UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA CIVIL

TEMA:

APLICACIÓN DE REDES NEURONALES RECURRENTES PARA ACELERAR EL ANÁLISIS NO LINEAL DE ESTRUCTURAS

AUTOR:

Coello Choez Bryan Xavier

Trabajo de titulación previo a la obtención del título de INGENIERO

TUTOR:

Barros Cabezas José Andrés

Guayaquil, Ecuador

14 de septiembre del 2021

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA CIVIL

CERTIFICACIÓN

Certificamos que el presente trabajo de titulación, fue realizado en su totalidad por **Coello Choez, Bryan Xavier**, como requerimiento para la obtención del título de **Ingeniero**.

TUTOR

f. _____
Ing. Barros Cabezas José Andrés, M. Sc.

DIRECTOR DE LA CARRERA

f. _____
Ing. Alcívar Bastidas Stefany Esther, M.Sc.

Guayaquil, a los 14 del mes de septiembre del año 2021

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA

CARRERA DE INGENERÍA CIVIL

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Coello Choez, Bryan Xavier**

DECLARO QUE:

El Trabajo de Titulación, **Aplicación de redes neuronales recurrentes para acelerar el análisis no lineal de estructuras** previo a la obtención del título de **Ingeniero**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

Guayaquil, a los 14 del mes de septiembre del año 2021

EL AUTOR

f. _____

Coello Choez Bryan Xavier

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA

CARRERA DE INGENERÍA CIVIL

AUTORIZACIÓN

Yo, **Coello Choez, Bryan Xavier**

Autorizo a la Universidad Católica de Santiago de Guayaquil a la **publicación** en la biblioteca de la institución del Trabajo de Titulación, **Aplicación de redes neuronales recurrentes para acelerar el análisis no lineal de estructuras,** cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

Guayaquil, a los 14 del mes de septiembre del año 2021

EL AUTOR:

f. _____

Coello Choez Bryan Xavier

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA CIVIL

REPORTE URKUND

URKUND

Urkund Analysis Result

Analysed Document: Coello_Bryan_FINAL.docx (D112186810)
Submitted: 9/7/2021 4:18:00 PM
Submitted By: claglas@hotmail.com
Significance: 2 %

Sources included in the report:

TESIS-2-12-70.pdf (D110500504)
https://www.cyberforum.ru/python-beginners/thread2626876.html
http://repositorio.ug.edu.ec/bitstream/redug/52642/1/B-CISC-PTG-1893-2021%20Guerrero%20Alvarado%20Marcos%20Daniel.pdf
https://www.codificandobits.com/blog/tutorial-overfitting-machine-learning-python/
https://qastack.mx/programming/61425296/why-neural-network-predicts-wrong-on-its-own-training-data
https://ichi.pro/es/pronostico-de-series-de-tiempo-predecir-los-precios-de-las-acciones-mediante-un-modelo-lstm-228721719369813
https://riull.ull.es/xmlui/bitstream/handle/915/20619/Aplicacion%20de%20tecnicas%20de%20Machine%20Learning%20a%20un%20problema%20practico%20de%20reposicion%20de%20inventario..pdf?sequence=1

Instances where selected sources appear:

8

Dedico este trabajo a mi madre la ing. Paola Choez y a mi padre el arq. Juan Coello quienes, con su amor, esfuerzo, cariño y paciencia me han permitido hacer realidad uno de mis tantos sueños, ser Ingeniero Civil.

A mi hermana, Nicole Coello por hacerme compañía a lo largo de este camino. Por ultimo, a mi a abuelita Brenda Navas y mi abuelito Santana Choez por su apoyo incondicional.

Las palabras no bastan para expresar lo que siento por mi familia, pero agradezco a Dios por darme una familia tan hermosa como la que tengo.

Esto es para ustedes

AGRADECIMIENTOS

En primer lugar, quiero agradecer a mi mamá, Paola Choez y a mi papá, Juan Coello quienes me han apoyado siempre en cada proyecto, idea u ocurrencia que he tenido. Alentándome a ser una mejor persona y enseñándome a que todo lo bueno en la vida se gana con esfuerzo y dedicación.

Por otro lado, quiero dar las gracias a mi hermana menor, Nicole Coello quien más que mi hermana ha sido como mi segunda madre, siempre cuidándome y siendo un ejemplo de una persona excelente.

A mis abuelitos, mi mamá Brenda y mi papá Santana, por el apoyo emocional y económico que he tenido por parte de ellos para poder hacer realidad este sueño. Les estoy eternamente agradecido.

A mi pareja, Verónica Espinoza por apoyarme siempre, por darme de su amor, por escucharme y más que nada por comprenderme en todo momento.

A mi grupo de amigos más cercanos "El Team", por las noches y madrugadas de estudio, por los consejos que me han brindado y por los viajes que hemos realizado juntos.

A los docentes de la facultad de Ingeniería de la Universidad Católica de Santiago de Guayaquil, por el conocimieno que han impartido a lo largo de la carrera y la experiencia compartida para darme una formación académica excelente.

De igual manera quiero agradecer a mi tutor el ingeniero José Barros, quien me ha brindado su confianza, tiempo, conocimiento y paciencia a lo largo de este ultimo semestre.

UNIVERSIDAD CATÓLICA

DE SANTIAGO DE GUAYAQUIL

FACULTAD DE INGENIERÍA

CARRERA DE INGENERÍA CIVIL

f. _____

Ing. Barros Cabezas José Andrés, M. Sc.

TUTOR

TRIBUNAL DE SUSTENTACIÓN

f. _____

Ing. Carlos Yldefonso Chon Diaz, M. Sc

DECANO DE LA CARRERA

f. _____

Ing. Ponce Vásquez Guillermo, M.Sc

DOCENTE DE LA CARRERA

f. _____

Ing. Casal Rodríguez Xavier Federico, M. Sc.

OPONENTE

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE CÓDIGOS

RESUMEN

El análisis no lineal de estructuras es una herramienta de la ingeniería basada en el desempeño que sirve para la validación de estructuras, ya sea para nuevas estructuras o para evaluar estructuras existentes. La aplicación de este análisis para sistemas estructurales compuestos por una gran cantidad de grados de libertad, como lo son edificios de gran altura, puentes e incluso presas, requieren de mucho tiempo computacional. Los tiempos de cálculo pueden resultar inadmisibles en estudios paramétricos. Para la reducción de este tiempo, se plantea en este trabajo una metodología práctica que consiste en la aplicación de redes neuronales y se da un primer paso en el uso de las redes neuronales recurrentes. Dentro del grupo de las redes neuronales recurrentes se aplica la memoria de largo y corto plazo (LSTM) para acelerar el análisis no lineal de estructuras. Con la creación de una base de datos para el entrenamiento y prueba de la red neuronal, que se utiliza no solo para aprender el comportamiento del análisis, sino también para hacer predicciones confiables en adición al objetivo principal que es la aceleración del análisis en sí. Por demás, la recomendación a las aplicaciones e investigaciones de los diferentes tipos de redes neuronales para la solución de problemas de ingeniería.

Palabras Claves: modelado no lineal, análisis no lineal, aceleración de análisis, redes neuronales, redes neuronales recurrentes, redes neuronales recurrentes LSTM

ABSTRACT

Nonlinear structural analysis is a performance-based engineering tool for the validation of proposed designs either for new structures or to evaluate existing structures. The application of this analysis for structural systems composed of a large number of degrees of freedom, such as high-rise buildings, bridges and even dams, the computational analysis can be time - consuming. Computational burden may result cumbersome for parametric studies. In order to reduce this computational cost, a practical methodology consisting in the application of neural networks and recurrent neural networks (RNN) is proposed in this work. Within the group of recurrent neural networks, long short-term memory (LSTM) is applied to accelerate the nonlinear analysis of structures. By creating a database for training and testing the neural network, it will be used not only to learn the behavior of the analysis, but also to make reliable predictions in addition to the main goal of reducing the computational burden. Furthermore, the recommendation to the applications and research of different types of neural networks for the solution of engineering problems.

Keywords: nonlinear modeling, nonlinear analysis, analysis acceleration, neural networks, recurrent neural networks, LSTM recurrent neural networks

1. INTRODUCCIÓN

1.1 ANTECEDENTES

El aprendizaje automatizado (en inglés, Machine Learning) no tiene una definición establecida; sin embargo, puede ser explicado en función del objetivo con el que se usará. De forma general éste busca realizar algoritmos que aprenden de ejemplos en lugar de algoritmos programados específicamente para realizar una tarea (Cohen, 2021). Entre los tipos de aprendizajes automatizados se encuentran las redes neuronales (NN). Las redes neuronales fueron introducidas en 1943 por Warren McCulloch y Walter Pitts (Wang & Fu, 2009). El nombre está inspirado en la funcionalidad de los cerebros humanos y la acción que realizan las neuronas para poder procesar la información.

Las redes neuronales, aunque no son tan conocidas han sido estudiadas, usadas y a la vez ignoradas debido al conocimiento que debe tenerse u obtenerse para poder crearlas, otra dificultad que existe es la falta o escasez de datos, ya que estos son necesariamente fundamentales para poder entrenar las redes neuronales. En la actualidad, con la facilidad existente para poder acceder a cursos, investigaciones, artículos y programas de uso libre, se ha logrado un incremento en el número de investigaciones y aplicaciones de las NN que anteriormente no se veía y esto se debe a la necesidad presente por querer mejorar y optimizar los métodos convencionales usados, ya sea para el análisis o el diseño de estructuras (Shoham et al., 2018).

Las aplicaciones más frecuentes de las redes neuronales se encuentran desarrolladas por empresas como Google, entre los usos resaltan la traducción de idiomas y el reconocimiento de imágenes. Por otro lado, en el campo de la ingeniería también se han realizado un sin número de aplicaciones (Salehi & Burgueño, 2018), que van desde la resolución numérica de ecuaciones diferenciales hasta temas con un grado de complejidad mayor tales como lo puede ser un análisis estructural e inclusive el desarrollo de redes neuronales de predicción de cargas de pandeo de

cascarones como un ejemplo de las aplicaciones, con un grado de dificultad y elaboración mayor.

La investigación de Sarkar (2008) muestra un estudio de la eficiencia de los sistemas neuronales artificiales en términos de su desempeño y la determinación amplia de las condiciones que son requeridas para su aplicación. Los resultados generales se compararon con los resultados obtenidos con programas convencionales en términos de facilidad de uso e implementación. Esta aplicación, aunque no tan reciente muestra que las redes neuronales son usadas en su mayoría para optimizar el tiempo de procesamiento.

De manera general las aplicaciones de las redes neuronales en la ingeniería son amplias al igual que el tipo de redes neuronales que pueden ser usadas e implementadas para diferentes ramas de la ingeniera. La aplicación de redes neuronales normales por Lavech du Bos, Balabdaoui y Heidenreich (2020) muestran un modelado de curvas de tensión – deformación para un sólido elastoplástico isotrópico. Realizar este modelo en el enfoque convencional, como es mencionado por los autores, requiere de un trabajo teórico y experimental sustancial. Los autores resolvieron el problema de falta de datos mediante la aplicación de una red neuronal normal lo que les permitió tener resultados muy precisos con respecto a la relación tensión – deformación. Una solución a esta escasez puede ser la generación de una base de datos, la misma que puede ser obtenida mediante ensayos. Tener en cuenta que esto a su vez repercute de cierta manera en un presupuesto que debe tenerse para hacer dichos ensayos; sin embargo, es una solución que a futuro será menos costosa con una base de datos ya generada y con ensayos que ya no serán necesarios para la obtención de resultados.

Otros de los tipos de redes neuronales que existen son las redes convolucionales, las mismas son usadas con más frecuencia puesto que estas trabajan con imágenes y pixeles, pero en lo que respecta al uso en la ingeniería son escasas las aplicaciones que se le han dado por el gran número de muestras que son necesarias para que logren dar buenos resultados. En una investigación actual de Yang et al (2020) se da a conocer

la aplicación de este tipo de redes neuronales para la predicción de curvas tensión-deformación de microestructura compuesta dando como conclusión resultados precisos y acelerando el proceso para la obtención del modelo, además de permitir el diseño compuesto con respecto a la respuesta mecánica más allá del límite elástico, que antes computacionalmente no era viable por el tiempo que este tomaba.

En lo que respecta el análisis de estructuras, el análisis no lineal cada vez es más usado. Según Krawinkler, H. (2006) en la ingeniería sísmica el uso del análisis no lineal va en creciente ya que es aplicado como herramienta para la evaluación del desempeño de las estructuras para los diferentes niveles de seguridad, siendo estos de vida y prevención de colapso. La problemática que más influye en el análisis no lineal de estructuras por métodos convencionales es el tiempo empleado para poder realizarlo, debido a que éste estará sujeto al tamaño de la estructura y el sistema estructural usado. Dado que la ingeniería busca replicar lo que ocurre con las estructuras durante un evento sísmico, los modelos numéricos se vuelven cada vez más complejos afectando en los tiempos de análisis; por lo tanto, conviene el uso de nuevas técnicas para poder acortar esos tiempos y poder apuntar a una evaluación eficiente del desempeño estructural.

1.2 OBJETIVOS

1.2.1 *Objetivo General*

Evaluar la posibilidad de acelerar el análisis no lineal de una estructura usando redes neuronales recurrentes

1.2.2 *Objetivo Específico*

- Generar una base de datos de entrenamiento para un modelo no lineal cíclico que pudiera incluir efectos de degradación
- Entrenar una red neuronal recurrente para imitar el comportamiento del modelo no lineal
- Evaluar las diferencias obtenidas entre el modelo con las redes neuronales (modelo tipo caja negra) y el modelo no lineal, además evaluar la posibilidad de usar el modelo recurrente para acelerar los resultados de un análisis más complejo

1.3 JUSTIFICACIÓN

Acelerar el cálculo computacional de modelos no lineales complejos. El siguiente trabajo de titulación busca mediante la aplicación de redes neuronales artificiales disminuir el tiempo que toma obtener resultados de modelos. Estos modelos tienen el objetivo de realizar análisis no lineal de un grado de complejidad alto, lo mismo que juega un papel importante en lo que a tiempo se refiere. Además, busca constituir un aporte para la rama de la ingeniería estructural ya que aparte de querer acelerar el proceso de estos modelos, tiene como fin que los ingenieros lleguen a considerar como una opción el uso de las redes neuronales.

CAPÍTULO II

2. MARCO TEÓRICO

2.1 ANÁLISIS NO LINEAL

Generalmente, los edificios son diseñados para cumplir con una resistencia por carga lateral, con estimaciones realizadas mediante análisis elásticos; sin embargo, la mayoría de estos edificios experimentará deformaciones inelásticas significativas bajo grandes terremotos. Los métodos de diseño modernos basados en desempeño requieren formas de determinar el comportamiento realista de las estructuras en dichas condiciones. Ahora, debido a los avances en las tecnologías computacionales y los datos de pruebas disponibles, los análisis no lineales proporcionan los medios para calcular la respuesta estructural más allá del rango elástico, incluyendo el deterioro de la resistencia y la rigidez asociado con el comportamiento inelástico de los materiales y los grandes desplazamientos (Deierlein et al., 2010).

El realizar un análisis no lineal implican un esfuerzo significativamente mayor y deben abordarse con objetivos específicos previamente planteados. Los casos típicos en los que se aplica el análisis no lineal en la práctica de la ingeniería sísmica estructural según Deierlein et al (2010) son:

1. Evaluar y diseñar soluciones de reacondicionamiento sísmico para edificios existentes

2. Diseñar nuevas construcciones que empleen materiales estructurales, sistemas u otras características que no se ajusten a los requisitos del código de construcción actual

3. Evaluar el desempeño de los edificios para los requisitos específicos de los propietarios o partes interesadas.

2.1.1 *Historia*

Las primeras aplicaciones prácticas generalizadas del análisis no lineal en la ingeniería sísmica fueron para evaluar y modernizar edificios existentes. Lo que fueron las primeras pautas importantes sobre la aplicación del análisis no lineal fueron las publicadas en FEMA 273 Directrices del NEHRP para la rehabilitación sísmica de edificios (FEMA, 1997) y ATC 40 Evaluación sísmica y reacondicionamiento de edificios de hormigón (Niewiarowski & Rojahn, 1996). A causa del estado del conocimiento y las tecnologías de la computación en el momento de su publicación (mediados de los años 90), estos documentos se centraron principalmente en el análisis estático no lineal (pushover). Posterior a eso, se han trasladado a ASCE 41 Rehabilitación sísmica de edificios existentes (ASCE, 2007), y se han propuesto mejoras en FEMA 440 Mejora de los procedimientos de análisis sísmico estático no lineal (FEMA, 2005) y FEMA P440A Efectos de la degradación de la resistencia y la rigidez en la respuesta sísmica (FEMA, 2009).

Es importante tener en consideración que, si bien la ASCE 41 y sus documentos relacionados al tema tienen un enfoque principal en la renovación de edificios existentes, la guía de análisis no lineal y el modelado de componentes, los criterios de aceptación presentados en dichos documentos se pueden aplicar al diseño de nuevas estructuras, siempre y cuando los criterios de aceptación elegidos proporcionen los niveles de desempeño esperados para los nuevos edificios diseño en ASCE 7 (Deierlein et al., 2010).

Al mismo tiempo se estaban desarrollando FEMA 273 y ATC 40, en donde también se estaban introduciendo conceptos de análisis no lineal en los métodos para la evaluación del riesgo sísmico, siendo el más conocido HAZUS. Particularmente, el módulo de evaluación de pérdidas específicas de edificios de HAZUS emplea métodos de análisis estático no lineal para desarrollar funciones de fragilidad de daños por terremotos para edificios en

“Metodología de estimación de pérdidas por terremotos”, HAZUS99-SR2, Módulo de construcción de ingeniería avanzada (FEMA, 2002).

2.1.2 *Consideraciones*

Una vez que los objetivos del análisis no lineal y la base del diseño son definidos, se debe identificar los parámetros de demanda y los criterios de aceptación, con la finalidad de evaluar cuantitativamente los niveles de desempeño. Generalmente, los parámetros de demanda incluyen fuerzas máximas, deformaciones tanto en componentes estructurales como no estructurales, desniveles y aceleración de pisos.

Realizar un análisis no lineal requiere pensar en el comportamiento inelástico y los estados límite, que a su vez dependerán de las fuerzas y deformaciones. En adición, requieren la definición de modelos capaces de capturar la respuesta fuerza – deformación de los componentes y sistemas en función de las propiedades de resistencia y rigidez.

Una consideración importante según Deirlein et al (2010) es tener expectativas previas y claras sobre aquellas zonas de la estructura donde se espera que exista deformaciones inelásticas y utilizar el análisis realizado para:

- Confirmar las zonas donde ocurrirán las deformaciones inelásticas.
- Caracterizar la demanda de las deformaciones en los elementos elásticos
- Caracterizar la demanda de fuerza en los elementos que no ceden.

Lo que buscan estas consideraciones es llegar a un diseño por capacidad y garantizar un rendimiento confiable. Puesto que, las incertidumbres en el cálculo de los parámetros de demanda aún se encuentran presentes en la medida en la que la estructura se vuelve menos lineal (o más no lineal). Para fines de diseño, se deben limitar las deformaciones a ubicaciones de comportamiento predecible, es decir, en donde no pueda producirse una

degradación repentina de la resistencia y rigidez de los elementos según los criterios de aceptación.

Como última consideración se recomienda el software de código abierto OpenSees para realizar análisis no lineal estático o dinámico. Al ser de código abierto permite tanto a los investigadores como a cualquier usuario aprovechar los logros de los demás, con un foro activo en donde se da la discusión y solución a ciertos problemas (McKenna et al., 2010).

Ahora, simplificando aún más las cosas, existe la herramienta OpenSeesPy que importa todos los comandos de OpenSees para que puedan ser usados dentro Python, que es un entorno científico cómodo, similar a MatLab, pero de acceso gratuito y de código abierto. Python es un lenguaje de programación de alto nivel y moderno que permite minimizar el tiempo de desarrollo de un código usando un lenguaje más limpio y simple (Johansson, 2015).

2.2 REDES NEURONALES (NN)

Antes de entrar a la creación de las redes neuronales es fundamental conocer los conceptos que existen detrás de esta tecnología computacional que ha ido desarrollándose con el pasar de los años. Lo que se suele pensar al tener una red neuronal, es un grupo de datos que es ingresado a un algoritmo y esperar que éste de resultados, pero es más importante conocer lo que ocurre dentro del algoritmo y con los datos cuando pasan por la red neuronal. De manera similar a las redes neuronales biológicas, las redes neuronales artificiales también tienen neuronas conectadas entre sí, que reciben información y la transforman para el siguiente paso.

Las redes neuronales no tienen un concepto establecido de manera general. Según Vemuri (1993), “Una red neuronal, es un conjunto de ecuaciones diferenciales ordinarias no lineales acopladas. Es decir, las ANN se pueden visualizar como sistemas dinámicos programables” (p. 207). En este sentido, posterior a la creación del sistema se da el proceso de entrenamiento o aprendizaje que podría verse relacionado a una calibración de dicho

sistema, el mismo que tendrá fin una vez que el objetivo en cuestión sea alcanzado.

2.2.1 *Historia*

En la década de 1940 se introdujo por primera vez el desarrollo de las redes neuronales que, como se conoce, buscan replicar el funcionamiento de las neuronas del cerebro humano. McCulloch & Pitts (1943) en la publicación de su trabajo muestran a través de teoremas el posible calculo lógico de la actividad que realizan las neuronas, dando paso a una investigación más profunda de las redes neuronales. A finales de la década, con la investigación de Hebb (1950), se completa la base de lo que es el aprendizaje autónomo con la propuesta de una ley de aprendizaje para las redes neuronales.

Posteriormente, en la década de 1950 y 1960, nuevos investigadores que comienzan a trabajar con redes neuronales compuestas de una sola capa, que recibieron el nombre de perceptrones y que poseen propiedad lineal. Sin embargo, la aplicación y el desarrollo de las redes neuronales se estancó por el problema de linealidad existente en los perceptrones. A finales de la década de 1960, fueron propuestas las redes neuronales de perceptrón multicapa (MLP, por sus siglas en inglés) con el objetivo de poder superar la limitante de la linealidad (Wang & Fu, 2009). El hecho de hacer uso de más capas ayudaba a que los puntos de datos contenidos dentro de las redes pudieran ser separados fácilmente dependiendo únicamente del ajuste del número de neuronas.

El desarrollo de las redes neuronales estuvo estancado aproximadamente dos décadas a causa de la falta de algoritmos de entrenamiento, ya que, sin algoritmos eficientes y sin importar la estructura de las redes neuronales, no darían resultados acertados. En la década de 1980, se presentó por primera vez un algoritmo de entrenamiento llamado “back propagation” o “retro propagación” el mismo que en tres pasos realizaba la tarea de retroalimentar los valores, calcular el error y propagarlo a las capas anteriores. “Las

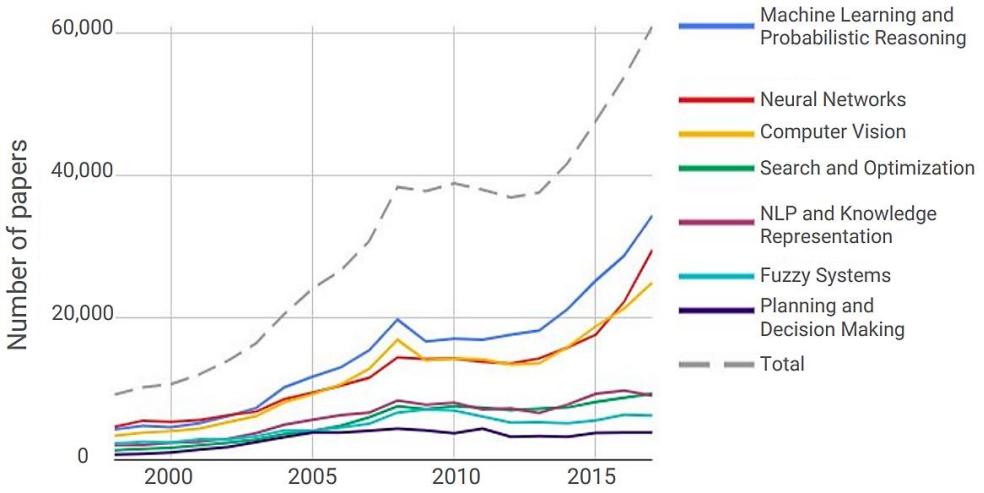
aplicaciones exitosas de redes neuronales en áreas científicas e industriales han atraído más atención en la investigación de redes neuronales artificiales para la resolución de problemas complejos" (Wang & Fu, 2009, p.181).

Según el reporte anual del Índice de Inteligencia Artificial (2018) para inicios del siglo XXI se dio el crecimiento de artículos de inteligencia artificial (IA), donde los artículos de categorías como el aprendizaje automático, visión artificial, redes neuronales, entre otros, tuvieron un crecimiento considerable durante el periodo del 2010 – 2014. Destacando la categoría de las redes neuronales con una tasa de crecimiento anual del 37% en el periodo del 2014 – 2017 como puede verse en la Figura 1

. Puede inferirse que el crecimiento de los artículos de redes neuronales presentado por el reporte anual se dan dentro del periodo indicado por el lanzamiento de las interfaces de programación Keras y Tensorflow en el año 2015 (Chollet, 2016).

Figura 1

Número de artículos de IA de Scopus por subcategoría (1998 – 2017)



Fuente: *Shoham et al* (2018).

2.2.2 *Aplicaciones de las Redes Neuronales*

Las aplicaciones de las redes neuronales a los problemas de la vida real han sido amplias con el pasar de los años. Estas van desde la educación, a los

negocios e incluso en la rama de la medicina. En el presente existen modelos que actualmente se están usando para abordar problemas con el reconocimiento de voz y patrones, alineación facial, visión artificial y detección de patrones e incluso enfermedades (Abiodun et al., 2018).

En lo que respecta al reconocimiento de voz en las últimas décadas, los algoritmos multi capa han sido empleados de forma amplia en áreas como el modelado acústico y el reconocimiento automático de voz o ASR por sus siglas en inglés. Un referente en esta área de aplicación es el trabajo de Lee et al (2004) donde muestra un proceso de aceleración al encriptado de videos MPEG. Las redes neuronales pueden identificar ruido dentro de este formato de video y eliminarlo.

Por otro lado, tenemos la visión artificial esta es una de las aplicaciones de redes neuronales más conocidas y al mismo tiempo que han sido de gran ayuda para los problemas de la actualidad. En la rama de la medicina se ha usado para la detección de cáncer de mama. Una investigación por Cueva (2017) expone el uso de la visión artificial para la detección de cáncer en la piel “Melanoma” que por medio del análisis de imágenes y clasificación de lunares por su asimetría, borde, color y diámetro se entrenaron hasta llegar a tener un rendimiento del 97.51%.

Entre las investigaciones más recientes y relevantes para un problema que aún persiste en la población, Rohila (2021) da a conocer el uso de las redes neuronales para la detección de COVID – 19 a partir de tomografías computarizadas del tórax de los pacientes en diferentes grados de infección con la finalidad de dar un diagnóstico mucho más rápido sin que dejara de ser correcto. Esta aplicación reciente de las redes neuronales llegó a tener un 94.9% de precisión para la detección de COVID – 19.

Por otro lado, tenemos las aplicaciones en la rama de la ingeniería. En particular, lo que respecta a la ingeniería estructural, fue creado un modelo basado en redes neuronales para evaluar el comportamiento histerético de materiales. Este modelo creado por Jin Yun et al. (2008), tuvo como punto clave la inserción de condiciones a las variables internas mediante una

relación matemática funcional que estaba presente entre los datos de entrada y salida. El uso de las redes neuronales en esta investigación fue implementado debido a la complejidad existente en el modelado del comportamiento de materiales bajo cargas cíclicas.

La visión artificial también ha sido usada en la rama de la ingeniería, especialmente en la inspección de estructuras y la seguridad de las mismas. La construcción de manera mundial es una de las industrias más peligrosas, debido a que los trabajadores son susceptibles a accidentes, lesiones e incluso, en el peor de los casos, la muerte en el lugar de trabajo. Aproximadamente el 7% de la fuerza laboral mundial está empleada en la construcción; sin embargo, la industria representa 30 - 40% de las muertes en el lugar de trabajo (Sunindijo & Zou, 2012). Esto indica el alto riesgo que tienen los trabajadores que se encuentran en obras civiles, por esto Fang et al (2020) en su investigación publicada en la revista “Automation in Construction” hace uso de la visión artificial para asegurar la seguridad en las construcciones basado en tres aspectos: comportamiento inseguro, obra insegura y defectos estructurales. Para poder llegar al estado de seguridad deseado fueron aplicados fusiones de datos y tecnologías digitales para poder identificar de manera exacta los aspectos establecidos, sin embargo, en la actualidad no se ha desarrollado ningún sistema automatizado de visión artificial para poder mejorar esta aplicación de manera considerable.

2.2.3 *Arquitectura de una red neuronal artificial*

Las redes neuronales forman parte del aprendizaje automático y se caracterizan por hacer uso de una cantidad grande de cálculos que se encuentran entrelazados. Según Krishna (2021), la red neuronal más simple consta de neuronas, pesos, capas y función de activación.

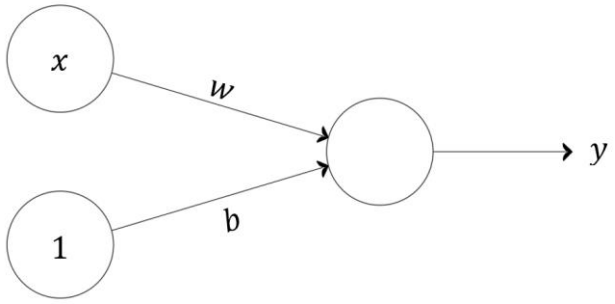
Las neuronas

También llamadas nodos, son las encargadas de contener o almacenar funciones matemáticas. Esta función matemática busca modelar el funcionamiento de una neurona biológica. De la manera más simple se muestra en la Figura 2

, la estructura de red neuronal con una neurona y una entrada.

Figura 2

Estructura de una red neuronal con una neurona y una entrada



Fuente: *Autor*

Se puede observar que la neurona tiene una entrada x, y también tiene un peso asociado w. El peso se da por las conexiones y es el parámetro esencial que el modelo usará para obtener un mejor ajuste para la salida, es decir, que estará en función de la importancia que tenga con respecto a otras entradas. Al pasar una entrada a una neurona, se da el producto de la entrada y el peso correspondiente y esto da como resultado $x * w$.

El elemento siguiente de la entrada se conoce como sesgo, éste se encarga de agregar al modelo un elemento de imprevisibilidad y está determinado por el valor de b, puesto que, el valor de la entrada es 1. El sesgo da al modelo la flexibilidad de adaptarse a las diferentes entradas.

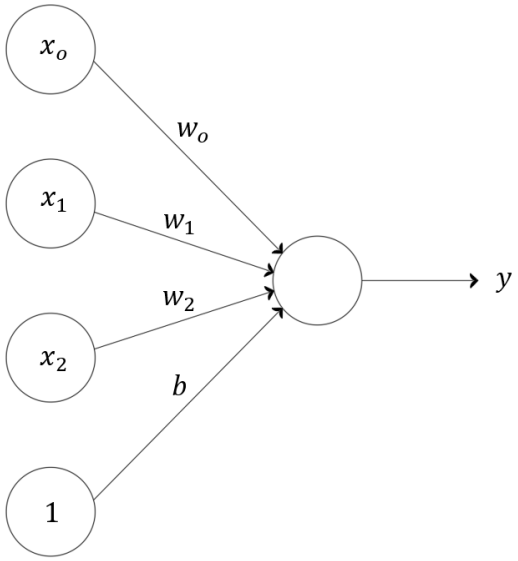
La combinación del sesgo y la entrada es lo que produce la salida quedando:

$$y = x.w + b$$

Generalmente, los conjuntos de datos que se usan no podrán ser analizados con una red neuronal tan simple como la mostrada anteriormente. Lo que se espera es una red neuronal con entradas múltiples que logren combinarse y así estimar una salida como se observa en la Figura 3.

Figura 3

Red neuronal con entradas múltiples



Fuente: *Autor*

Formulando la ecuación de la salida se obtiene lo siguiente:

$$y = x_0 w_0 + x_1 w_1 + x_2 w_2 + b$$

Y de manera general:

$$Y = \Sigma\,(\,pesos * entradas) + sesgo$$

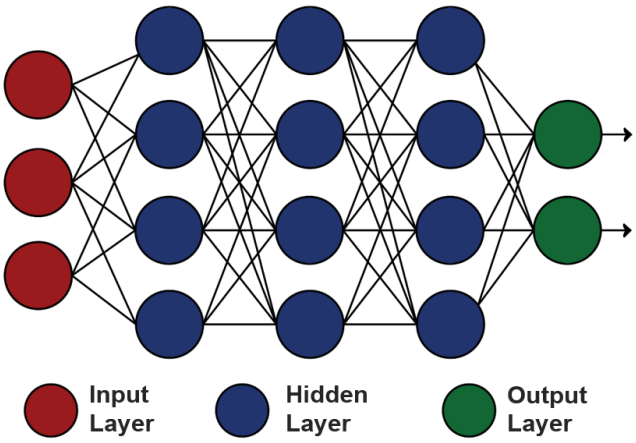
En otras palabras, lo que sucede en las neuronas es el cálculo de una suma ponderada de las entradas, la adición de un sesgo y por último, la decisión de disparar o no el resultado a la siguiente capa.

Las capas

La organización de las neuronas dentro de la red neuronal se realiza por medio de capas que pueden ser de tres tipos, como se muestra en la Figura 4.

Figura 4

Tipos de capas de una red neuronal



Fuente: *Autor*

Capa de Entrada (Input Layer)

Las neuronas que pertenecen a esta capa cumplen la función de recibir las entradas reales y enviar éstas a las neuronas de la capa siguiente, denominada “capa oculta”.

Capa oculta (Hidden Layer)

Dentro de la red neuronal pueden crearse varias capas ocultas. La capa oculta es donde se realiza el cálculo principal de la red neuronal. Las neuronas de esta capa reciben entradas de la capa de entrada o de la capa oculta anterior y pasan por nuevas funciones para terminar enviando el resultado a la capa siguiente.

Capa de Salida (Output Layer)

Las neuronas de esta capa reciben las entradas de las capas anteriores y finalmente muestran la información que podría ser el resultado final o podría ser usado para un nuevo bucle dentro de la misma red.

Funciones de activación

El propósito principal de las funciones de activación radica en la adición de una propiedad no lineal a la red neuronal. Sin estas funciones, la red neuronal no realizará más que asignaciones lineales, es decir que, la única operación matemática que se estará realizando es el producto de un vector de entrada y una matriz. Dado que un producto escalar no es nada más que una operación lineal, los siguientes cálculos sucesivos serán múltiples operaciones lineales.

Las redes neuronales para poder realizar operaciones complejas necesitan de una función de activación sin esta no podrán resolver las tareas que se desean resolver. Dependiendo del tipo de red neuronal y de los datos a obtener se escoge dicha función. Las funciones de activación más usadas según Oppermann (2021) son:

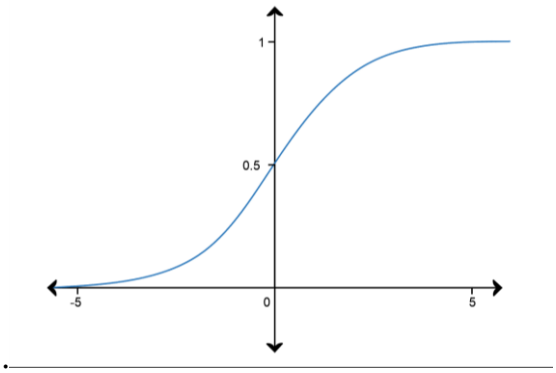
- *Sigmoide o logística*

Está función tiene forma de S (ver Figura 5) y es una de las funciones más utilizadas por ser de naturaleza no lineal. La expresión matemática esta dado por:

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

Figura 5

Gráfico de función Sigmoide



Fuente: *Autor*

El uso de esta función transforma los valores entre el rango de 0 y 1. Por esta misma razón se utiliza con más frecuencia para modelos en los que debe predecir la probabilidad, dado que, la probabilidad de cualquier cosa existe entre dicho rango.

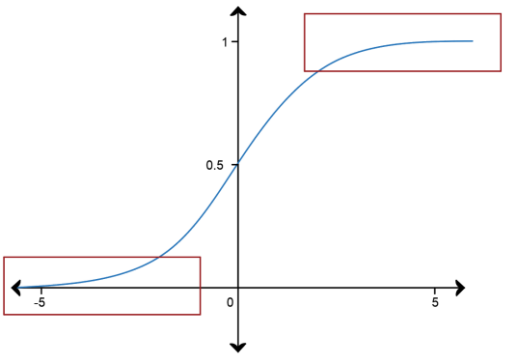
La función sigmoidea a pesar de ser muy utilizada tiene dos grandes inconvenientes:

- *Desaparecen gradientes*

El primer inconveniente radica específicamente en que la derivada de la función da resultados muy pequeños en los extremos, como puede verse en la Figura 6 (zonas rojas). De esta manera, la derivada cercana a 0 causa que el gradiente de la función de pérdida sea muy pequeño, esto lleva a que la actualización de los pesos se evite y que el entrenamiento se atasque, es decir que, la red se negará a aprender más o el entramiento se volverá excesivamente lento.

Figura 6

Gráfico de función Sigmoide - Extremos



Fuente: *Autor*

- *No está centrada en cero*

Esta es una propiedad indeseable de la función Sigmoide y causa que el entrenamiento de la red neuronal sea más complejo e inestable. Para la compresión de este problema, se considerará una neurona con entradas x_1 y x_2 multiplicada con sus respectivos pesos w_1 y w_2:

$$y = x_1 w_1 + x_2 w_2$$

Se tiene a x_1 y x_2 como salidas de una capa oculta anterior con función de activación sigmoide. Estas dos salidas siempre serán positivas puesto que, sigmoide nunca está centrada en 0. Los gradientes de la expresión anterior causarán dependencia en el sentido de los gradientes con respecto a los pesos, es decir que, si los gradientes son positivos para dichas salidas, los pesos también serán positivos y viceversa si son negativos.

El verdadero inconveniente se hace presente cuando la red neuronal busca un ajuste óptimo de los pesos, ya que se requiere un aumento en el peso w_1 y una disminución en el peso w_2. Por lo tanto, sí x_1 y x_2 son siempre positivos, no se puede aumentar y disminuir los pesos a la par. Necesariamente se extiende el número de pasos o capas en la red neuronal, debido a, que los pesos solo pueden ser aumentados o disminuidos al mismo tiempo.

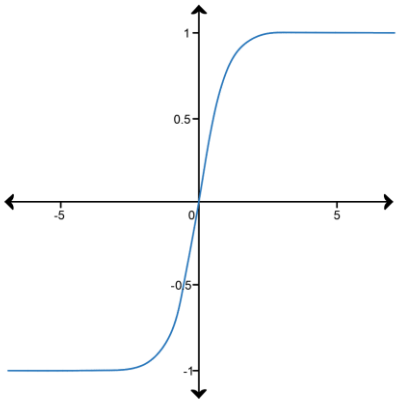
- *Tangente o Tangente Hiperbólica (Tanh)*

Una función de activación muy similar a la sigmoide y que se diferencia de la misma por la simetría con respecto al origen. El rango de valores que transforma está entre -1 y 1. Se define esta función como:

$$f(x) = tanh\,(x) = \frac{2}{1 + e^{-2x}} - 1$$

Figura 7

Gráfico de función Tangente Hiperbólica (Tanh)

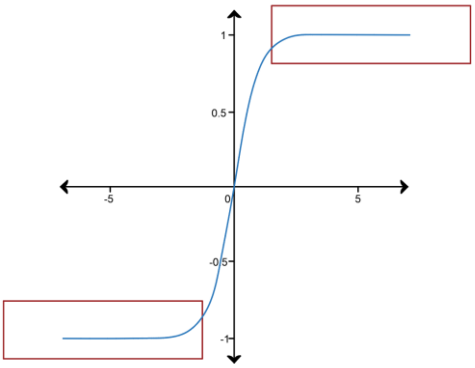


Fuente: *Autor*

La función tangente hiperbólica tiene el mismo problema de los gradientes en los extremos como se muestra en la Figura 8 (zonas rojas) de tal manera que, las neuronas se saturarán para obtener grandes valores negativos y positivos en los extremos, esto resulta conveniente porque, al tener las salidas centradas en 0 aportarán a un mejor ajuste de los pesos.

Figura 8

Gráfico de función Tangente Hiperbólica (Tanh) - Extremos



Fuente. *Autor*

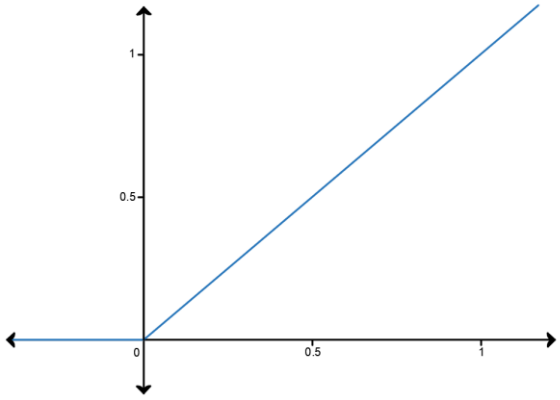
- *Unidad Lineal Rectificada (ReLU)*

La función de activación ReLU de naturaleza no lineal, ha ganado popularidad a causa de una ventaja que posee sobre las otras funciones y esta es que no activa todas las neuronas al mismo tiempo debido a que la activación se limita a 0. Se muestra la Figura 9 para una mejor comprensión.

$$\begin{cases} x & si\ x \geq 0 \\ 0 & de\ lo\ contrario \end{cases}$$

Figura 9

Gráfico de función ReLU



Fuente: *Autor*

La condición de no activación de las neuronas dependerá de los valores de entrada, sí estos son negativos el resultado será 0. Esto genera una aceleración en la convergencia del descenso del gradiente, haciendo que las pérdidas sean menores. Desde un punto de vista computacional, la función ReLU implica operaciones matemáticas simples y menos costosas, en comparación a las otras dos funciones anteriormente explicadas, ya que, hacen uso de exponenciales. Esta ventaja genera que la red neuronal sea mucho más rápida debido a la activación escasa de neuronas.

Todo indica que la mejor función de activación es la ReLU pero esto no es del todo cierto, ya que puede producirse la “muerte” de las neuronas, es decir, que durante el ajuste o actualización de los pesos, la red neuronal puede hacer un ajuste en el que ciertas neuronas siempre tendrán entradas menores a 0 y esto lleva a que estos valores no contribuyan al proceso de entrenamiento.

- *Softmax*

La función de activación softmax a diferencia de las otras funciones, se aplica únicamente en la última capa y sólo cuando se desea que la red neuronal haga predicciones acerca de las puntuaciones de probabilidad a lo largo de tareas de clasificación. La expresión matemática de esta función es:

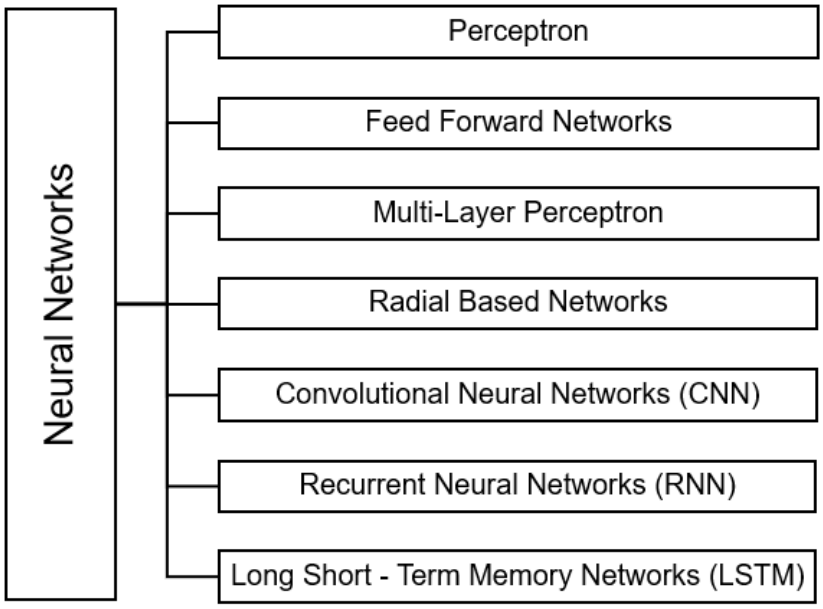
$$y_i = \frac{e^{z_i}}{\sum_{i=0}^{m} e^{z_i}}$$

2.2.4 *Tipos de Redes Neuronales*

Existen muchos tipos de redes neuronales, como pueden observarse en la Figura 10, donde se incluyen las más populares según Vadapalli (2020) :

Figura 10

Tipos de redes neuronales



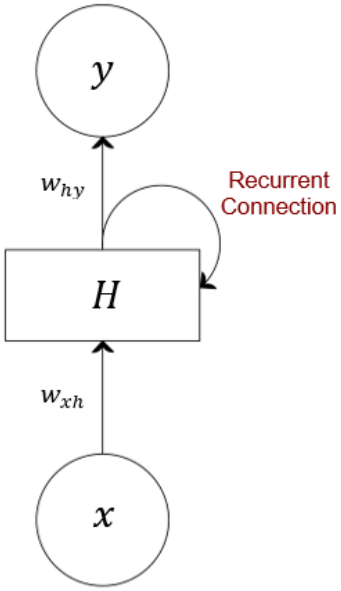
Fuente: *Autor*

El presente trabajo de titulación hace énfasis en las redes neuronales recurrentes por sus siglas en inglés (RNN), también conocidas como Auto asociativas o Feedback Network. Se caracterizan por usar su propia memoria interna para procesar las secuencias arbitrarias de entradas. Las señales viajan hacia adelante y hacia atrás, introduciendo un bucle a la red (Poznyak et al., 2019).

Si bien la mayoría de las redes neuronales logran recordar y mejorar sus resultados mediante el entrenamiento, las redes neuronales recurrentes pueden recordar el pasado y las decisiones que tome estarán influenciadas por lo que ha aprendido en el pasado. Ahora, lo que las diferencia es que las RNN recuerdan las cosas aprendidas mientras generan los resultados (Venkatachalam, 2019). La arquitectura de una red neuronal recurrente se muestra en la Figura 11 para una mejor comprensión.

Figura 11: Red Neuronal Recurrente

Red Neuronal Recurrente



Fuente: *Autor*

La conexión recurrente hace mención a un circuito de retroalimentación que se encuentra conectado a las decisiones pasadas, para que la red neuronal use sus propias salidas, momento tras momento como entradas.

CAPÍTULO III

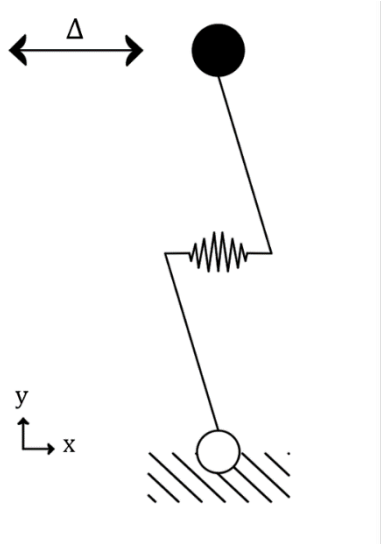
3. RESULTADOS

3.1 GENERACIÓN DE BASE DE DATOS

Para la generación de la base de datos se hizo uso de Python. Dentro de este software se importaron todos los comandos de OpenSees mediante la biblioteca de OpenSeesPy. Se creó un modelo conformado por un elemento tipo armadura, el mismo que cumple la función de un resorte, como se muestra en la Figura 12. Entre las cosas más importantes a destacar del modelo creado, es el comando para especificar el tipo de material y el comando usado para la construcción del algoritmo que dicta la secuencia de solución de las ecuaciones no lineales.

Figura 12

Modelo de Elemento Tipo Armadura



Fuente: *Autor*

El comando “uniaxialMaterial” se usó para construir un objeto que representara relaciones uniaxiales de fuerza – deformación. Dentro de este comando se especifica el tipo de material y, el comando usado para escoger el tipo de material fue “Hysteretic”, debido a que se busca construir un elemento histerético bilineal uniaxial con el efecto de “pinching” de fuerza y deformación, que permite incluir también el deterioro a causa de la demanda

de ductilidad y de la energía absorbida por deformación, a lo que se añade también la pérdida de rigidez de descarga degrada basada en ductilidad.

El comando “algorithm” y dentro de este, el tipo de algoritmo que se uso fue “Newton” que crea un algoritmo de Newton – Raphson. Su formulación simple permite una programación fácil y la convergencia que posee es una de las más rápidas.

Una vez terminado el modelo se estableció la cantidad de repeticiones que el modelo se ejecutaría para producir resultados, almacenarlos y con esto crear la base de datos, conformada por valores de historias de fuerza-deformación. Las repeticiones se establecieron en 10.000, con la finalidad de generar una base con una cantidad considerable de datos para que la red neuronal pueda usar un porcentaje de dichos datos para entrenar y posterior al entrenamiento y usar el porcentaje restante para testear la red.

3.2 RED NEURONAL PREALIMENTADA (FEEDFORWARD)

3.2.1 *Importación de Módulos*

Un módulo es un archivo que está compuesto de código Python, permite definir variables y funciones, además de poder incluir un código ejecutable. Los módulos importados para la creación de la red neuronal fueron los siguientes:

```
# Module Import
import os
import numpy as np
import pickle as pk
import tensorflow as tf
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping
import time
```

Código 1: *Módulos importados para el modelo FNN*. Fuente: *Autor*

Los módulos importados más relevantes son:

- Pickle: cumple la función de importar los datos para el entrenamiento y testeo de la red neuronal.
- Tensorflow: se encarga de agrupar una pila lineal de capas.
- Tensorflow.keras.layers: permite la creación de las capas con su respectiva cantidad de neuronas y capa de activación.
- Tensorflow.keras.callbacks: este módulo permite que los pesos del modelo logren guardarse, es decir que, permite guardar el progreso del entrenamiento.

3.2.2 *Importación de datos de entrenamiento y testeo*

Con la base datos ya generada se estableció un porcentaje del 75% destinado a la etapa de entrenamiento de la red neuronal y el 25% restante de los datos de la base fueron destinados a usarse para el testeo de la red. De esta manera, de la base datos se generan 4 archivos que serán usados en el código de la red neuronal:

- Training data (75%)
  - x_train
  - y_train
- Test data (25%)
  - x_test
  - y_test

```
# Data Import
# Training Data
n = 3000
x_train1 = pk.load(open('xtrain1.dat', 'rb'))
x_train2 = pk.load(open('xtrain2.dat', 'rb'))
x_train3 = pk.load(open('xtrain3.dat', 'rb'))
x_train4 = pk.load(open('xtrain4.dat', 'rb'))
x_train5 = pk.load(open('xtrain5.dat', 'rb'))
x_train6 = pk.load(open('xtrain6.dat', 'rb'))
x_train7 = pk.load(open('xtrain7.dat', 'rb'))
x_train8 = pk.load(open('xtrain8.dat', 'rb'))

X_train = []
for i in x_train1:
    X_train.append(i[:n])
for i in x_train2:
    X_train.append(i[:n])
for i in x_train3:
    X_train.append(i[:n])
for i in x_train4:
    X_train.append(i[:n])
for i in x_train5:
    X_train.append(i[:n])
for i in x_train6:
    X_train.append(i[:n])
for i in x_train7:
    X_train.append(i[:n])
for i in x_train8:
    X_train.append(i[:n])


y_train1 = pk.load(open('ytrain1.dat', 'rb'))
y_train2 = pk.load(open('ytrain2.dat', 'rb'))
y_train3 = pk.load(open('ytrain3.dat', 'rb'))
y_train4 = pk.load(open('ytrain4.dat', 'rb'))
y_train5 = pk.load(open('ytrain5.dat', 'rb'))
y_train6 = pk.load(open('ytrain6.dat', 'rb'))
y_train7 = pk.load(open('ytrain7.dat', 'rb'))
y_train8 = pk.load(open('ytrain8.dat', 'rb'))

Y_train = []
for i in y_train1:
    Y_train.append(i[:n])
for i in y_train2:
    Y_train.append(i[:n])
for i in y_train3:
    Y_train.append(i[:n])
for i in y_train4:
    Y_train.append(i[:n])
for i in y_train5:
    Y_train.append(i[:n])
for i in y_train6:
    Y_train.append(i[:n])
for i in y_train7:
    Y_train.append(i[:n])
for i in y_train8:
    Y_train.append(i[:n])
```

```
# Test Data
x_test = pk.load(open('xtest.dat', 'rb'))
X_test = []
for i in x_test:
    X_test.append(i[:n])


y_test = pk.load(open('ytest.dat', 'rb'))
Y_test = []
for i in y_test:
    Y_test.append(i[:n])
```

Código 2: *Importación de datos de entrenamiento y prueba*. Fuente: *Autor*

Como logra verse en el código, los datos de entrenamiento han sido divididos en varios archivos, puesto que, al cargar una base demasiado grande el consumo de memoria puede llegar a ser elevado y con la fragmentación de los datos se da solución a este contratiempo.

La variable $\boldsymbol{n}$ está definida como 3000 debido a que el modelo usado para crear la base de datos crea aleatoriamente datos de desplazamientos y fuerzas, lo que genera listas de datos con una cantidad de datos desiguales. Especificar la cantidad de datos que se requieren en la lista busca prevenir problemas en el proceso de aprendizaje, que será realizado posteriormente.

3.2.3 *Arquitectura de la Red Neuronal*

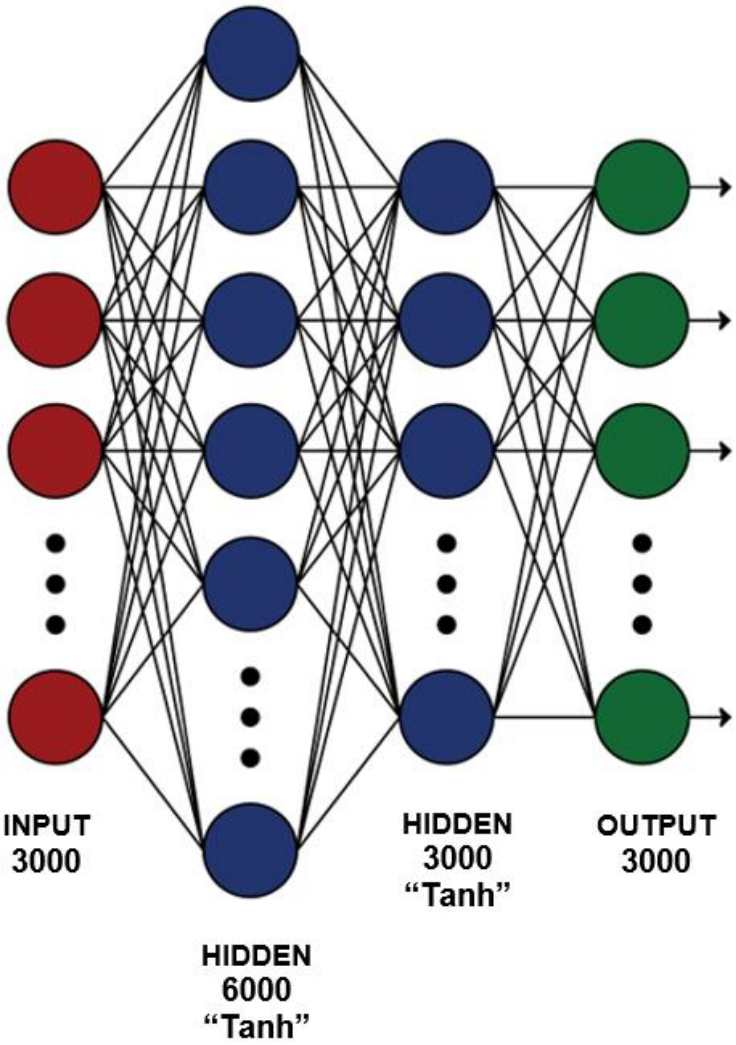
La arquitectura está conformada por la capa de entrada, dos capas ocultas y finalmente la capa de salida, como puede observarse en la Figura 13. La primera capa oculta formada con un número de 6000 neuronas y la segunda capa con 3000 neuronas. La función de activación usada para ambas capas ocultas fue tangente hiperbólica (tanh).

```
# FeedForward Neural Network - Sequential Model
n = 3000
model = tf.keras.models.Sequential()
model.add(Dense(n*2, activation='tanh'))
model.add(Dense(n, activation='tanh'))
```

Código 3: *Creación del modelo de Red Neuronal FeedForward*. Fuente: *Autor*

Figura 13

Arquitectura de Red Neuronal FeedForward



Fuente: *Autor*

El número de capas ocultas necesario para que la red neuronal logre arrojar buenos resultados no está definido, por lo que en el proceso de entrenamiento se apunta a agregar capas a la red hasta que los resultados alcanzados cumplan con un nivel de precisión dentro de lo satisfactorio y una vez que se llegue a esta calibración dejar agregar capas hasta que dichos resultados dejen de mejorar, esto también aplica a la cantidad de neuronas que conforman cada capa oculta.

3.2.4 *Compilación del modelo*

La fase final del modelo de una red neuronal se da en este paso. El proceso de compilación es el más importante porque se establece la función de pérdida y el optimizador. Para el código de esta red se usó como función de pérdida el error medio cuadrado o MSE, por sus siglas en inglés y como optimizador se usó el optimizador de Adam.

```
# Model Compile
model.compile(optimizer='adam',
loss='mse')
```

Código 4: *Compilación del modelo*. Fuente: *Autor*

3.2.5 *Creación de punto de guardado (Checkpoint)*

La creación del punto de guardado es necesaria para poder salvar los pesos del modelo y el progreso del entrenamiento. Esto ayudará a que, si el entrenamiento se desea continuar después, puede hacerse desde el punto guardado ya que los pesos podrán ser cargados y la red procederá a cambiar dichos pesos a partir de dicho punto.

```
# Checkpoint
cp = tf.keras.callbacks.ModelCheckpoint('cp.ckpt',
     save_weights_only = True,
     verbose=1)
```

Código 5: *Creación de Checkpoint*. Fuente: *Autor*

3.2.6 *Definir parada anticipada (EarlyStopping)*

Esta parte del código es implementada en el modelo con el objetivo de parar el proceso de aprendizaje cuando la cantidad que deseamos monitorizar deja de mejorar, de aumentar o disminuir. Como se busca minimizar las pérdidas la cantidad a monitorizar sería ‘loss’ y esto se logra especificando el cambio mínimo que puede considerarse como mejora dentro de la parada anticipada.

La aplicación más importante de la parada anticipada es evitar el “overfitting” que se da por un sobre entrenamiento de la red neuronal, lo que conlleva a que esta considere únicamente como valores válidos y correctos a los valores dentro de la base de entrenamiento. Por esto, la parada anticipada

se encarga de revisar que la base de datos de entrenamiento mejore al igual que contribuye a que la base de datos de testeo no empeore.

```
# Early Stopping
monitor = EarlyStopping(monitor='val_loss',
min_delta=1e-8,
patience=5,
verbose=1,
restore_best_weights = True)
```

Código 6: *Parada anticipada*. Fuente: *Autor*

3.2.7 *Entrenamiento del modelo*

El proceso de aprendizaje de la red neuronal depende mucho de la base de datos, puesto que, mientras mayor sea la cantidad de datos con los que la red logre hacer este proceso dará buenos resultados. Sin embargo, algo que radica mucho en los resultados es la capacidad de ajuste de la red. Para este proceso dentro del código se usaron devoluciones de llamada (callbacks) puesto que, esto permite realizar acciones dentro y en varias etapas del entrenamiento.

```
# Model Training
history = model.fit(X_train,
Y_train,
batch_size=10,
epochs=100,
Callbacks = monitor,
validation_data=(X_test,Y_test))
```

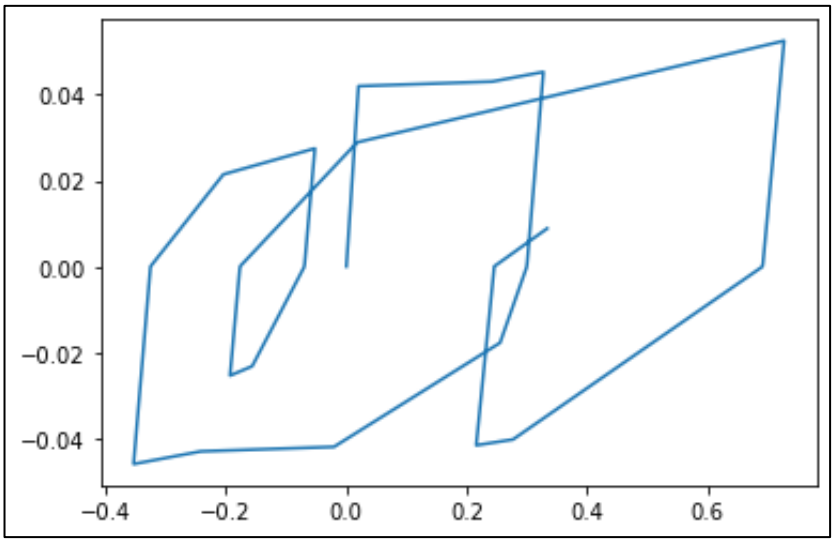
Código 7: *Proceso de entrenamiento de Red Neuronal FeedForward .* Fuente: *Autor*

3.2.8 *Resultados de red neuronal FeedForward (Prealimentada)*

Para examinar el resultado de la red neuronal, con el modelo de OpenSees que se usó para crear la base de datos también se obtuvieron valores aleatorios de desplazamiento y fuerza, que se encuentran fuera de los datos de entrenamiento y prueba, de esta manera se logran ver en la Figura 14 los valores graficados

Figura 14

Resultado con Modelo de OpenSees (FNN)

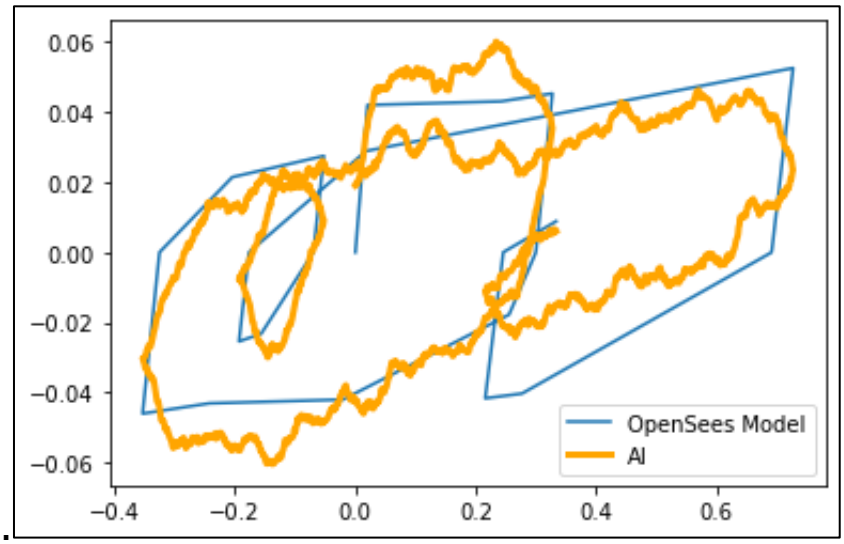


Fuente: *Autor*

El objetivo de usar nuevamente el modelo de OpenSees, es que la red haga una predicción con valores que no se encuentran en la base de datos ya generada, dando más complejidad al modelo de la red neuronal para predecir valores con los que nunca entrenó e hizo pruebas. En la Figura 15 se muestra el resultado.

Figura 15

Modelo de OpenSees y Modelo Inteligencia Artificial (IA - *FNN*)

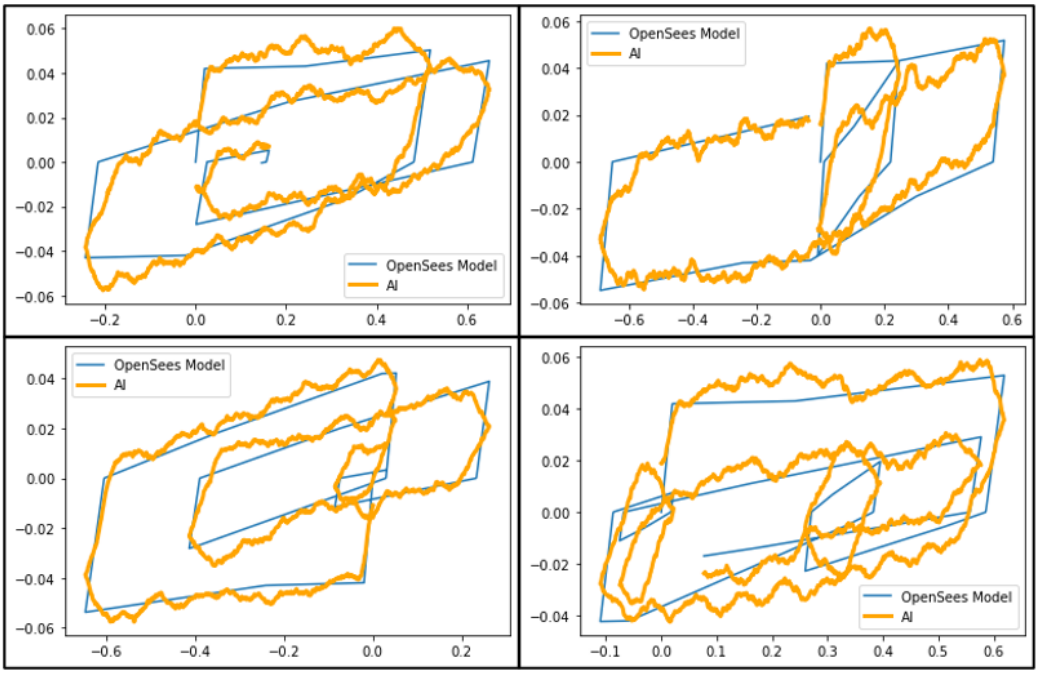


Fuente: *Autor*

Como logra observarse en la figura 15, la predicción que hizo la red neuronal prealimentada resulta ser muy acertada. Para corroborar los resultados entregados por el modelo de IA, se procedió en la ejecución reiterada del modelo de OpenSees obteniendo como resultado buenas predicciones (ver Figura 16 **Recopilación de Resultados**) por parte del modelo IA.

Figura 16

Recopilación de Resultados



Fuente: *Autor*

Las predicciones realizadas por el modelo IA en su mayoría son correctas. Se puede resaltar la presencia de ruido dentro de las gráficas; sin embargo, a primera vista se puede tener una idea clara de los valores; puesto que, el margen de error no es grande.

Desde otra perspectiva, el tiempo que toma la red neuronal prealimentada para predecir el comportamiento no lineal del modelo planteado resultó ser considerablemente menor. Para llegar a esta premisa fueron recopilados los tiempos en segundos de 100 pruebas aleatorias con el modelo de OpenSees vs el modelo de inteligencia artificial, con el objetivo de realizar un promedio. Finalmente, se obtuvo un porcentaje de diferencia del 51.10% en lo que respecta a el tiempo. Se debe mencionar también, que el modelo utilizado es

muy simple, por lo que esa diferencia pudiera resultar considerablemente mayor en otros casos.

3.3 RED NEURONAL RECURRENTE (RECURRENT)

3.3.1 Importación de Módulos

Los módulos importados para la red neuronal recurrente como pueden observarse en el Código 8: *Módulos importados para el modelo RNN*. Fuente: *Autor*, se diferencian a los mostrados en el Código 1: *Módulos importados para el modelo FNN*. Fuente: *Autor* por un módulo adicional.

```
# Module Import
import os
import matplotlib.pyplot as plt
import pickle as pk
import numpy as np
from tensorflow.keras.layers import Dense
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM
```

Código 8: *Módulos importados para el modelo RNN*. Fuente: *Autor*

El módulo añadido para la creación del modelo en esta ocasión es para la aplicación de las capas recurrentes, específicamente de memoria de largo y corto plazo, LSTM por sus siglas en inglés (Long Short – Term Memory). Este tipo de capa recurrente es implementada para permitir que la red tenga la capacidad de aprender la dependencia del orden de los datos y en su mayoría es aplicado para problemas de traducción y predicción de secuencias.

3.3.2 Importación de datos de entrenamiento y testeo

La base de datos conserva los mismos porcentajes para la etapa de entrenamiento (75%) y para la etapa de prueba (25%). Con respecto a la importación de los datos, es similar a la mostrada anteriormente (ver Código 2: *Importación de datos de entrenamiento y prueba*. Fuente: *Autor*) con la excepción de que la variable $\boldsymbol{n}$ está definida como 500.

3.3.3 Cambio de forma de los datos

El cambio de forma es necesario en este tipo de red neuronal por la capa LSTM, puesto que, esta solo permite la entrada de datos con forma 3D. La forma que los datos de entrenamiento y prueba poseen es de (7500, n) y (2500, n), respectivamente. Para cambiar de esta forma 2D a una 3D se plantea lo mostrado en el Código 9.

```
# Reshape training and test data
num_steps = n
# training set
X_train_shaped = np.reshape(X_train, newshape=(-1, num_steps, 1))
Y_train_shaped = np.reshape(Y_train, newshape=(-1, num_steps, 1))
assert X_train_shaped.shape[0] == Y_train_shaped.shape[0]
# test set
X_test_shaped = np.reshape(X_test, newshape=(-1, num_steps, 1))
Y_test_shaped = np.reshape(Y_test, newshape=(-1, num_steps, 1))
assert X_test_shaped.shape[0] == Y_test_shaped.shape[0]
```

Código 9: *Cambio de forma 2D a 3D*. Fuente: *Autor*

Se usa la función *reshape* para lograr cambiar la forma a (7500, n, 1) para los datos de entrenamiento y para los datos de prueba la forma se cambió a (2500, n, 1).

3.3.4 Arquitectura de la Red Neuronal

La arquitectura de la red neuronal recurrente está compuesta por dos capas recurrentes (LSTM) con 20 y 50 neuronas, y finalmente con dos capas densas de 20 y 1 neurona. La unidad de la neurona al final de la arquitectura se hace para que la forma de la salida coincida con la entrada, debido a que, si la salida tiene una forma diferente a la de la entrada, el modelo puede arrojar un tamaño mayor al que debería haciendo que la predicción se vuelva lenta.

```
# Recurrent Neural Network - Sequential Model
model = Sequential()
model.add(LSTM(units = 20,
               activation = 'tanh',
              input_shape = (num_steps, 1),
    return_sequences = True))
model.add(LSTM(units = 50,
               activation = 'tanh',
    return_sequences = True))
```

```
model.add(Dense(units = 20,
           activation = 'tanh'))
model.add(Dense(units = 1,
           activation = 'tanh'))
```

Código 10: *Creación del modelo de Red Neuronal Recurrente*. Fuente: *Autor*

3.3.5 Compilación del modelo

En la compilación del modelo de la red recurrente se usó como función de pérdida el error medio cuadrado o MSE, por sus siglas en inglés y como optimizador se usó el optimizador de Adam. Esta misma función de pérdida y optimizador fueron usados para la red neuronal prealimentada (ver Código 4**: *Compilación del modelo*. Fuente: *Autor***).

3.3.6 Definir parada anticipada (EarlyStopping)

Al igual que en el modelo de la red neuronal prealimentada el código aplicado para la parada anticipada en el modelo de la red recurrente fue el mismo (ver Código 6**: *Parada anticipada*. Fuente: *Autor***)

3.3.7 Entrenamiento del modelo

El entrenamiento para las redes recurrentes depende mucho de la compatibilidad en las formas de los datos de entrada y salida. Para esta etapa de aprendizaje también se aplicaron las devoluciones de llamada (callbacks). El Código 11**: Proceso de entrenamiento de Red Neuronal Recurrente. Fuente: Autor** muestra cómo se estableció la etapa de entrenamiento.

```
# Model Training
history = model.fit(X_train_shaped,
Y_train_shaped,
epochs = 200,
callbacks = monitor,
batch_size = 1,
validation_data=(X_test_shaped,Y_test_shaped))
```

Código 11: *Proceso de entrenamiento de Red Neuronal Recurrente.* Fuente: *Autor*

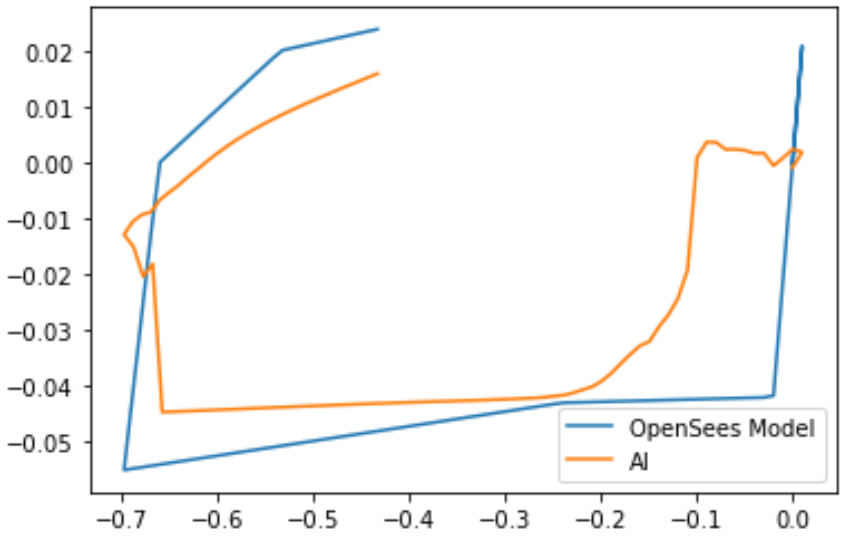
3.3.8 Resultados de Red Neuronal Recurrente (Recurrent)

Al igual que en las redes neuronales prealimentadas, se usó nuevamente el modelo de OpenSees, para examinar la red con valores fuera de la base de

datos. Para el caso de la red neuronal recurrente, como se puede ver en la figura 17, se empezó con una cantidad de datos pequeña correspondiente a 100 datos de desplazamiento y fuerza, para examinar la arquitectura establecida en el Código 10: *Creación del modelo de Red Neuronal Recurrente*. Fuente: *Autor*.

Figura 17

Modelo de OpenSees y Modelo Inteligencia Artificial (IA - RNN) – (d=100)

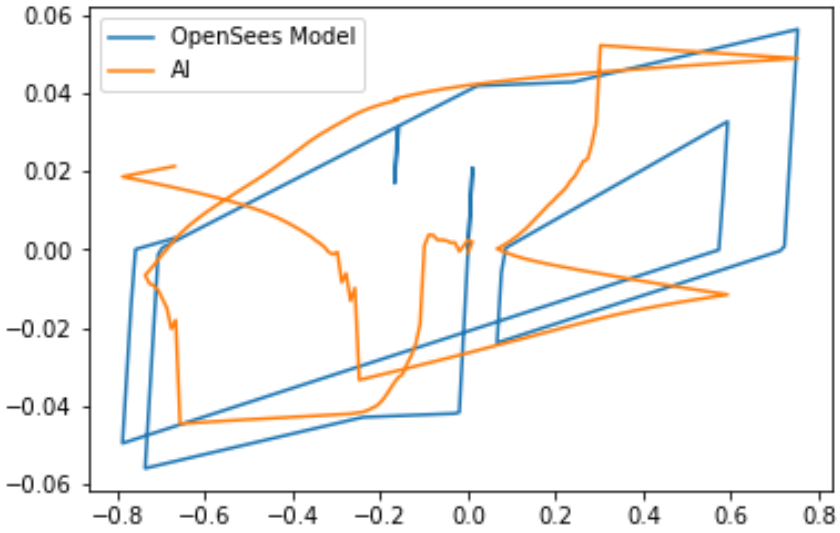


Fuente: *Autor*

Se observan resultados con un margen de error muy grande; sin embargo, a diferencia de la FNN los resultados no presentaron ruido. El margen de error presente puede ocurrir a que la arquitectura de la red no es la correcta y deben quitarse o agregarse capas y neuronas. Aun así, se usó la misma arquitectura con una cantidad de datos 5 veces mayor a la anterior.

Figura 18

Modelo de OpenSees y Modelo Inteligencia Artificial (IA - RNN) – (d=500)



Fuente: *Autor*

Como logra observarse en la figura 18, el margen de error se mantuvo debido a que la arquitectura también se conservó; sin embargo, este margen puede estar presente por otras razones influyentes como lo son el tamaño del lote de entrenamiento o la cantidad de épocas usadas en la etapa de aprendizaje.

4. CONCLUSIONES

- El procedimiento se probó utilizando una base de datos creada por OpenSees de la cual se extrajeron las siguientes conclusiones:
- Se demostró que las redes neuronales prealimentadas y recurrentes pueden imitar el comportamiento de modelos no lineales
- En el caso de la red neuronal prealimentada se logró acelerar el análisis no lineal de una estructura en más de la mitad del tiempo que es aplicado para el modelo de OpenSees.
- Este trabajo hace un primer acercamiento a la aplicación de las redes recurrentes para resolver problemas no-lineales.

5. RECOMENDACIONES

- Se recomienda crear una base de datos que contenga muchos datos de desplazamiento y fuerza, para que a partir de ésta se limiten los datos que serán usados para la etapa de aprendizaje, esto contribuirá en el futuro para aprovechar el tiempo en la calibración de los modelos de las redes neuronales.
- Con respecto a la calibración del modelo de la red neuronal prealimentada (FNN), se recomienda realizar un aumento progresivo de los datos y en función de esto ajustar el número de capas y neuronas que serán necesarias para obtener valores acertados.
- Por otro lado, en materia de las redes neuronales recurrentes (RNN) es recomendable la correcta organización de los datos de entrenamiento y prueba en lotes (batches), con el objetivo de agilizar la etapa de entrenamiento. Además, de establecer una arquitectura apropiada para obtener resultados con un margen de error menor.

6. REFERENCIAS

Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A. E., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, *4*(11), e00938. https://doi.org/10.1016/j.heliyon.2018.e00938

ASCE. (2007). *Seismic rehabilitation of existing buildings*. https://ascelibrary.org/doi/abs/10.1061/9780784408841

Chollet, F. (2016, April 11). *Introducing Keras 1.0*. https://blog.keras.io/introducing-keras-10.html

Cohen, S. (2021). The basics of machine learning: strategies and techniques. In *Artificial Intelligence and Deep Learning in Pathology*. Elsevier Inc. https://doi.org/10.1016/b978-0-323-67538-3.00002-6

Cueva, W. F., Munoz, F., Vasquez, G., & Delgado, G. (2017, Octubre 20). Detection of skin cancer "Melanoma" through computer vision. *Proceedings of the 2017 IEEE 24th International Congress on Electronics, Electrical Engineering and Computing, INTERCON 2017*. https://doi.org/10.1109/INTERCON.2017.8079674

Deierlein, G. G., Reinhorn, A. M., & Willford, M. R. (2010). Nonlinear Structural Analysis For Seismic Design. *NEHRP Seismic Design Technical Brief No. 4*, *4*, 1–32.

du Bos, M. L., Balabdaoui, F., & Heidenreich, J. N. (2020). Modeling stress-strain curves with neural networks: a scalable alternative to the return mapping algorithm. *Computational Materials Science*, *178*(May 2019), 109629. https://doi.org/10.1016/j.commatsci.2020.109629

Fang, W., Ding, L., Love, P. E. D., Luo, H., Li, H., Peña-Mora, F., Zhong, B., & Zhou, C. (2020). Computer vision applications in construction safety assurance. *Automation in Construction*, *110* (Septiembre 2019), 103013. https://doi.org/10.1016/j.autcon.2019.103013

FEMA. (1997). NEHRP Commentary on the Guidelines for the Seismic Rehabilitation of Buildings. In *Federal Emergency Management Agency, Washington, DC, developed by the Applied Technology Council* (Issue Octubre). www.atcouncil.org

FEMA. (2002). Earthquake Loss Estimation Methodology. Hazus - MH2.1. In

Technology Acting Branch Chief. Claire Drury. www.msc.fema.gov

FEMA. (2005). Improvement of Nonlinear Static Seismic Analysis Procedures. In *FEMA 440, Federal Emergency Management Agency, Washington DC* (Vol. 440, Issue Junio).

FEMA. (2009). FEMA P440 - Effects of Strength and Stiffness Degradation on Seismic Response. In *Fema P440a* (Issue June). www.ATCouncil.org

Hebb, D. (1950). The Organization of Behavior; A Neuropsychological Theory. *The American Journal of Psychology*, *63*(4), 633. https://doi.org/10.2307/1418888

Jin Yun, G., Ghaboussi, J., & Elnashai, A. S. (2008). A new neural network-based model for hysteretic behavior of materials. *International Journal for Numerical Methods in Engineering*, *May*, 447–469. https://doi.org/10.1002/nme

Johansson, R. (2015). Introduction to Scientific Computing and Visualization in Python. *Python*.

Krawinkler, H. (2006). Importance of good nonlinear analysis. *Structural Design of Tall and Special Buildings*, *15*(5), 515–531. https://doi.org/10.1002/tal.379

Krishna, D. (2021, January 5). *The Components of a Neural Network*. https://towardsdatascience.com/the-components-of-a-neural-network-af6244493b5b

Lee, S., Choon, A. S., & Budiarto, R. (2004). *Lightweight and Cost-Effective MPEG Video Encryptionr*.

McCulloch, W. S., & Pitts, W. (1943). Learning based industrial bin-picking trained with approximate physics simulator. *Advances in Intelligent Systems and Computing*, *867*, 786–798. https://doi.org/10.1007/978-3-030-01370-7_61

McKenna, F., Scott, M. H., & Fenves, G. L. (2010). Nonlinear Finite-Element Analysis Software Architecture Using Object Composition. *Journal of Computing in Civil Engineering*, *24*(1), 95–107. https://doi.org/10.1061/(asce)cp.1943-5487.0000002

Niewiarowski, R. W., & Rojahn, C. (1996). *Seismic Evaluation and Retrofit of Concrete Buildings*.

Oppermann, A. (2021, Marzo 1). *Activation Functions in Deep Neural*

Networks. https://towardsdatascience.com/activation-functions-in-deep-neural-networks-aae2a598f211

Poznyak, T. I., Chairez Oria, I., & Poznyak, A. S. (2019). Background on dynamic neural networks. *Ozonation and Biodegradation in Environmental Engineering*, 57–74. https://doi.org/10.1016/b978-0-12-812847-3.00012-3

Rohila, V. S., Gupta, N., Kaul, A., & Sharma, D. K. (2021). Deep learning assisted COVID-19 detection using full CT-scans. In *Internet of Things* (Vol. 14, p. 100377). https://doi.org/10.1016/j.iot.2021.100377

Salehi, H., & Burgueño, R. (2018). Emerging artificial intelligence methods in structural engineering. *Engineering Structures*, *171* (Mayo) , 170–189. https://doi.org/10.1016/j.engstruct.2018.05.084

Sarkar, K. (2008). *APPLICATION OF ARTIFICIAL NEURAL NETWORKS FOR OPTIMUM STRUCTURAL Project Report on APPLICATION OF ARTIFICIAL NEURAL NETWORKS FOR OPTIMUM STRUCTURAL ANALYSIS Submitted to In partial fulfillment of requirement for the award of degree Master of Engineering i* (Issue Julio).

Shoham, Y., Perraulr, R., Brynjolfsson, E., Clark, J., Manyika, J., Niebles, J. C., Lyons, T., Etchemendy, J., & Grosz, B. (2018). Artificial intelligence index. *Angewandte Chemie International Edition, 6(11), 951–952.*, 1–94.

Sunindijo, R. Y., & Zou, P. X. W. (2012). Political Skill for Developing Construction Safety Climate. *Journal of Construction Engineering and Management*, *138*(5), 605–612. https://doi.org/10.1061/(asce)co.1943-7862.0000482

Vadapalli, P. (2020, Diciembre 29). *7 Types of Neural Networks in Artificial Intelligence Explained | upGrad blog*. https://www.upgrad.com/blog/types-of-neural-networks/

Vemuri, V. (1993). Artificial Neural Networks in Control Applications. In *Advances in Computers* (Vol. 36, Issue C). https://doi.org/10.1016/S0065-2458(08)60272-7

Venkatachalam, M. (2019, Febrero 28). *Recurrent Neural Networks. Remembering what's important*. https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

Wang, L., & Fu, K. (2009). Artificial neural networks. *Wiley Encyclopedia of Computer Science and Engineering*, *130*, 181–188. https://doi.org/10.1533/9780857099440.275

Yang, C., Kim, Y., Ryu, S., & Gu, G. X. (2020). Prediction of composite microstructure stress-strain curves using convolutional neural networks. *Materials and Design*, *189*, 108509. https://doi.org/10.1016/j.matdes.2020.108509

DECLARACIÓN Y AUTORIZACIÓN

Yo, Coello Choez Bryan Xavier, con C.C: #0921939388 autor/a del trabajo de titulación: **Aplicación de redes neuronales recurrentes para acelerar el análisis no lineal de estructuras** previo a la obtención del título de **ingeniero** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 14 de septiembre de 2021

f. _____

Nombre: Coello Choez Bryan Xavier

C.C: **0921939328**

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA			
FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN			
TEMA Y SUBTEMA:	Aplicación de redes neuronales recurrentes para acelerar el análisis no lineal de estructuras		
AUTOR(ES)	Bryan Xavier Coello Choez		
REVISOR(ES)/TUTOR(ES)	José Andrés Barros Cabezas		
INSTITUCIÓN:	Universidad Católica de Santiago de Guayaquil		
FACULTAD:	Facultad de Ingeniería		
CARRERA:	Ingeniería Civil		
TITULO OBTENIDO:	Ingeniero		
FECHA DE PUBLICACIÓN:	14 de septiembre de 2021	**No. DE PÁGINAS:**	44
ÁREAS TEMÁTICAS:	Análisis no lineal, Modelado no Lineal, Redes neuronales		
PALABRAS CLAVES/ KEYWORDS:	*modelado no lineal, análisis no lineal, aceleración de análisis, redes neuronales, redes neuronales recurrentes, redes recurrentes LSTM*		
El análisis no lineal de estructuras es una herramienta de la ingeniera fundamentada en el desempeño para la validación de diseños propuestos ya sea para nuevas estructuras o para evaluar estructuras existentes. La aplicación de este análisis para sistemas estructurales compuestos por una gran cantidad de grados de libertad como lo son edificios de gran altura, puentes e incluso presas, requieren de mucho tiempo. Los tiempos de cálculo extensos ocurren con más frecuencia en estudios paramétricos. Para la reducción de este tiempo, se plantea en este trabajo una metodología practica que consiste en la aplicación de redes neuronales recurrentes. Dentro del grupo de las redes neuronales recurrentes se aplica la memoria de largo y corto plazo (LSTM) para acelerar el análisis no lineal de estructuras. Con la creación de una base de datos para el entrenamiento y prueba de la red neuronal, se utilizará no solo para aprender el comportamiento del análisis, sino también hacer predicciones confiables en adición al objetivo principal que es la aceleración del análisis en sí. Por demás, la recomendación a las aplicaciones e investigaciones de los diferentes tipos de redes neuronales para la solución de problemas de ingeniería.			
ADJUNTO PDF:	☒ SI	☐ NO	
CONTACTO CON AUTOR/ES:	**Teléfono:** +593–95–898–3863	E-mail: bryanxavierr189@hotmail.com	
CONTACTO CON LA INSTITUCIÓN (C00RDINADOR DEL PROCESO UTE):	**Nombre:** Glas Cevallos, Clara Catalina		
	Teléfono: +593–98–461–6792		
	E-mail: clara.glascu.ucsg.edu.ec		
SECCIÓN PARA USO DE BIBLIOTECA			
Nº. DE REGISTRO (en base a datos):			
Nº. DE CLASIFICACIÓN:			
DIRECCIÓN URL (tesis en la web):			