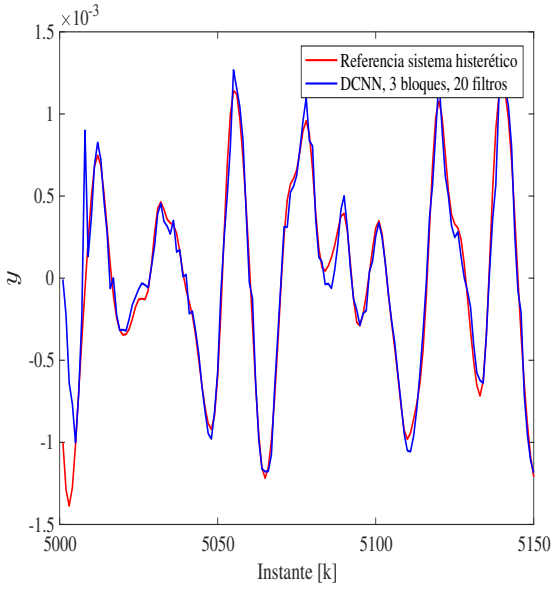
En este grupo se observa de manera similar al anterior que aumentar un bloque convo-



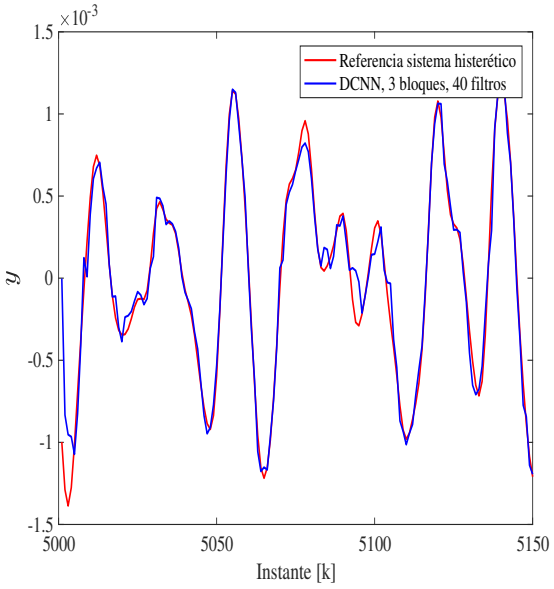
(a) 2 bloques y 10 filtros.



(b) 2 bloques y 20 filtros.



(c) 3 bloques y 20 filtros.



(d) 3 bloques y 40 filtros.

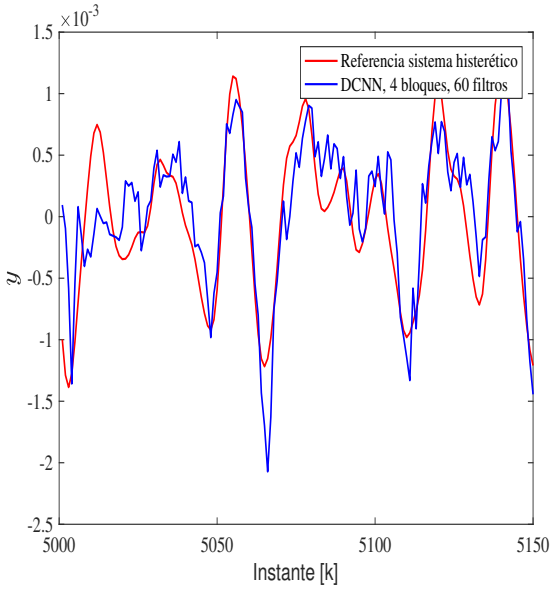
Figura 3.8: Resultados de simulación, Sistema con histéresis con dos y tres bloques convolucionales.

lucional también requiere de un aumento de la cantidad de *hiper-parámetros* de la red así como del tiempo requerido de cómputo.

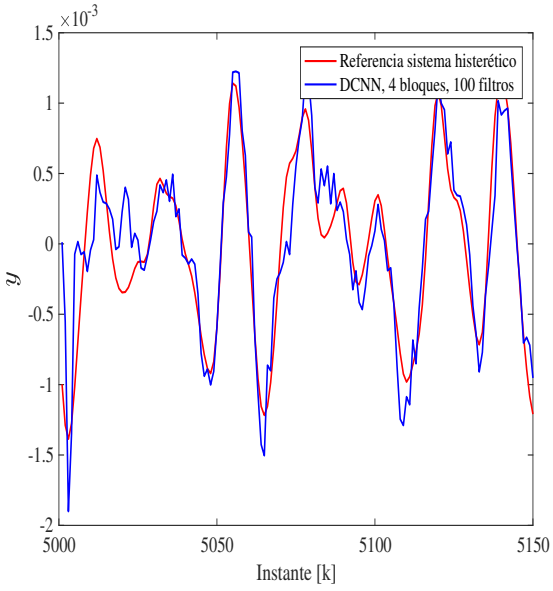
El último grupo de simulaciones corresponden a utilizar 4 bloques convolucionales, teniendo una $CNNt$ de 9 capas en total, 4 capas convolucionales , 4 de submuestreo y una completamente conectada. La cantidad de filtros utilizados para este grupo son $h = 60, 100, 600$. En este grupo, la cantidad mínima para que se realice la generalización dentro de un rango aceptable fue de 50 como se puede observar en la Figura 3.9a.

Para este sistema en particular, por más que se aumente la cantidad de filtros, la $CNNt$ no logra mejorar su desempeño. se muestra el resultado obtenido para 100 filtros en la Figura 3.9b, se realizaron simulaciones utilizando hasta 600 filtros, que es el valor en el cual su desempeño de la CNNt mejora, ver Figura 3.9c. La gráfica de los errores de seguimiento de combinaciones distintas se presenta en la Figura 3.9d, donde se puede notar que las cuatro señales tiene un comportamiento muy parecido entre ellas, se puede notar mejor la diferencia si se compara el índice de desempeño MSE presentado en la Tabla 3.2. La comparación de estos tres grupos de simulaciones para el sistema con histéresis puede encontrarse en la Tabla 3.2. En esta tabla se puede observar que utilizar 2 bloques con 20 filtros convolucionales sería la mejor opción a escoger ya que el valor MSE esta un poco por arriba de la combinación de 3 bloques y 40 filtros, pero es aproximadamente un 70 % mas rápido, esto puede deberse a distintas causas, una de ellas puede ser que debido a que al aumentar la cantidad de *hiper-parámetros* a la estructura podemos provocar el sobre ajuste de los datos lo cual no empeora la etapa de generalización. Otra causa puede ser la misma naturaleza aleatoria de la estructura, como todos los *hiper-parámetros* son inicializados al azar, puede que esa combinación para la combinacion de 2 bloques y 20 filtros haya encontrado un mínimo local más pequeño que el resto de las combinaciones.

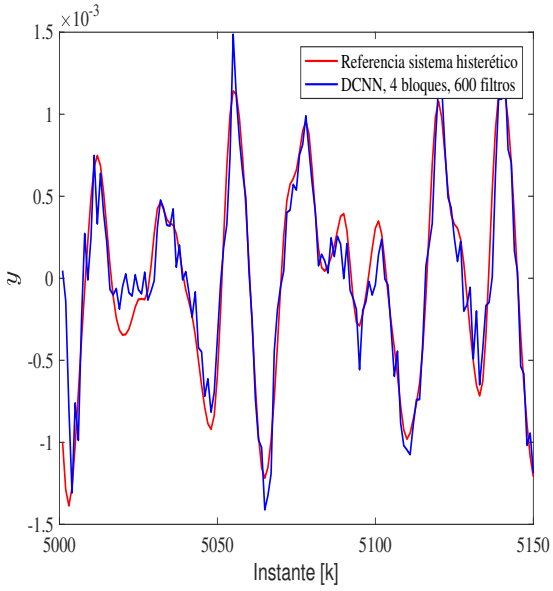
Como se observó en estos dos sistemas anteriores, la arquitectura de $CNNt$ propuesta utilizando únicamente dos bloques convolucionales con 20 filtros por capa puede ser utilizada para el modelado de sistemas no lineales.



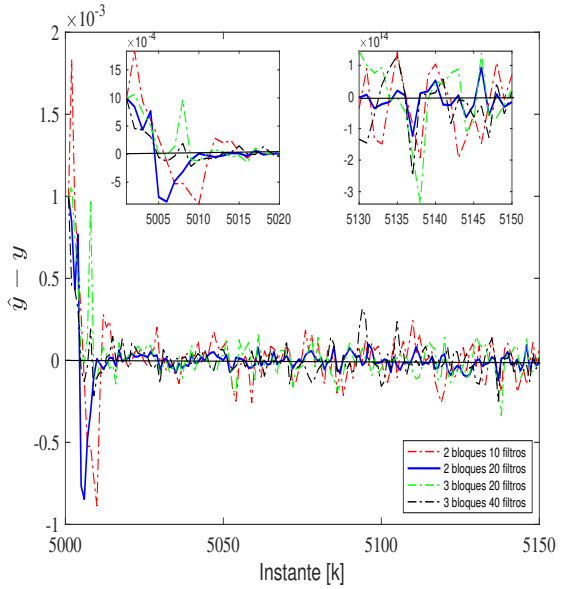
(a) 4 bloques y 60 filtros.



(b) 4 bloques y 100 filtros.



(c) 4 bloques y 600 filtros.



(d) Error de seguimiento para el sistema con histéresis.

Figura 3.9: Resultados de simulación, Sistema con histéresis con cuatro bloques convolucionales.

Tabla 3.2: Comparación de resultados para el sistema con histéresis.

No. filtros	MSE	Tiempo $[s]$
	2 bloques	
5	1.0372×10^{-7}	25.6713
10	3.8802×10^{-8}	29.1682
20	1.8649×10^{-8}	42.5135
	3 bloques	
10	9.4496×10^{-8}	40.8731
20	2.0789×10^{-8}	50.1582
40	1.0407×10^{-8}	60.1519
	4 bloques	
60	8.4877×10^{-8}	69.9346
100	3.6154×10^{-8}	100.7928
600	3.025×10^{-8}	498.2714

3.3. Simulaciones con pesos aleatorios

En esta sección se utilizan 2 ejemplos de prueba para mostrar la eficacia en el modelado de sistemas no lineales de la $CNNt$ con pesos aleatorios. Tres casos se muestran

- Pesos aleatorios sin pre-entrenamiento, inicializados en el intervalo $[0, 1]$.
- Pesos aleatorios sin pre-entrenamiento, inicializados en el intervalo $[-1, 1]$.
- Pesos aleatorios con pre-entrenamiento, inicializados en el intervalo $[-1, 1]$.

Los primeros dos casos consisten en variar el rango de inicialización de los pesos, mientras que en el tercer se pretende mostrar la eficacia de los pesos aleatorio comparando el tiempo requerido para el entrenamiento incluyendo el pre-entrenamiento. Este pre-entrenamiento consiste en utilizar el BP para actualizar todos parámetros de la $CNNt$. Para estas simulaciones también únicamente se utiliza el modelo serie-paralelo (2.34) para alimentar a la CNN.

3.3.1. Sistema no lineal

Este ejemplo de prueba fue propuesto en Narendra y Parthasarathy [84]. El sistema está descrito por:

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + u^3(k) \tag{3.21}$$

donde $u(k)$ es la entrada periódica, la cual se diferencia en la etapa de entrenamiento y en la de generalización, $u(k) = A\sin(\frac{\pi k}{50}) + B\sin(\frac{\pi k}{20})$. En la etapa de entrenamiento $A = B = 1$, mientras que en la etapa de generalización $A = 0{,}9, B = 1{,}1$. Este ejemplo de prueba es muy utilizado para modelos de predicción, donde la salida $y(k-1)$ no está disponible y únicamente las señales de entrada. Para este sistema se propone utilizar $\hat{\varpi} = [u(k) \cdots u(k-5)]$. Se realiza una comparación con un red neuronal multicapa (MLP), la cual consta con dos capas, y la capa oculta tiene 20 neuronas. Esta comparación se puede ver en la Figura 3.10. Se puede observar que la $CNNt$ puede modelar el sistema con mejor precisión que la MLP.

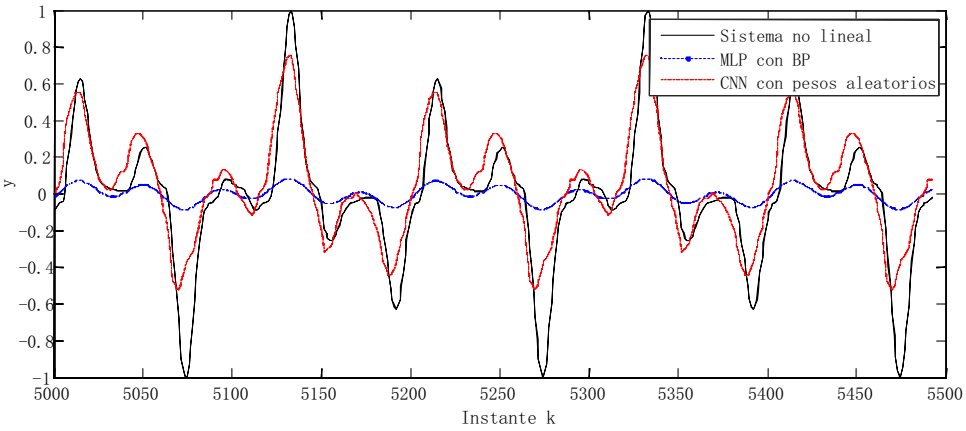


Figura 3.10: Resultados de simulación de modelado del sistema no lineal.

Los errores de generalización se reducen al utilizar el pre-entrenamiento, como se muestra en la Figura 3.11. El error utilizado es el mismo que se ha utilizado en capítulos anteriores, el error medio cuadrático acumulado. Se puede observar que utilizar un pre-entrenamiento mejor el desempeño de la CNN pero no lo suficiente para compensar el tiempo requerido para ello.

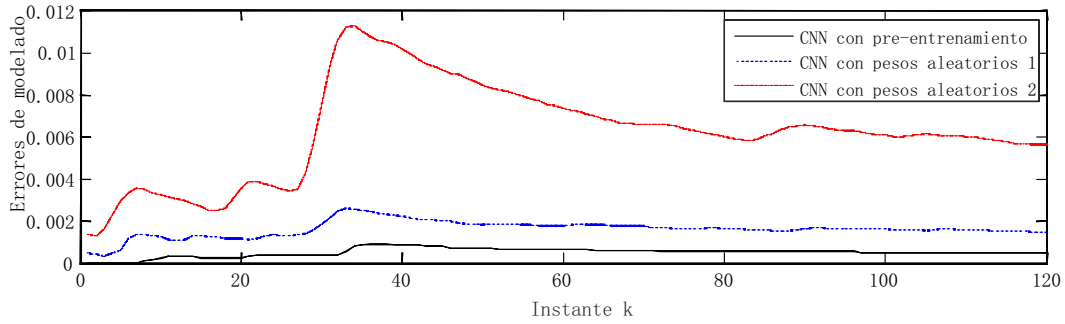


Figura 3.11: Errores de generalización.

Ahora la distribución de los pesos se cambia, de $[0, 1]$ a $[-5, 5]$. La Tabla. 3.3 muestra como la precisión del modelado es afectada al cambiar la distribución de los pesos. Se puede observar que diferentes distribuciones de pesos pueden mejor el desempeño de la $CNNt$. El

rango $[1, 3]$ es el que presenta mejores resultados.

Tabla 3.3: Errores medios cuadráticos para el modelado del sistema no lineal ($\times 10^{-3}$).

Distribución de filtros	Aleatorios	Pre-entrenamiento,
$[-5, 3]$	4.11	5.98
$[-3, 0]$	3.26	3.57
$[0, 1]$	1.42	4.64
$[1, 3]$	1.37	1.02
$[-5, 5]$	3.51	5.95

3.3.2. Horno de gas

El conjunto de datos del horno de gas es otro problema de identificación muy usado [92]. El conjunto consiste de 296 ejemplares muestreados cada 9s. Box et al. [92] utilizan una aproximación por series de tiempo para desarrollar un modelo lineal, Sugeno y Yasukawa [93] y Wang y Langari [94] utilizan estos datos evaluar su modelo difuso. Para estas simulaciones se utiliza el modelo paralelo (2.33) con $\hat{\varpi} = [u(k) \cdots u(k-10)]$. 200 muestras son utilizadas para el entrenamiento y las restantes para la generalización. Los errores de generalización se muestran en la Figura 3.12. Para este ejemplo se puede ver que la CNN tiene un mejor resultado que la MLP. El pre-entrenamiento mejora ligeramente el desempeño de la CNN.

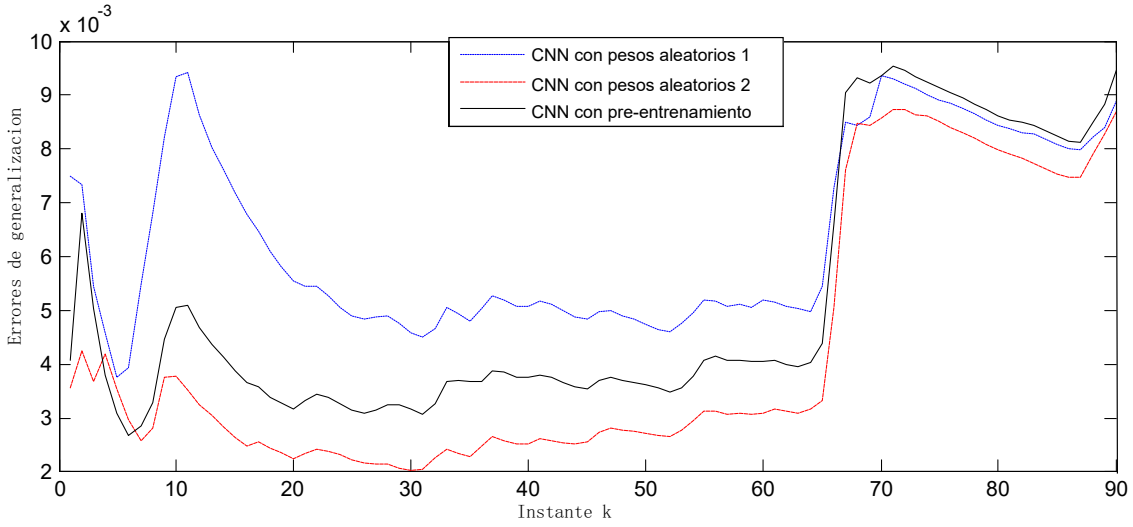


Figura 3.12: Errores de generalización.

De igual forma que para el ejemplo anterior, la distribución de los pesos es cambiada de $[0, 1]$ a $[-5, 5]$. La Tabla. 3.4 muestra los errores de modelado para el horno de gas, para sistema en particular, el rango $[0, 1]$ presenta el mejor resultado.

Tabla 3.4: Errores cuadráticos medios para el modelado del horno de gas($\times 10^{-3}$).

Distribución de filtros	Aleatorios	Pre-entrenamiento,
$[-5, 3]$	27.6	24.1
$[-3, 0]$	12.7	21.1
$[0, 1]$	9.5	8.9
$[1, 3]$	11.7	9.1
$[-5, 5]$	24.3	31.2

3.4. Detección de daño en estructuras de edificios basada en datos de vibración y el modelo de Bouc Wen

Los edificios son estructuras civiles susceptibles a diversos tipos de daños durante su vida útil debido principalmente al continuo deterioro, fatiga, arrastramiento y cargas pesadas inesperadas causadas por fenómenos naturales, como terremotos, fuertes vientos, tifones, entre otros. En este sentido, la tecnología de Monitoreo de Salud Estructural (SHM) es un tema emergente en la ingeniería civil que evita fallas estructurales catastróficas y prolonga la vida útil, mediante evaluaciones continuas y periódicas de la integridad de las infraestructuras civiles. La mayoría de los métodos SHM se basan en el análisis de vibraciones que estudia características globales como frecuencias naturales, formas de modales y curvaturas modales. Se puede encontrar una extensa revisión de estos métodos en [95, 96] mientras que una amplia revisión de los métodos vibracionales con énfasis en aplicaciones de ingeniería estructural se describe en [97, 98]. Del mismo modo, los métodos presentados en [99, 100, 101] también son técnicas para la detección de daños mediante el análisis de los cambios en sus frecuencias de vibración. Dichos cambios se comparan con un ancho de banda en una situación sin daños. Sin embargo, en ambos casos, estos métodos requieren excitar un edificio a altas frecuencias, lo cual no es fácil de lograr y, por lo tanto, el daño puede pasar inadvertido. Además, las influencias ambientales como la lluvia y la temperatura pueden inducir cambios en las frecuencias de vibración incluso si no hay daños [102, 103, 104].

Por otro lado, el daño progresivo inducido en los componentes estructurales podría empujar significativamente la respuesta estructural al régimen no lineal. Por lo tanto, recientemente se han desarrollado enfoques más generales para la identificación de sistemas estructurales no lineales, basados en la equivalencia de entrada-salida. En estudios de identificación basados en un enfoque paramétrico, varios investigadores han adoptado el modelo de Bouc Wen [105, 106, 107]. Sin embargo, estos métodos deben satisfacer la condición de excitación persistente, que es difícil de lograr, de lo contrario, no se puede garantizar la convergencia

paramétrica. Además, la complejidad de los métodos crece de acuerdo con el número de parámetros a identificar, de modo que ajustar los parámetros puede resultar en una tarea difícil. Otros trabajos relacionados son [108, 109] donde se discuten diferentes técnicas de identificación no lineal para la detección de daños.

Entre los esquemas de identificación no paramétricos, el enfoque de red neuronal artificial se ha utilizado recientemente para caracterizar sistemas no lineales de estructura desconocida. Se puede encontrar una revisión exhaustiva de estos métodos en [110]. Bajo este enfoque, se presenta un método para la localización de daños utilizando modelos autorregulados y un conjunto de registros de aceleración en [111]. De manera similar, en [112] se presenta un algoritmo de ubicación de daños, donde las señales de vibración se suponen como la salida de un modelo de promedio móvil autorregresivo (ARMA). La hipótesis es que la reducción de la rigidez induce cambios en los coeficientes de la parte autorregresiva del modelo. Se pueden consultar otros algoritmos similares para la detección de daños en [113]. En [114] se presenta un sistema de detección de daños que utiliza redes neuronales convolucionales de una dimensión que fusionan tanto la extracción de características como los bloques de clasificación en un cuerpo de aprendizaje único y compacto. Las CNN introducidas en [54] se utilizan principalmente para el reconocimiento y clasificación de imágenes.

En esta sección, se desarrollan procesos de identificación y extracción de características del sistema no lineal como una solución alternativa a los métodos de análisis modal para la evaluación de la salud de edificios de varios pisos. Se introduce una CNN para estimar el el parámetro correspondiente al *hysteretic displacement* del modelo de Bouc Wen, analizando los datos provenientes de los sensores de aceleración. La hipótesis principal es que después de dañar el edificio, se reduce su capacidad de disipación de energía. Este hecho puede ser un claro indicador de la presencia de daño. En este sentido, el modelo de Bouc Wen es capaz de capturar, en forma analítica, una gama de formas de ciclos de histéresis que coinciden con la energía del sistema. Por lo tanto, los modelos de histéresis degradantes están relacionados con la pérdida de energía del sistema. Los resultados experimentales de una construcción a escala reducida de cinco pisos confirman que el método propuesto es prometedor para aplicaciones prácticas.

3.4.1. Modelo matemático de una estructura

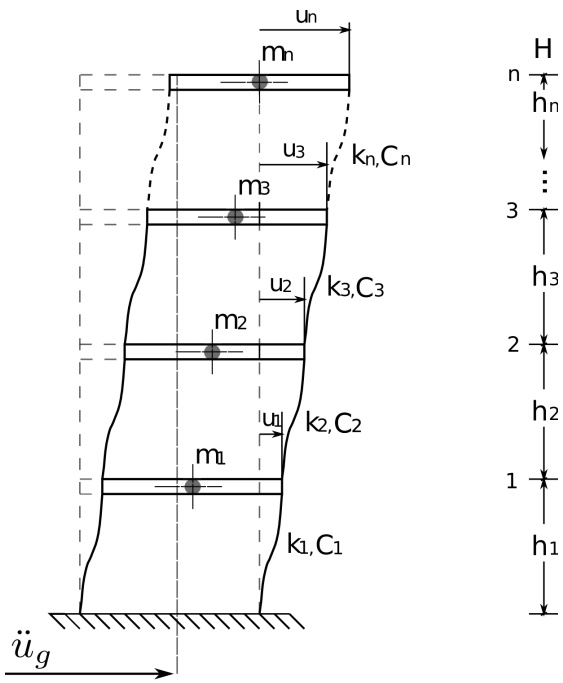


Figura 3.13: Edificio de múltiples grados de libertad

El modelo de estructuras de construcción de grados múltiples de libertad ($MDOF$), ver Figura 3.13 ,excitado sísmicamente está dado por (3.22)

$$M(\ddot{x} + l\ddot{x}_g) + C\dot{x} + Kx = 0 \tag{3.22}$$

donde M, C y K son las matrices de masa, amortiguamiento y de rigidez, respectivamente,

y están definidas como:

$$
\begin{aligned}
M &= \begin{bmatrix} m_1 & 0 & \cdots & 0 \\ 0 & m_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & m_n \end{bmatrix} > 0 \qquad\qquad \in R^{n\times n} \\
C &= \begin{bmatrix} c_1+c_2 & -c_2 & \cdots & 0 \\ -c_2 & c_2+c_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_n \end{bmatrix} \geq 0 \quad \in R^{n\times n} \\
K &= \begin{bmatrix} k_1+k_2 & -k_2 & \cdots & 0 \\ -k_2 & k_2+k_3 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k_n \end{bmatrix} > 0 \quad \in R^{n\times n}
\end{aligned} \tag{3.23}
$$

$$
\begin{aligned}
x = [x_1, x_2, \ldots, x_n]^T, \;\; \dot{x} = [\dot{x}_1, \dot{x}_2, \ldots, \dot{x}_n]^T \;\; \in R^{n\times 1}, \\
\ddot{x} = [\ddot{x}_1, \ddot{x}_2, \ldots, \ddot{x}_n]^T \;\; \in R^{n\times 1}, \;\; l = [1, 1, 1, \ldots, 1] \in R^n
\end{aligned} \tag{3.24}
$$

donde, n es el número de pisos, c_i y k_i con $i = 1, 2, \ldots n$, son respectivamente, el amortiguamiento lateral y la rigidez entre los pisos (i) e $(i-1)$. Las entradas del vector $x,\ \dot{x},\ \ddot{x}$ son el desplazamiento relativo, la velocidad y la aceleración de cada piso, respectivamente, medidos con respecto al sótano, mientras que $\ddot{x}_g$ es la aceleración del suelo inducida por el terremoto. Suponiendo que existe daño estructural en el edificio, el comportamiento no lineal se agrega a la rigidez en (3.22):

$$M(\ddot{x} + l\ddot{x}_g) + C\dot{x} + \rho(x, z) = 0 \tag{3.25}$$

$$\rho(x, z) = \begin{bmatrix} \rho_1(x_1, z_1) & \rho_2(xi, z_2) & \ldots & \rho_i(x_i, z_i) \end{bmatrix}^T \tag{3.26}$$

donde:

$$\rho_i(x_i, z_i) = ak_i x_i - (1-\alpha)Dk_i z_i, \tag{3.27}$$

$$\dot{z}_i = \frac{A_i \dot{x}_i - \nu_i(\beta_i|\dot{x}_i||z|_i^{r-1} z_i - \gamma_i \dot{x}_i |z|_i^r)}{\eta_i} \tag{3.28}$$

Tenga en cuenta que (3.28) representa el parámetro del modelo de Bouc Wen correspondiente al *hysteretic displacement* del edificio de corte no lineal. Desde el modelo de Bouc Wen, A, β y η son parámetros que controlan la forma de los bucles generados [115]. En general, β y γ afectan el tamaño, mientras que $r > 1$ influye en la suavidad del ciclo de histéresis. Además, la degradación de la resistencia y la rigidez se modela mediante el parámetro ν y η, que varían en función de la duración y la amplitud de la respuesta sísmica [116].

En general, la degradación depende de la duración y la gravedad de la respuesta. Una medida conveniente del efecto combinado de duración y gravedad es la energía disipada a través de la histéresis desde el tiempo inicial $t = 0$ hasta el tiempo actual t, como

$$e_i(t) = \int_0^t z_i \dot{u}_i \mathrm{d}u \tag{3.29}$$

Hay que considerar que la función de degradación de la rigidez $\eta_i(e)$ y la función de degradación de la resistencia $\nu_i(e)$, se pueden obtener configurando

$$\eta_i(e) = 1{,}0 + \delta_\eta e_i \tag{3.30}$$

$$\nu_i(e) = 1{,}0 + \delta_\nu e_i \tag{3.31}$$

donde δ_η y δ_ν son parámetros no negativos que se conocen como relación de degradación de rigidez y resistencia, respectivamente. Estas variables, tanto los parámetros como el estado interno z son desconocidos, por lo tanto, ambos deben estimarse. Tradicionalmente, esto se lograría empleando un observador adaptable. Sin embargo, teniendo en cuenta el número de parámetros a identificar, es difícil de lograr. Además, suponiendo ruido de medición, no se puede asegurar la convergencia paramétrica. Por esta razón, la aplicación de la CNN para superar estos problemas es propuesta.

3.4.2. Propiedad de estabilidad del modelo de Bouc Wen

Esta subsección está basada en el trabajo presentado en [117] donde analizan la estabilidad del modelo de Bouc Wen.

Considere el sistema físico con un componente de histéresis que puede ser representado por un mapeo $x(t) \longmapsto \Phi_s(x)(t)$ el cual es conocido como histéresis real. El llamado modelo de Bouc Wen representa la histéresis real en la forma [118] [1]:

$$\Phi_{BM}(x)(t) = \alpha k x(t) + (1-\alpha) D k z(t), \tag{3.32}$$

$$\dot{z} = D^{-1}(Ax - \beta|\dot{x}||z|^{n-1}z - \gamma\dot{x}|\dot{z}|^n) \tag{3.33}$$

donde $\dot{z}$ denota la derivada temporal, $n > 1$, $D > 0$, $k > 0$ y $0 < \alpha < 1$. También, considere que $\beta + \gamma \neq 0$. Se introduce el siguiente conjunto:

$$\begin{aligned}\Omega_{\alpha,k,D,A,\beta,\gamma,n} = \{&z(0) \in R \text{ tal que } \Phi_{BM} \text{ es BIBO estable} \\ &\text{para todo } C^1 \text{ señales de entrada } x(t) \text{ con valores fijos} \\ &\text{the los parámetros } \alpha, k, D, A, \beta, \gamma, n\}\end{aligned} \tag{3.34}$$

Si el conjunto $\Omega_{\alpha,k,D,A,\beta,\gamma,n}$ es vacío, entonces, para cualquier condición inicial $z(0)$ existe una señal acotada $x(t)$ tal que, la salida de histéresis correspondiente Φ_{BM} no está acotada. Esto es, el conjunto de parámetros $\{\alpha, k, D, A, \beta, \gamma, n\}$ no corresponde a un modelo Bouc Wen $BIBO$ estable

Por el contrario, sea $z(0)$ un elemento de $\Omega_{\alpha,k,D,A,\beta,\gamma,n}$. Entonces para cualquier C^1 con entrada acotada $x(t)$, la salida $\Phi_{BM}(x)(t)$ está acotada. Esto quiere decir que el conjunto

$$\begin{aligned}\Omega_{\alpha,\beta,\gamma,n} = \{&z(0) \in R \text{ tal que } z(t) \text{ está acotado para cualquier} \\ &C^1 \text{ señales de entrada acotadas } x(t) \text{ con valores fijos} \\ &\text{de los parámetros } \alpha, \beta, \gamma, n\}\end{aligned} \tag{3.35}$$

es tal que $\Omega_{\alpha,k,D,A,\beta,\gamma,n} \subset \Omega_{\alpha,\beta,\gamma,n}$. La importancia de esta igualdad se deriva del hecho de que es más fácil determinar el conjunto $\Omega_{\alpha,\beta,\gamma,n}$ [117]. Para mayor detalle consultar la Tabla 3.5.

[1] el subíndice (i) se omite por facilidad de lectura

Tabla 3.5: Clasificación de modelos Bouc Wen $BIBO$ estables [117]

Caso	Ω	Cota superior	Clase
$A > 0$, $\beta + \gamma > 0$ y $\beta - \gamma \geq 0$	R	$\max(\lvert z(0)\rvert, z_0)$	I
$\beta - \gamma < 0$ y $\beta \geq 0$	$[-z_1, z_1]$	$\max(\lvert z(0)\rvert, z_0)$	II
$A < 0$, $\beta - \gamma > 0$ y $\beta + \gamma \geq 0$	R	$\max(\lvert z(0)\rvert, z_1)$	III
$\beta + \gamma < 0$ y $\beta \geq 0$	$[-z_0, z_0]$	$\max(\lvert z(0)\rvert, z_1)$	IV
$A = 0$, $\beta + \gamma > 0$ y $\beta - \gamma \geq 0$	R	$\lvert z(0)\rvert$	V
Otros casos	0		

3.4.3. CNN para la identificación del parámetro z del modelo de Bouc Wen

Se propone utilizar una red neuronal convolucional en el dominio del tiempo ($CNNt$) para identificar el parámetro z del modelo de Bouc Wen de una estructura de edificio. Se utiliza la arquitectura propuesta en la sección 3, ver Figura 3.14, utilizando bloques convolucionales conectados en cascada seguidos de una capa complemente conectada para así obtener la señal de salida.

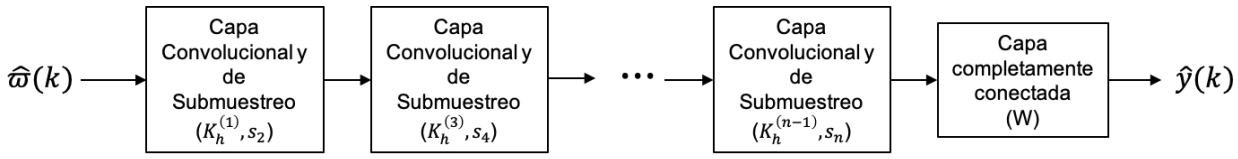


Figura 3.14: Arquitectura CNNt para la identificación del parámetro de histéresis

3.4.4. Resultados experimentales

Los experimentos para evaluar el rendimiento del algoritmo de estimación propuesto se llevaron a cabo utilizando un prototipo de construcción de cinco pisos, que se muestra en la Figura 3.15. El edificio está hecho de aluminio, con la excepción de 3 columnas que están hechas de latón. Las dimensiones de las columnas del primer piso y el resto de los pisos son $(0{,}635 \times 0{,}635 \times 31{,}5)$ cm y $(0{,}635 \times 0{,}635 \times 36)$ cm, respectivamente. Las dimensiones del edificio son de $(60 \times 50 \times 180)$ cm y se monta en una mesa de batido accionada por servomotores de Parker, modelo 406T03LXR. La mesa está equipada con acelerómetros con tecnología MEMS, modelo ADXL203E con un rango de medición de $\pm 1{,}7$ g y banda de ancho $(0 - 50)$ Hz. La adquisición de datos se realizó utilizando dos tarjetas electrónicas de la serie PCI-6221-3 de National Instruments. La comunicación entre estos tableros y Simulink se realizó utilizando Matlab Real-Time Windows Target Toolbox. Los experimentos se llevaron a cabo con un tiempo de muestreo de 0,001 s. Para representar el daño en la estructura la sección transversal de una de las columnas fue reducida en el piso 2 y posteriormente en el piso 5, la reducción fue de 0.635 mm a 0.500 mm mediante el fresado de las mismas. Reducir la sección transversal afecta directamente a la rigidez de la columna en ese piso en particular, lo mismo ocurre cuando se presenta algún daño en la estructura. Debido a este proceso de fresado, toda la adquisición de datos del del sistema sin daños estructural se realizó previo a este proceso., luego se realizaron nuevas mediciones con la columna dañada en el piso 2 y finalmente con ambos columnas dañadas, en los pisos 2 y 5.

La excitación sísmica utilizada en los experimentos es el componente Norte-Sur del terremoto de la Ciudad de México de 1985, registrado en la Secretaría de Comunicaciones y Transportes (SCT). La señal de excitación se escala para que coincida con el prototipo experimental, como se ilustra en la Figura 3.16.

Se realizan seis experimentos, dos cuando no se presenta daño en el edificio, dos cuando se detecta daño en el piso dos y otros dos cuando se detecta daño en los pisos dos y cinco. De cada pareja, se toma un experimento para la etapa de entrenamiento, utilizando los 25000 datos para ello, y el otro se utiliza para la generalización. Para poder desarrollar la detección de daño es necesario tener cierta información previa acerca de las condiciones nominales de la

Figura 3.15: Prototipo a escala de un edificio de 5 pisos.

estructura del edificio. Para esto, el sistema de identificación con $CNNt$ propuesto es usado, donde $\hat{\varpi}$ es un vector formado por las señales de velocidad de la estructura,$\dot{x}$ y la salida de la $CNNt$ en iteraciones pasadas $\hat{z}$.

$$\hat{\varpi}(k) = [\dot{x}(k-1), \cdots, \dot{x}(k-r_1), \hat{z}(k), \cdots, \hat{z}(k-r_2)]^T \tag{3.36}$$

El orden de regresión para cada señal es $n_1 = 10$ and $n_2 = 10$.

Como ya se mencionó, hay dos bloques convolucionales conectados en cascada , cada uno con 10 filtros por cada capa convolucional, $h = 10$. Para las capas de submuestreo, la reducción en cada una de ella es $s_2 = s_4 = 2$. Esta arquitectura permite recuperar la respuesta estructural del edificio como se presenta en la Figura 3.17a, donde la estimación del parámetro z del segundo piso está suficientemente cerca del valor de referencia, con un error cuadrático medio acumulado de $8{,}2572 \times 10^{-4}$. Los valores de referencia son generados analíticamente. Resultados similares se presentan con los pisos restantes. Los valores numéricos del edificio

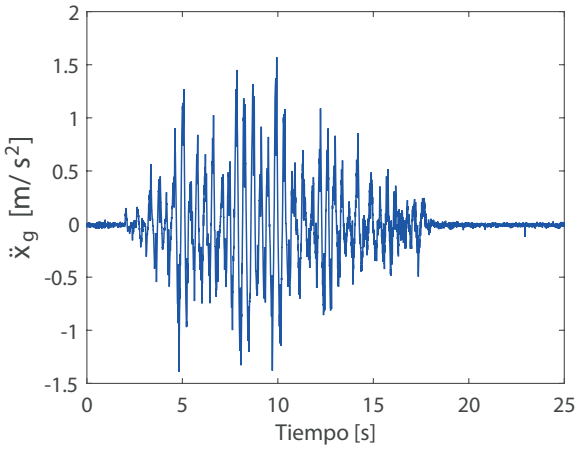
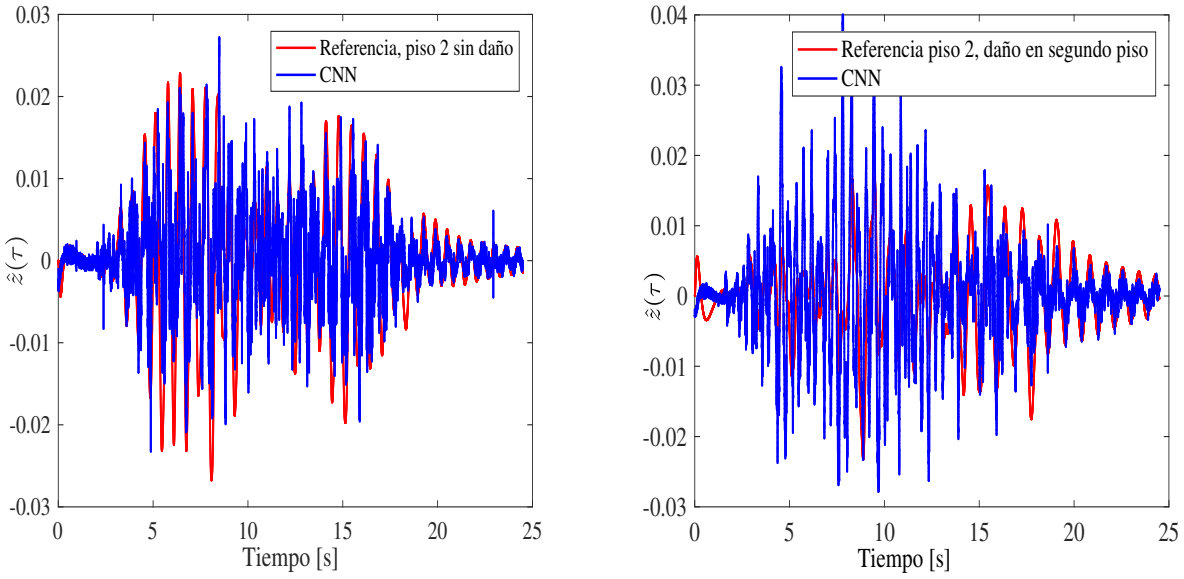


Figura 3.16: Componente Norte-Sur del terremoto de la Ciudad de México de 1985 a escala.

son: $m = [9{,}17, 9{,}17, 9{,}14, 9{,}12, 9{,}08]$ kg, $k = [12011, 12108 \times 0{,}77, 11966, 11850, 19406 \times 0{,}77]$ N/m, donde el amortiguamiento c (Ns/m) se considera como el 5 % del valor de la rigidez en cada piso. Estos valores son utilizados únicamente para obtener la señal de referencia en la etapa de entrenamiento.

Detección de daño en el piso dos, y pisos dos y cinco

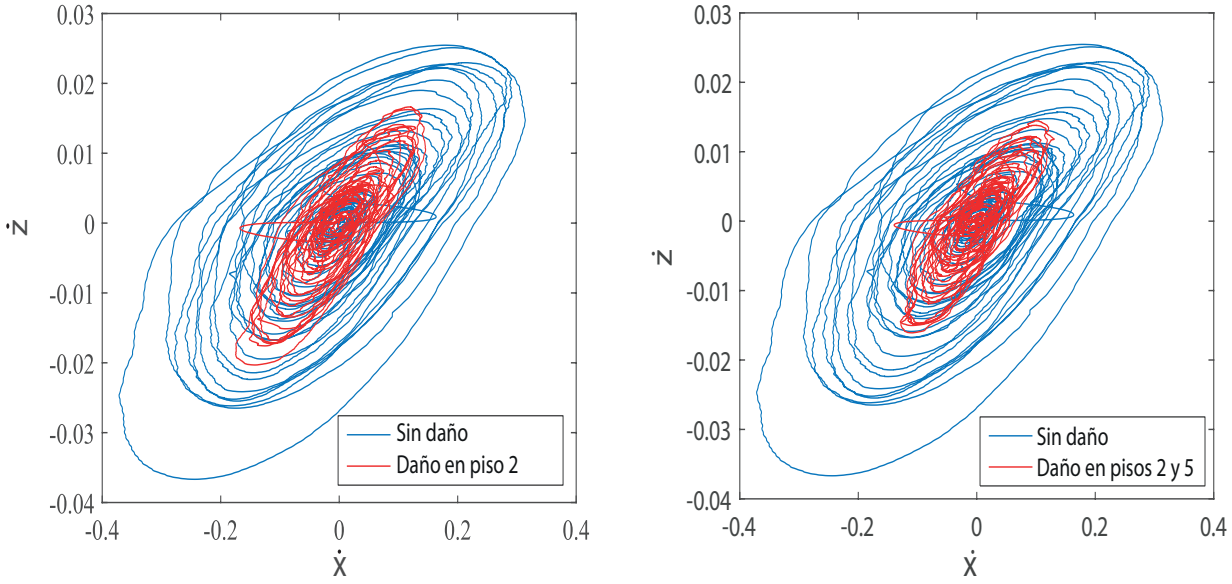
En este apartado se investiga la detección de daños en el segundo piso, en dos escenarios diferentes. El primero de ellos consiste en daño inducido en el segundo piso, mientras que para el segundo caso el daño se extiende en el segundo y quinto piso. En ambos casos, la sección transversal cuadrada de estas columnas se redujo aproximadamente 23 %, cambiando de 6,35 a 5 mm. Para detectar daños, se utilizaron el esquema de identificación propuesto con la $CNNt$ y tres tipos diferentes de datos de vibración. El primero de ellos corresponde a una situación nominal, donde el edificio está libre de daños, otro, donde el piso dos tiene daños, y uno más donde los daños se encuentran en el segundo y quinto piso. Para ilustrar este algoritmo, la salida de la $CNNt$, $\hat{z}$ correspondiente al segundo piso se presenta en la Figura 3.17b con un error medio cuadrático acumulado de 0,0071. Esta figura corresponde al



(a) Piso 2 sin daño.

(b) Piso 2 con daño en el segundo piso.

Figura 3.17: Parámetro z del modelo de Bouc Wen para el segundo piso.



(a) Quinto piso en presencia de daño en el segundo piso.

(b) Quinto piso en presencia de daño en el segundo y quinto piso.

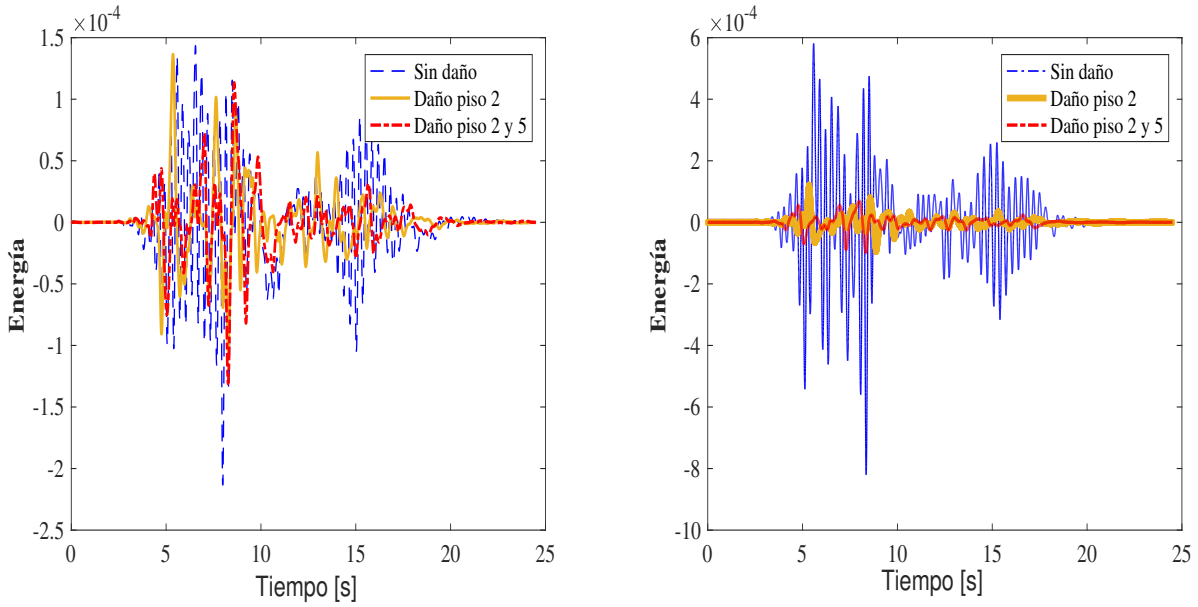
Figura 3.18: Diagrama de histéresis del quinto piso.

segundo escenario, donde el daño solo ocurre en el segundo piso. Esta imagen fue seleccionada dado el caso de estudio. Sin embargo, los resultados de estimación son similares para los pisos restantes.

Además, a partir de la estimación de $\hat{z}$, se calculan los ciclos de histéresis, donde se puede apreciar la pérdida de energía para el edificio dañado con respecto al modelo de no daño. Una comparación entre los ciclos de histéresis con daños y sin daños correspondientes al quinto piso se ilustra en la Figura 3.18a y en la Figura 3.18b.

En ambos casos se observa que en presencia de daño, los ciclos de histéresis reducen su área, concentrándose en el centro del ciclo de histéresis sin daño. Se obtuvieron resultados similares en los pisos restantes.

Una vez obtenidos los resultados de la $CNNt$, se calcula la energía disipada por el edificio



(a) Energía en el segundo piso.

(b) Energía en el quinto piso.

Figura 3.19: Comparativa de energías entre estructura sin daño y con daño.

usando la ecuación (3.29). Estos resultados se pueden ver en las Figuras 3.19a y 3.19b. Es evidente que debido al daño, la capacidad de la estructura para disipar energía se reduce, lo cual indica que la estructura ha cambiado de una zona elástica a una plástica. Adicionalmente, cuando ocurre el daño en otro piso, la energía disminuye, como se muestra en las Figuras 3.19a y 3.19b. Esto es, las magnitudes para las líneas rojas son menores, la cuales corresponden cuando hay daño en el segundo y quinto piso, en comparación con las lineas naranja, las cuales corresponde a daño en el segundo piso. Por lo tanto, la aproximación propuesta basada en la disipación de la energía puede ser una herramienta alternativa útil para la detección de daño, dado que nos permite observar la dinámica de la estructura en términos energéticos.

Los resultados experimentales han confirmado que cuando un edificio tiene daño estructural, en este debido a una reducción de la rigidez de las columnas de la estructura,se reduce su capacidad de disipación de energía, lo que está directamente relacionado con la pérdida de rigidez. En este sentido, el modelo Bouc Wen ha sido una herramienta útil, dado que nos permite realizar una estimación de la energía del sistema mediante el parámetro z del modelo de Bouc Wen de la estructura. Además, los resultados experimentales utilizando la $CNNt$ para estimar el z confirman que el método propuesto es prometedor para aplicaciones prácticas, únicamente necesitando mediciones de posición y de velocidad para calcular la energía sin la necesidad de realizar alguna prueba extra a la estructura.

Capítulo 4

CNN para la identificación de sistemas: Dominio de la frecuencia

En este capítulo las redes neuronales convolucionales son utilizadas para el modelado de sistemas no lineales utilizando la operación de convolución en el dominio de la frecuencia, lo cual conlleva ventajas respecto a la propuesta del capítulo 3 como lo es el tiempo de cómputo y la robustez ante el ruido en los datos.

Dentro del área de modelado de sistemas, un problema muy común son las perturbaciones externas, ya sea ruido de medición o alguna perturbación generada por alguna otra razón. El procesamiento de datos se puede llevar a cabo para reducir su efecto sobre el proceso. También podemos reducir este efecto a través de una red neuronal, por ejemplo para un MLP, aumentar el número de nodos y capas es una solución viable, pero esto implica que tenemos que entrenar un mayor número de *hiper-parámetros* que podrían causar más problemas de modelado. En el caso de las CNN, esta puede reducir el ruido directamente en las capas convolucionales, haciéndolas efectivas en este tipo de casos. Como la CNN se usa con más frecuencia para otras tareas [119], no hay tantas referencias en las que CNN pueda usarse para modelado de sistemas no lineales.

Se realizan comparaciones entre ambas arquitecturas de red, observándose que la CNN en el dominio de la frecuencia disminuye el error de identificación, reduce el tiempo de

cómputo y brinda cierta robustez ante la presencia de ruido en los datos de entrada. Se añade un teorema que utiliza la transformada de Fourier para mostrar que para filtros aleatorios dentro de la CNN, existe una entrada óptima que al interactuar con los filtros se obtiene la máxima respuesta. La utilización de filtros aleatorios durante la etapa de entrenamiento ayudan a disminuir el tiempo requerido obteniendo resultados muy parecidos que cuando estos se van modificando mediante un algoritmo de aprendizaje.

4.1. Transformada Discreta de Fourier

La transformada de Fourier discreta (DFT), denotada por $\mathcal{F}(\cdot)$ y definida en (4.1), es una herramienta poderosa además de ser una transformación lineal.

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-\mathrm{i}2\pi}{N} kn} \tag{4.1}$$

donde x_n y X_k son una secuencia de N números complejos, $k = 1, 2, \cdots, N-1$,$n = 1, 2, \cdots, N-1$. La transformación inversa está dada por:

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k e^{\frac{\mathrm{i}2\pi}{N} kn} \tag{4.2}$$

La DFT tiene muchas aplicaciones en varias ramas del conocimiento. En el análisis espectral de señales, convierte muestras espaciales en una secuencia de muestras de valor complejo en el dominio de la frecuencia para lo cual primero es necesario convertir nuestras muestras de tiempo continuo a discreto, lo que por lo general conlleva varios tipos de distorsión en la señal como el aliasing y leakage [120]. En el procesamiento de señales digitales también es muy utilizado [121], principalmente para la compresión de imágenes y sonido. Otras aplicaciones incluyen la multiplicación de polinomios y para resolver ecuaciones diferenciales parciales [122].

4.1.1. Teorema de convolución

Este teorema indica que la convolución de dos funciones puede ser obtenida mediante la transformación inversa del producto de la transformación de cada una de las dos secuencias. Matemáticamente, sea f, g dos funciones, se tiene:

$$f \otimes g = \mathcal{F}^{-1}\left(\mathcal{F}(f) \odot \mathcal{F}(g)\right) \tag{4.3}$$

donde $\otimes$ es la operación de convolución y $\odot$ es el producto elemento a elemento. Este teorema simplifica esta operación, en cuestión de complejidad computacional, es mas conveniente transformar las señales utilizando la DFT y luego realizar la multiplicación para regresar posteriormente al dominio del tiempo que realizar la operación de convolución en si.

4.1.2. Matriz DFT

Otra forma de ver esta transformación es mediante una matriz. La matriz DFT de n-puntos definida como $A = \mathcal{F}(a)$ tiene la forma siguiente

$$F_n = \frac{1}{\sqrt{n}} \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ 1 & \omega^2 & \omega^3 & \dots & \omega^{2(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)(n-1)} \end{bmatrix} \tag{4.4}$$

donde $\omega = e^{\frac{-2\pi \mathrm{i}}{n}}$

Dos de las propiedades importantes de la matriz DFT son su linealidad y que es unitaria [123]. Su transformada inversa está dada por $\mathcal{F}^{-1}(\cdot) = \mathcal{F}(\cdot)^*$ que es el conjugado de la propia transformación. Esta última propiedad es muy importante para implementar de manera rápida la etapa de entrenamiento en el dominio de la frecuencia, ya que al utilizar el algoritmo BP, es necesario pasar del dominio de la frecuencia a la del tiempo en varias ocasiones.

4.2. Impacto de los pesos aleatorios en la identificación de sistemas no lineales usando redes neuronales convolucionales

Los algoritmos aleatorios fueron presentados inicialmente por Schmidt et al [124] y fueron estudiados a profundidad por Igelnik y Pao [125]. Sus pesos ocultos se eligen al azar y se aplica el enfoque pseudoinversa (o el método de mínimos cuadrados) para calcular los pesos de salida. Muestran que la optimización de los *hiper-parámetros* de la capa oculta no mejora significativamente el comportamiento de generalización, mientras que la actualización de los pesos de salida es más efectiva [126]. La importancia del alcance para la asignación aleatoria de parámetros ocultos se describe en Li y Wang [127]. El modelo neuronal puede ser generado por el método de configuración estocástica, esta red de configuración estocástica [35] necesita menos intervención humana y su aprendizaje es más rápido. Se han aplicado con éxito algoritmos aleatorios a la identificación de sistemas no lineales en Tapson y van Schaik [128]. En de la Rosa y Yu [89] los pesos ocultos aleatorios se ajustan por las características estadísticas de la entrada del sistema a través de máquinas de Boltzmann restringidas. Cheney et al. [129] muestra que la CNN tiene un grado significativo de robustez en la clasificación de imágenes con filtros aleatorios. Para el procesamiento de imágenes, Saxe et al. [130] explican por qué la agrupación de cuadrados en CNN funciona bien con pesos aleatorios.

A continuación se presenta el estudio de la CNN con pesos aleatorios que genera resultados sorprendentes para la identificación de sistemas. Se combinan las propiedades de la CNN y de los algoritmos aleatorios. El comportamiento del modelo propuesto se analiza en el dominio de la frecuencia, con lo cual se llega a la existencia de entradas óptimas (o pesos óptimos) y que la precisión con pesos aleatorios es muy parecida a la de los óptimos.

4.2.1. Análisis en el dominio de la frecuencia

De acuerdo a la arquitectura propuesta en la sección 3.1, en los bloques convolucionales, después de realizar la operación de convolución, una función de activación es aplicada seguida

de una operación *max-pool*. Ahora se considera que no existe una función de activación entre estas dos capas. La operación *max-pool* puede verse como el producto de un vector por una matriz S, donde esta matriz contiene ceros en la mayoría de sus posiciones y en las posiciones en las cuales se encuentra la respuesta mas grande de la entrada se colocan unos. Además, la convolución puede verse como el producto de matrices donde K_m es la matriz de convolución. Este nuevo bloque convolucional queda definido, utilizando estas dos nuevas formas de definir las operaciones, como:

$$y^{(2)} = S\varpi^T K_m \tag{4.5}$$

Se utiliza el conjunto de datos de un horno de gas, el cual es el conjunto de prueba en [92]. El vector de entrada con su orden de regresión correspondiente es de $\hat{\varpi}(k) = [y(k-1)\cdots y(k-4)u(k)\cdots u(k-5)]$, este corresponde al modelo serie-paralelo. Al resultado de la convolución se le aplica la transformada discreta de Fourier y su espectro en frecuencia se presenta en la Figura 4.1, donde varios filtros son aplicados en la entrada. Comparando la figura anterior con la Figura 4.2, se puede notar que ciertos filtros aumentan la magnitud mientras que otros la disminuye, pero ninguno cambia la forma de esta, esto es, las características de $\hat{\varpi}(k)$ pueden extraerse utilizando filtros aleatorios. Esta propiedad no puede encontrarse en el dominio del tiempo.

Desde el punto de vista de la identificación, se espera que la amplitud alcance un valor máximo, para que la propiedad pueda ser extraída con la mayor energía. En el dominio del tiempo, queremos encontrar el mejor filtro posible $K^{(\ell)}$ tal que:

$$\max_{K} [\hat{\varpi} \otimes K] \tag{4.6}$$

Es una tarea complicada. Sin embargo, en el dominio de la frecuencia, (4.6) se convierte en:

$$\max_{K} \left[\hat{\vartheta} s) \odot K(s)\right] \tag{4.7}$$

donde $\hat{\vartheta}(s) = \mathcal{F}(\hat{\varpi})$, $K(s) = \mathcal{F}(K)$. Dado que los datos de entrada son conocidos, es fácil hacer un cambio a:

$$\max_{\hat{\vartheta}} \left[\hat{\vartheta}(s) \odot K(s)\right] \tag{4.8}$$

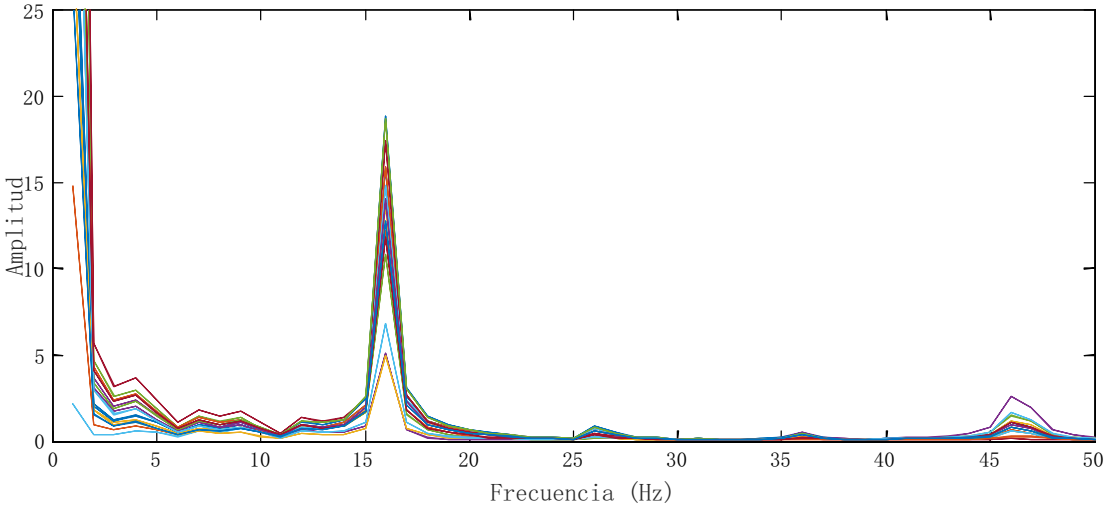


Figura 4.1: El espectro en frecuencia de $\hat{\varpi}(k) \otimes K_k^{(1)}$ con pesos aleatorios.

(4.8) significa que podemos encontrar la entrada óptima en lugar de los pesos óptimos. El siguiente teorema provee la entrada óptima, tal que (4.6) se maximiza.

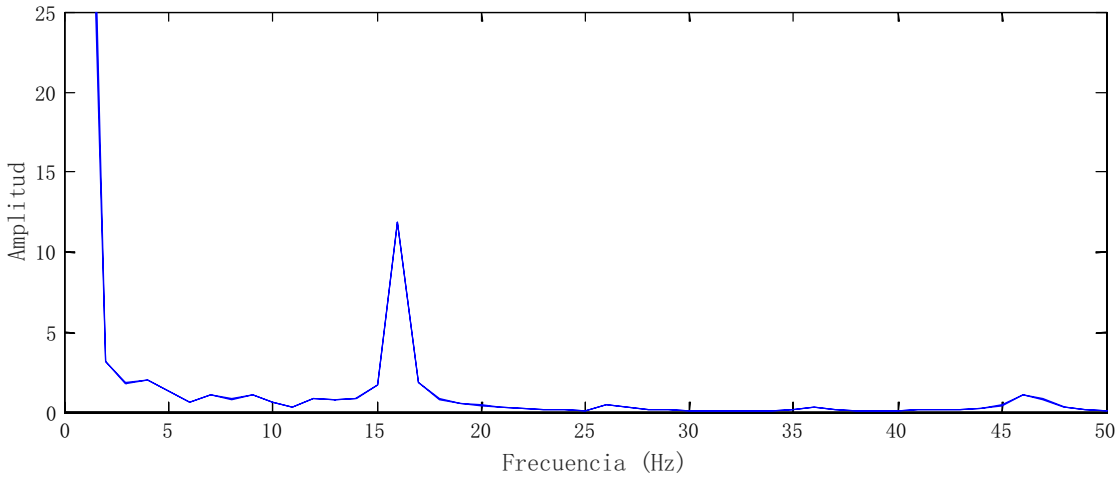


Figura 4.2: El espectro en frecuencia de la entrada $\hat{\varpi}(k)$.

Teorema 4.1 *Los datos de entrada* $\hat{\varpi}(k)$, $k = 1, 2, \cdots N$ *pasan a través de un filtro* $K = [K(1) \cdots K(p)]$, $p < N$. *Existe una entrada óptima que maximiza el bloque convolucional, i.e.* (4.6) *o* (4.8) *utilizando una operación max-pool como:*

$$\hat{\varpi}^* = \frac{\sqrt[2]{2}}{N} \cos\left(\frac{2\pi f_0}{N} + \phi\right) \tag{4.9}$$

donde f_0 *es la frecuencia óptima y* ϕ *es la fase.*

Demostración. Cada bloque convolucional incluye la operación de convolución (4.6) y la operación *max-pool*. El problema de optimización (4.8) es:

$$S \max_{\hat{\varpi}} [K_m \hat{\varpi}] \text{ o } S \max_{X} [K_m X] \tag{4.10}$$

donde X es la normalización de $\hat{\varpi}$, $||X||^2 = 1$, S está definida en (4.5) y K_m como ya se definió anteriormente es una matriz circular, i.e. cada fila es la cambio circular de la fila anterior [131]. EL problema puede ser reescrito como

$$S \max_{X, X \neq 0} \frac{(X^T K_m^T K_m X)^{\frac{1}{2}}}{X^T X} = S \max_{X, X \neq 0} J \tag{4.11}$$

Aquí $X \neq 0$ está bajo el supuesto de que la frecuencia cero puede ser la frecuencia máxima f_0. La solución óptima debe ser el vector propio asociado con el valor máximo de $K_m^T K_m$. Como vimos anteriormente, la transformada de Fourier discreta se puede usar para colocar los valores propios de $K_m^T K_m$ en la matriz diagonal. (4.11) está en una forma cuadrática semi-definida positiva. Las soluciones óptimas son los vectores propios asociados con el valor máximo de $K_m^T K_m$. Usamos la DFT en (4.11) para obtener:

$$J = \frac{(X^T K_m^T K_m X)^{\frac{1}{2}}}{X^T X} = \frac{(X^T F^T F K_m^T K_m F^T F X)^{\frac{1}{2}}}{X^T F^T F X} \tag{4.12}$$

La matriz de la transformada de Fourier F satisface $F^T F = 1$. Definimos $q = FX$ y reescribiendo la ecuación anterior se tiene:

$$J = \frac{(q^T F K_m^T K_m F^T q)^{\frac{1}{2}}}{q^T q} \tag{4.13}$$

Considerando que K_m es una matriz circular, entonces $F K_m F^T = \Lambda$. Finalmente el problema de optimización (4.10) es:

$$S \max_q \frac{(q^T |\Lambda|^2 q)^{\frac{1}{2}}}{q^T q} \quad \text{Sujeto a: } F^T q \in \Re \tag{4.14}$$

La restricción $F^T q \in \Re$ garantiza que $\hat{\varpi} \in \Re$. La matriz Λ es diagonal , $\Lambda_{ii} = \beta |F K_m|_i$, β es un factor de escalamiento. Entonces, los eigenvalores son $\lambda_i = \beta |\Lambda|_{ii}$. Dado que ϑ debe ser real, los coeficientes de la DFT correspondientes a las frecuencias negativas deben ser el complejo conjugado de las positivas, i.e. $q_{-i} = \bar{q_{-i}}$ y $q_{-j} = \bar{q_j}$. Para que se satisfaga esta condición, una posible solución es:

$$q_j = \begin{cases} v^{i \text{sign}(j) b_{|j|}} a_{|j|}, & \lambda_j \in \max(\lambda) \\ 0 & \text{otros casos} \end{cases} \tag{4.15}$$

con a, b vectores de valores arbitrarios, uno de ellos es la frecuencia máxima f_0, $||a|| = 1$, v es el eigenvector de (4.14). La condición de realidad es:

$$a_{|j|} e^{isign(-j)\phi_{|j|}} = \overline{a_{|j|} e^{ising(j)\phi_{|j|}}} = \bar{q_j} \tag{4.16}$$

Para q_j y $\bar{q_j}$ no ceros, la condición de realidad se satisface. Dado que (4.16) alcanza su máximo sin restricciones de realidad, también es la solución para el problema completo. Convirtiendo q al dominio de tiempo, el máximo de $\hat{\varpi}$ es (4.9). ■

4.3. CNN para el modelado de sistemas no lineales en el dominio de la frecuencia

Como se vio en la sección 3.4, la $CNNt$ puede utilizarse para la identificación del parámetro de histéresis z del modelo de Bouc Wen en estructuras de edificios. Una nueva arquitectura de red convolucional en el dominio de la frecuencia ($CNNf$) es propuesta para resolver este problema, la cual presenta ventajas sobre la $CNNt$, como el tiempo de entrenamiento es mucho menor para arquitecturas similares y la $CNNf$ puede modelar el sistema aun en la presencia de ruido de frecuencia mayor a la del sistema a modelar, evitando el uso de filtros para reducirlo y que pueden afectar las propiedades del sistema.

4.3.1. Arquitectura de la $CNNf$

La principal diferencia entre la CNNt propuesta en secciones anteriores y la propuesta ahora, es la aplicación de la DFT a la entrada de la $CNNf$ y a los filtros en capas convolucionales, por lo que las operaciones se vuelven más simples desde el punto de vista computacional. Además de que no es necesario utilizar funciones de activación en toda la arquitectura.

Considere el sistema a identificar representado en (2.31), lo que se busca es utilizar una $CNNf$ para modelar dicho sistema, para ello se propone una red convolucional en el dominio de la frecuencia:

$$\hat{y}_F(k) = V^T \Upsilon \tag{4.17}$$

donde $\hat{y}_F(k)$ es la salida escalar de la $CNNf$, V son los pesos sinápticos y Υ es la salida del último bloque convolucional.

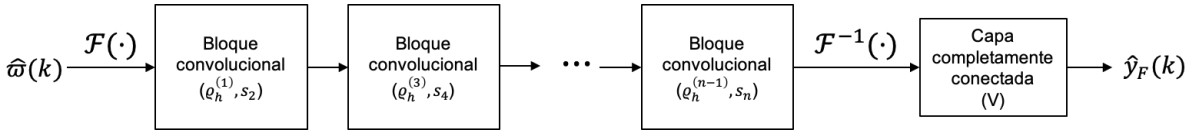


Figura 4.3: Arquitectura de la $CNNf$ para el modelado de sistemas.

La arquitectura se presenta en la Figura 4.3, tiene una entrada $\hat{\varpi}$ definida como:

$$\hat{\varpi}\left(k\right) = \left[y\left(k-1\right), \cdots, y\left(k-n_y\right), u\left(k\right), \cdots, u\left(k-n_u\right)\right]^T \tag{4.18}$$

n_y y n_u son el orden de regresión para cada señal, los cuales son diferentes para cada sistema. Se utiliza el modelo serie-paralelo para definir la arquitectura de la $CNNf$.

Una DFT se aplica a la entrada, de manera que se obtenga su representación en frecuencia de la misma longitud, i.e. $\Phi^{(0)} = \mathcal{F}(\hat{\varpi})$. En esta nueva representación se asume que el componente de frecuencia cero se encuentre en el centro.

Al finalizar los bloques convolucionales, a cada salida se la aplica la transformada inversa de Fourier para obtener su representación en el dominio del tiempo, i.e. $\psi_i^{(\ell)} = \mathcal{F}^{-1}\left(\Psi_i^{(\ell)}\right)$ y estas son apiladas en un vector Υ

$$\Upsilon = \left[\psi_1^{(\ell)T}, \psi_2^{(\ell)T}, \cdots, \psi_{h_2}^{(\ell)T}\right]^T \tag{4.19}$$

4.3.2. Bloque convolucional

El bloque convolucional consta de dos capas, una de convolución y otra de submuestreo. En la primera de ella se define los *hiper-parámetros*, los cuales son los filtros y su dimensión. Por cada capa convolucional hay h_F filtros $\varrho^{(\ell)}$ cuya dimensión es f_ℓ, donde ℓ indica la capa actual. A estos filtros también es necesario aplicarles la DFT, $\Gamma_i^{(\ell)} = \mathcal{F}(\varrho_i^{(\ell)})$ para $i = 1, \ldots, h_F$, su representación en frecuencia debe coincidir con la entrada del bloque. La operación de convolución queda definida como:

$$\Psi_i^{(\ell)} = \Psi_i^{(\ell-1)} \odot \Gamma_i^{(\ell)} \tag{4.20}$$

donde $\odot$ es el producto elemento a elemento de los vectores. La dimensión de $\Psi_i^{(\ell)}$ es la misma que la entrada de este bloque. En la Figura 4.4 se observa la diferencia de la operación de convolución, mientras que en el dominio del tiempo la dimensión de los filtros es menor que en los de la frecuencia, la cantidad de operaciones que se realizan en el tiempo son mayores que en el de la frecuencia haciendo mas rápido el proceso en el dominio de la frecuencia. Además, no se aplica ninguna función de activación reduciendo también el tiempo.

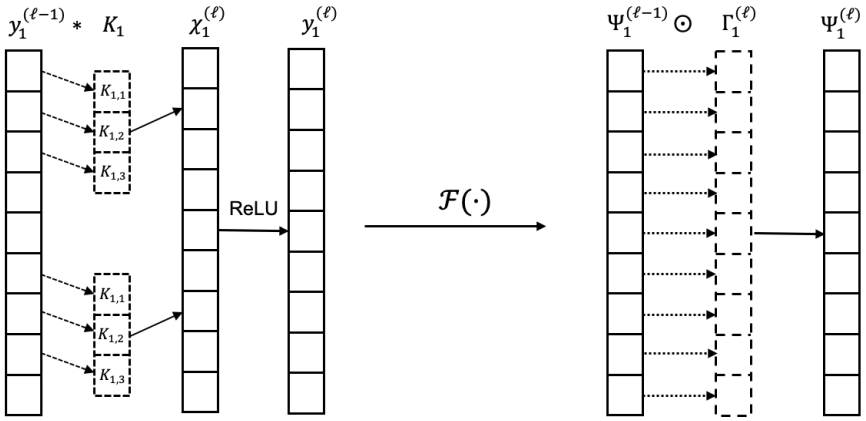


Figura 4.4: Operaciones de convolución.

Para la capa de submuestreo, la operación de agrupamiento espectral es realizada [132]. La idea de está técnica es remover los componentes de alta frecuencia para reducir su representación por un factor s_ℓ. La salida de esta capa está dada por:

$$\Psi_i^{(\ell)} = \text{Shrink}(\Psi_i^{(\ell-1)}, s_\ell) \tag{4.21}$$

Para facilitar esta operación , se diseña la $CNNf$ de tal manera que todas las representaciones en el dominio de la frecuencia sea de dimensión impar.

4.3.3. Entrenamiento de la CNNf

El proceso de aprendizaje se realiza mediante el algoritmo de retro propagación. La función de costo a utilizar es:

$$J(k) = \frac{1}{2}e^2(k) = [\hat{y}_F(k) - y(k)]^2 \tag{4.22}$$

Para la capa completamente conectada, el gradiente de J respecto a los pesos sinápticos V es:

$$V(k+1) = V(k) - \eta\frac{\partial J}{\partial V} = V(k) - \eta e\Upsilon \tag{4.23}$$

donde $\eta_F^{(\ell)}$ es la tasa de aprendizaje y se define una para cada capa de la CNNf.

El gradiente propagado hacia las capas anteriores está definido por:

$$\frac{\partial J}{\partial \Upsilon} = \frac{\partial J}{\partial e}\frac{\partial e}{\partial \hat{y}_F}\frac{\partial \hat{y}_F}{\partial \Upsilon} = eV \tag{4.24}$$

Dado que Υ es el apilamiento de las salidas del bloque convolucional, se tiene que descomponer para cada salida $\psi_1^{(\ell)}$. Una vez separado el gradiente, se utiliza la DFT para cambiar su representación y seguir con el algoritmo de propagación. Para las capas de submuestreo, solo es necesario aumentar la representación en frecuencia para igualar la dimensión de la entrada, i.e.:

$$\frac{\partial J}{\partial \Psi_i^{(\ell-1)}} = up\left(\frac{\partial J}{\partial \Psi_i^{(\ell)}}\right) \tag{4.25}$$

donde $up(\cdot)$ es la operación realizada para aumentar la representación espectral. En esta capa no se necesita actualizar ningún hiper-parámetro y solo se calcula el gradiente a la capa anterior.

Para la capa convolucional, la actualización de los pesos se realiza de la siguiente manera, primero se calcula el gradiente respecto a cada filtro Γ, el cual está dado por:

$$\frac{\partial J}{\partial \Gamma_i^{(\ell)}} = \frac{\partial J}{\partial \Psi^{(\ell)}} \odot \Psi^{(\ell-1)} \tag{4.26}$$

Esto es el producto elemento a elemento entre el gradiente del error de la salida y la entrada del bloque convolucional. Dado que cada elemento del filtro puede ser actualizado de manera

separada, (4.26) puede reescribirse como:

$$\frac{\partial J}{\partial \Gamma_{i,a}^{(\ell)}} = \frac{\partial J}{\partial \Gamma_{i,a}^{(\ell)}} \Psi_{i,a}^{(\ell-1)} \tag{4.27}$$

donde $i = 1, \cdots, h_F$ y $a = 1, \cdots, f_\ell$. La actualización de los filtros se hace mediante la regla delta

$$\Gamma_i^{(\ell)}(k+1) = \Gamma_i^{(\ell)}(k) - \eta \frac{\partial J}{\partial \Gamma_i^{(\ell)}} = V(k) - \eta \frac{\partial J}{\partial \Psi^{(\ell)}} \odot \Psi^{(\ell-1)} \tag{4.28}$$

El gradiente del error propagado a capas a bloques convolucionales anteriores se obtiene con

$$\frac{\partial J}{\partial \Psi^{(\ell-1)}} = \frac{\partial J}{\partial \Psi^{(\ell)}} \odot \Gamma^{(\ell)} \tag{4.29}$$

el cual también es el producto elemento a elemento entre los filtros y el gradiente de la capa actual.

4.3.4. Resultados experimentales

Para probar la nueva arquitectura $CNNf$, se propone identificar el parámetro de histéresis z del modelo de Bouc Wen de un prototipo de una estructura de edificio de dos pisos, la cual se encuentra en el laboratorio del departamento de control automático del CINVESTAV-IPN. Este prototipo de dos pisos está construido de aluminio con dimensiones $(32{,}5 \times 53) cm$ y una altura de 1,2 m. Todas las columnas tiene un sección transversal rectangular de $(0{,}635 \times 2{,}54)\, cm$ con 58 cm de altura el primer piso y 62 cm el siguiente, ver Figura 4.5. El edificio está montado sobre una mesa accionada por servomotores de Quanser, modelo l-40. Durante los experimentos, la estructura se excita con el terremoto de Northridge, que se ajusta en amplitud para estar de acuerdo con la estructura y se muestra en la Figura 4.6. El edificio está equipado con acelerómetros Analog Devices modelo XL403A, con un rango de medición de 1 a 15 g y banda de ancho $[1 \times 800]\, Hz$, para medir las respuestas en cada piso y en la base. La adquisición de datos se realizó mediante el uso de placas electrónicas de la serie RT-DAC / USB2 de Inteco. Los programas de adquisición fueron operados en Windows 7 con Matlab 2011a / Simulink. La comunicación entre estos tableros y Simulink se llevó

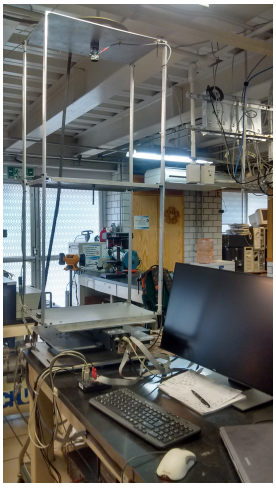


Figura 4.5: Prototipo DCA

a cabo utilizando el compilador C. Los experimentos se llevaron a cabo con un tiempo de muestreo de $0:005$s. Las frecuencias de vibración son $f_i = 1{,}758Hz$ y $f_2 = 4{,}0Hz$, como resultado de los experimentos de respuesta de frecuencia. Los parámetros estructurales como la masa, la rigidez y el amortiguamiento son necesarios para calcular el parámetro z que se utiliza para el entrenamiento de la red.Las masas fueron medidas directamente dando como valor $m_1 = 2{,}034$Kg y $m_2 = 2{,}534$kg. la rigidez $k_1 = 12011$N/m y $k_2 = 363{,}24$N/m fueron calculadas usando sus valores nominales de las propiedades mecánicas de los materiales. El amortiguamiento propuesto es del $3\,\%$

Para identificar z, el proceso de aprendizaje se muestran en la Figura 4.7 para ambas CNN, se utiliza el algoritmo BP para el entrenamiento, 2 bloques convolucionales son utilizadas en cada CNN y la tasa de aprendizaje se coloca en 0.3 para todas las capas de ambas redes, con 15 filtros por capa convolucional de dimensión 3.

Para la CNNt, $s_2 = s_4 = 2$, lo cual implica que de cada 2 elementos se elige el de mayor valor para seguir en la red, la entrada se construye de la siguiente manera, 4 datos de la salida de $CNNt$ en iteraciones anteriores, 4 datos de la aceleración del piso, 4 datos de la

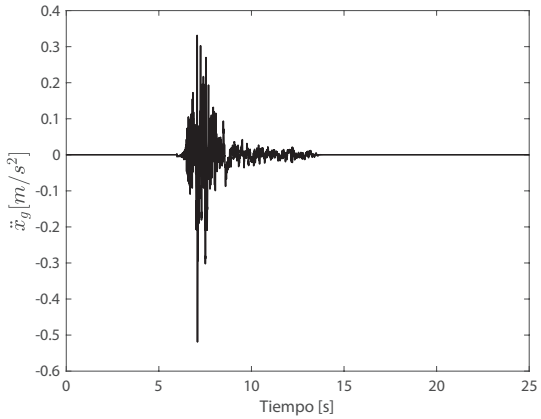


Figura 4.6: Prototipo DCA

señal de excitación, 4 datos de la velocidad y 4 de la posición del piso:

$$\begin{aligned}\hat{\varpi}\left(k\right) =&[\hat{y}\left(k-1\right),\cdots,\hat{y}\left(k-4\right),\ddot{x}\left(k-1\right),\cdots,\ddot{x}\left(k-4\right),\\ &\dot{x}\left(k-1\right),\cdots,\dot{x}\left(k-4\right),x\left(k-1\right),\cdots,x\left(k-4\right),\\ &\ddot{x}_g\left(k\right),\cdots,\ddot{x}_g\left(k-3\right)]^T\end{aligned} \tag{4.30}$$

La inicialización para ambas CNN se realiza en el intervalo $[1, 1]$. Dada las limitaciones físicas, las mediciones de las aceleraciones de cada piso son las únicas disponibles, las restantes deben ser estimadas. Por esta razón dos filtros pasa bandas junto con un integrador son usados para obtener las señales restantes, estos filtros están definidos como:

$$f(s) = \underbrace{\frac{s^2}{s^2+3{,}77+3{,}55}}_{hp} \times \underbrace{\frac{s^2}{s^2+3{,}77+3{,}55}}_{hp} \times \frac{1}{s} \tag{4.31}$$

con una frecuencia de corte de 0.3 Hz.

Se realizan 11 experimentos , 9 de ellos son usados para entrenamiento y los dos restantes para la generalización, cada experimento dura 25s muestreados cada 5ms.

En la Figura 4.8a se muestran los resultados de simulación para la $CNNt$ en el primer piso de la estructura, los datos de aceleración tienen ruido de medición, su MSE es de

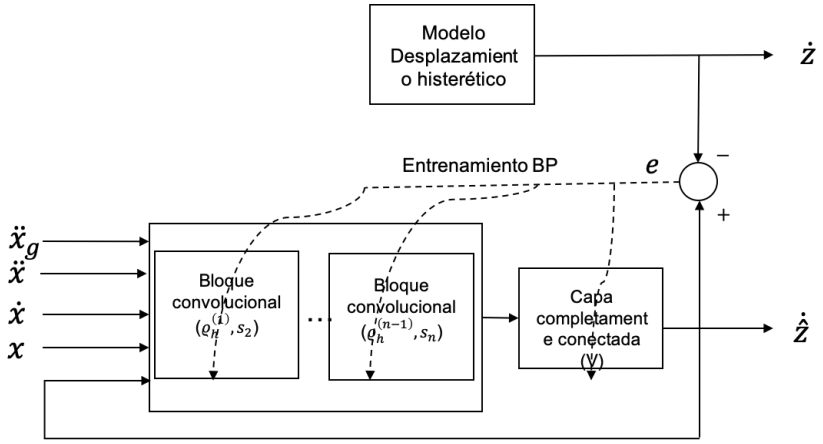


Figura 4.7: Proceso de identificación

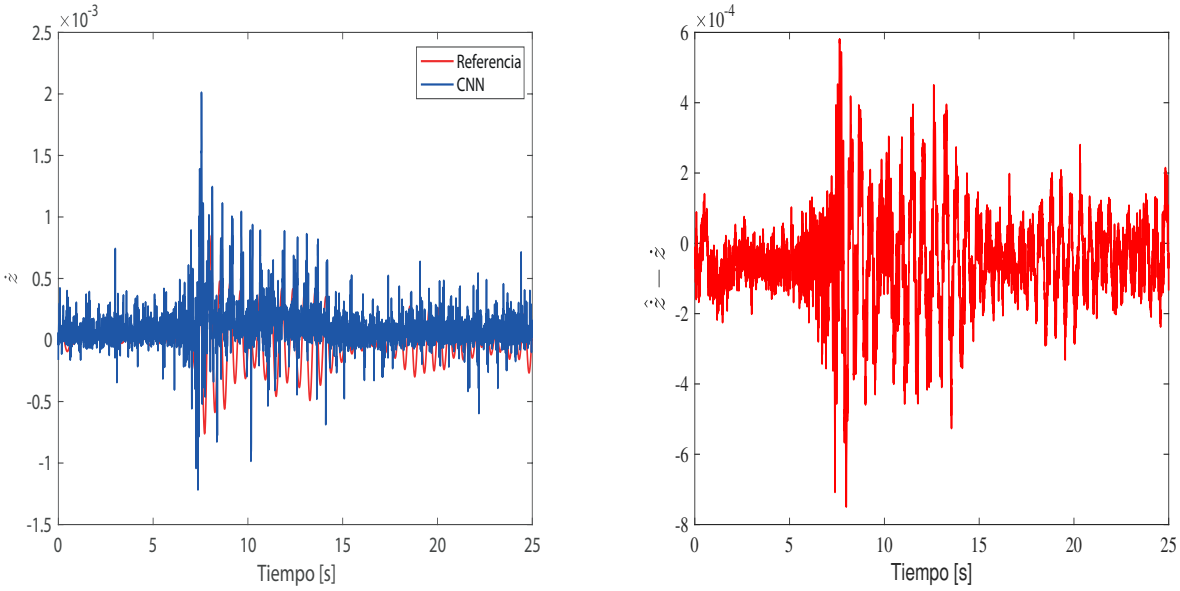
$1{,}0127 \times 10^{-7}$. Se requiere de 158.52 s para la etapa de entrenamiento con 5 épocas. La señal del error de seguimiento se presenta en la Figura 4.8b, de la cual se nota que el error es grande, pero sigue siendo útil la señal obtenida para calcular la energía de la estructura.

Ahora se utiliza un filtro Butterworth para limpiar la señal, reduciendo componentes de alta y baja frecuencia, el ancho de banda va de 0.3 a 5 Hz. Un conocimiento previo sobre las frecuencias naturales del sistema es necesario para poder diseñar el filtro. La Figura 4.9a muestra los resultados para la $CNNt$ utilizando el filtro mencionado en el segundo piso de la estructura, se observa que el error de generalización disminuye a $2{,}38 \times 10^{-8}$ con un tiempo de entrenamiento de 165.73s. Mientras que el error de seguimiento se muestra en la Figura 4.9b.

Para la CNNf, $s_2 = s_4 = 4$, la entrada esta dado similarmente , pero en lugar de utilzar orden de regresión 4, se utiliza 3, i.e.

$$\begin{aligned}\hat{\varpi}(k) =&[\hat{y}(k-1), \cdots, \hat{y}(k-3), \ddot{x}(k-1), \cdots, \ddot{x}(k-3),\\ &\dot{x}(k-1), \cdots, \dot{x}(k-3), x(k-1), \cdots, x(k-3),\\ &\ddot{x}_g(k), \cdots, \ddot{x}_g(k-2)]^T\end{aligned} \tag{4.32}$$

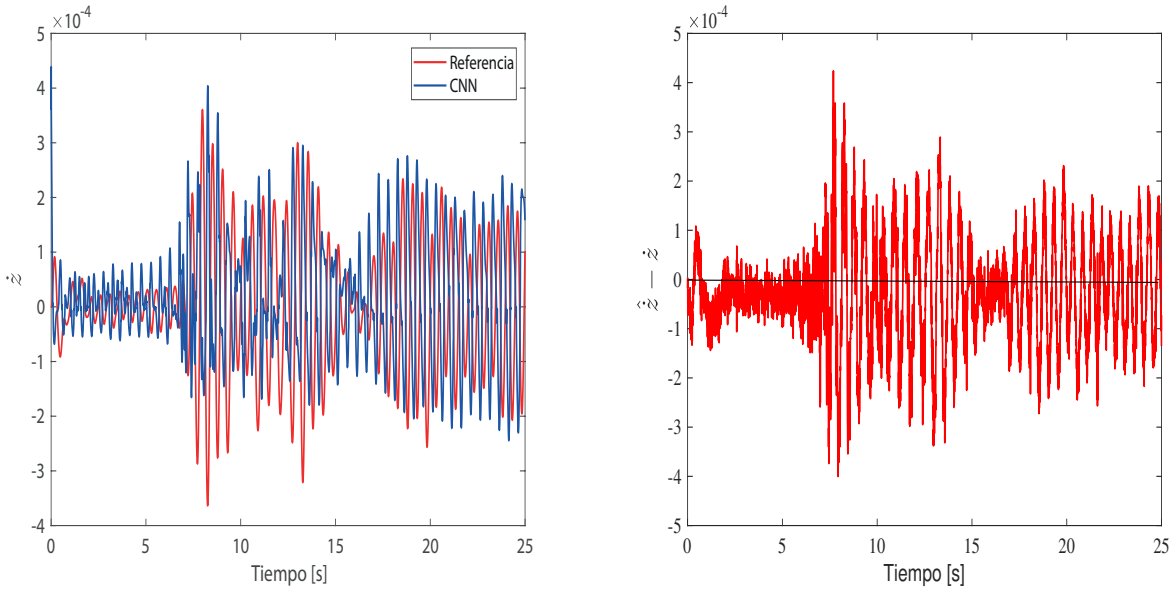
Los mismos 11 experimentos son utilizados para realizar pruebas con la $CNNf$. La $CNNf$



(a) Parámetro z del primer piso usando $CNNt$ con ruido en los datos.

(b) Error de seguimiento primer piso usando $CNNt$ con ruido en los datos.

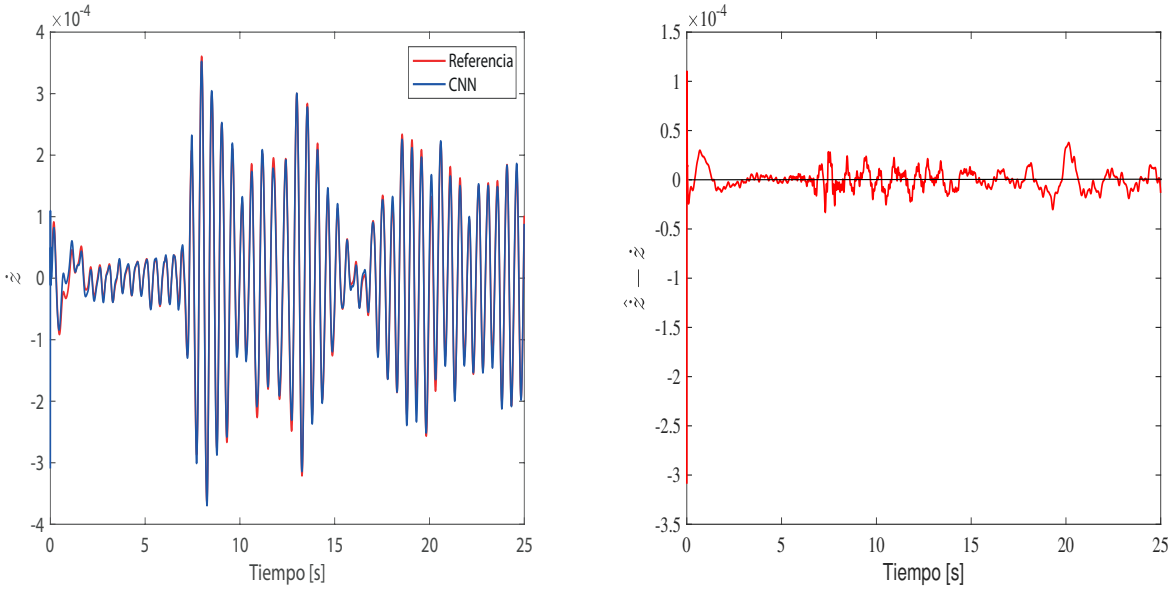
Figura 4.8: Resultados de simulación.



(a) Parámetro z del segundo piso usando $CNNt$ sin ruido en los datos.

(b) Error de seguimiento segundo piso usando $CNNt$ sin ruido en los datos.

Figura 4.9: Resultados de simulación.



(a) Parámetro z del segundo piso usando $CNNf$ con ruido en los datos.

(b) Error de seguimiento segundo piso usando $CNNf$ con ruido en los datos..

Figura 4.10: Resultados de simulación.

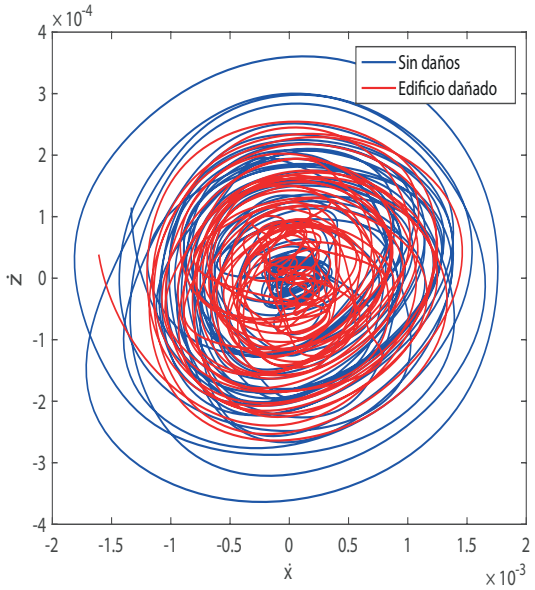
identificará el parámetro z del modelo de Bouc Wen de los pisos uno y dos del edificio realizando dos simulaciones para cada piso, una con ruido en los datos de entrada y otra utilizando filtros pasa bandas para limpiar la señal. En la Figura 4.10a se muestran los resultados obtenidos al aplicar la $CNNf$ para modelar z del segundo piso, se observa que aunque los datos presenten ruido el modelo se realiza de manera correcta obteniendo un MSE de 2,0901 $\times 10^{-9}$ requiriendo un tiempo de 43,97 s para 5 épocas de entrenamiento utilizando 9 experimentos. La gráfica del error se presenta en la Figura 4.10b, en este caso se tiene un error grande al inicio del modelado y manteniéndose por debajo de 0,5 $\times 10^{-4}$ en el resto del tiempo.

Una comparación completa se puede encontrar en la Tabla. 4.1, donde se puede apreciar que la $CNNf$ tiene mejores resultados de generalización al comparar los errores medios

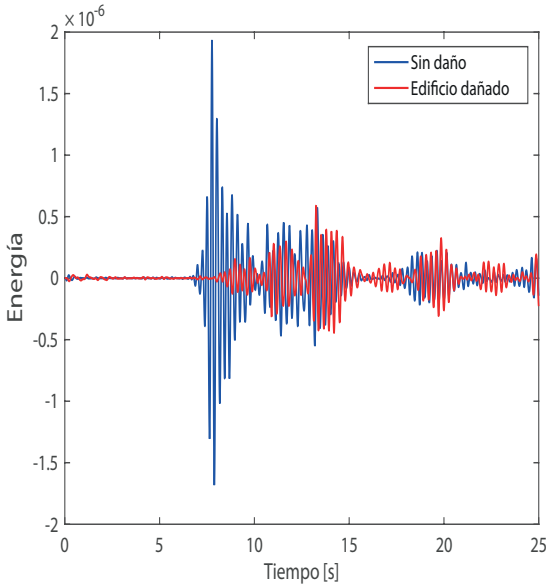
cuadráticos para ambos pisos así como también reduciendo aproximadamente 3.5 veces el tiempo requerido para el entrenamiento. Esta reducción en el tiempo tanto de entrenamiento como de generalización es debido principalmente a la estructura de la $CNNf$, al no contar con la operación de convolución como tal, y realizando un producto directo en las capas convolucionales, en cada iteración se requiere realizar menos cantidad de operaciones. Respecto al índice de desempeño $RMSE$ de la $CNNf$, aún con los datos sin filtrado, su valor es de un orden de magnitud más pequeño que la $CNNt$, con datos filtrados que sería el mejor de los casos para el modelado. Comparando ambas situaciones para la $CNNf$, de igual manera existe un orden de magnitud de diferencia en el valor del $RMSE$ para ambos pisos, pero se tendría que considerar si esa diferencia haga que valga la pena realizar el filtrado de los datos. La principal ventaja que se obtiene al emplear la $CNNf$ es poder reducir el efecto del ruido en el modelado de sistemas, así como también en reducir el tiempo y al carga computacional.

Tabla 4.1: Comparación de la CNNf contra CNNt.

Caso	Tiempo de entrenamiento (s)	Tiempo de generalización(s)	RMSE piso 1	RMSE piso 2
CNNt sin ruido	164.4124	1.3247	$2{,}34 \times 10^{-8}$	$3{,}6151 \times 10^{-8}$
CNNt con ruido	157.2137	1.3122	$1{,}8245 \times 10^{-7}$	$1{,}0127 \times 10^{-7}$
CNNf sin ruido	44.2361	1.0652	$2{,}4728 \times 10^{-10}$	$1{,}9167 \times 10^{-10}$
CNNf con ruido	42.9391	1.0352	$2{,}3511 \times 10^{-9}$	$2{,}0901 \times 10^{-9}$



(a) Ciclo de histéresis del primer piso con y sin daño.



(b) Comparación de energías del segundo piso.

Figura 4.11: Ciclo de histéresis y energías de la estructura con y sin daño.

4.3.5. Análisis de daño en estructuras de edificios

Para determinar si la estructura presenta daño, hacemos uso de los ciclos de histéresis, que se forman al trazar la velocidad contra el desplazamiento histérico para cada piso. Se ejecuta una simulación más con la $CNNf$, aquí se usan los datos de construcción de estructuras dañadas del mismo prototipo para que la $CNNf$ pueda obtener su desplazamiento histérico. Estos datos se obtienen aflojando una tuerca en el piso superior, esto se considera como pérdida de rigidez. En la Figura 4.11a se presenta una comparación entre los ciclos obtenidos cuando no hay daños y cuando hay daños en el edificio en el primer piso. Aquí puede verse que cuando hay daños, el ciclo se vuelve más pequeño, lo cual es un gran indicador de que la estructura sufrió daños.

Calculando la energía del sistema con la salida de la $CNNf$, se puede mostrar la diferencia

de estas cuando hay y cuando no hay daño en la estructura. En la Figura 4.11b se puede notar como la energía es mucho menor cuando la estructura presenta daño.

Hasta este punto solo se ha mostrado que la $CNNf$ puede modelar el parámetro de histéresis del modelo de Bouc Wen y con los ciclos de histéresis determinar si existe daño o no en la estructura, pero todavía no se ha identificado en donde se localiza el daño. La siguiente sección consiste en un clasificador basado en la $CNNf$ para localizar daños en estructuras utilizando señales de aceleración y de z.

4.4. CNN como clasificador

La principal aplicación de la CNN es como un clasificador. Debido a las propiedades de invarianza que presenta es muy utilizada en la clasificación de imágenes. En esta sección, además de utilizarse para identificar el parámetro z del modelo de Bouc Wen, la $CNNf$ es utilizada como un clasificador de señales de aceleración para determinar si existe daño o no en estructuras de edificios e indicar en que piso de la estructura se encuentra en caso de que exista daño, esta clasificación se realiza utilizando la señal de aceleración de cada piso para así determinar en conjunto si algún piso presenta daño.

4.4.1. Detección de daños en edificios usando CNN

Para la tarea de clasificación, una nueva arquitectura de $CNNf$ es propuesta ($CNNfc$), la cual consiste en bloques convolucionales ya descritos seguidos de un bloque de clasificación. Para resolver este problema de localización de daño en estructuras de edificios, una $CNNfc$ con dos bloques convolucionales seguidos de un bloque clasificador es utilizado, ver Figura 4.12.

La $CNNfc$ cuenta con una salida para cada clase de dato a clasificar p_i. En este sentido, 4 diferentes clases son designadas para la evaluación y localización de daño en el estructura:

- Clase 1. Evaluación global del edificio, si no existe daño esta salida tendrá valor igual a 1.

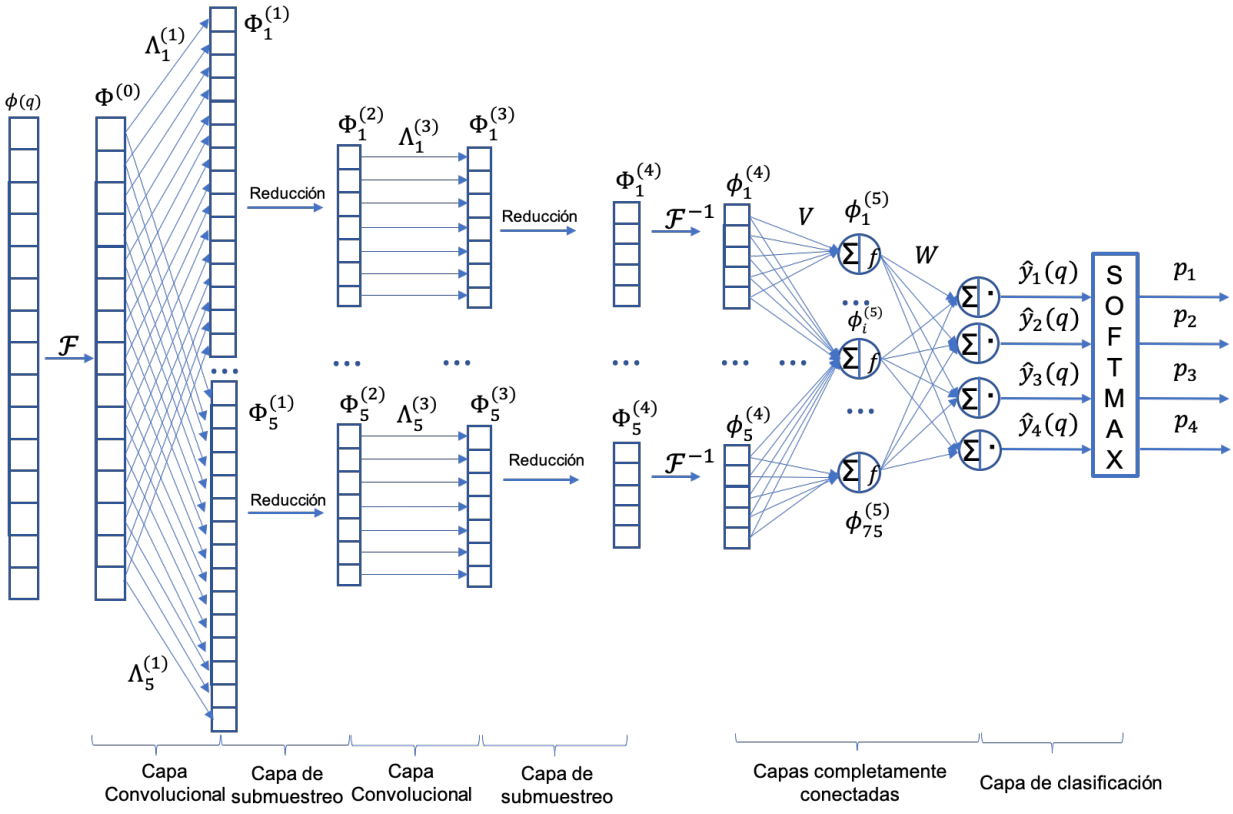


Figura 4.12: Arquitectura $CNNfc$ para clasificación.

- Clase 2. Evaluación del piso 1, solamente si en el primer piso se presenta el daño esta salida lo indicará.
- Clase 3. Evaluación del piso 2, solamente si en el segundo piso se presenta el daño esta salida lo indicará.
- Clase 4. Evaluación del piso 1 y 2, Si ambos pisos presentan daño con esta clase se podrá identificar.

Por lo tanto, cada grupo de datos tendrá la etiqueta correspondiente a la clase a la que pertenece.

Por otro lado, la entrada de la $CNNfc$ $\phi(q)$ para la clasificación se crea mediante el uso del algoritmo de Análisis de componentes principales (PCA) aplicado al desplazamiento histérico y los datos de aceleración con el fin de obtener los componentes principales para cada piso, incluida la base del edificio. Para cada señal obtenemos una matriz (3×3) que se convierte en vectores de tamaño 9, como se muestra en la Figura 4.13.

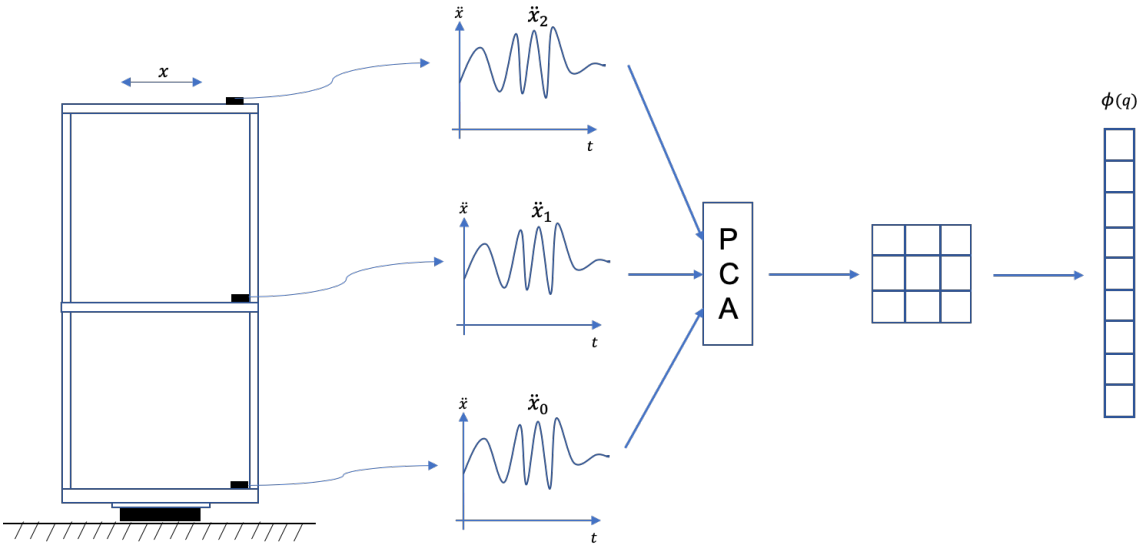


Figura 4.13: Diseño de entrada a la CNN

Una vez obtenido este vector, se procede a utilizar la DFT para obtener su representación en el dominio de la frecuencia, $\Phi^{(0)} = \mathcal{F}(\phi(q))$.

4.4.2. Bloque convolucional

El bloque convolucional consta de dos capas, una de convolución y otra de submuestreo. Para la capa convolucional, filtros aleatorios son definidos como $\lambda_i^{(\ell)} \in \Re^{f_\ell}$, donde $i = 1, 2, \cdots, r$, r es el total de filtros en la capa actual ℓ. Estos filtros pasan a través de una DFT para coincidir con la dimensión del vector de entrada del bloque (para el primer bloque, la transformación deberá coincidir con la dimensión de $\Phi^{(0)}$), i.e. $\Lambda_i^{(\ell)} = \mathcal{F}(\lambda_i^{(\ell)})$.

Por consiguiente, la salida de una capa convolucional se define como el producto elemento a elemento ($\odot$) de la salida del bloque anterior con los filtros, esto es:

$$\Phi_i^{(\ell)} = \Phi_i^{(\ell-1)} \odot \Lambda_i^{(\ell)} \tag{4.33}$$

Para la capa de submuestreo, una operación de agrupamiento espectral es realizada. Con esto obtenemos una representación mas pequeña de tamaño $s^{(\ell)}$, removiendo las frecuencias altas. La salida del bloque de convolución está definida como:

$$\Phi_i^{(\ell)} = Shrink(\Phi_i^{(\ell-1)}, s^{(\ell)}) \tag{4.34}$$

4.4.3. Bloque de clasificación

Este bloque consta de dos capas completamente conectadas seguidas de un capa de clasificación. En seguida de los dos bloques convolucionales, la salida se necesita regresar al dominio del tiempo, i.e. $\phi_i^{(\ell)} = \mathcal{F}^{-1}\left(\Phi_i^{(\ell)}\right)$ y se apila en un vector

$$\Psi = \left[\phi_1^{(\ell)T}, \phi_2^{(\ell)T}, \cdots, \phi_r^{(\ell)T}\right]^T \tag{4.35}$$

Este vector se utiliza como entrada para la primer capa completamente conectada, de donde su salida está definida por:

$$\phi^{(\ell)} = V^{\mathrm{T}}\Psi + b \tag{4.36}$$

donde b es el sesgo en esta capa y V son los pesos sinápticos de la capa. Se utilizan LL nodos ocultos, i.e. $\phi^{(\ell)} \in \Re^{LL}$. Las dimensiones de b y V deben ser adecuadas para las operaciones. Una función de activación es aplicada, en este caso la función $ReLU$ es la elegida:

$$y^{(F_1)} = max(0, \phi^{(\ell)}) \tag{4.37}$$

y se aplica a cada elemento del vector por separado. Finalmente, las salidas de la $CNNfc$ $\hat{y}(q)$ son:

$$\hat{y}(q) = W^T y^{(F_1)} + b_2 \tag{4.38}$$

donde $\hat{y}(q)$ es un vector de cuatro elementos, cada uno corresponde a las diferentes clases a clasificar, W is la matriz de pesos sinápticos de la capa de salida, con dimensiones adecuadas, y b_2 es el sesgo de la capa de salida.

Una vez obtenida la salida de la $CNNfc$, se utiliza un clasificador softmax, el cual calcula las "probabilidades"para todas las clases, esto significa que la suma de todas las salidas del clasificador deben sumar uno. La función softmax está definida como:

$$p_i = \frac{e^{\hat{y}_i}}{\sum_j^C e^{\hat{y}_j}} \tag{4.39}$$

donde C son el total de clases y p_i las probabilidades predichas para cada clase.

4.4.4. Entrenamiento $CNNfc$

Para entrenar la $CNNfc$ también se usa el algoritmo BP, pero en este caso se cambia la función de costo o pérdida, comúnmente cuando se utiliza softmax, la entropía cruzada es la indicada como función de costo. La función de entropía cruzada entre dos distribuciones de probabilidad (m, n) es:

$$xEnt(m, n) = -\sum_j m_j \log(n_j)) \tag{4.40}$$

donde j recorre cada clase en C. Considerando que cada grupo de datos tiene una etiqueta, se define $Y(j)$ como la probabilidad "verdadera"de la clase dada por los datos. Aquí, debe notarse que Y en general, contiene únicamente un 1 en la clase verdadera y ceros en los demás componentes, para que la suma de sus valores den 1. Definiendo la entropía cruzada para la $CNNfc$ se tiene:

$$J(Y, p) = -\sum_j Y_j \log(p_j) \tag{4.41}$$

Su gradiente para cada una de las salidas de la $CNNfc$ se obtiene mediante:

$$\frac{\partial J}{\partial \hat{y}_j} = p_j - Y_j \tag{4.42}$$

Esta expresión disminuye el valor de la función de costo cuando la salida de corresponde a la clase etiquetada Una vez obtenido el gradiente respecto a las salidas de la $CNNfc$, el proceso de entrenamiento se procede de la misma manera que se indica en la sección 4.3.3.

4.4.5. Detección de daño en estructuras mediante CNNfc

Dos clasificadores son propuestos, uno utilizando los datos de aceleraciones medidas de los sensores y otro utilizando además el parámetro de histéresis del modelo de Bouc Wen generados por la $CNNf$. Una vez obtenidos resultados de identificación de la $CNNf$ para el parámetro z, se utilizan esos datos como entrada para la $CNNfc$.

El primer experimento consiste en inducir daños reduciendo la rigidez k_1 del primer piso aflojando solo un tornillo en una columna que se conecta con la losa. Las 3 columnas restantes no se modifican. El siguiente paso consiste en extraer las características de la construcción de daños. Primero, el edificio está excitado nuevamente por el terremoto del Centro. A partir de la medición de la aceleración, se observa que el daño inducido produce una reducción del ancho de banda, cambiando a $f_1 = 1{,}709$ Hz, $f_2 = 4{,}0$ Hz. Estos cambios son un buen indicador de detección de daños para métodos basados en análisis vibracional, sin embargo, aquí utilizamos curvas de deformación de carga y cambios de energía disipativa. A través de $CNNf$ y la identificación basada en el modelo, es posible confirmar el daño estructural y la ubicación del mismo.

La tasa de aprendizaje para la capa convolucional es 0.0001 y para ambas capas completamente conectadas se establece en 0.007.

Considere 1 de los 28 experimentos realizados con daño en el primer piso, los resultados obtenidos al usar la $CNNfc$ con $z$como entrada además de los datos de aceleración confirman que existe daño en el primer piso con una probabilidad de 99,95 % como se observa en la Figura 4.14a. En general, la precisión de ambos clasificadores para daño en el primer piso es de 96,43 % al utilizar únicamente datos de aceleración y 100 % al utilizar z.

Ahora se investiga la ubicación del daño en el segundo piso. Similar al apartado anterior, el daño en el edificio se induce al reducir la rigidez k_2 del segundo piso al aflojar solo un tornillo en una columna que se conecta con la losa. Los elementos y configuraciones restantes no son modificadas. Una vez más, el edificio está excitado por el terremoto del Centro para extraer características de daños. La frecuencia de vibración fundamental y el ancho de banda también cambian debido al daño inducido, produciendo $f_1 = 1{,}733$ Hz, $f_2 = 3{,}97$ Hz. Para el segundo piso ambos clasificadores obtuvieron una precisión del 100 % sobre el conjunto de

datos. Para uno de los 35 experimentos de prueba, el segundo clasificador obtuvo que existe daño en el segundo piso con 99,99 % de precisión, ver Fig 4.14b.

Por último, el daño se localiza en ambos piso. Como se mostró anteriormente, el daño en el edificio es inducido reduciendo la rigidez k_1 y k_2 al aflojar tornillos en columnas de ambos pisos. El edificio es excitado con el terremoto del el Centro para extraer las características del daño. Los resultados indican que que debido al daño estructural, la frecuencia de vibración fundamental y el ancho de banda se reduce, produciendo $f_1 = 1{,}66$Hz y $f_2 = 35{,}97$ Hz, lo cual es un indicador claro de la presencia de daño.

Se utilizan 25 experimentos para los clasificadores, el primer clasificador obtiene una precisión de 92 % comparado con 100 % que obtiene el segundo. Como se observa en la Figura 4.14c, el clasificador que utiliza ambas señales como entrada indica que existe daño en ambos pisos con un 99,57 % de probabilidad.

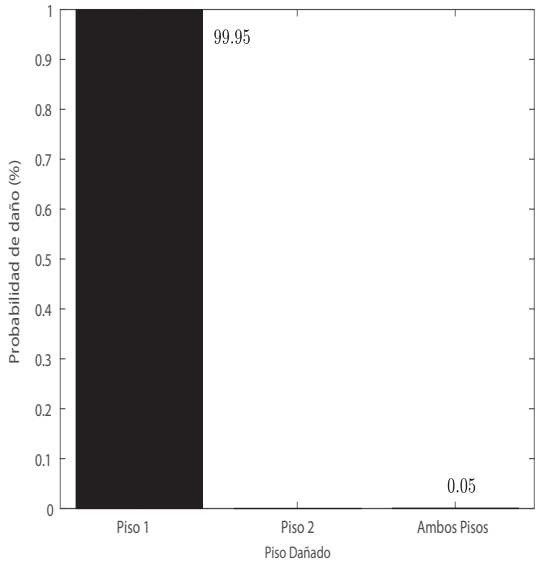
En resumen, se utilizaron 74 experimentos para entrenamiento que incluyen datos sin daño y con daño que caen en las 4 clases definidas y un total de 121 experimentos para la generalización, se requiere de 5500 épocas para el entrenamiento. Los resultados para el primer clasificador se muestran en la Tabla. 4.2, donde se puede notar que tiene una precisión del 95,04 % sobre todos los datos.

Tabla 4.2: Resultados de clasificación utilizando datos de aceleración.

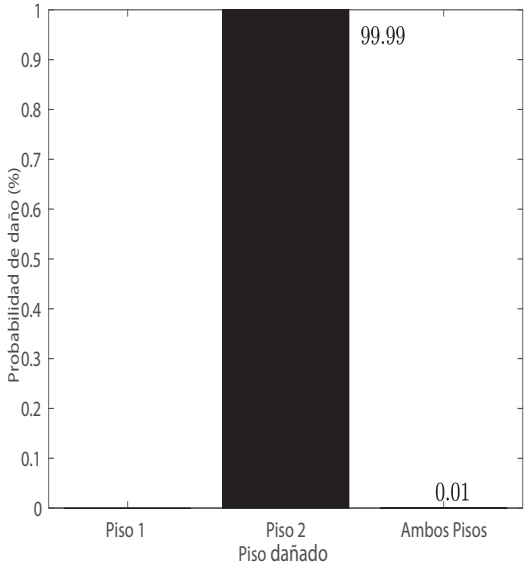
	Número de pruebas	**Aciertos**	**Precisión (%)**
Sin daño	33	30	90.91
Daño piso 1	28	27	96.43
Daño piso 2	35	35	100
Daño ambos pisos	25	23	92
Precisión total	121	115	95.04

En la Tabla. 4.3 agregando la salida de la $CNNf$ como datos de entrada se incrementa la precisión del clasificador a 100 %

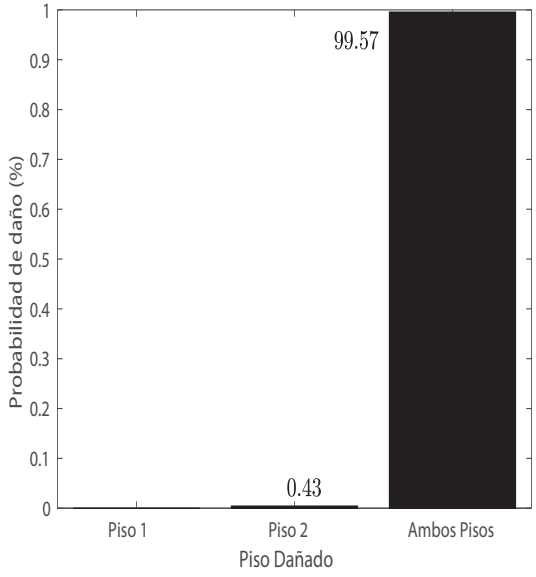
Una gran diferencia entre el clasificador propuesto con otros, es que no solo usa datos sin



(a) Localización de daño en el primer piso.



(b) Localización de daño en el segundo piso.



(c) Localización de daño en ambos pisos.

Figura 4.14: Resultados de clasificación del daño en la estructura.

Tabla 4.3: Resultados de clasificación utilizando datos de aceleración y el parámetro z_i.

	Número de pruebas	**Aciertos**	**Precisión (%)**
Sin daño	33	33	100
Daño piso 1	28	28	100
Daño piso 2	35	35	100
Daño ambos pisos	25	25	100
Precisión total	121	121	100

procesar sin saber nada sobre ellos, se tiene conocimiento previo de un modelo que puede contribuir a un mejor rendimiento como se demostró. Los resultados experimentales han demostrado que la inclusión de las señales del parámetro z en el esquema clasificador por $CNNfc$ mejora la precisión para localizar el daño estructural, en contraste cuando solo se utilizan datos de aceleración.

No se presenta una comparación de resultados con métodos ya publicados debido a que estamos empleando una técnica de clasificación basada en les mediciones del sistema como también de la señal generada por la $CNNf$ correspondiente al parámetro z del modelo de Bouc Wen. Además, el prototipo utilizado se encuentra en el laboratorio del departamento de control automático del CINVESTAV, por lo que no es de fácil acceso para que prueben sus algoritmos sobre éste sistema.

Capítulo 5

CNN para la identificación de sistemas: Arquitectura con valores complejos

La mayoría de los algoritmos de aprendizaje profundo funcionan con valores reales, como redes multicapa, máquina de vectores de soporte, memorias de largo y corto plazo y redes neuronales convolucionales. Por otro lado, tenemos los números complejos, un conjunto más completo con el que se puede trabajar, por ejemplo, hay redes neuronales multicapa con valores complejos que funcionan mejor que su contraparte real [133].

Las redes neuronales de valor complejo no se limitan a MLP. Hay otros tipos, como las redes RBF de valor complejo [134] y las redes neuronales convolucionales con valores complejos ($CVCNN$) [135, 136]. En la mayoría de los casos, los modelos de redes neuronales normales se modifican para trabajar con números complejos. Las modificaciones se encuentran tanto en la estructura del modelo como en los métodos de aprendizaje [137]. En [135] se introduce un algoritmo de entrenamiento para las redes neuronales de valor complejo. Una red neuronal de valor complejo puede considerarse como dos redes neuronales de valor real [138], que amplían la información que se puede aprender. Es más difícil dar un análisis teórico para valores complejos que para números reales como se menciona en [139]. En áreas

como el procesamiento de imágenes, la $CVCNN$ ha demostrado ser un método útil para eliminar el ruido de imágenes [140] y para la clasificación de imágenes SAR polarimétricas [141].

En el caso de imágenes automotrices, en [142] se propone un $CVCNN$ como clasificador basado en el tensor de rango-haz-Doppler generado por radares automotrices, útil para otros sistemas integrados dentro del automóvil. En [143] se propone la clasificación de las actividades humanas mediante un $CVCNN$, donde se capturan los movimientos de los humanos transformando estas imágenes al dominio de frecuencia para obtener información micro-Doppler, que se utiliza como datos de clasificación.

La principal ventaja de la $CVCNN$ es que se puede tener la misma estructura que la CNN de valor real, pero al mismo tiempo mejorar el rendimiento de la segunda en el sentido de modelado de sistemas no lineales. Con el fin de crear una red neuronal que pueda modelar sistemas no lineales con datos faltantes y grandes incertidumbres, en este capítulo se presenta una nueva arquitectura de $CVCNN$ para el modelado de sistemas no lineales, junto con su algoritmo de aprendizaje.

Para mostrar las ventajas del $CVCNN$, las comparaciones con los otros modelos de redes neuronales se realizan utilizando sistemas de referencia no lineales utilizando 5 tipos de índices de desempeño, comparándolo también con otros métodos actuales que se emplean para el mismo fin.

5.1. Propiedades de una red neuronal convolucional con valores complejos.

La arquitectura de una red neuronal convolucional para el modelado de sistemas se presentó con anterioridad en el capítulo 3.1. Para agregar la parte compleja a la estructura, es necesario cambiar todos los *hiper-parámetros* de la red, i.e. filtros y pesos sinápticos a valores complejos. Para modelar sistemas no lineales complejos, es necesario que la estructura de la red neuronal sea lo suficientemente grande, lo cual conlleva una gran cantidad de *hiper-*

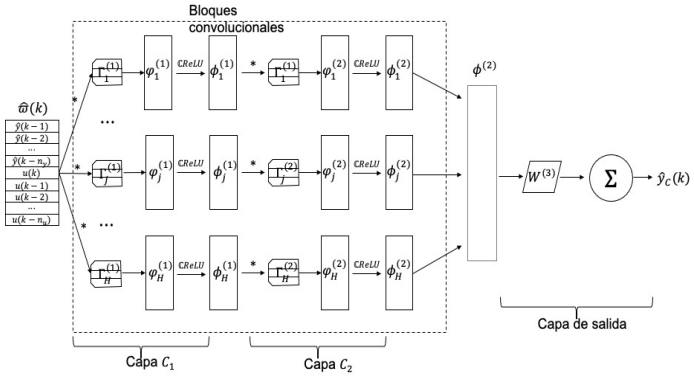


Figura 5.1: Arquitectura de una red neuronal convolucional para el modelado de sistemas no lineales

parámetros. Dado esto último, se generan ventajas sobre otros métodos que a continuación se enlistan:

- Las CNN poseen la propiedad de pesos compartidos, la cual permite disminuir la cantidad de filtros y pesos comparándola con una red neuronal multicapa por ejemplo. Mas aún, al contar los *hiper-parámetros* de la red con parte real e imaginaria, se obtiene información extra sobre el sistema sin tener que aumentar la cantidad de ellos. De igual manera, en la etapa de entrenamiento, como se verá mas adelante, la actualización de ellos no requerirá de operaciones extras.

- Cuando se extrae información de los datos de entrada, la propiedad de conectividad dispersa juega un papel importante, ya que los filtros buscan sobre todo el conjunto características para las cuales estos reaccionen de mejor manera. Estas características serán mas diversas debido a que los filtros son números complejos, adquiriendo mayor cantidad de información, o información distinta de la que puede obtener su contra-parte en el dominio real, generando mapas de características mas enriquecidos con información del sistema [144].

- La mayoría de las funciones de activación generan el problema del desvanecimiento del

gradiente [145], más cuando la estructura es profunda. Para reducir este problema se hace uso de la función $ReLU$ modificada para el dominio complejo, la cual mantiene la propiedad de reducir el desvanecimiento del gradiente.

- El sobreajuste de los datos es otro de los problemas recurrentes en este tipo de algoritmos, la capa de submuestro es la encargada de reducir este efecto, al utilizar una función compleja para la reducción de los datos, estamos eliminando características que no tienen mucha relevancia en el sistema quedándose con las que mayor respuestas se generen a partir de los filtros, lo cual implica que la probabilidad de presentar sobreajuste en los datos es mínima [146].

5.2. Arquitectura con valores complejos

Para modelar sistemas no lineales complejos, se emplea la arquitectura mostrada en la Figura 5.1 con las siguientes operaciones características.

5.2.1. Bloque convolucional

Para el bloque convolucional con valores complejos definimos la operación de convolución como:

$$\varphi_j^{(\ell)} = \phi_j^{(\ell-1)} \otimes \Gamma_j^{(\ell)} \tag{5.1}$$

donde φ es el $j-$ésimo mapa de características en la capa ℓ, producido por la convolución entre la entrada $\phi^{(\ell-1)}$, y el filtro de la capa actual $\Gamma_j^{(\ell)} \in \mathbb{C}$. En cada capa hay h filtros con una longitud de f_ℓ. Cada capa convolucional se denota con C_ℓ.

En la primera capa, $\phi^{(\ell-1)} = \hat{\varpi}$, la cual es la entrada de la $CVCNN$.

Una vez se han generado los mapas de características, se utiliza una función de activación, en este caso es necesario sustituir la función $ReLU$ por la función $\mathbb{C}ReLU$ definida como:

$$\begin{aligned} &\mathbb{C}\text{ReLU}(x) = \text{ReLU}(x_{\Re}) + i\text{ReLU}(x_{\Im}) \\ &x = x_{\Re} + ix_{\Im} \in \mathbb{C} \end{aligned} \tag{5.2}$$

La función $\mathbb{C}ReLU$ satisface las ecuaciones de Cauchy-Riemann, cuando ambas partes, real e imaginaria, son estrictamente positivas o negativas. Esta condición también se cumple cuando $\theta_x \in \left[0, \frac{\pi}{2}\right]$ or $\theta_x = \in \left[\pi, \frac{3}{2}\pi\right]$, $\theta_x = arg(x)$, θ_x es la fase o argumento del numero complejo x.

Aplicando la función de activación $\mathbb{C}ReLU$ al mapa de características (5.1),

$$\phi_j^{(\ell)} = \mathbb{C}ReLU(\varphi_j^{\ell}) \tag{5.3}$$

donde $\phi_j^{(\ell)}$ es la salida de la capa convolucional.

Para terminar el bloque convolucional, viene una capa de submuestreo. Dado que los números complejos tiene mas propiedades que un numero real, las operaciones de agrupamiento tales como el *max-pool* tienen un significado distinto. Por la propiedad cíclica, no podemos evaluar únicamente números complejos por su fase. La magnitud es una propiedad más confiable en el submuestreo. La operación a ocupar es conocida como agrupamiento por magnitud ($max - by - magnitude$), y se define como:

$$\arg \max_{x \in conjunto} M(x) \tag{5.4}$$

de donde:

$$M(x) = |x| \quad M : \mathbb{C} \to \Re \tag{5.5}$$

la expresión (5.4) permite mantener valores complejos a los largo de la $CVCNN$ ya que la salida de esta función es del mismo dominio que el de su entrada.

5.2.2. Capa completamente conectada

La capa final de la $CVCNN$ es una capa completamente conectada con pesos sinápticos $W \in \mathbb{C}^n$. La salida de la $CVCNN$ se define como:

$$\hat{y}_C(k) = W\phi^{(\ell-1)} \tag{5.6}$$

5.2.3. Algoritmo de retro propagación en el dominio complejo

La etapa de entrenamiento se realiza mediante el algoritmo de retro propagación modificado para esta red neuronal en particular.

Para lograr el objetivo del modelado de sistemas no lineales usando una red neuronal convolucional con valores complejos, es necesario actualizar los *hiper-parámetros* de la $CVCNN$, los filtros $\Gamma_j^{(\ell)}$ y los pesos sinápticos W, de tal manera que la salida de la $CVCNN$ $\hat{y}_C(k)$ converja a la salida del sistema $y(k)$ de (2.31):

$$\arg \min_{\Gamma_j^{(\ell)}, W^{(\ell)}} \sum_{k=1}^{N} \left[\hat{y}_C(k) - y(k)\right]^2, \quad \ell = 1, 2, \cdots, m$$

donde N es total de elementos en el conjunto de entrenamiento. Para alcanzar este objetivo, el algoritmo de retro propagación basado en el gradiente descendiente es empleado, el cual busca minimizar:

$$\arg \min_{\Gamma_j^{(\ell)}, W^{(\ell)}} \left[\hat{y}_C(k) - y(k)\right]^2$$

donde $\ell = 1, 2, \cdots, m$, $k = 1, 2, \cdots, N$.

La función de costo está definida como:

$$J(k) = \frac{1}{2} e_o(k)^2, \quad e_o(k) = \hat{y}_C(k) - y(k) \tag{5.7}$$

donde $e_o(k)$ es el error de modelado. La salida se descompone de la siguiente manera:

$$\hat{y}_C = \hat{y}_{\Re} + i\hat{y}_{\Im}$$

La estructura de la $CVCNN$ cuenta con $n-1$ bloques convolucionales, tanto capas convolucionales y de submuestreo, mas la capa de salida, donde cada capa convolucional cuenta con h filtros. Por lo que al final de los bloques convolucionales tendremos $h \times (n-1)$ filtros para actualizar, más los pesos sinápticos de la capa de salida, ver Figura 5.2.

Para aplicar este algoritmo a la $CVCNN$ es necesario definir conceptos que se utilizarán para ello.

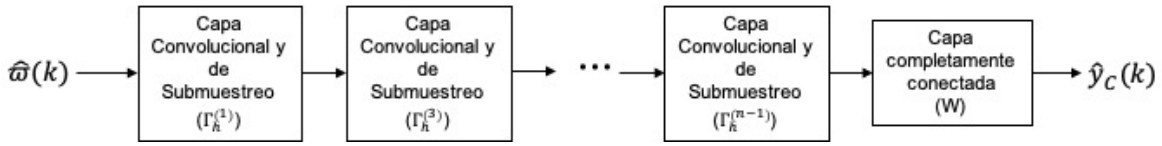


Figura 5.2: Estructura jerárquica de la CVCNN para el modelado de sistemas.

Para valores complejos, es necesario calcular el gradiente de funciones complejas, las cuales se definen como $f : \mathbb{C} \to \mathbb{C}$. La derivada de una función compleja puede ser vista como si se tratara de una función multivariable, definida como $f : \Re^2 \to \Re^2$.

Si $f(x)$ y $f(x, y)$ son diferenciables en el punto $a \in \Re^1$ y $(a, b) \in \Re^2$, entonces:

$$\begin{array}{l} \lim_{x\to\alpha} \frac{f(x)-f(\alpha)}{x-\alpha} = \beta \\ \lim_{(x,y)\to(a,b)} \frac{\|f(x,y)-f(a,b)-F(x-a,y-b)\|}{\|(x-a,y-b)\|} = 0 \end{array} \tag{5.8}$$

donde $x, \alpha, \beta \in \Re$.

Si $f(x, y) = f\,[u(x, y), v(x, y)]$ y se denota la derivada de f como Df, definida como:

$$Df = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix} \tag{5.9}$$

Aplicando el Teorema de Cauchy-Riemann [147], una función compleja $f(x) = r(x) + is(x)$, $r, s : \Re \to \Re$ es diferenciable si y solo si $f(x_\Re, x_\Im) = (r(x_\Re, x_\Im), s(x_\Re, x_\Im))$ es diferenciable en $\Re^2 \to \Re^2$, y sus derivadas parciales satisfacen:

$$\frac{\partial r}{\partial x_\Re} = \frac{\partial s}{\partial x_\Im}, \quad \frac{\partial s}{\partial x_\Re} = -\frac{\partial r}{\partial x_\Im} \tag{5.10}$$

donde $x_\Re$ y $x_\Im$ son las partes reales e imaginarias del número complejo x. Considerando un punto $a + ib$, las derivada de Wirtinger está dada por:

$$\lim_{x_\Re + ix_\Im \to a+ib} \frac{f(x_\Re + ix_\Im) - f(a + ib)}{x_\Re + ix_\Im - (a + ib)} = \varsigma_\Re + i\varsigma_{Im} \tag{5.11}$$

El número complejo $\varsigma = \varsigma_\Re + i\varsigma_{Im}$. (5.11) se convierte en:

$$\lim_{\substack{x_\Re + ix_\Im \\ \to a+ib}} \frac{\begin{bmatrix} f(x_\Re + ix_\Im) - f(a + ib) \\ -(\varsigma_\Re + i\varsigma_{Im})(x_\Re + ix_\Im - (a + ib)) \end{bmatrix}}{x_\Re + ix_\Im - (a + ib)} = 0 \tag{5.12}$$

Por lo tanto, la derivada compleja de la función f puede verse como una transformación lineal, la cual es equivalente a la derivada de la función en $\Re^2 \to \Re^2$. La derivada compleja son funciones lineales especiales con matrices ortogonales y con determinante positivo.

El diferencial de una función con valores complejos $f(x) : S \to \mathbb{C}$, $S \subseteq \mathbb{C}$ puede expresarse mediante:

$$df = \frac{\partial f(x)}{\partial x} dx + i \frac{\partial f(x)}{\partial x^*} dx^* \tag{5.13}$$

donde x^* se encuentra en el conjunto de los complejos. En este caso la derivada de la función real genera una matriz ortogonal con determinante positivo.

Actualización de los *hiper-parámetros*

Comenzando por la capa de salida, los pesos sinápticos se actualizan siguiendo la regla delta:

$$W(k+1) = W(k) - \eta \frac{\partial J}{\partial W} \tag{5.14}$$

con $\eta > 0$ siendo el factor de aprendizaje.

Las ecuaciones de Cauchy-Riemann se satisfacen para e_o con $r = \hat{y}_\Re - y$ y $s = \hat{y}_\Im$:

$$\begin{vmatrix} \frac{\partial r}{\partial \hat{y}_\Re} = 1 & \frac{\partial r}{\partial \hat{y}_\Im} = 0 \\ \frac{\partial s}{\partial \hat{y}_\Re} = 0 & \frac{\partial s}{\partial \hat{y}_\Im} = 1 \end{vmatrix} > 0, \quad \frac{\partial e_0}{\partial \hat{y}_C} = 1 \tag{5.15}$$

Es necesario calcular dos gradientes en esta capa, uno para actualizar los pesos sinápticos y otro para pasar el error a capas anteriores, primero se verifica que:

$$\begin{vmatrix} \frac{\partial r}{\partial W_\Re} = \Phi_\Re^{(n-1)} & \frac{\partial r}{\partial W_\Im} = -\Phi_\Im^{(n-1)} \\ \frac{\partial s}{\partial W} = \Phi_\Im^{(n-1)} & \frac{\partial s}{\partial W_\Im} = \Phi_\Re^{(n-1)} \end{vmatrix} > 0 \tag{5.16}$$

de aquí que, la derivada parcial de $\hat{y}_C$ respecto a W es:

$$\frac{\partial \hat{y}_C}{\partial W} = \frac{\partial r}{\partial W_\Re} + i \frac{\partial s}{\partial W_\Re} = \Phi_\Re^{(n-1)} + i\Phi_\Im^{(n-1)} = \Phi^{(n-1)} \tag{5.17}$$

donde $\Phi^{(n-1)} = [\phi_{n_1,1} \cdots \phi_{n_1,h}]^T$ es la salida de todos los filtros del último bloque convolucional del modelo (5.6).

Aplicando la regla de la cadena, se obtiene la expresión de la derivada de la función de costo respecto a W como:

$$\frac{\partial J}{\partial W} = \frac{\partial J}{\partial e_o}\frac{\partial e_o}{\partial \hat{y}_C}\frac{\partial \hat{y}_C}{\partial W} = (e_o)(1)(\Phi_{\Re}^{(n-1)} + i\Phi_{\Im}^{(n-1)}) = e\Phi^{(n-1)} \tag{5.18}$$

El gradiente de $\hat{y}_C$ respecto a $\Phi^{(n-1)}$ se obtiene con ayuda de la ecuación de Cauchy-Riemann con $r = \hat{y}_{\Re}$ y $s = \hat{y}_{\Im}$:

$$\begin{vmatrix} \frac{\partial r}{\partial \Phi_{\Re}^{(n-1)}} = W_{\Re} & \frac{\partial r}{\partial \Phi_{\Im}^{(n-1)}} = -W_{\Im} \\ \frac{\partial s}{\partial \Phi_{\Re}^{(n-1)}} = W_{\Im} & \frac{\partial s}{\partial \Phi_{\Im}^{(n-1)}} = W_{\Re} \end{vmatrix} > 0 \tag{5.19}$$

Por lo tanto:

$$\frac{\partial \hat{y}}{\partial \Phi^{(n-1)}} = \frac{\partial r}{\partial \Phi_{\Re}^{(n-1)}} + i\frac{\partial s}{\partial \Phi_{\Re}^{(n-1)}} = W_{\Re} + iW_{\Im} = W \tag{5.20}$$

El error propagado hacia capas anteriores proveniente de la capa n a la capa $n-1$ se describe como:

$$\frac{\partial J}{\partial \Phi^{(n-1)}} = e_o W \tag{5.21}$$

Para la capa de submuestreo, que es la siguiente en orden inverso, simplemente el error pasa a través de esta capa sin modificación de manera similar a lo que ocurre con la función *max-pool* (3.11) .

Después de la capa de submuestreo el error tienen que pasar por la función de activación. Para este caso se utiliza:

$$r = \text{ReLU}(\Psi_{\Re})^{(m)}, \quad s = \text{ReLU}(\Psi_{\Im}^{(m)})$$

para expresar $\Psi_{\Re}^{(n-1)}$, lo cual nos lleva a:

$$\begin{vmatrix} \frac{\partial r}{\partial \Psi_{\Re}^{(n-1)}} = \Psi_{\Re}^{(n-1)} & \frac{\partial r}{\partial \Psi_{\Im}^{(n-1)}} = 0 \\ \frac{\partial s}{\partial \Psi_{\Re}^{(n-1)}} = 0 & \frac{\partial s}{\partial \Psi_{\Im}^{(n-1)}} = \Psi_{\Im}^{(n-1)} \end{vmatrix} > 0 \tag{5.22}$$

$$\frac{\partial \Phi^{(n-1)}}{\partial \Psi^{(n-1)}} = \Psi_{\Re}^{(n-1)} \tag{5.23}$$

El gradiente a través de la función de activación se determina agregando $\Psi_{ij}^{(n-1)}$ al vector correspondiente $\varphi_j^{(n-1)}$, dando como resultado:

$$\frac{\partial J}{\partial \varphi_\imath} = \frac{\partial J}{\partial \phi_\imath^{(n-1)}} \varphi_\imath^{(n-1)} \tag{5.24}$$

La capa convolucional es similar a la capa completamente conectada, por el hecho de que la definición de convolución es una suma de productos, en la cual tenemos:

$$\frac{\partial J}{\partial \gamma_\jmath^{(\ell)}} = \sum_{a=0}^{N-f_\ell} \varphi_a^{(\ell)} \phi_{\jmath+a}^{(\ell-1)} \tag{5.25}$$

y el gradiente a través de esta capa se calcula como:

$$\frac{\partial J}{\partial \phi_\jmath^{(\ell-1)}} = \sum_{a=0}^{f_\ell-1} \frac{\partial J}{\partial \varphi_{\jmath-a}^{(\ell)}} \frac{\partial \varphi_{\jmath-a}^{(\ell)}}{\partial \phi_\jmath^{(\ell-1)}} = \sum_{a=0}^{f_\ell-1} \frac{\partial j}{\partial \varphi_{\jmath-a}^{(\ell)}} \gamma_a^{(\ell)} \tag{5.26}$$

En la siguiente subsección se detallan las operaciones realizadas en la etapa hacia adelante de la red.

Etapa hacia adelante

Los filtros y los pesos sinápticos para la $CVCNN$ están definidos como:

$$\Gamma^{(\ell)} = \begin{bmatrix} \gamma_{1,\Re}^{(\ell)} + i\gamma_{1,\Im}^{(\ell)} \\ \vdots \\ \gamma_{p,\Re}^{(\ell)} + i\gamma_{p,\Im}^{(\ell)} \end{bmatrix} \in \mathbb{C} \tag{5.27}$$

y

$$W = [W_{1,\Re} + iW_{1,\Im} \cdots W_{m,\Re} + iW_{m,\Im}]^T \in \mathbb{C}^m \tag{5.28}$$

La entrada a la $CVCNN$ es $\hat{\varpi} = \begin{bmatrix} \phi_1^{(0)} \\ \phi_2^{(0)} \\ \cdots \\ \phi_l^{(0)} \end{bmatrix}$.

En la primera capa, la operación de convolución genera los primeros mapas de características $\varphi^{(1)}$,

$$\begin{aligned}\varphi_1^{(1)} &= \phi^{(0)} * \Gamma_1^{(1)} = \begin{bmatrix}\Psi_{1,1}^{(1)}\\ \vdots \\ \Psi_{1,q}^{(1)}\end{bmatrix} \\ &= \begin{bmatrix}\phi_1^{(0)}\\ \vdots \\ \phi_q^{(0)}\end{bmatrix} * \begin{bmatrix}\gamma_{1,\Re}^{(1)} + i\gamma_{1,\Im}^{(1)}\\ \vdots \\ \gamma_{p,\Re}^{(1)} + i\gamma_{p,\Im}^{(1)}\end{bmatrix}\end{aligned} \tag{5.29}$$

donde cada elemento $\Psi^{(1)} \in \mathbb{C}$ se obtiene mediante la operación

$$\Psi_{1,j}^{(1)} = \sum_{a=0}^{q} \phi_a^{(0)} \gamma_{j-a} \tag{5.30}$$

Posteriormente, la función de activación $\mathbb{C}ReLU$ se aplica a estos mapas de características,

$$\phi_1^{(1)} = \mathbb{C}\mathrm{ReLU}(\varphi_1^{(1)}) \tag{5.31}$$

Los elementos de este vector son:

$$\begin{aligned}\phi_1^{(1)} &= \begin{bmatrix}\Phi_{1,1}^{(1)}\\ \vdots \\ \Phi_{1,q}^{(1)}\end{bmatrix} = \begin{bmatrix}\mathbb{C}\mathrm{ReLU}(\Psi_{1,1}^{(1)})\\ \vdots \\ \mathbb{C}\mathrm{ReLU}(\Psi_{1,q}^{(1)})\end{bmatrix} \\ &= \begin{bmatrix}\mathrm{ReLU}(\Psi_{1,1,\Re}^{(1)}) + i\mathrm{ReLU}(\Psi_{1,1,\Im}^{(1)})\\ \vdots \\ \mathrm{ReLU}(\Psi_{1,3,\Re}^{(1)}) + i\mathrm{ReLU}(\Psi_{1,p,\Im}^{(1)})\end{bmatrix}\end{aligned} \tag{5.32}$$

Esta operación se repite para todos los bloques convolucionales que se tengan hasta llegar a la capa completamente conectada, en la cual la operación para obtener la salida de la $CVCNN$ es:

$$\hat{y}_C = W\phi^{(n-1)} = [W_1 \cdots W_p] \begin{bmatrix}\Phi_1^{(n-1)}\\ \cdots \\ \Phi_p^{(n-1)}\end{bmatrix} \tag{5.33}$$

La salida se descompone de la siguiente manera:

$$\hat{y}_C = \hat{y}_{\Re} + i\hat{y}_{\Im}$$

donde $\hat{y}_{\Re}$ y $\hat{y}_{\Im}$ se definen como:

$$\begin{aligned} \hat{y}_{\Re} &= W_{1,\Re}\Phi_{1,\Re}^{(n-1)} - W_{1,\Im}\Phi_{1,\Im}^{(n-1)} + \cdots \\ &\quad + W_{p,\Re}\Phi_{p,\Re}^{(n-1)} - W_{p,\Im}\Phi_{p,\Im}^{(n-1)} \\ &\quad \vdots \\ \hat{y}_{\Im} &= W_{1,\Re}\Phi_{1,\Im}^{(n-1)} + W_{1,\Im}\Phi_{1,\Re}^{(n-1)} + \cdots \\ &\quad + W_{p,\Re}\Phi_{p,\Im}^{(n-1)} + W_{p,\Im}\Phi_{p,\Re}^{(n-1)} \end{aligned} \tag{5.34}$$

aquí $\Phi_{1,\Re}^{(n-1)}$ y $\Phi_{1,\Im}^{(n-1)}$ son las partes real e imaginaria del elemento $\Phi_1^{(n-1)}$, respectivamente.

5.3. Simulaciones

En esta sección se utilizan 3 ejemplos de prueba para mostrar la efectividad de la red neuronal convolucional con valores complejos. Los resultados de simulación se comparan con la red mostrada en el capítulo 3.1 y con una MLP.

La inicialización de los filtros y pesos sinápticos de la $CVCNN$ se realiza de manera aleatoria en el intervalo $[-1, 1]$ tanto para la parte real como para la parte imaginaria. mientras que para la CNN la inicialización de los *hiper-parámetros* se hace en el intervalo $[-1, 1]$. Los pesos sinápticos de la MLP se inicializan en el mismo intervalo $[-1, 1]$.

Para propósitos comparativos, se utilizará la misma estructura para la $CVCNN$ y para la CNN, i.e. la misma cantidad de bloques convolucionales. La diferencia radicará en la cantidad de *hiper-parámetros*, los filtros y pesos sinápticos, y en el hecho que unos serán valores complejos y otros reales. Esta cantidad varía de acuerdo al sistemas que estemos utilizando. Ambas CNN tendrán dos bloques convolucionales, con sus respectivas funciones de activación y de agrupamiento, $\mathbb{C}$ReLU con *max-by-magnitude* y $ReLU$ con *max-pool*, respectivamente.

En el caso de la MLP, está es una red neuronal de dos capas, con la función de activación $tanh\,(\cdot)$. El número de neuronas o nodos en la capa oculta cambia de acuerdo al sistema utilizado buscando siempre la mejor configuración posible, se especifica en cada sistema la cantidad utilizada.

Se utilizan 4 tipos de modelo de la entrada a los métodos empleados para las simulaciones:

1. Modelo serie-paralelo (2.34)

$$\hat{y}(k) = NN[y\,(k-1)\,, \cdots y\,(k-n_y)\,, u\,(k)\,, \cdots u\,(k-n_u)] \tag{5.35}$$

   tanto la entrada $u\,(k)$ y la salida $y\,(k-1)$ del sistema a identificar están disponible para alimentar a la red neuronal $NN\,[\cdot]\,$.

2. Modelo paralelo (2.33)

$$\hat{y}(k) = NN[\hat{y}(k-1), \cdots \hat{y}(k-n_y), u\,(k)\,, \cdots u\,(k-n_u)] \tag{5.36}$$

   únicamente se cuenta con la entrada $u\,(k)$ del sistema a identificar como entrada de la red neuronal $NN\,[\cdot]\,$. En lugar de la salida real del sistema, la salida de la red neuronal $\hat{y}(k-1)$ es utilizada como entrada.

3. Modelo serie-paralelo con ruido en los datos.

$$\hat{y}(k) = NN[y\,(k-1) + \rho, \cdots y\,(k-n_y) + \rho, u\,(k)\,, \cdots u\,(k-n_u)] \tag{5.37}$$

   donde ρ es el ruido aleatorio agregado a los datos de entrenamiento.

4. Modelo serie-paralelo con pérdida de datos.

$$\hat{y}(k) = NN[\bar{y}\,(k-1)\,, \cdots \bar{y}\,(k-n_y)\,, \bar{u}\,(k)\,, \cdots \bar{u}\,(k-n_u)] \tag{5.38}$$

   donde $\bar{y}\,(k-1)$ y $\bar{u}\,(k)$ son pares de datos elegidos de manera aleatoria del conjunto original $\{y\,(1)\,, \cdots y\,(N)\}$ y $\{u\,(1)\,, \cdots u\,(N)\}$. Para las simulaciones, se utilizan $70\,\%$ del total de datos y el $30\,\%$ restante se considera como perdido.

Se describe a continuación cada sistema a identificar utilizado junto con los *hiper-parámetros* de la red que mejor resultados presentaron.

Horno de gas

El conjunto de datos para la primera parte de simulación de la $CVCNN$ es el mismo que se utiliza en la sección 3.3.2.

Para los modelos serie-paralelo (5.35), (5.37) y (5.38), el vector de entrada de los modelos neuronales está definido como

$$[y(k-1), \cdots y(k-5), u(k), \cdots u(k-4)]$$

Para el modelo paralelo (5.36), el vector de entrada es $[\hat{y}(k-1), u(k), \cdots u(k-10)]$. Para la $CVCNN$ y CNN, cada capa convolucional tiene 3 filtros convolucionales de tamaño 3. La MLP tiene 50 neuronas en la capa oculta.

En el caso de los datos con ruido, la potencia de este es de un 10 % de la amplitud de los datos.

Los resultados de simulación utilizando el modelo paralelo se muestra en la Figura 5.3 para el sistema del horno de gas, de la gráfica se nota que los tres algoritmos realizan el seguimiento de la mejor manera.

La gráfica del error de seguimiento se presenta en la Figura 5.4, como se esperaba, el error entre la MLP y los datos reales es muy grande comparada con las de las CNN, recordando que se esta utilizando el modelo paralelo para las simulaciones.

La principal métrica de desempeño utilizada para evaluar los sistemas es la del error cuadrático medio MSE. Para este caso la $CVCNN$ tiene un valor de 0.0069 mientras que la CNN tiene un valor de 0.0215 y la MLP de 0.0158.

Los resultados de las distintas métricas de desempeño se muestran en la siguiente sección para todos los sistemas a identificar.

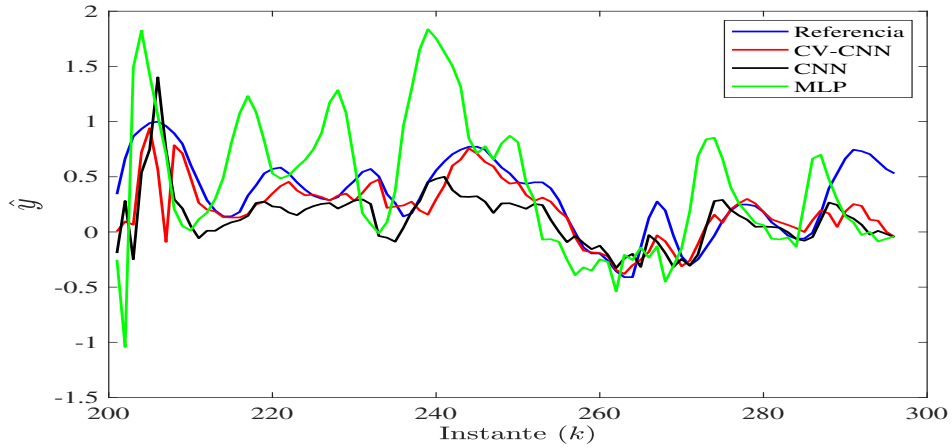


Figura 5.3: Resultados de simulación para el sistema del horno de gas utilizando el modelo paralelo.

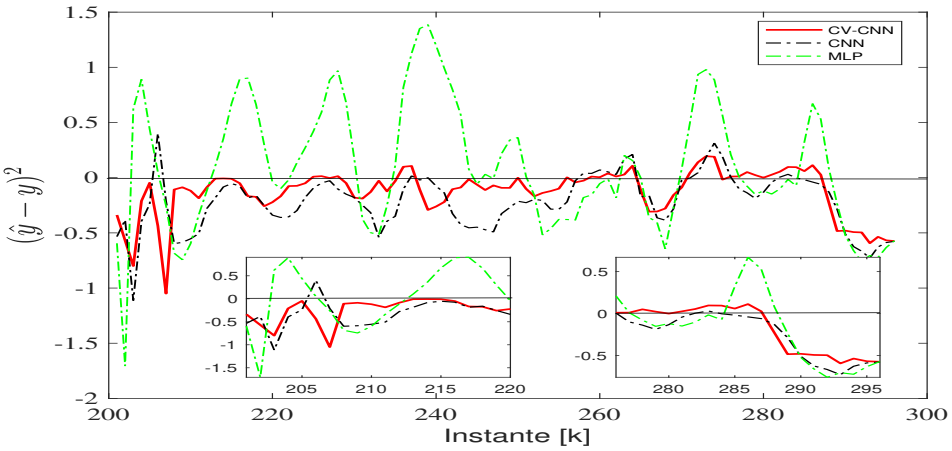


Figura 5.4: Error de seguimiento sistema del horno de gas utilizando el modelo paralelo.

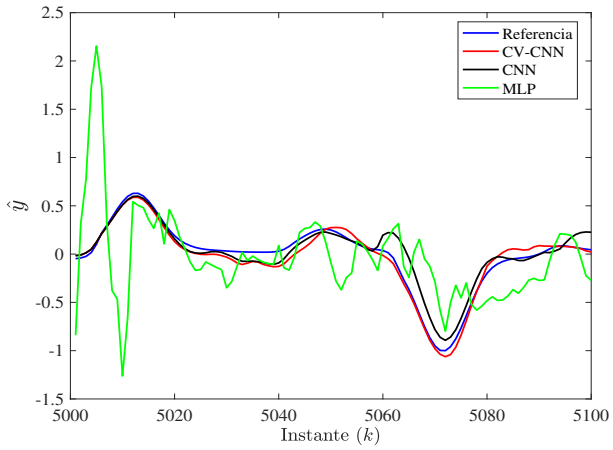


Figura 5.5: Resultados de simulación para el sistema no lineal de primer orden utilizando el modelo paralelo.

Sistema no lineal

El siguiente sistema a utilizar se presentó en la sección 3.3.1. Para los modelos serie-paralelo (5.35), 5.37 y (5.38)), el vector de entrada a los modelos neuronales es:

$$[y(k-1), \cdots y(k-10), u(k), \cdots u(k-13)]$$

Para el modelo paralelo (5.36), el vector de entrada es $[\hat{y}(k-1), u(k), \cdots u(k-12)]$. La $CVCNN$ y la CNN tienen cada una 8 filtros en cada capa convolucional, para la MLP se utilizan 35 nodos en la capa oculta. Cuando los datos presentan ruido, este es como máximo un 10 % de la amplitud de los datos.

Los resultados de la identificación para este sistema se presenta en la Figura 5.5, en la que se observa que cuando se utiliza el modelo paralelo, la $CVCNN$ tiene el mejor desempeño, considerando la métrica MSE, 0.0050, mientras que la CNN y MLP obtuvieron un valor de 0.1743 y 0.3822, respectivamente.

Los errores mostrados en la Figura 5.6 muestran también que ambas CNN tiene un seguimiento mejor que la MLP para este sistema, una mejor comparación se realiza en la

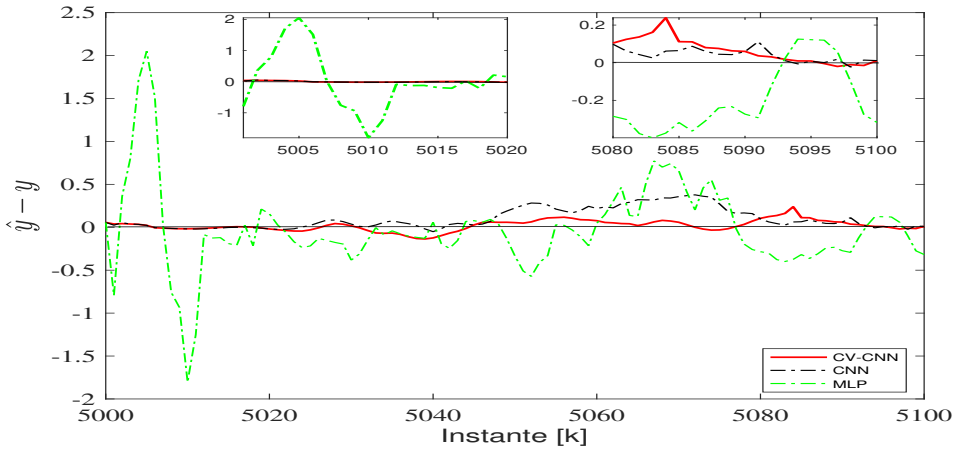


Figura 5.6: Error de seguimiento sistema no lineal de primer orden utilizando el modelo paralelo.

siguiente sección utilizando distintos índices de desempeño.

Los resultados de simulación de los otros modelos se presentan en la siguiente sección.

Sistema Wiener-Hammerstein

Este sistema ya fue descrito en la sección 3.2.1. Ahora se utilizan 12000 datos para entrenamiento y lo restante para generalización. Para los modelos serie-paralelo (5.35), (5.37) y (5.38), el vector de entrada a los modelos neuronales es:
$[y(k-1), \cdots y(k-4), u(k), \cdots u(k-5)]$.

Para el modelo paralelo (5.36), el vector de entrada está definido por:

$[\hat{y}(k-1), \cdots \hat{y}(k-3), u(k), \cdots u(k-80)]$.

La cantidad de filtros utilizados para este sistema es de 15 de tamaño 6 para ambas CNN, mientras que para la MLP se utilizan 50 neuronas en la capa oculta.

Los resultados de la identificación del sistema cuando se utiliza el modelo paralelo (5.36) se muestra en Figura 5.7.

De la Figura 5.8 podemos observar que tanto la CNN como la MLP presentan un error

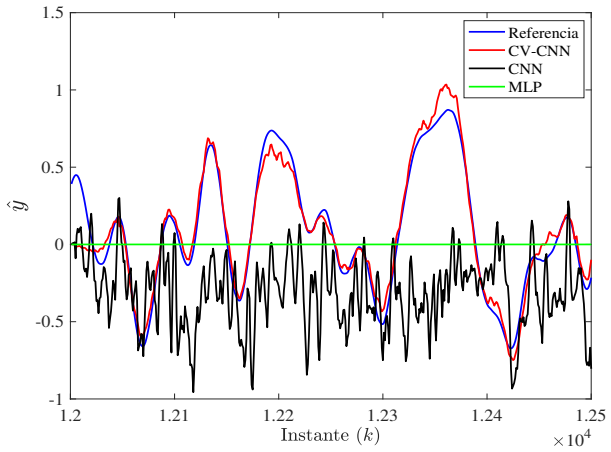


Figura 5.7: Resultados de simulación para el sistema Wiener-Hammerstein utilizando el modelo paralelo.

de seguimiento muy grande comparado con la $CVCNN$, de aquí que para este sistema en particular solo la $CVCNN$ es la que está realizando la identificación correctamente. El valor del índice MSE que se obtuvo para la $CVCNN$ es 0.0042, mientras que para la CNN 0.0106 y la MLP 0.0544. Para los tres sistemas mostrados, la $CVCNN$ tiene un mejor desempeño que los otros dos, haciéndola una opción factible para la identificación de sistemas.

Una comparación mas detallada se presenta en la siguiente sección.

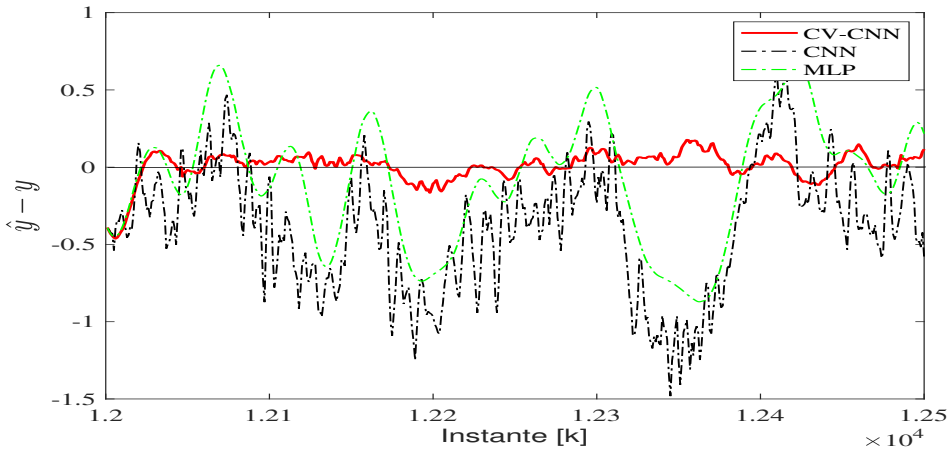


Figura 5.8: Error de seguimiento sistema Wiener-Hammerstein utilizando el modelo paralelo.

5.3.1. Discusión de los resultados

Se presentan a continuación tablas comparativas de las distintas métricas de desempeño utilizadas sobres nuestros algoritmos propuestos. Se presentan primero las correspondientes al sistema del horno de gas

Tabla 5.1: Métricas de desempeño para el sistema del horno de gas usando el modelo serie-paralelo.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.8784	0.9669	0.0069	0.0729	0.0829
MLP	0.7207	0.9166	0.0158	0.1279	0.1257
CNN	0.6200	0.8833	0.0215	0.1313	0.1466

Las Tablas 5.1, 5.2, 5.3 y 5.4 muestran los resultados de simulación para el sistema del horno de gas utilizando 5 métricas de desempeño comparando tres modelos distintos, hay que tener en cuenta que únicamente se está comparando este índice y no se está considerando

Tabla 5.2: Métricas de desempeño para el sistema del horno de gas usando el modelo paralelo.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.4042	0.8270	0.0337	0.1914	0.1835
MLP	-1.1887	0.0983	0.1237	0.3222	0.3518
CNN	0.5052	0.8657	0.0280	0.1742	0.1672

Tabla 5.3: Métricas de desempeño para el sistema del horno de gas usando el modelo serie-paralelo con ruido en los datos.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.8785	0.9669	0.0074	0.0980	0.0860
MLP	-3.4390	0.4211	13.9291	1.7413	3.7322
CNN		0.7255	0.9420	0.1015	0.1293

Tabla 5.4: Métricas de desempeño para el sistema del horno de gas usando el modelo serie-paralelo con pérdida de datos.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.9305	0.9841	0.0042	0.0736	0.0650
MLP	-0.6885	-0.3736	0.0953	0.3530	0.3087
CNN	0.6803	0.9454	0.0195	0.1041	0.1395

la parte computacional requerida para obtener esos resultados, como lo es el tiempo de entrenamiento o de generalización ni los recursos utilizados por el procesador. Como se puede apreciar en estas tablas, la $CVCNN$ propuesta se desempeña mejor que los otros dos métodos con los que se compara. Por otro lado, en la Tabla 5.5, los resultados del estado de arte para este sistema se presentan comparando únicamente la métrica $RMSE$ [148]. Para estos resultados, los datos se utilizaron sin ninguna modificación, como se menciona en cada

trabajo.

Hasta donde se tiene conocimiento, no hay resultados publicados para los casos que se presentan donde se generen datos faltantes o haya alteraciones en ellos con los cuales se pueda hacer una comparación.

Tabla 5.5: Métricas RMSE para el sistema del horno de gas encontradas en la literatura.

modelo	RMSE
Arima	0.843
Tong's model	0.685
Xu's model	0.573
Sugeno's model	0.596
surmans's model	0.4
ANFIS	0.405
Generalized fuzzy NN	0.273
PEC-WNN	0.0589
OSELM-K [149]	0.3341
LSTM	0.9730

Los resultados obtenidos muestran que el método propuesto tiene un mejor desempeño que los otros métodos, comparando la métrica $RMSE$, únicamente superado por PEC-WNN.

Para este modelo en particular se concluye:

- En el caso de ruido o de perdida de datos, la $CVCNN$ propuesta presenta un mejor desempeño que la MLP o la CNN con valores reales, la cual es una alternativa adecuada para estos casos.

- Cuando se utiliza el modelo paralelo, tanto la $CVCNN$ como la CNN logran modelar el sistema mucho mejor que la MLP, esto puede notarse en los valores obtenidos en la Tabla 5.3

- En el caso del modelo serie-paralelo, las tres redes neuronales tienen valores cercanos en sus métricas, haciendo que todas sean alternativas para el modelado, este es el formato de modelado mas utilizado.

Las Tablas 5.6, 5.7, 5.8 y 5.9 muestran las comparaciones de las métricas de desempeño para las tres redes neuronales con los cuatro tipos de modelo, i.e. serie-paralelo, paralelo, serie-paralelo con ruido en los datos y serie-paralelo con pérdida de datos, respectivamente para el sistema Wiener-Hammerstein.

Tabla 5.6: Métricas de desempeño para el sistema Wiener-Hammerstein usando el modelo serie-paralelo.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.9991	0.9998	0.000076	0.0.0018	0.0087
MLP	0.9551	0.9884	0.0038	0.0641	0.0617
CNN	0.9975	0.9994	0.00021	0.0127	0.0146

Tabla 5.7: Métricas de desempeño para el sistema Wiener-Hammerstein usando el modelo paralelo.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.9503	0.9867	0.0042	0.0684	0.0649
MLP	0.3588	0.7217	0.0544	0.2458	0.2322
CNN	0.8747	0.9734	0.0106	0.0992	0.1031

Como puede notarse en las tablas presentadas para el sistema Wiener-Hammerstein, la $CVCNN$ propuesta en este trabajo tiene mejores resultados en las métricas de desempeño que las otras dos redes, en algunas pruebas como lo es el modelo serie-paralelo, las tres redes neuronales tienen un desempeño muy parecido, mientras que en el caso del modelo serie-

Tabla 5.8: Métricas de desempeño para el sistema Wiener-Hammerstein usando el modelo serie-paralelo con ruido en los datos.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.9734	0.993	0.00018	0.0155	0.0138
MLP	-0.0350	-0.0237	0.0878	0.3445	0.2963
CNN	0.1471	0.5112	0.0723	0.3082	0.2690

Tabla 5.9: Métricas de desempeño para el sistema Wiener-Hammerstein usando el modelo serie-paralelo con pérdida de datos.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.8753	0.9702	0.00089	0.0100	0.0298
MLP	0.7890	0.9344	0.0015	0.0430	0.0388
CNN	0.7129	0.9341	0.0020	0.0106	0.0453

paralelo con ruido o con pérdida de datos, la CNN y la MLP tienen un desempeño muy pobre, comparándolo con el obtenido por la $CVCNN$.

Tabla 5.10: Métricas RMSE para el sistema Wiener-Hammerstein encontradas en la literatura.

modelo	RMSE
LSTM [150]	0.011
BLA [151]	0.00607
SVM [152]	0.047
PNLSS [153]	0.00042

La Tabla 5.10 muestra los resultados obtenidos por otros métodos propuestos en la literatura para este mismo sistema, la comparación se realiza únicamente con la métrica $RMSE$.

De esta tabla podemos observar que la $CVCNN$ tienen resultados similares para el modelo serie-paralelo, el cual es el utilizado por los otros métodos. De manera similar que para el sistema del horno de gas, resultados con ruido o pérdida de datos no se encontraron en la literatura para poder realizar la comparación correspondiente. Como en el sistema anterior, únicamente se compara el valor $RMSE$ y se toma en cuenta otras cuestiones computacionales que pueden beneficiar o perjudicar a los métodos propuestos.

Para el sistema no lineal de primer orden los resultados para los mismos cuatro casos se presentan en la Tabla 5.11, Tabla 5.12, Tabla 5.13 y Tabla 5.14. Para este último sistema, los resultados fueron consistentes con los anteriores, presentando mejor desempeño la $CVCNN$ en las cuatro comparaciones con la mayoría de los índices de desempeño

Tabla 5.11: Métricas de desempeño para el sistema no lineal de primer orden usando el modelo serie-paralelo.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.9993	0.9998	3.97×10^{-5}	0.0061	0.0063
MLP	0.9501	0.9884	0.0030	0.0585	0.0551
CNN	0.9936	0.9984	3.90×10^{-4}	0.0222	0.0198

Tabla 5.12: Métricas de desempeño para el sistema no lineal de primer orden usando el modelo paralelo.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.9179	0.9829	0.0050	0.0799	0.0707
MLP	-1.3991	0.7553	0.1461	0.3007	0.3822
CNN	0.5012	0.9167	0.0304	0.2033	0.1743

Con base en las tablas, se puede notar que tanto la $CVCNN$ como la MLP tienen buen desempeño cuando se trata del modelo serie-paralelo y paralelo. Cuando se presentan pertur-

Tabla 5.13: Métricas de desempeño para el sistema no lineal de primer orden usando el modelo serie-paralelo con ruido en los datos.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.9122	0.9758	0.3111	0.6106	0.5577
MLP	0.8710	0.9645	0.4572	0.7495	0.6761
CNN	0.8915	0.9710	0.3846	0.6767	0.6201

Tabla 5.14: Métricas de desempeño para el sistema no lineal de primer orden usando el modelo serie-paralelo con pérdida de datos.

	R^2	WI	MSE	MAE	RMSE
CVCNN	0.9760	0.9940	0.0850	0.2423	0.2915
MLP	-2.1640	0.5257	11.2123	1.0461	3.3485
CNN	0.9560	0.9895	0.1561	0.1512	0.3951

baciones en los datos, ambas CNN se desempeñan mejor que la MLP. Estas afirmaciones se realizan considerando el índice $RMSE$.

Dado la naturaleza arbitraria de este sistema, encontrar resultados para este en la literatura es complicado, por lo que la comparación con métodos existentes es omitida para este último sistema.

En general se puede ver las ventajas de la CVCNN respecto de la MLP y de la CNNt. Cuando trabajamos con datos de entrada únicamente, es decir ocupamos un modelo paralelo, tanto la MLP y la CNN su estructura es insuficiente para poder realizar la tarea con un buen desempeño. Caso similar cuando se ocupa el modelo serie-paralelo con perturbaciones o que los datos estén incompletos, la CVCNN presenta mejor desempeño, utilizando los índices presentados en las tablas, que la MLP y la CNNt. Esta es una de las contribuciones de este trabajo, presentar la estructura de la CVCNN para utilizar en los casos ya mencionados.

Capítulo 6

Conclusiones

Este trabajo sirvió para mostrar ventajas de las redes neuronales convolucionales para la identificación de sistemas no lineales

En el contexto de la identificación de sistemas mediante modelos de caja como lo son las redes neuronales, las perturbaciones externas, ya sean ruido de medición o la pérdida de datos, son de los principales problemas con los que nos enfrentamos para poder definir un modelo, podemos realizar un procesamiento de los datos antes de emplearlos para la obtención del modelo, como puede ser una etapa de filtrado para disminuir el ruido o alguna extrapolación de datos para completarlos. Sin embargo, este procesamiento extra a los datos requiere que se tenga un conocimiento más profundo del sistema al momento de aplicarlo, dado que podemos alterar las interrelaciones de los datos al aplicar técnicas de filtrado correctamente.

Los algoritmos propuestos en este trabajo intentan reducir estos problemas, al ocupar los datos como fueron obtenidos directamente de los sistemas a modelar. Como se pudo observar en las secciones de simulación. La CNN tanto en el dominio de la frecuencia como la CNN con valores complejos presentaron un mejor desempeño cuando los datos presentaban ruido de medición comparado con otras redes neuronales.

El tipo de datos que tengamos serán de suma importancia para elegir qué arquitectura es la mas conveniente para utilizar. En el caso de tener ruido de medición, como es el caso

de la detección de daños en edificios, la arquitectura mas conveniente será la del dominio de la frecuencia como se pudo observar.

Utilizar la CNN en el dominio de la frecuencia conlleva ventajas sobre la CNN en el dominio del tiempo, como lo es la simplificación de la operación de convolución por un producto reduce el tiempo de computo de manera considerable, lo cual es importante en la aplicación de detección de daños.

En general, la CNN para la identificación de sistemas resulta una buena alternativa para esta tarea en particular, podemos obtener valores muy competitivos en cuanto a las métricas de desempeño se refiere comparándolas con distintos métodos propuestos en otros trabajos los cuales tengan el mismo objetivo.

Capítulo 7

Trabajo a futuro

Ya que se ha verificado que los tres tipos de arquitecturas propuestas de la CNN son una alternativa para la identificación de sistemas, podemos dar el paso para aplicarla como controlador neuronal.

El planteamiento del controlador sería utilizando dos CNN, una para el modelado del sistema y otra como el controlador. Cada una de las redes entrenará utilizando el algoritmo de retro propagación, ambos con el mismo objetivo, minimizar el error de seguimiento entre nuestro modelo y la señal de referencia. Como conocemos la estructura de la CNN para el modelado del sistema, podemos pasar el error a través de ella y así entrenar al controlador.

Bibliografía

[1] L. Ljung, "System identification," *Wiley Encyclopedia of Electrical and Electronics Engineering*, pp. 1–19, 1999.

[2] G. Horvath, "Neural networks in system identification," *Nato Science Series Sub Series III Computer And Systems Sciences*, vol. 185, pp. 43–78, 2003.

[3] K. Parthasarathy and K. Narendra, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[4] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[5] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[6] S. Cohen, "Chapter 2 - the basics of machine learning: strategies and techniques," in *Artificial Intelligence and Deep Learning in Pathology*, S. Cohen, Ed. Elsevier, 2021, pp. 13–40. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780323675383000026

[7] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[8] B. Fernandez, A. G. Parlos, and W. K. Tsai, "Nonlinear dynamic system identification using artificial neural networks (anns)," in *1990 IJCNN International Joint Conference on Neural Networks*. IEEE, 1990, pp. 133–141.

[9] H. V. H. Ayala and L. dos Santos Coelho, "Cascaded evolutionary algorithm for nonlinear system identification based on correlation functions and radial basis functions neural networks," *Mechanical Systems and Signal Processing*, vol. 68, pp. 378–393, 2016.

[10] I. Livieris and P. Pintelas, "A survey on algorithms for training artificial neural networks," *Department of Mathematics, University of Patras. Tech. Rep*, 2008.

[11] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön, "Deep learning and system identification," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1175–1181, 2020.

[12] A. Brusaferri, M. Matteucci, P. Portolani, and S. Spinelli, "Nonlinear system identification using a recurrent network in a bayesian framework," in *2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, vol. 1. IEEE, 2019, pp. 319–324.

[13] H. Khodabandehlou and M. S. Fadali, "Nonlinear system identification using neural networks and trajectory-based optimization," *arXiv preprint arXiv:1804.10346*, 2018.

[14] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[15] F. Rosenblatt, "Principles of neurodynamics. perceptrons and the theory of brain mechanisms," Cornell Aeronautical Lab Inc Buffalo NY, Tech. Rep., 1961.

[16] D. E. Rumelhart, "Parallel distributed processing: Explorations in the microstructure of cognition," *Learning internal representations by error propagation*, vol. 1, pp. 318–362, 1986.

[17] [Online]. Available: https://jarroba.com/introduccion-a-las-redes-neuronales-el-perceptron-video/

[18] R. Manger and K. Puljić, "Multilayer perceptrons and data compression," *Computing and Informatics*, vol. 26, no. 1, pp. 45–62, 2012.

[19] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.

[20] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[21] O. Nelles, *Nonlinear system identification: from classical approaches to neural networks and fuzzy models*. Springer Science & Business Media, 2013.

[22] W. R. Gilks, S. Richardson, and D. Spiegelhalter, *Markov chain Monte Carlo in practice*. Chapman and Hall/CRC, 1995.

[23] H. K. Khalil and J. W. Grizzle, *Nonlinear systems*. Prentice hall Upper Saddle River, NJ, 2002, vol. 3.

[24] K. J. Aström and R. M. Murray, *Feedback systems: an introduction for scientists and engineers*. Princeton university press, 2010.

[25] J.-P. Noël and G. Kerschen, "Nonlinear system identification in structural dynamics: 10 more years of progress," *Mechanical Systems and Signal Processing*, vol. 83, pp. 2–35, 2017.

[26] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[27] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[28] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[29] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[30] K. Godfrey and J. DiStefano III, "Identifiability of model parameter," *IFAC Proceedings Volumes*, vol. 18, no. 5, pp. 89–114, 1985.

[31] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu, "Deep blue," *Artificial intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002.

[32] S. Mor-Yosef, A. Samueloff, B. Modan, D. Navot, and J. G. Schenker, "Ranking the risk factors for cesarean: logistic regression analysis of a nationwide study." *Obstetrics and gynecology*, vol. 75, no. 6, pp. 944–947, 1990.

[33] H. Zhang, "The optimality of naive bayes," *AA*, vol. 1, no. 2, p. 3, 2004.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[35] Y. Chen, Z. Lin, X. Zhao, G. Wang, and Y. Gu, "Deep learning-based classification of hyperspectral data," *IEEE Journal of Selected topics in applied earth observations and remote sensing*, vol. 7, no. 6, pp. 2094–2107, 2014.

[36] E. Busseti, I. Osband, and S. Wong, "Deep learning for time series modeling," *Technical report, Stanford University*, 2012.

[37] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[38] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *European conference on machine learning*. Springer, 1998, pp. 137–142.

[39] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D. Haussler, "Support vector machine classification and validation of cancer tissue samples using microarray expression data," *Bioinformatics*, vol. 16, no. 10, pp. 906–914, 2000.

[40] O. Martins, G. Braz Junior, A. Corrêa Silva, A. Cardoso de Paiva, M. Gattass *et al.*, "Detection of masses in digital mammograms using k-means and support vector machine," *ELCVIA: electronic letters on computer vision and image analysis*, vol. 8, no. 2, pp. 039–50, 2009.

[41] S. P. Das and S. Padhy, "A novel hybrid model using teaching–learning-based optimization and a support vector machine for commodity futures index forecasting," *International Journal of Machine Learning and Cybernetics*, vol. 9, no. 1, pp. 97–111, 2018.

[42] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.

[43] G. Dahl, A.-r. Mohamed, G. E. Hinton *et al.*, "Phone recognition with the mean-covariance restricted boltzmann machine," in *Advances in neural information processing systems*, 2010, pp. 469–477.

[44] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[45] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted boltzmann machines for collaborative filtering," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 791–798.

[46] F. Q. Lauzon, "An introduction to deep learning," in *2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA)*. IEEE, 2012, pp. 1438–1439.

[47] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[48] F. Ordóñez and D. Roggen, "Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition," *Sensors*, vol. 16, no. 1, p. 115, 2016.

[49] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[50] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5-6, pp. 602–610, 2005.

[51] C. Zhou, C. Sun, Z. Liu, and F. Lau, "A c-lstm neural network for text classification," *arXiv preprint arXiv:1511.08630*, 2015.

[52] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzel, "Learning to diagnose with lstm recurrent neural networks," *arXiv preprint arXiv:1511.03677*, 2015.

[53] P. Sermanet and Y. LeCun, "Traffic sign recognition with multi-scale convolutional networks." in *IJCNN*, 2011, pp. 2809–2813.

[54] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[55] D. H. Hubel and T. N. Wiesel, "Receptive fields and functional architecture of monkey striate cortex," *The Journal of physiology*, vol. 195, no. 1, pp. 215–243, 1968.

[56] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, "Subject independent facial expression recognition with robust face detection using a convolutional neural network," *Neural Networks*, vol. 16, no. 5-6, pp. 555–559, 2003.

[57] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[58] Y. Wang, C. Xu, S. You, D. Tao, and C. Xu, "Cnnpack: Packing convolutional neural networks in the frequency domain." in *NIPS*, vol. 1, 2016, p. 3.

[59] L. Mao, X. Li, D. Yang, and R. Zhang, "Convolutional feature frequency adaptive fusion object detection network," *Neural Processing Letters*, pp. 1–16, 2021.

[60] S. Soleymanpour, H. Sadr, and H. Beheshti, "An efficient deep learning method for encrypted traffic classification on the web," in *2020 6th International Conference on Web Research (ICWR)*. IEEE, 2020, pp. 209–216.

[61] F. Sultana, A. Sufian, and P. Dutta, "Advancements in image classification using convolutional neural network," in *2018 Fourth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. IEEE, 2018, pp. 122–129.

[62] S. S. Yadav and S. M. Jadhav, "Deep convolutional neural network based medical image classification for disease diagnosis," *Journal of Big Data*, vol. 6, no. 1, pp. 1–18, 2019.

[63] G. Altan, A. Yayık, and Y. Kutlu, "Deep learning with convnet predicts imagery tasks through eeg," *Neural Processing Letters*, pp. 1–16, 2021.

[64] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[65] N. Hatami, Y. Gavet, and J. Debayle, "Classification of time-series images using deep convolutional neural networks," in *Tenth international conference on machine vision*

(ICMV 2017), vol. 10696. International Society for Optics and Photonics, 2018, p. 106960Y.

[66] A. Borovykh, S. Bohte, and C. W. Oosterlee, "Conditional time series forecasting with convolutional neural networks," *arXiv preprint arXiv:1703.04691*, 2017.

[67] H. J. Sadaei, P. C. d. L. e Silva, F. G. Guimarães, and M. H. Lee, "Short-term load forecasting by using a combined method of convolutional neural networks and fuzzy time series," *Energy*, vol. 175, pp. 365–377, 2019.

[68] M. Binkowski, G. Marti, and P. Donnat, "Autoregressive convolutional neural networks for asynchronous time series," in *International Conference on Machine Learning*. PMLR, 2018, pp. 580–589.

[69] S. Namasudra, S. Dhamodharavadhani, and R. Rathipriya, "Nonlinear neural network based forecasting model for predicting covid-19 cases," *Neural Processing Letters*, pp. 1–21, 2021.

[70] S. Genc, "Parametric system identification using deep convolutional neural networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2112–2119.

[71] Y. Kang, S. Chen, X. Wang, and Y. Cao, "Deep convolutional identifier for dynamic modeling and adaptive control of unmanned helicopter," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 2, pp. 524–538, 2018.

[72] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[73] S. Kim and R. Casper, "Applications of convolution in image processing with matlab," *University of Washington*, pp. 1–20, 2013.

[74] Z. Xiang, R. Zhang, and P. Seeling, "Chapter 19 - machine learning for object detection," in *Computing in Communication Networks*, F. H. Fitzek, F. Granelli,

and P. Seeling, Eds. Academic Press, 2020, pp. 325–338. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128204887000347

[75] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[76] J. Bouvrie, "Notes on convolutional neural networks," 2006.

[77] Y. LeCun *et al.*, "Generalization and network design strategies," in *Connectionism in perspective*. Citeseer, 1989, vol. 19.

[78] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[79] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International conference on artificial neural networks*. Springer, 2010, pp. 92–101.

[80] R. Briega, "Matemáticas, análisis de datos y python," 2018.

[81] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," in *European Conference on Computer Vision*. Springer, 2012, pp. 73–86.

[82] K. Fukushima and S. Miyake, "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position," *Pattern recognition*, vol. 15, no. 6, pp. 455–469, 1982.

[83] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[84] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on neural networks*, vol. 1, no. 1, pp. 4–27, 1990.

[85] D. Ruta and B. Gabrys, "Neural network ensembles for time series prediction," in *2007 International Joint Conference on Neural Networks*. IEEE, 2007, pp. 1204–1209.

[86] C.-M. Kuan and K. Hornik, "Convergence of learning algorithms with constant learning rates," *IEEE Transactions on Neural Networks*, vol. 2, no. 5, pp. 484–489, 1991.

[87] J. C. A. Barata and M. S. Hussein, "The moore–penrose pseudoinverse: A tutorial review of the theory," *Brazilian Journal of Physics*, vol. 42, no. 1-2, pp. 146–165, 2012.

[88] M. D. S. Torres, "La inversa de penrose," in *Anales de estudios económicos y empresariales*, no. 1. Servicio de Publicaciones, 1986, pp. 299–308.

[89] E. De la Rosa and W. Yu, "Randomized algorithms for nonlinear system identification with deep learning modification," *Information Sciences*, vol. 364, pp. 197–212, 2016.

[90] J. Schoukens and L. Ljung, "Wiener-hammerstein benchmark," 2009.

[91] J. Noël and M. Schoukens, "Hysteretic benchmark with a dynamic nonlinearity," in *Workshop on nonlinear system identification benchmarks*, 2016, pp. 7–14.

[92] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[93] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Transactions on fuzzy systems*, vol. 1, no. 1, p. 7, 1993.

[94] L. Wang and R. Langari, "Complex systems modeling via fuzzy logic," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 100–106, 1996.

[95] S. W. Doebling, C. Farrar, and M. B. Prime, "A summary review of vibration-based damage identification methods," *The Shock and Vibration Digest*, vol. 30(2), pp. 1–34, 1998.

[96] E. P. Carden and P. Fanning, "Vibration based condition monitoring: A review," *Structural Health Monitoring*, vol. 3, pp. 355–377, 2004.

[97] S. Das, P. Saha, and S. Patro, "Vibration-based damage detection techniques used for health monitoring of structures: a review," *Journal of Civil Structural Health Monitoring*, vol. 6(3), pp. 477–507, 2016.

[98] X. Kong, C.-S. Cai, and J. Hu, "The state-of-the-art on framework of vibration-based structural damage identification for decision making," *Applied Sciences*, vol. 7(5), pp. 497–510, 2017.

[99] H. Y. Hwang and C. Kim, "Damage detection in structures using a few frequency response measurements," *Journal of Sound and Vibration*, vol. 270(1-2), pp. 1–14, 2004.

[100] J.-T. Kim, Y.-S. Ryu, H.-M. Cho, and N. Stubbs, "Damage identification in beam-type structures: frequency-based method vs mode-shape-based method," *Engineering Structures*, vol. 25(1), pp. 57–67, 2003.

[101] Q. Huang, Y. Xu, J. Li, Z. Su, and H. Liu, "Structural damage detection of controlled building structures using frequency response function," *Journal of Sound and Vibration*, vol. 331(15), pp. 3476–3492, 2012.

[102] J. F. Clinton, S. C. Bradford, T. H. Heaton, and J. Favela, "The observed wander of the natural frequencies in a structure," *Bulletin of the Seismological Society of America*, vol. 96, pp. 237–257, 2006.

[103] R. D. Nayeri, S. F. Masri, R. G. Ghanem, and R. L. Nigbor, "A novel approach for the structural identification and monitoring of a full-scale 17-story building based on

ambient virbation measurement," *Smart Materials ans Structures*, vol. 17(2), pp. 1–19, 2008.

[104] M. Herak and D. Herak, "Continuous monitoring of dynamic parameters of the dgfsm building (zagreb, croatia)," *Bulletin of Earthquake Engineering*, vol. 8(3), pp. 657–669, 2010.

[105] A. W. Smyth, S. F. Masri, A. G. Chassiakos, and T. K. Caughey, "On-line parametric identification of mdof nonlinear hysteretic systems," *Journal of Engineering Mechanics*, vol. 125, pp. 133–142, 1999.

[106] F. Ikhouane, V. Mañosa, and J. Rodellar, "Adaptive control of a hysteretic structural system," *Automatica*, vol. 41, pp. 225–231, 2005.

[107] R. Garrido and F. J. Rivero-Angeles, "Hysteresis and parameter estimation of mdof systems by a continuous-time least squares method," *Journal of Earthquake Engineering*, vol. 10, pp. 237–264, 2006.

[108] C. R. Farrar, K. Worden, M. D. Todd, G. Park, J. Nichols, D. E. Adams, M. T. Bement, and K. Farinholt, "Nonlinear system identification for damage detection," *Report LA-14353, Los Alamos National Laboratory (LANL), Los Alamos, NM.*, pp. 1–161, 2007.

[109] E. N. Chatzi, A. W. Smyth, and S. F. Masri, "Experimental application of on-line parametric identification for nonlinear hysteretic systems with model uncertainty," *Structural Safety*, vol. 32, pp. 326–337, 2010.

[110] H. Sohn, C. Farrar, F. Hemez, D. S. Devin, W. S. Daniel, R. N. Brett, and J. C. Jerry, "A review of structural health monitoring literature: 1996-2001," *Los Alamos National Laboratory Report, LA-13976-MS*, pp. 1–331, 2003.

[111] H. Sohn and C. R. Farrar, "Damage diagnosis using time series analysis of vibration signals," *Smart Materials and Structures,*, vol. 10(3), pp. 446–451, 2001.

[112] K. K. Nair, A. S. Kiremidjian, and K. H. Law, "Time series-based damage detection and localization algorithm with application to the asce benchmark structure," *Journal of Sound and Vibration*, vol. 291, pp. 349–368, 2006.

[113] Y.-Y. Liu, Y.-F. Ju, C.-D. Duan, and X.-F. Zhao, "Structure damage diagnosis using neural network and feature fusion," *Engineering Applications of Artificial Intelligence*, vol. 24, pp. 87–92, 2011.

[114] O. Abdeljaber, O. Avci, S. Kiranyaz, M. Gabbouj, and D. J. Inman, "Real-time vibration-based structural damage detection using one-dimensional convolutional neural networks," *Journal of Sound and Vibration*, vol. 388, pp. 154–170, 2017.

[115] Y. K. Wen, "Method for random vibration of hysteretic system," *Journal of the Engineering Mechanics Division*, vol. 102(2), pp. 249—-263, 1976.

[116] F. Ma, C. H. Ng, and N. Ajavakom, "On system identification and response prediction of degrading structures," *Structural Control and Health Monitoring*, vol. 13, pp. 347–364, 2006.

[117] F. Ikhouane, V. M. nosa, and J. Rodellar, "Dynamic properties of the hysteretic bouc-wen model," *Systems and Control Letters*, vol. 56(3), pp. 197–205, 2007.

[118] Y.-K. Wen, "Method for random vibration of hysteretic systems," *Journal of the engineering mechanics division*, vol. 102, no. 2, pp. 249–263, 1976.

[119] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," *Mechanical systems and signal processing*, vol. 151, p. 107398, 2021.

[120] E. Jacobsen and R. Lyons, "The sliding dft," *IEEE Signal Processing Magazine*, vol. 20, no. 2, pp. 74–80, 2003.

[121] S. K. Mitra and J. F. Kaiser, *Handbook for digital signal processing*. John Wiley & Sons, Inc., 1993.

[122] D. A. Kopriva, *Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers*. Springer Science & Business Media, 2009.

[123] J. W. Cooley, P. A. Lewis, and P. D. Welch, "The fast fourier transform and its applications," *IEEE Transactions on Education*, vol. 12, no. 1, pp. 27–34, 1969.

[124] W. F. Schmidt, M. A. Kraaijveld, R. P. Duin *et al.*, "Feed forward neural networks with random weights," in *International Conference on Pattern Recognition*. IEEE COMPUTER SOCIETY PRESS, 1992, pp. 1–1.

[125] B. Igelnik and Y.-H. Pao, "Stochastic choice of basis functions in adaptive function approximation and the functional-link net," *IEEE Transactions on Neural Networks*, vol. 6, no. 6, pp. 1320–1329, 1995.

[126] S. Scardapane and D. Wang, "Randomness in neural networks: an overview," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 2, p. e1200, 2017.

[127] M. Li and D. Wang, "Insights into randomized algorithms for neural networks: Practical issues and common pitfalls," *Information Sciences*, vol. 382, pp. 170–178, 2017.

[128] J. Tapson and A. van Schaik, "Learning the pseudoinverse solution to network weights," *Neural Networks*, vol. 45, pp. 94–100, 2013.

[129] N. Cheney, M. Schrimpf, and G. Kreiman, "On the robustness of convolutional neural networks to internal architecture and weight perturbations," *arXiv preprint arXiv:1703.08245*, 2017.

[130] A. M. Saxe, P. W. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Y. Ng, "On random weights and unsupervised feature learning." in *ICML*, vol. 2, no. 3, 2011, p. 6.

[131] R. M. Gray *et al.*, "Toeplitz and circulant matrices: A review," *Foundations and Trends® in Communications and Information Theory*, vol. 2, no. 3, pp. 155–239, 2006.

[132] O. Rippel, J. Snoek, and R. P. Adams, "Spectral representations for convolutional neural networks," in *Advances in neural information processing systems*, 2015, pp. 2449–2457.

[133] P. Virtue, X. Y. Stella, and M. Lustig, "Better than real: Complex-valued neural nets for mri fingerprinting," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3953–3957.

[134] T. Xiong, Y. Bao, Z. Hu, and R. Chiong, "Forecasting interval time series using a fully complex-valued rbf neural network with dpso and pso algorithms," *Information Sciences*, vol. 305, pp. 77–92, 2015.

[135] J. S. Dramsch, M. Lüthje, and A. N. Christensen, "Complex-valued neural networks for machine learning on non-stationary physical data," *Computers & Geosciences*, vol. 146, p. 104643, 2021.

[136] N. Guberman, "On complex valued convolutional neural networks," *arXiv preprint arXiv:1602.09046*, 2016.

[137] N. Benvenuto and F. Piazza, "On the complex backpropagation algorithm," *IEEE Transactions on Signal Processing*, vol. 40, no. 4, pp. 967–969, 1992.

[138] A. Hirose and S. Yoshida, "Comparison of complex-and real-valued feedforward neural networks in their generalization ability," in *International Conference on Neural Information Processing*. Springer, 2011, pp. 526–531.

[139] A. Hirose, *Complex-valued neural networks: theories and applications*. World Scientific, 2003, vol. 5.

[140] Y. Quan, Y. Chen, Y. Shao, H. Teng, Y. Xu, and H. Ji, "Image denoising using complex-valued deep cnn," *Pattern Recognition*, vol. 111, p. 107639, 2021.

[141] Z. Zhang, H. Wang, F. Xu, and Y.-Q. Jin, "Complex-valued convolutional neural network and its application in polarimetric sar image classification," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 12, pp. 7177–7188, 2017.

[142] M. Meyer, G. Kuschk, and S. Tomforde, "Complex-valued convolutional neural networks for automotive scene classification based on range-beam-doppler tensors," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–6.

[143] X. Yao, X. Shi, and F. Zhou, "Human activities classification based on complex-value convolutional neural network," *IEEE Sensors Journal*, vol. 20, no. 13, pp. 7169–7180, 2020.

[144] H. Gu, G. Qing, Y. Wang, S. Hong, G. Gui, H. Gacanin, and F. Adachi, "Deep complex-valued convolutional neural network for drone recognition based on rf fingerprinting," 2020.

[145] R. Hongyo, Y. Egashira, and K. Yamaguchi, "Deep neural network based predistorter with relu activation for doherty power amplifiers," in *2018 Asia-Pacific Microwave Conference (APMC)*. IEEE, 2018, pp. 959–961.

[146] X. Tan, M. Li, P. Zhang, Y. Wu, and W. Song, "Complex-valued 3-d convolutional neural network for polsar image classification," *IEEE Geoscience and Remote Sensing Letters*, vol. 17, no. 6, pp. 1022–1026, 2019.

[147] G. James and D. Burley, *Matemáticas avanzadas para ingeniería*. Pearson Educación, 2002.

[148] B. B. Ustundag and A. Kulaglic, "High-performance time series prediction with predictive error compensated wavelet neural networks," *IEEE Access*, vol. 8, pp. 210 532–210 541, 2020.

[149] K. George and P. Mutalik, "A multiple model approach to time-series prediction using an online sequential learning algorithm," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 5, pp. 976–990, 2017.

[150] J. Gonzalez and W. Yu, "Non-linear system modeling using lstm neural networks," *IFAC-PapersOnLine*, vol. 51, no. 13, pp. 485–489, 2018.

[151] M. A. H. Shaikh and K. Barbé, "Wiener–hammerstein system identification: A fast approach through spearman correlation," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 5, pp. 1628–1636, 2019.

[152] A. Marconato and J. Schoukens, "Identification of wiener-hammerstein benchmark data by means of support vector machines," *IFAC Proceedings Volumes*, vol. 42, no. 10, pp. 816–819, 2009.

[153] J. Paduart, L. Lauwers, R. Pintelon, and J. Schoukens, "Identification of a wiener–hammerstein system using the polynomial nonlinear state space approach," *Control Engineering Practice*, vol. 20, no. 11, pp. 1133–1139, 2012.