Métodos de Aprendizaje Automático para la Predicción de Series Temporales Caóticas

por

M. C. Astrid Maritza González Zapata

Tesis presentada en cumplimiento parcial de los requisitos para el grado de:

Doctora en Ciencias en la Especialidad de Electrónica

en

Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE)

Octubre, 2024

Santa María Tonantzintla, Puebla

Asesor:

Dr. Israel Cruz Vega

Departamento de Electrónica INAOE

Co-Asesor:

Dr. Esteban Tlelo Cuautle

Departamento de Electrónica INAOE

Resumen

En los últimos años, los métodos de aprendizaje automático, han ganado popularidad debido a su capacidad para abordar problemas complejos y no lineales. Dentro de este contexto, la predicción de series temporales caóticas ha captado un interés creciente, ya que estos sistemas, caracterizados por su comportamiento aparentemente impredecible, tienen aplicaciones prácticas en campos como la economía, la meteorología y la biología. El uso de técnicas avanzadas de aprendizaje automático para predecir el comportamiento de estas series ofrece una herramienta poderosa para anticipar dinámicas complejas y mejorar la toma de decisiones en escenarios de alta incertidumbre, lo que resulta especialmente relevante en esta época de rápida transformación tecnológica.

Existen dos enfoques principales para la predicción de series temporales caóticas: sin forzamiento del maestro y con forzamiento del maestro. En el primero, se predice un paso hacia adelante y, en la siguiente iteración, el valor predicho se utiliza como entrada para estimar el siguiente paso. Aunque este enfoque puede llevar a una divergencia progresiva entre los datos predichos y los reales, permite extender el horizonte de predicción a largo plazo. El segundo enfoque, con forzamiento del maestro, establece una ventana de tamaño definido que contiene n datos históricos, a partir de los cuales se predicen h pasos adelante, siendo h la cantidad de salidas generadas por el método de aprendizaje automático. En esta tesis, exploraremos

ambos enfoques utilizando diferentes estrategias para cada uno.

Para la predicción de series temporales caóticas sin forzamiento del maestro, utilizamos redes de estado eco (ESN), que son ampliamente reconocidas por sus buenos resultados y bajo costo computacional en esta tarea. Para ampliar el horizonte de predicción alcanzado con las ESN, empleamos la técnica de decimación, particularmente útil en series temporales con un comportamiento lento. Además, optimizamos los hiperparámetros de la ESN, ya que una selección adecuada de estos influye directamente en el error y el horizonte de predicción. Gracias a estas estrategias, logramos extender el horizonte de predicción de la serie temporal del sistema de Lorenz en comparación con los trabajos reportados en la literatura. También implementamos una ESN en FPGA para la predicción de series temporales caóticas. Para ello, exploramos diferentes topologías en la conexión de las neuronas del depósito de la ESN, ya que estas conexiones afectan directamente el costo computacional y los recursos necesarios para su implementación en FPGA. Encontramos que la topología en anillo presenta mejores resultados en la predicción de series temporales caóticas en comparación con la topología completamente conectada. Cabe destacar que la topología en anillo utiliza menos del $\frac{1}{N} \times 100\,\%$ de conexiones, siendo N el número de neuronas del depósito, por ejemplo, para 50 neuronas la topología anillo utiliza menos del $2\,\%$ de conexiones en relación con la topología completamente conectada. Esta reducción es directamente proporcional a los recursos requeridos para la implementación de cada una de las topologías en FPGA.

Para la predicción de series temporales caóticas con forzamiento del maestro, empleamos redes LSTM, que son ampliamente utilizadas en tareas de predicción y clasificación. Sin embargo, estas redes tienen un alto costo computacional durante la fase de entrenamiento, dado que se deben entrenar 12 matrices por cada celda LSTM. Por esta razón, proponemos un modelo híbrido LSTM-ESN, donde las compuertas que conforman la celda LSTM son reemplazadas por redes de estado eco.

Este enfoque mantiene las propiedades de la ESN, cuyas matrices se generan aleatoriamente y se re-escalan, lo que reduce significativamente el costo computacional en el entrenamiento. En el modelo propuesto, la única matriz que se entrena es la matriz de salida, lo que implica una reducción de más del 90 % en los parámetros que deben ser entrenados por cada celda. Inicialmente, entrenamos esta matriz utilizando el método de gradiente descendente y el algoritmo Adam. Posteriormente, optamos por un método de gradiente descendente de orden fraccionario, ya que los métodos numéricos para resolver este tipo de ecuaciones poseen propiedades de memoria (infinita o corta), lo que puede contribuir a mejorar el rendimiento del modelo híbrido. Los modelos propuestos LSTM-ESN y LSTM-ESN con gradiente descendente de orden fraccionario presentan mejores resultados en la predicción de un paso adelante de series temporales caóticas de los sistemas de Lorenz, Chen y el Producto Interno Bruto de México, en comparación con el método tradicional LSTM, a pesar de la reducción en la cantidad de parámetros entrenables.

Abstract

In recent years, machine learning methods have gained popularity due to their ability to address complex and nonlinear problems. Within this context, the prediction of chaotic time series has attracted increasing interest, as these systems characterized by their seemingly unpredictable behavior have practical applications in fields such as economics, meteorology, and biology. The application of advanced machine learning techniques for predicting the behavior of these time series provides a powerful tool for anticipating complex dynamics and enhancing decision-making in high-uncertainty scenarios, which is especially pertinent in this era of rapid technological transformation.

There are two primary approaches to predicting chaotic time series: without master forcing and with master forcing. In the first approach, one step ahead is predicted, and in the next iteration, the predicted value is used as input to estimate the subsequent step. While this method may lead to a gradual divergence between predicted and actual data, it allows for an extended long-term prediction horizon. The second approach, with master forcing, establishes a defined-sized window containing n historical data points, from which h steps ahead are predicted, with h representing the number of outputs generated by the machine learning method. This thesis will explore both approaches, employing different strategies for each.

For the prediction of chaotic time series without master forcing, we utilize Echo

State Networks (ESNs), widely recognized for their strong performance and low computational cost in this task. We employ the decimation technique to extend the prediction horizon achieved with ESNs, which is particularly beneficial for time series exhibiting slow behavior. Additionally, we optimize the hyperparameters of the ESN, as an appropriate selection of these parameters directly influences both error and prediction horizon. Through these strategies, we successfully extend the prediction horizon of the Lorenz system's time series compared to the work reported in the literature. We also implemented an ESN on an FPGA for the prediction of chaotic time series. In this context, we explored various topologies for the connections of the ESN reservoir neurons, as these connections directly impact the computational cost and resources required for FPGA implementation. We found that the ring topology yields superior results in predicting chaotic time series compared to the fully connected topology. Notably, the ring topology uses less than $\frac{1}{N} \times 100\,\%$ of the connections, where N is the number of reservoir neurons; for instance, with 50 neurons, the ring topology utilizes less than $2\,\%$ of connections relative to the fully connected topology. This reduction is directly proportional to the resources required for the implementation of each topology on FPGA.

For the prediction of chaotic time series with master forcing, we employ Long Short-Term Memory (LSTM) networks, which are widely used in prediction and classification tasks. However, these networks incur a high computational cost during the training phase, as 12 matrices must be trained for each LSTM cell. For this reason, we propose a hybrid LSTM-ESN model, in which the gates that comprise the LSTM cell are replaced by Echo State Networks. This approach retains the beneficial properties of the ESN, whose matrices are generated randomly and rescaled, significantly reducing the computational cost during training. In the proposed model, the only matrix that is trained is the output matrix, resulting in a reduction of over $90\,\%$ in the parameters that need to be trained for each cell. Initially, we train this matrix using the gradient descent method and the Adam algorithm. Subse-

quently, we adopt a fractional-order gradient descent method, as numerical methods for solving such equations possess memory properties (either infinite or short), which may enhance the performance of the hybrid model. The proposed LSTM-ESN and LSTM-ESN with fractional-order gradient descent demonstrate superior results in one-step-ahead prediction of chaotic time series from the Lorenz systems, Chen, and Mexico's Gross Domestic Product compared to the traditional LSTM method, despite the reduction in the number of trainable parameters.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Actualmente, la predicción de series temporales caóticas sigue siendo un desafío abierto. Algunas aplicaciones bien conocidas son, por ejemplo: la predicción del clima [1, 2, 3], patologías relacionadas con la salud [4], procesos financieros [5], entre otros. En particular, estos casos exhiben un comportamiento caótico que a menudo se modela mediante conjuntos de ecuaciones diferenciales no lineales fuertemente acopladas [6], de modo que su solución depende de las características dinámicas, las cuales tienen una alta sensibilidad a las condiciones iniciales, es decir, las soluciones de dos sistemas caóticos idénticos con ligeras diferencias en las condiciones iniciales serán significativamente diferentes después de un breve tiempo de evolución [7]. Afortunadamente, algunos trabajos proporcionan directrices para resolver este tipo de sistemas caóticos aplicando métodos numéricos apropiados [8].

Debido a la dificultad de modelar sistemas caóticos, se emplean técnicas prometedoras como el Aprendizaje Automático (ML, por sus siglas en inglés), que como aproximaciones universales de funciones no lineales, se utilizan para aprender el comportamiento de los sistemas caóticos a partir de datos experimentales, haciendo predicciones de la serie temporal sin un conocimiento específico del modelo [9].

Algunas técnicas de ML que se aplican a la predicción de series temporales caóticas son, por ejemplo: Máquinas de Vectores de Soporte (Support Vector Machines, SVM) [10], Redes Neuronales Artificiales (ANN), Redes Neuronales Recurrentes con memoria a largo y corto plazo (RNN-LSTM) [6], Redes Neuronales de propagación hacia adelante (FFNN) [11, 12], Computación de Reservorio (RC) [13, 14, 15, 16, 17], entre otras.

La familia de la Computación de Reservorio (RC) ha sido ampliamente estudiada para la predicción de series temporales caóticas, entre las cuales se encuentra la Red de Estado de Eco (ESN) [18, 19, 6, 20]. Estas redes presentan muchas ventajas sobre otras redes neuronales recurrentes, como su bajo costo computacional y su fácil entrenamiento [21]. Algunas mejoras en ESN, como aquellas que incluyen un modelo basado en conocimiento (modelo híbrido) [22], aumentan el tiempo de predicción casi al doble. En este sentido, los principales desafíos están relacionados con cómo aumentar el horizonte de predicción y reducir el costo computacional. Para abordar estos problemas, se han explorado diferentes alternativas: en [23], los autores proponen una ESN robusta, reemplazando la función de verosimilitud Gaussiana en la regresión bayesiana por una función de verosimilitud de Laplace; en [24], los autores realizan una optimización bayesiana de los hiperparámetros del depósito (tasa de aprendizaje, radio espectral, etc.), ya que seleccionarlos es difícil debido a todas las posibles combinaciones; en [25], los autores también utilizan optimización bayesiana para encontrar un buen conjunto de hiperparámetros para la ESN, y además proponen una validación cruzada para validar sus resultados; en [26], se puede ver la aplicación del algoritmo de optimización de búsqueda hacia atrás (BSA) y sus variantes para determinar los pesos de salida más apropiados de una ESN, dado que el problema de optimización es complejo; en [27], los autores optimizan la matriz de entrada de la ESN, definida aleatoriamente en la propuesta original [28], utilizando el algoritmo de lobo gris de oposición selectiva; recientemente, y en [29], se propone una ESN con topología mejorada para una predicción de series temporales preci-

sa y eficiente, utilizando una función de activación del depósito suave. Los tiempos de predicción aumentan en todos estos casos, pero los sistemas de estudio aún son escasos en algunos aspectos. Por ejemplo: no incluyen sistemas caóticos de comportamiento lento, como las neuronas artificiales, donde el cambio en la amplitud entre el punto actual y el anterior es pequeño, tendiendo a ser monótono.

La mayoría de los trabajos realizan la simulación de redes neuronales para la predicción de series temporales en lenguajes como C o Python [30, 31], sin embargo, la implementación de ESN en sistemas digitales como FPGA (Field-Programmable Gate Array) sigue siendo un desafío debido a factores como la implementación de la función de activación (tangente hiperbólica, sigmoide, etc.), el error de redondeo, el tipo de conexión de las neuronas, etc. [32, 33].

Con el fin de mejorar la topología de una red ESN, en [29] se propone una nueva función de activación y una nueva topología para la conexión de las neuronas del depósito, utilizando diferentes sistemas caóticos como Mackey-Glass y Rössler. Sin embargo, la implementación de la función de activación no es factible en un sistema digital debido a su complejidad. En [34] se realiza un estudio de diferentes topologías, observando la calidad de las predicciones al variar el número de neuronas y conexiones en cada una de sus topologías, donde concluyen que las redes con topologías simples y conectividad esparcida presentan una calidad y comportamiento similar al de otras redes más complejas. Respecto a los problemas de baja conectividad en los depósitos, los autores de [24] estudian varias topologías de baja conexión, pero su enfoque es la optimización de hiperparámetros con el objetivo de mejorar la predicción, concluyendo que la optimización de hiperparámetros es una gran herramienta para mejorar la predicción, aunque no realizan un análisis sobre cómo mejorar las redes para una implementación digital.

Cabe destacar que la predicción de una red ESN no es multi-salida, sino que la

predicción de un paso adelante se retroalimenta hasta alcanzar h-pasos adelante, hasta que el error acumulado supera un umbral. Otros tipos de redes neuronales recurrentes como LSTM, GRU, ELMAN RNN, entre otras, no presentan buenos resultados en la predicción de h-pasos adelante, ya que utilizan múltiples salidas; es decir, h-pasos adelante corresponden a h salidas de la red. Según [35], a mayor cantidad de pasos predichos utilizando salidas múltiples en estas redes, mayor es el error. En [36], Youming Lei et al. proponen un método híbrido que combina redes profundas LSTM con un modelo empírico inexacto de sistemas dinámicos para predecir sistemas caóticos de alta dimensión. Los modelos analizados son los sistemas Mackey-Glass y Kuramoto-Sivashinsky, los autores presentan resultados numéricos donde el método propuesto evita la divergencia rápida del modelo LSTM multicapa al reconstruir el atractor caótico, demostrando también la viabilidad del modelo híbrido. Otro desafío interesante estudiado en la literatura es la predicción de series temporales caóticas con ruido. En [37], los autores utilizan diferentes métodos de aprendizaje profundo, incluyendo LSTM, para resolver el problema mencionado anteriormente, logrando una predicción de hasta 48 pasos adelante; sin embargo, su error para este horizonte de predicción es grande.

1.1. Principales Retos en la Predicción de Series Temporales Caóticas

Como se ha mencionado previamente, se han utilizado una gran variedad de métodos de aprendizaje automático para la predicción de series temporales caóticas. Generalmente, el criterio para escoger el método de aprendizaje depende del enfoque o el reto que se quiera afrontar. A continuación se describen los retos al momento de realizar la predicción de series temporales caóticas:

- **Aumentar el horizonte de predicción:** (Ya sea aumentar los tiempos de Lyapunov[1] o la cantidad de pasos predichos) [23], [38], [39], [40]. La técnica de decocción es ampliamente usada en el ámbito de las telecomunicaciones, sin embargo, en [33] proponen utilizar esta técnica como una estrategia para aumentar el horizonte de predicción de series temporales caóticas. Esta propuesta es útil en series temporales que presentan un comportamiento lento, sin embargo, en aquellos datos con cambios abruptos, como podrían ser datos obtenidos de fenómenos reales, esta estrategia podría eliminar datos característicos e importantes del fenómeno de interés. Por otro lado, encontramos la optimización de los hiperparámetros de los métodos de aprendizaje automático, este enfoque permite reducir el error en la predicción pero también ayuda a incrementar el horizonte de predicción alcanzado.

- **Disminuir el costo computacional:** Cuando se habla de disminuir el costo computacional, generalmente se busca poder realizar la predicción de las series temporales en menor tiempo o realizar la implementación en sistemas digitales como FPGA's. Para el primer punto se suelen hacer propuestas de modelos híbridos[2] donde se busca reducir el costo computacional debido al entrenamiento de una gran cantidad de parámetros pero que mantengan la calidad de la predicción. Para el segundo punto se suelen explorar diferentes topologías, conexiones [24, 41, 42] o estrategias de diseño [43] que permitan reducir la cantidad de recursos necesarios para la implementación.

- **Disminuir el error en la predicción:** En este caso, generalmente se realiza la predicción de un paso adelante o muy pocos pasos adelante, y se busca que tenga el error más pequeño posible [42], [44].En este punto se suele realizar la optimización de los hiperparámetros de los métodos de aprendizaje automático, pero el enfoque es exclusivo en reducir el error en la predicción, lo cual no

[1] En el anexo A de esta tesis, en la pág. 88, se describen los tiempos de Lyapunov.
[2] Como la propuesta realizada en esta tesis.

implica mejorar el horizonte de predicción.

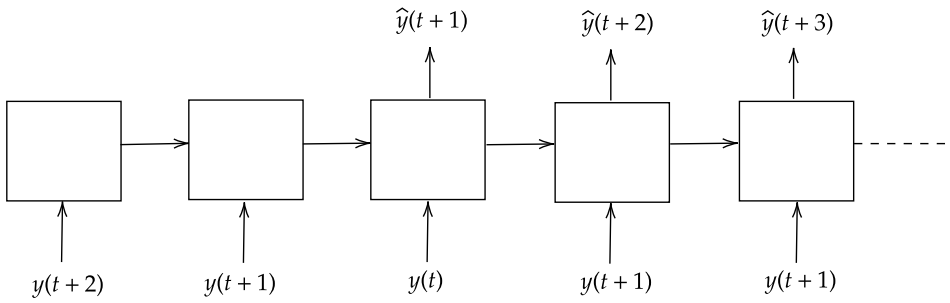
1.2. Predicción de Series Temporales Caóticas con y sin Forzamiento del Maestro

En la predicción de series temporales caóticas podemos definir dos tipos de lazos, independientemente del método de aprendizaje automático que se utilice. El primer lazo es con forzamiento del maestro, donde la entrada del sistema es obtenida del conjunto de datos de prueba. En esta configuración, generalmente se realiza la predicción de un paso hacia adelante, y el proceso se repite iterativamente sin una realimentación de los datos predichos. Esto implica que la predicción no diverge de los datos objetivo. En la figura 1.1a se muestra el lazo de predicción con forzamiento del maestro.

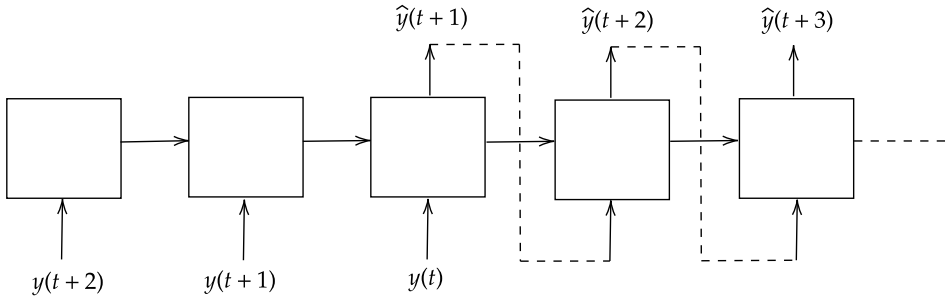
El siguiente lazo, es sin forzamiento del maestro, donde para la predicción de un paso se utiliza como entrada el paso predicho previamente, como se muestra en la figura 1.1b. En este caso se introduce un error acumulativo que puede hacer que eventualmente la predicción diverja de los datos objetivo. Este tipo de lazo es comúnmente usado en la predicción de h-pasos adelante [45].

La predicción de series temporales caóticas sin forzamiento del maestro se utiliza cuando se busca maximizar el horizonte de predicción. Por otro lado, la predicción con forzamiento del maestro se emplea para reducir el error en los resultados. La elección del método depende de la aplicación y de la naturaleza de los datos. Es decir, si se prioriza predecir una gran cantidad de datos o si es preferible predecir uno o pocos valores con mayor precisión.

Para la predicción sin forzamiento del maestro, utilizamos redes de estado eco



(a)



(b)

Figura 1.1: Lazos de predicción de series temporales caóticas. (a): Con forzamiento del maestro, (b): Sin forzamiento del maestro.

(ESN), conocidas por su eficacia en esta tarea y su bajo costo computacional. Con el fin de aumentar el horizonte de predicción de las ESN, implementamos la técnica de decimación, particularmente útil en series temporales con comportamientos lentos. Además, optimizamos los hiperparámetros de la ESN, ya que una selección adecuada de estos influye directamente en el error y el horizonte de predicción alcanzado. Estas estrategias nos permitieron mejorar el horizonte de predicción de la serie temporal de Lorenz, en comparación con trabajos reportados en la literatura. También implementamos una ESN en FPGA para la predicción de series temporales caóticas. Para ello, exploramos diferentes topologías en la conexión de las neuronas del depósito de la ESN, ya que estas conexiones afectan directamente el costo computacional y los recursos necesarios para su implementación en FPGA.

En cuanto a la predicción con forzamiento del maestro, empleamos redes LSTM, un método popular en el aprendizaje automático. Sin embargo, su costo computacional en la fase de entrenamiento es elevado, debido a la necesidad de entrenar 12 matrices por cada celda LSTM. Para reducir este costo, proponemos un modelo híbrido LSTM-ESN, en el cual las matrices de la celda LSTM son reemplazadas por redes de estado eco. Este enfoque mantiene las características de la ESN, donde las matrices se generan aleatoriamente y se reescalan, lo que disminuye significativamente el costo computacional del entrenamiento.

En el modelo propuesto, la única matriz que debe entrenarse es la matriz de salida, e inicialmente se hace mediante el método de gradiente descendiente y el algoritmo Adam. Posteriormente, se propone utilizar un método de gradiente descendiente de orden fraccionario, aprovechando las propiedades de memoria (infinita o corta) de los métodos para resolver ecuaciones de orden fraccionario, lo que creemos puede mejorar los resultados del modelo híbrido. Los modelos propuestos LSTM-ESN y LSTM-ESN con gradiente descendente de orden fraccionario presentan mejores resultados en la predicción de un paso adelante de series temporales caóticas de los sistemas de Lorenz, Chen y el Producto Interno Bruto de México, en comparación con el método tradicional LSTM, a pesar de la reducción en la cantidad de parámetros entrenables.

1.3. Objetivo General

- Proponer nuevas técnicas de aprendizaje automático que mejoren la predicción de series temporales caóticas con respecto al estado del arte actual y determinar aplicaciones potenciales para su implementación en software y hardware.

1.3.1. Objetivos Específicos

- Comparar técnicas actuales de aprendizaje automático en la predicción de series temporales caóticas.
- Proponer un modelo basado en aprendizaje profundo para la predicción de series temporales caóticas.
- Estudiar la viabilidad de aplicaciones potenciales como el monitoreo de arritmias cardíacas para prevenir infartos, monitoreo de patrones de activación neuronal para detectar picos neuronales, predicción de temblores, entre otros.
- Optimización de los modelos propuestos para la predicción de series temporales caóticas.
- Implementación en hardware de modelos de aprendizaje automático para la predicción de series temporales caóticas.

1.4. Organización de la Tesis

Esta tesis está organizada de la siguiente manera: en el capítulo 2 se describen las series temporales caóticas utilizadas, así como las técnicas aplicadas, como son la decimación y el escalado. En el capítulo 3 se detalla la red de estado eco (ESN) utilizada para la predicción de series temporales caóticas, sus parámetros y el proceso de optimización, además de las consideraciones para su implementación en FPGA. El capítulo 4 aborda la red LSTM y proponemos dos modelos híbridos: la combinación LSTM-ESN y LSTM-ESN con gradiente descendiente, ambos aplicados a la predicción de series temporales caóticas. En el capítulo 5 se exponen los resultados de las predicciones de series temporales caóticas sin forzamiento del maestro, realizadas

con la red ESN, y de las predicciones con forzamiento del maestro, en las que se utilizaron las redes LSTM, LSTM-ESN y LSTM-ESN FR. Finalmente, en el capítulo 6 se presentan las principales conclusiones de esta tesis, así como las contribuciones más destacadas.

Capítulo 2

Series Temporales Caóticas

En este capítulo, nos adentramos en la descripción de los sistemas caóticos empleados como base fundamental en esta tesis. Nos enfocaremos en el proceso de escalado, un requisito esencial para la predicción de series temporales caóticas usando métodos de Aprendizaje Automático basados en neuronas. Este proceso, que implica ajustar el rango dinámico de los datos para que se sitúe dentro de un intervalo específico, como $[-1, 1]$, juega un papel crítico mantener las propiedades intrínsecas de las series temporales caóticas.

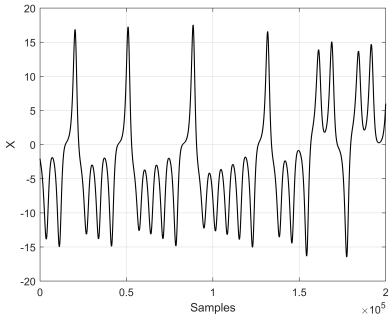
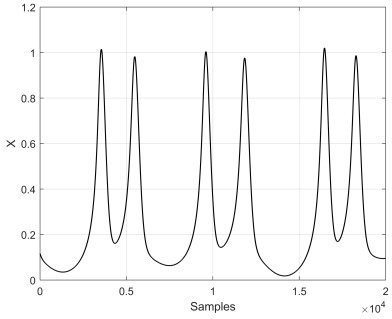
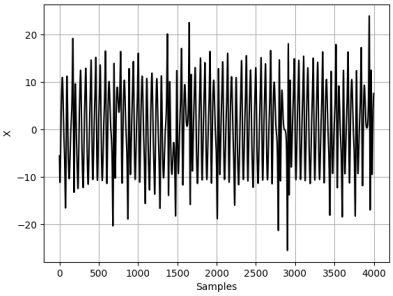
Además, exploraremos el proceso de decimación, una estrategia que permite modificar la ventana de visualización de las series temporales sin alterar sus propiedades fundamentales. La decimación se presenta como una herramienta para ampliar el horizonte de predicción especialmente en sistemas caóticos con un comportamiento lento.

2.1. Sistemas Caóticos

Para que un sistema pueda ser categorizado como caótico, es necesario que cumpla con una serie de requisitos fundamentales. En primer lugar, debe exhibir aperiodicidad en un tiempo infinito, lo que implica que a medida que el tiempo tiende hacia el infinito ($t \to \infty$), la serie temporal no converge hacia un punto específico, una trayectoria periódica o semi-periódica. Además, el sistema debe ser sensible a las condiciones iniciales, lo que implica que incluso una variación de una millonésima en estas condiciones provocará una variación significativa en la trayectoria de la serie temporal. Por último, debe ser determinista, lo que significa que el sistema no puede estar sujeto a entradas o variables que dependan del ruido externo [46].

La variedad de sistemas caóticos abarca tanto aquellos que se encuentran en tiempo continuo como en tiempo discreto, también conocidos como mapas caóticos. En este trabajo usaremos como modelos de referencia sistemas caóticos continuos y datos obtenidos de fenómenos reales. En la tabla 2.1 se describen los sistemas caóticos de interés, donde los sistemas de Lorenz y Chen fueron resueltos utilizando el método numérico de Runge-Kutta 4 con los siguientes parámetros: tamaño de paso $h = 0.0001$ y condiciones iniciales $x_0 = 0.1, y_0 = 0.1, z_0 = 0.1$ para el sistema de Lorenz, tamaño de paso $h = 0.01$ y condiciones iniciales $x_0 = 0.1, y_0 = 0.1, z_0 = 0.1$ para el sistema de Chen, y la neurona HR fue resuelta utilizando el método de Forward-Euler, con un tamaño de paso de $h = 0.01$ y condiciones iniciales $x_0 = 0.1169, y_0 = 0.0356, z_0 = 0.0103$.

Tabla 2.1: Series temporales caóticas usadas en esta tesis.

Sistema Caótico	Ecuaciones Diferenciales	Serie Temporal
Lorenz [46]	$\dot{x} = \sigma(y - x)$ $\dot{y} = x(\rho - z) - y$ $\dot{z} = xy - \beta z$	
Neurona Hindmarsh Rose [47]	$\dot{x} = -s(-ax^3 + x^2) - y - bz$ $\dot{y} = \varphi(x^2 - y)$ $\dot{z} = \varepsilon(sa_1x + b_1 - kz)$	
Chen [48]	$\dot{x} = a(y - x)$ $\dot{y} = (c - a)x - xz + cy$ $\dot{z} = xy - bz$	

2.2. Series Temporales Caóticas

Se pueden obtener series temporales caóticas utilizando datos de fenómenos reales como el clima, las finanzas, el transporte, etc. Un ejemplo es el Producto Interno Bruto de México (PIB). En la figura 2.1 se puede observar el comportamiento del PIB desde 1980 hasta el segundo trimestre de 2023, el cual es un dato decisivo en la economía del país, y es afectado por una gran variedad de factores, lo que implica

que su comportamiento no es periódico. El PIB en México es reportado cada trimestre [49], por tanto, cada dato de la figura representa el valor del PIB trimestral en MDP (Millones de Pesos Mexicanos) base 2018.

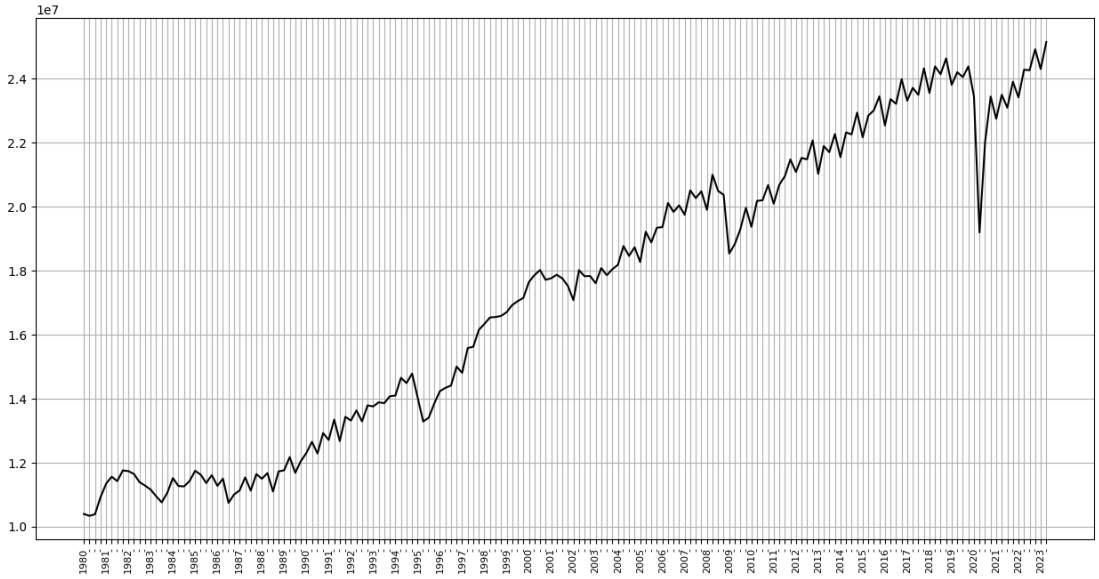


Figura 2.1: Producto Interno Bruto (PIB) de México.

2.3. Escalamiento de Series Temporales Caóticas

Para la predicción de series temporales caóticas mediante métodos de aprendizaje automático basados en neuronas que utilizan funciones de activación con la tangente hiperbólica ($\tanh(\cdot)$), se recomienda que el rango dinámico de los datos se ajuste al intervalo $[-1:1]$ [33, 30]. Este requisito puede lograrse mediante el re-escalamiento del sistema dinámico, ya sea a partir de sus ecuaciones de estado o mediante la manipulación de los datos de la serie temporal. Se sugiere re-escalar las ecuaciones de estado, dada la importancia de preservar las propiedades inherentes de los sistemas

caóticos, las cuales pueden verse comprometidas por cambios bruscos en la escala de los datos. Este enfoque permite ajustar el rango dinámico de los datos de manera eficiente sin sacrificar la integridad del sistema.

El re-escalamiento de un sistema caótico implica la aplicación de un cambio de variables conforme se muestra en la ecuación (2.3.1) sobre el sistema de interés.

$$\begin{aligned} x &= k_1 u \\ y &= k_2 v \\ z &= k_3 w \end{aligned} \tag{2.3.1}$$

La modificación de los valores de k_1, k_2 y k_3 en el nuevo sistema permite ajustar su rango dinámico. Es importante destacar que los valores de estos coeficientes (k) están intrínsecamente ligados a las características específicas de cada sistema en cuestión. En consecuencia, determinar los valores óptimos de k para lograr el ajuste deseado requiere un proceso iterativo.

Este proceso iterativo implica la evaluación sistemática de diversas combinaciones de valores para k_1, k_2 y k_3, seguido de la observación de cómo estas modificaciones afectan el comportamiento dinámico del sistema. El objetivo es encontrar una configuración de coeficientes que permita alcanzar el rango dinámico deseado sin comprometer las propiedades fundamentales del sistema.

Para ilustrar el proceso de escalado de sistemas caóticos, tomaremos como referencia el sistema de Lorenz. El conjunto de ecuaciones diferenciales que describen este sistema es mostrado en la tabla 2.1, el primer paso consiste en aplicar el cambio de variables mostrado en la ecuación (2.3.1), lo que nos permite obtener un nuevo conjunto de ecuaciones que describe el sistema transformado (2.3.2).

$$\begin{aligned}\dot{u} &= \sigma(\tfrac{k_2}{k_1}v - u)\\ \dot{v} &= \tfrac{k_1}{k_2}u(\rho - k_3 w) - v\\ \dot{w} &= \tfrac{k_1 k_2}{k_3}uv - \beta w\end{aligned} \tag{2.3.2}$$

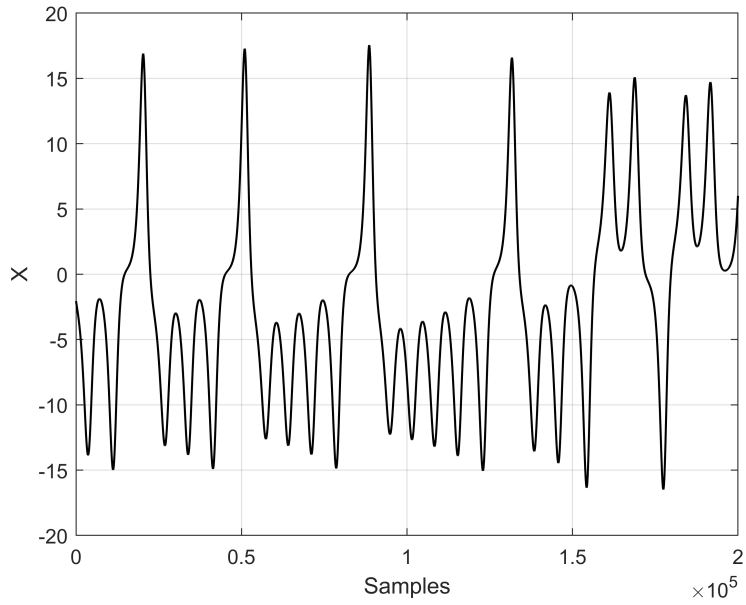
Con el nuevo conjunto de ecuaciones, procedemos a realizar una búsqueda sistemática de los valores de k que aseguren que el rango dinámico del sistema transformado se sitúe dentro del intervalo $[-1, 1]$. Este proceso implica explorar diferentes combinaciones de valores para k, evaluando cómo estas modificaciones afectan el rango dinámico del sistema.

La figura 2.2 muestra el sistema original de Lorenz (2.2a) y su versión escalada (2.2b), donde podemos notar que el sistema escalado sigue presentando un comportamiento caótico.

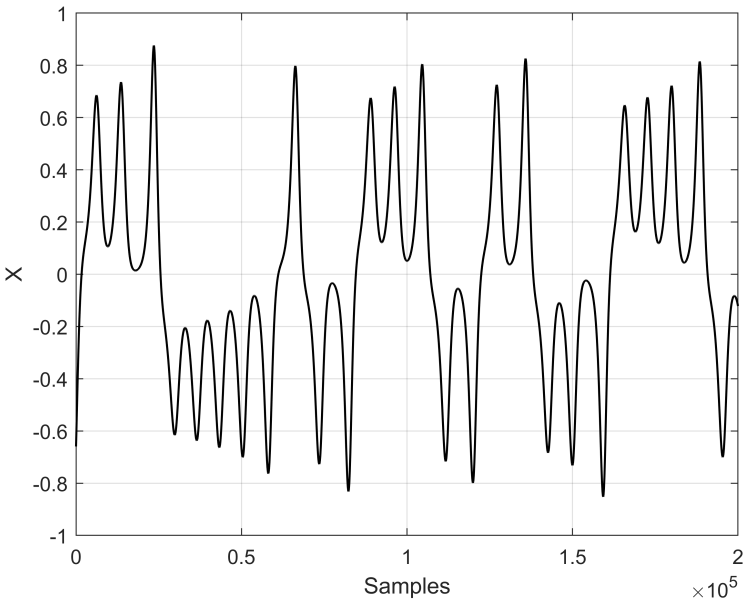
En el caso de utilizar series temporales caóticas de algún sistema real, como el PIB, se suelen utilizar funciones nativas del lenguaje de programación que se este utilizando que re-escalen directamente los datos ya que no se cuenta con el modelo matemático. La figura 2.3 muestra el PIB original (2.3a) y el PIB escalado (2.3b) usando la función *MinMaxScaler* de Python.

2.4. Decimación de Series Temporales Caóticas

Una estrategia presentada en la literatura [33] para mejorar la predicción de series temporales caóticas, especialmente en aquellas que presentan un comportamiento lento, es el uso de la técnica de decimación. La decimación por un entero $D > 1$ en una secuencia $x[n]$ consiste en mantener cada D_{th} muestra de $x(n)$ y eliminar $D - 1$



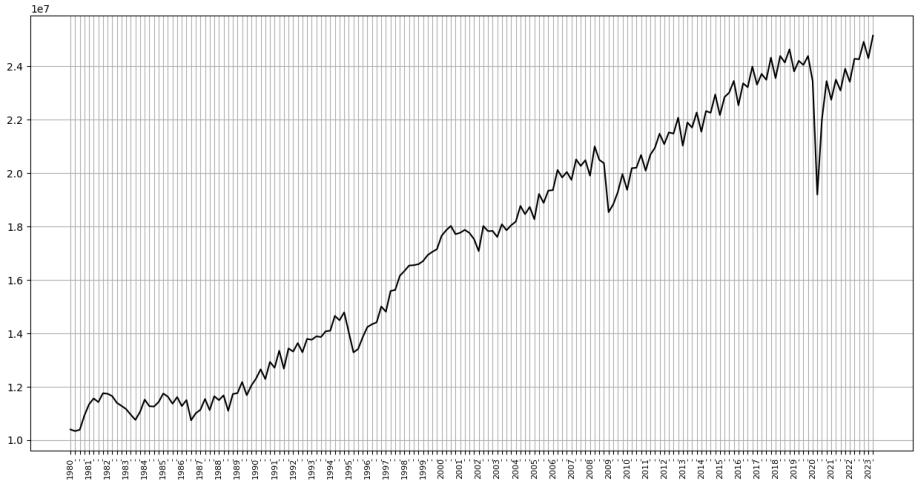
(a)



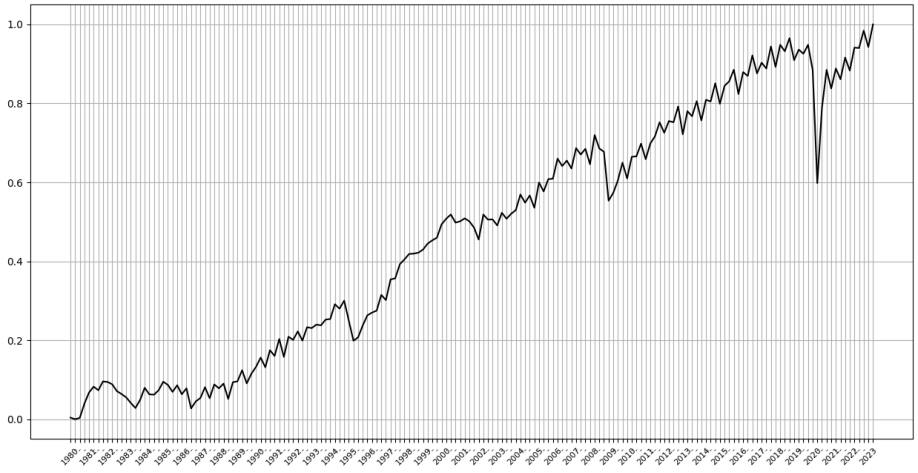
(b)

Figura 2.2: Series temporales caóticas. (a): Sistema de Lorenz original, (b): Sistema de Lorenz escalado.

muestras intermedias, generando así una secuencia de salida $y[d]$ [50]. Para utilizar esta técnica de manera efectiva, es crucial preservar las principales características



(a)



(b)

Figura 2.3: Series temporales caóticas. (a): Producto Interno Bruto de México original, (b): Producto Interno Bruto de México escalado.

de las series temporales caóticas. Esto implica que no se deben alterar significativamente los exponentes de Lyapunov, la dimensión Kaplan-Yorke (DKY), y la serie temporal debe mantener sus dinámicas sin que se observe una deformación de estas.

Los exponentes de Lyapunov pueden ser determinados a partir del modelo matemático del sistema o directamente de los datos. Una recomendación general es calcularlos a partir de los datos, ya que esto elimina la necesidad de conocer el modelo matemático del sistema y de manera implícita se consideran parámetros importantes del sistema que pueden ser omitidos en el modelo matemático. El software TISEAN facilita esta tarea [51], ya que permite calcular los exponentes de Lyapunov utilizando solo las series temporales. A continuación se presenta un ejemplo de cómo utilizar este software:

- *lyap_spec.exe serieTemporal.txt -c2,3,4 -m3,1 -k5000 -oResultados.txt*

Donde *lyap_spec.exe* es el ejecutable disponible en la pagina de TISEAN, *serieTemporal.txt* contiene las series temporales, *-c2,3,4* son las columnas que contienen las variables X,Y,Z, *-m3,1* es la dimensión embebida, *-x5000* elimina 5000 datos que generalmente corresponden a la parte transitoria, *-oResultados.txt* almacena los resultados obtenidos. Dependiendo de las dinámicas de las series temporales y la cantidad de datos disponibles se pueden ajustar otros parámetros los cuales dependen de cada sistema y se pueden consultar en la pagina web del software.

Una vez obtenidos los exponentes de Lyapunov, se puede utilizar la ecuación (2.4.1) para calcular la dimensión Kaplan-Yorke [52]. Este proceso debe hacerse tanto para las series temporales antes y después de realizar la decimación, con el fin de verificar que esta técnica no afecte las propiedades fundamentales de las series temporales caóticas.

$$DKY = (M-1) + \frac{\sum_{i=1}^{M-1} \lambda_i}{|\lambda_M|} \tag{2.4.1}$$

Donde M es el orden del sistema, λ_M es el mínimo exponente de Lyapunov y λ_i

corresponde a los exponentes de Lyapunov restantes. La DKY es un valor que oscila entre $[M-1, M]$, entre más cercano sea el valor a M mayor es la caoticidad del sistema.

Proponemos realizar la decimación de la serie temporal de Lorenz y de la serie temporal de la neurona HR, ya que estas series temporales se van a utilizar en la predicción sin forzamiento del maestro. Se incrementa el valor de la decimación hasta que se dejen de cumplir los requisitos mencionados previamente. En la tabla 2.2 se presenta el máximo exponente de Lyapunov y la dimensión Kaplan-Yorke para la serie temporal de Lorenz sin decimación y con decimaciones iguales a 10, 25 y 50, complementariamente, en la figura 2.4 se observa que para estos valores de decimación la serie temporal conserva su dinámica caótica sin distorsiones.

Tabla 2.2: Máximo exponente de Lyapunov (MLE) y dimensión Kaplan-Yorke (D_{KY}) del sistema de Lorenz para diferentes valores de decimación.

D	**0**	**10**	**25**	**50**
MLE	0.0061	0.0016	0.0022	0.0038
$\mathbf{D_{KY}}$	2.7995	2.2518	2.1265	2.1079

Una de las principales características de la neurona HR es su comportamiento lento, lo que la hizo una gran candidata para el método de decimación, ya que al cambiar el tamaño de paso del método numérico para obtener los datos, se pierde fácilmente su caoticidad. La tabla 2.3 presenta el máximo exponente de Lyapunov y la dimensión Kaplan-Yorke para la serie temporal de la neurona HR sin decimación y con decimaciones igual a 10, 25 y 50, complementariamente, en la figura 2.4 se observa que para estos valores de decimación la serie temporal conserva su dinámica caótica sin distorsiones.

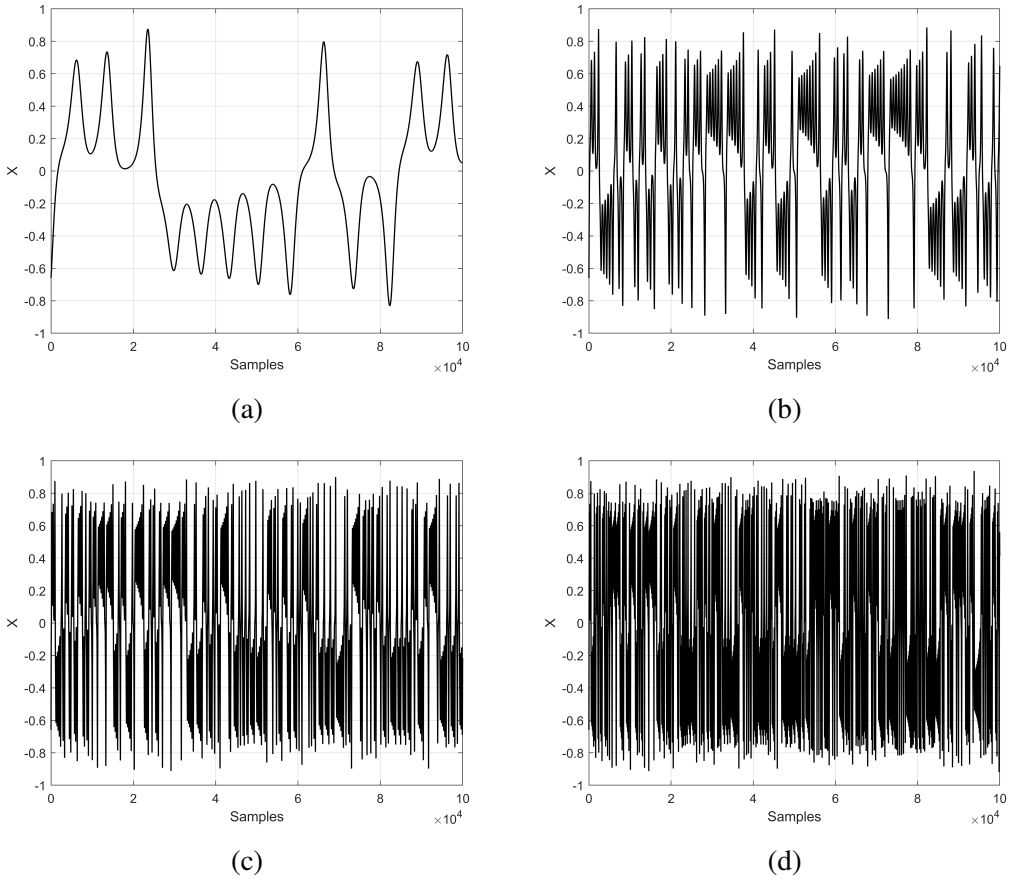


Figura 2.4: Series temporales del sistema de Lorenz con decimación. (a): $D = 0$, (b): $D = 10$, (c): $D = 25$, (d): $D = 50$.

Tabla 2.3: Máximo Exponente de Lyapunov (MLE) y Dimensión Kaplan-Yorke (D_{KY}) de la neurona HR para diferentes valores de decimación.

D	**0**	**10**	**25**	**50**
MLE	0.0012	0.0113	0.0283	0.0566
$\mathbf{D_{KY}}$	2.3401	2.2754	2.2748	2.2736

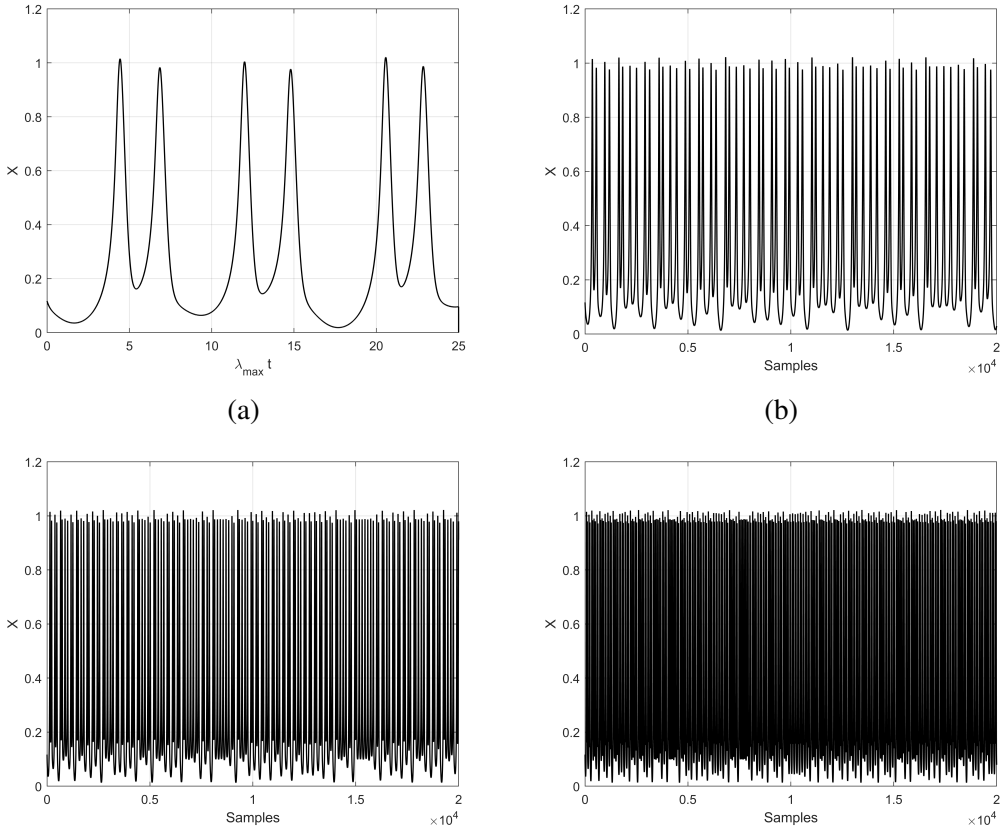


Figura 2.5: Series temporales de la neurona HR con decimación. (a): $D = 0$, (b): $D = 10$, (c): $D = 25$, (d): $D = 50$.

Capítulo 3

Predicción de Series Temporales Caóticas usando redes ESN

Como se mencionó en el capítulo 1, existen dos esquemas generales para la predicción de series temporales caóticas, la primera es sin forzamiento del maestro que consiste en realizar una retroalimentación de los datos predichos como entrada para la siguiente predicción, y la segunda es con forzamiento del maestro, donde las entradas para realizar las predicción provienen del conjunto de datos de prueba. En este capítulo nos enfocaremos en las Redes de Estado Eco (ESN), que han sido ampliamente utilizadas en la predicción de series temporales caóticas sin forzamiento del maestro, lo que permite alcanzar mayores horizontes de predicción.

Se ha utilizado una gran variedad de métodos de aprendizaje automático para la predicción de series temporales caóticas sin forzamiento del maestro. En la Tabla 3.1 se presentan algunos trabajos relacionados, donde se puede observar que las ESN han sido ampliamente utilizadas en diversos sistemas caóticos y también se han explorado diferentes modelos híbridos de ESN con otros métodos de aprendizaje automático.

Tabla 3.1: Predicción de series temporales caóticas usando diferentes métodos de aprendizaje automático.

Método	**Conjunto de datos**	**Pasos Predichos**	$\lambda_{\text{máx}}^{-1}$	**RMSE**
ESN [28]	Lorenz	300	—	—
ESN [6]	Lorenz	460	10.35	—
RNN-LSTM [6]	Lorenz	180	4.05	—
ANN [6]	Lorenz	120	2.7	—
CEEMDAN-LSTM [53]	Lorenz	—	2	1.327
ESN [39]	Lorenz	700	—	—
RESN [23]	Lorenz	500	—	0.2238
	Rossler	500	—	0.1128
AESN [54]	Lorenz	1	—	6×10^{-3}
	Rossler	1	—	1.65×10^{-2}
HESN [22]	Lorenz	—	12	—
DPM [55]	Lorenz	300	—	3.3×10^{-3}
Fuzzy [56]	Mackey-Glass	1000	—	3.8×10^{-3}
ADRC [40]	Rossler	40	—	1.41×10^{-2}
ALM-ESN [44]	Lorenz	1	—	2.21×10^{-5}
	Mackey-Glass	84	—	1.9268×10^{-4}
HESN [38]	Rossler	28	—	0.8890
FESN [42]	Mackey-Glass	20	—	—
NARX [33]	Chaotic Serie	600	—	6.81×10^{-2}
ESN [33]	Chaotic Serie	600	—	2.94×10^{-2}

En este capítulo se describe una ESN en su forma clásica, posteriormente, se explica el proceso de optimización de una ESN utilizando PSO y los parámetros que pueden ser optimizados. Luego abordamos el problema de la implementación de una ESN en FPGA y algunas alternativas para reducir el costo computacional de la implementación, mostrando como las propuestas afectan el horizonte de predicción y finalmente, algunas alternativas para la implementación de la función de activación.

3.1. Red de Estado Eco

Las Redes de Estado Eco (ESN, por sus siglas en ingles), son un tipo de red neuronal recurrente conformada principalmente por tres capas: la capa de entrada, la capa oculta o también conocida como depósito (Reservoir) que contiene N neuronas y la capa de salida, como se muestra en la figura 3.1.

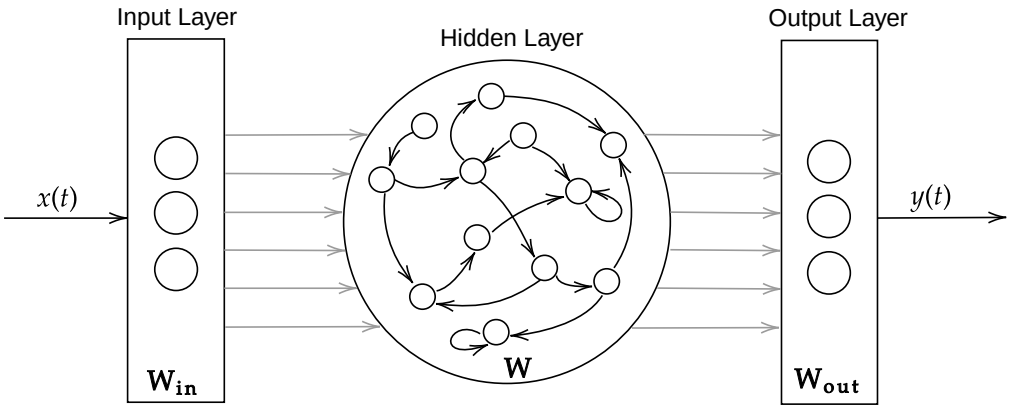


Figura 3.1: Diagrama de una Red de Estado Eco.

La ESN con Integrador Permeable, es una de las variantes más usadas [57], la ecuación (3.1.1) describe los estados internos, donde a es la tasa de aprendizaje, que controla la velocidad de actualización del vector de estado del depósito; $b_{in} = 1$ es el valor de sesgo de entrada; $\mathbf{W_{in}} \in \mathbb{R}^{N_x \times (1+N_u)}$ es la matriz de entrada; $\mathbf{W} \in \mathbb{R}^{N_x \times N_x}$ la matriz de conexiones internas; estas matrices son generadas aleatoriamente con una distribución uniforme entre $[-1, 1]$; $f(\cdot)$ es la función de activación, en este caso $\tanh(\cdot)$; γ es la ganancia, se recomienda que sea 1; $\mathbf{W_{bf}} \in \mathbb{R}^{N_x \times N_y}$ es la matriz de pesos de realimentación, y generalmente es igual a 0.

$$\mathbf{h(t+1)} = (1 - a\gamma)\mathbf{h(t)} + \gamma f(\mathbf{W_{in}}[b_{in}; x(t+1)] + \mathbf{Wh(t)} + \mathbf{W_{bf}}y(t)) \qquad (3.1.1)$$

La matriz de conexiones internas $\mathbf{W}$ es generada aleatoriamente como se mencionó previamente, sin embargo, se debe re-escalar para que tenga un radio espectral especifico, el escalamiento se realiza utilizando la ecuación (3.1.2), donde $\mathbf{W_r}$ es la matriz de conexiones internas re-escalada; ρ_r es el radio espectral deseado; $\rho_r(\mathbf{W})$ es el radio espectral de la matriz $\mathbf{W}$. Además, se recomienda que el radio espectral sea un valor menor o igual a 1 para que se cumpla la propiedad de estado Eco [58, 59].

$$\mathbf{W_r} = \frac{\rho_r}{\rho_r(\mathbf{W})}\mathbf{W} \tag{3.1.2}$$

Solo la capa de salida es entrenada, donde $\mathbf{W_{out}}$ es la matriz cuyos pesos son 'entrenables'. La capa de salida esta definida por la ecuación (3.1.3). Donde b_{out} es el sesgo de salida; la matriz de pesos de salida $\mathbf{W_{out}}$ esta disponible después de la etapa de entrenamiento, en esta etapa se busca tener el menor error posible entre $y_i(t)$ y el valor objetivo Y_{target}. El calculo de la matriz $\mathbf{W_{out}}$ esta dado en la ecuación (3.1.4) [60]; donde λ es el coeficiente de regularización, usado para prevenir el sobre-entrenamiento; $\mathbf{X} \in \mathbb{R}^{(1+N_u+N_x)\times T}$ es la matriz que concatena horizontalmente todos los vectores de estado $\mathbf{x}$ a lo largo de $t = 1, ..., T$.

$$y_i(t+1) = \mathbf{W_{out}}[b_{out}; x(t); \mathbf{h(t)}] \tag{3.1.3}$$

$$\mathbf{W_{out}} = \mathbf{Y}_{target}\mathbf{H}^T(\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I})^{-1} \tag{3.1.4}$$

Los principales hiperparámetros de la ESN deben ser ajustados para cada conjunto de datos sobre el que se desea realizar la predicción. La selección de estos hiperparámetros determina la precisión y el horizonte de predicción que se puede

alcanzar, y aunque existen recomendaciones generales para algunos de ellos, no hay una fórmula exacta que permita seleccionar el mejor conjunto de hiperparámetros.

- *Tasa de aprendizaje* α: Es el porcentaje de cambio con el que se actualizan los pesos en cada iteración.

- *Radio Espectral (SR):* Corresponde al máximo auto-valor de la matriz de conexiones internas ($\mathbf{W}$). Se recomienda que este parámetro se encuentre entre$(0, 1]$ para asegurar la propiedad de Estado Eco [60].

- *Tamaño del depósito (N):* Representa el número de neuronas dentro del depósito. Este parámetro es muy importante ya que decide el número máximo de posibles conexiones en el depósito (N^2) [61]. Jaeger ha sugerido que N se encuentre en el rango ($\frac{T}{10} \leq N \leq \frac{T}{2}$) con T como la longitud del conjunto de datos de entrenamiento.

- *Sesgo Entrada/salida:* El peso de la matriz de entrada ($\mathbf{W_{in}}$) influencia el nivel de linealidad de la respuesta del depósito. Para una $\mathbf{W_{in}}$ que esta uniformemente distribuida el sesgo de entrada b_{in} se representa en un rango $[-b; b]$.

- *Función de Activación:* Para la ESN, la función de activación es no lineal, en la mayoría de trabajos se utiliza la función tangente hiperbólica $tanh(.)$ o la función signo $sign(.)$ [62].

- *Parámetro de Regularización:* Suele tener como objetivo reducir la sensibilidad al ruido de la red y evitar el sobre-ajuste [63].

3.1.1. Optimización de las ESN para la Predicción de Series Temporales Caóticas

Las metaheurísticas son técnicas y algoritmos iterativos que utilizan algún nivel de aleatoriedad para descubrir soluciones óptimas o casi óptimas a problemas computacionalmente complejos [64]. Entre las más comúnmente aplicadas se encuentran la Evolución Diferencial (DE) [65], los Algoritmos Genéticos (GA) [66] y el Enjambre de Partículas (PSO) [20]. Se ha demostrado en diversos artículos que la optimización de los hiperparámetros de la ESN permite reducir el error en la predicción de series temporales caóticas y/o aumentar el horizonte de predicción [64]. Esto se debe a que los algoritmos realizan una búsqueda exhaustiva o aleatoria en el espacio de los hiperparámetros, encontrando la mejor combinación según la función de aptitud o función objetivo establecida. En la Tabla 3.2 se presentan algunos ejemplos.

Tabla 3.2: Optimización de una ESN para la predicción de series temporales caóticas.

Optimizador	**Serie Temporal**	**Función de aptitud**	**Valor**	**Datos de Prueba**
BSA[26]	Canadan Lynx	MSE	5.17×10^{-2}	14
PSO[20]	Lorenz	RMSE	2.01×10^{-3}	500
	Mackey-Glass	RMSE	2.97×10^{-3}	500
CS[67]	Mackey-Glass	MSE	8.74×10^{-9}	1000
FOA[18]	Lorenz	RMSE	8.20×10^{-3}	4000
	Mackey-Glass	RMSE	9.60×10^{-3}	1000
SOGWO[27]	Mackey-Glass	RMSE	1.46×10^{-4}	800
WOA[68]	Lorenz	FF	7.37×10^{-8}	600
GA[68]	Lorenz	FF	5.54×10^{-6}	600
PSO[69]	Lorenz	RMSE	3.24×10^{-2}	500
BGWO[70]	Mackey-Glass	RMSE	1.03×10^{-2}	500
DE[71]	Mackey-Glass	NRMSE	2.59×10^{-4}	2000
PSO[72]	Mackey-Glass	MSE	8.12×10^{-5}	1000
Neuro-Evolucion[73]	Hénon	NMSE	4.23×10^{-3}	3000
MBBO[74]	Lorenz	RMSE	2.28×10^{-4}	2000

Para optimizar los diferentes parámetros de la ESN se pueden utilizar diversas metaheurísticas, como se mencionó previamente. PSO es uno de los algoritmos más utilizados debido a sus buenos resultados en la optimización de métodos de aprendizaje automático. Por esta razón, vamos a profundizar en este algoritmo.

Un conjunto de soluciones forma la población inicial del PSO, esta población se genera en un espacio específico, cada partícula p está marcada por un par de posición y velocidad (x_i, v_i), debe actualizarse según las ecuaciones (3.1.5) y (3.1.6). Cada partícula i se mueve según su correspondiente vector v_i. En cada paso de tiempo, la calidad de las soluciones se evalúa de acuerdo con una función de aptitud o función objetivo [75, 76].

$$v_{ij} \leftarrow v_{ij} + c_1 \text{rand}()(p_{ij} - x_{ij}) + c_2 \text{rand}()(g_j - x_{ij}) \tag{3.1.5}$$

$$x_{ij} \leftarrow x_{ij} + v_{ij} \tag{3.1.6}$$

En las ecuaciones (3.1.5) y (3.1.6), c_1 y c_2 son coeficientes de aceleración, g es otra partícula que guarda la mejor posición global encontrada, y x_i, v_i son la posición y velocidad de la partícula, respectivamente.

El algoritmo 1 describe el proceso general para optimizar la matriz de conexiones internas $\mathbf{W}$ de una ESN, para la predicción de series temporales caóticas. En este caso de estudio, no tenemos restricciones, sin embargo, se debe considerar re-escalar la matriz $\mathbf{W}$ con un radio espectral menor o igual a uno para garantizar la propiedad de estado eco.

Algoritmo 1 Optimización de una ESN para predecir series temporales caóticas con PSO.

Inicializar la primera partícula de la población con parámetros conocidos, las demás partículas aleatoriamente $(\mathbf{x})$.
Inicializar la velocidad de las partículas v.
for $(counter = 1; counter \leq G; counter + +)$ **do**
    for $(i = 1; i \leq N_p; i + +)$ **do**
        for $(j = 1; j \leq D; j + +)$ **do**
            Para cada conjunto de partículas(p), entrenar la ESN, re-escalar la matriz $\mathbf{W}$ acorde al nuevo radio espectral.
            Calcular la matriz de salida W_{out} para cada conjunto de partículas (p).
            Predecir el sistema de Lorenz para cada conjunto de partículas (p).
            Calcular la función de aptitud MSE, entre los datos predichos y los datos objetivo
            Encontrar el mejor valor de p y guardarlo en g
            Evaluar la nueva velocidad usando (3.1.5).
            Evaluar la nueva posición usando (3.1.6).
        end for
        $f_x \leftarrow func(x_i)$
        if f_x es mejor que $score_i$ **then**
            $score_i \leftarrow f_x$
            $p_i \leftarrow x_i$
            if p_i es mejor que g **then**
                $g \leftarrow p_i$
            end if
        end if
    end for
end for
return x, p, g y $score$

La función de aptitud utilizada es la Media del Error Cuadrático (MSE, por sus siglas en ingles) entre los valores predichos (y_i) y los valores objetivo (y_{target}), ver anexo A.

3.2. Implementación en FPGA de una ESN para la Predicción de Series Temporales Caóticas

La implementación de redes neuronales en FPGA es un desafío de gran interés actualmente, debido a las múltiples aplicaciones y ventajas de integrar redes neuronales en hardware, especialmente desarrollarlas directamente en chips. Las Redes de Estado Eco (ESN) presentan la ventaja de tener un costo computacional relativamente bajo en comparación con otros métodos tradicionales. No obstante, un gran número de neuronas en el depósito de la ESN puede requerir una cantidad significativa de multiplicadores, siendo este un recurso limitado en FPGA.

En la siguiente sección, se describen varias topologías que ayudan a reducir la cantidad de multiplicadores necesarios para implementar ESNs en FPGA, lo que se traduce en un menor costo computacional. Además, se presentan diversas alternativas para implementar la función de activación, específicamente, la función tangente hiperbólica ($\tanh(\cdot)$).

3.2.1. Topologías y Horizonte de Predicción

Las topologías de conexión en las ESN se refieren a cómo las neuronas están interconectadas dentro del depósito, y estas conexiones se representan en la matriz $\mathbf{W}$. La topología más común es la *Completamente Conectada*, donde cada neurona está conectada con todas las demás, resultando en una matriz $\mathbf{W}$ completamente llena. Otra topología ampliamente utilizada especifica el porcentaje de conexiones, lo que define el número de valores *No Cero* en la matriz $\mathbf{W}$. Sin embargo, esta topología presenta el inconveniente de que las neuronas tienen diferentes números de entradas, lo que impide reutilizar un diseño de neurona estándar y requiere un diseño específico

para cada neurona según su número de entradas.

Vamos a describir diversas topologías junto con sus matrices de conexiones internas correspondientes. En todos los casos, los pesos de la matriz $\mathbf{W}$ se generan aleatoriamente con una distribución uniforme en el rango $[-1, 1]$ y se reescalan según la ecuación (3.1.2) para asegurar la propiedad de estado eco.

La Figura 3.2 ilustra la topología 1, en la cual se determina el número de entradas para cada neurona k. Cada fila de la matriz $\mathbf{W}$ representa las entradas de una neurona; por lo tanto, si $k = 2$, cada fila tendrá exactamente dos valores no nulos, cuyas posiciones se seleccionan aleatoriamente. Los valores en la diagonal de la matriz indican el peso de la conexión de la neurona consigo misma.

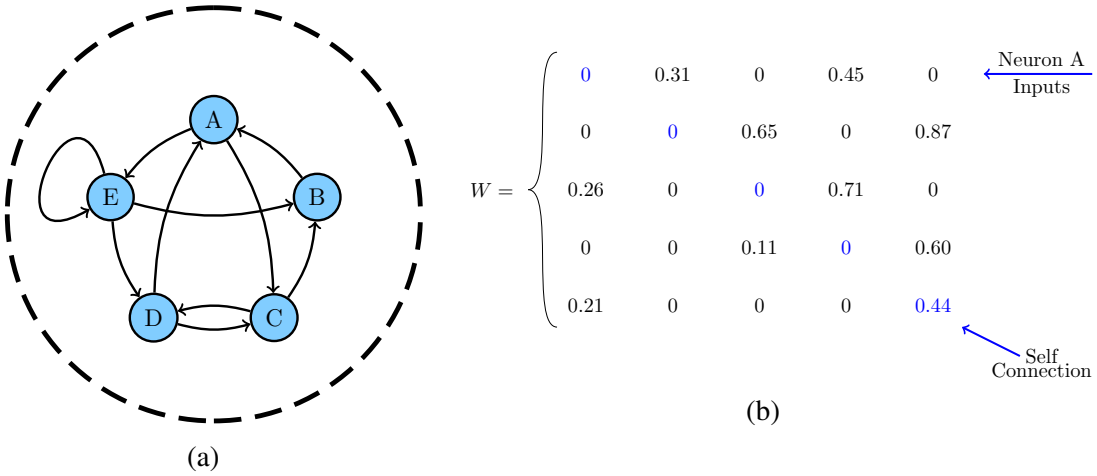
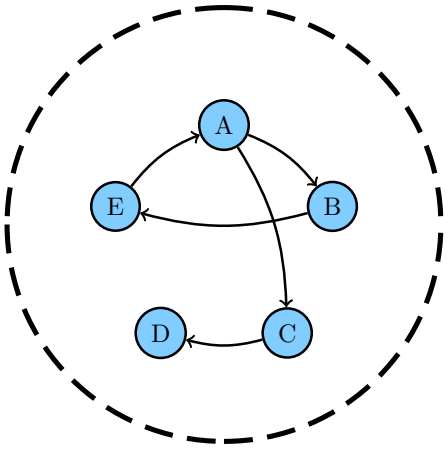


Figura 3.2: Topología 1 con 5 neuronas y $k = 2$. (a): Conexiones de las neuronas del depósito, (b): Matriz de conexiones internas.

La topología 2 es similar a la topología 1, en este caso $k = 1$, lo que indica que cada neurona tiene solo una entrada, y no hay un patrón definido en las conexiones, como se muestra en la figura 3.3 las conexiones son aleatorias.



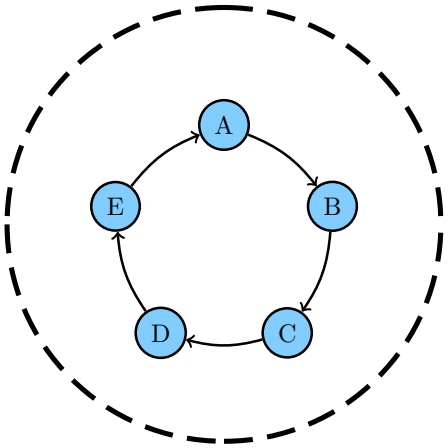
(a)

$$W = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0.65 \\ 0 & 0 & 0 & 0.18 & 0 \\ 0.21 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.11 & 0 & 0 \\ 0 & 0.54 & 0 & 0 & 0 \end{Bmatrix}$$

(b)

Figura 3.3: Topología 2 con 5 neuronas y $k = 1$. (a): Conexiones de las neuronas del depósito, (b): Matriz de conexiones internas.

La topología 3 es la topología anillo, donde $k = 1$ pero las conexiones de las neuronas siguen un patrón en forma de anillo como se muestra en la figura 3.4.



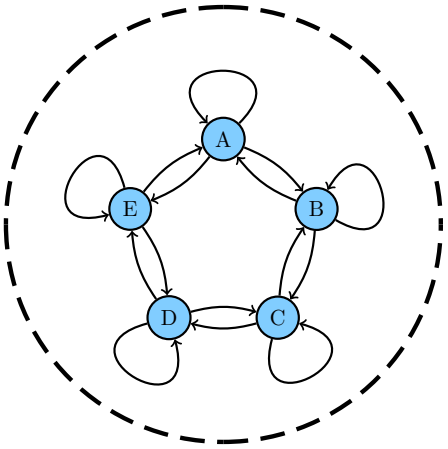
(a)

$$W = \begin{Bmatrix} 0 & 0 & 0 & 0 & 0.65 \\ 0.18 & 0 & 0 & 0 & 0 \\ 0 & 0.21 & 0 & 0 & 0 \\ 0 & 0 & 0.11 & 0 & 0 \\ 0 & 0 & 0 & 0.54 & 0 \end{Bmatrix}$$

(b)

Figura 3.4: Topología 3 con 5 neuronas, conexión de anillo. (a): Conexiones de las neuronas del depósito, (b): Matriz de conexiones internas.

La topología 4 tiene una conexión de anillo doble donde $k = 3$, y cada neurona tiene una conexión consigo misma como se muestra en 3.5.



(a)

$$W = \begin{Bmatrix} 0.32 & 0.15 & 0 & 0 & 0.65 \\ 0.18 & 0.21 & 0.36 & 0 & 0 \\ 0 & 0.11 & 0.72 & 0.45 & 0 \\ 0 & 0 & 0.11 & 0.59 & 0.27 \\ 0.46 & 0 & 0 & 0.33 & 0.52 \end{Bmatrix}$$

(b)

Figura 3.5: Topología 4 con 5 neuronas, conexión de anillo doble. (a): Conexiones de las neuronas del depósito, (b): Matriz de conexiones internas.

Para cada una de las topologías mencionadas previamente, la cantidad de conexiones internas se define según la tabla 3.3, donde N representa el número de neuronas en el depósito. Cabe destacar que las topologías 2 y 3 presentan el menor número de conexiones internas, con un valor equivalente al número de neuronas utilizadas.

Tabla 3.3: Cantidad de conexiones para cada topología (valores no cero de la matriz **W**.)

Topología	**Conexiones**
1	$k \times N$
2	N
3	N
4	$3N$

El horizonte de predicción se define como el intervalo de tiempo durante el cual el error normalizado es menor que un umbral k_n [77, 22], la ecuación (3.2.1) muestra como calcular el error normalizado [25], donde HP es el horizonte de predicción, y_{target} son los datos objetivo, y_i son los datos predichos. Para nuestras pruebas escogimos $k_n = 0.4$, ya que es un valor comúnmente usado por otros autores.

$$\frac{\|y_{target}(i) - y(i)\|}{\sqrt{\frac{1}{HP}\sum_{j=1}^{HP} \|y(j)\|^2}} \leq k_n \tag{3.2.1}$$

3.2.1.1. Complejidad Espacial

La complejidad espacial permite determinar la cantidad de recursos o espacio necesario para un algoritmo. En este caso, se consideran únicamente los recursos relacionados con las topologías del depósito en una ESN. La complejidad espacial de una ESN con una topología completamente conectada es de $O(N^2)$, ya que cada neurona está conectada con todas las demás, lo que genera un total de $N \times N$ conexiones. En cambio, una ESN con topología de anillo tiene una complejidad espacial de $O(N)$, dado que cada neurona tiene solo una conexión, resultando en N conexiones en el depósito, donde N es el número de neuronas.

Estos datos son cruciales para la implementación de redes ESN, ya que el número de conexiones en el depósito determina el número de multiplicadores que deben implementarse. Por ejemplo, si se utiliza un depósito con 50 neuronas completamente conectadas, se requieren $50 \times 50 = 2500$ multiplicadores para representar todas las conexiones. Suponiendo que se utiliza el mismo modelo de neurona, cada una de ellas tendría 50 multiplicadores. En contraste, con una topología de anillo, si el depósito cuenta con 50 neuronas, solo se necesitarían 50 multiplicadores en total, con cada neurona usando un único multiplicador.

La figura 3.6 ilustra la complejidad espacial de las dos topologías, mostrando cómo la cantidad de conexiones necesarias en el depósito -equivalente al número de multiplicadores requeridos— se ve afectada a medida que aumenta el número de neuronas en el depósito.

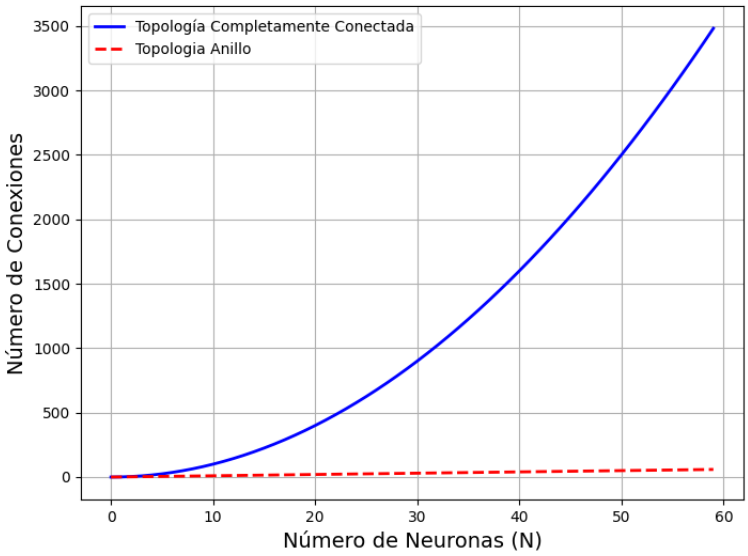


Figura 3.6: Complejidad espacial de las topologías de la ESN, topología anillo, $O(N)$, vs topología completamente conectada, $O(N^2)$.

3.2.2. Función de Activación

La función tangente hiperbólica es una de las más utilizadas como función de activación en métodos de aprendizaje automático, como es el caso de la ESN. Para la implementación de esta función en sistemas embebidos como una FPGA se deben utilizar aproximaciones matemáticas a dicha función, estas aproximaciones introducen un error en nuestros resultados, en el caso de la predicción de series temporales caóticas, este error disminuye la cantidad de pasos que se pueden predecir. A continuación presentamos algunas propuestas para aproximar la función tangente hiperbólica, todas ellas están definidas como funciones por partes.

La primera opción es aproximar la función tanh utilizando PWL (Piece Wise Linear), definiendo la función por partes mostrada en la ecuación (3.2.2), donde $L = 2$. La ecuación (3.2.3), representa la parte central de la función tangente hiperbólica

utilizando dos ecuaciones de segundo grado, donde $\theta = 0.25$ y $\beta = 1$ [78].

$$f(x) = \begin{cases} 1 & for \quad L \leq x \\ H(x) & for \quad -L < x < L \\ -1 & x \leq -L \end{cases} \tag{3.2.2}$$

$$H(x) = \begin{cases} x(\beta - \theta x) & for \quad 0 \leq x \leq L \\ x(\beta + \theta x) & -L \leq x < 0 \end{cases} \tag{3.2.3}$$

La siguiente propuesta es utilizar Splines y PCHIP (Shape-Preserving Piecewise Cubic Interpolation) para la aproximación de la tangente hiperbólica, ambos realizan la aproximación de una forma similar, definiendo la función por partes mostrada en la ecuación (3.2.4), donde $L = 3$, y dividiendo la parte central positiva de la función correspondiente al rango $[0, L]$ (ya que es una función impar) en intervalos de tamaño (Δx), en este caso $\Delta x = 0.2$. Dado que los coeficientes polinómicos obtenidos son coeficientes locales para cada intervalo, debe restar el extremo inferior del intervalo correspondiente para usar los coeficientes en una ecuación polinomial convencional. En otras palabras, para los coeficientes $[a, b, c, d]$ en el intervalo $[x_1, x_2]$, el polinomio correspondiente está definido en la ecuación (3.2.5) [79], esto significa que para el rango entre $[0, L]$, hay que definir $(L/(\Delta x))$ polinomios, o utilizar multiplexores para definir que variables se deben implementar cada que se evalúe la función.

$$f(x) = \begin{cases} 1 & for \quad L \leq x \\ \text{sign}(x)H(|x|) & for \quad -L < x < L \\ -1 & x \leq -L \end{cases} \tag{3.2.4}$$

$$H(x) = a(x - x_1)^3 + b(x - x_1)^2 + c(x - x_1) + d \tag{3.2.5}$$

La principal diferencia entre los Splines y PCHIP es la forma en la que calculan la pendiente de x_j.

La ultima propuesta es aproximar la función tanh usando funciones polinómicas, para esto primero se define la función por partes mostrada en la ecuación (3.2.4), donde $L = 3$, luego se generan polinomios de grado 3, 4 y 6, descritos por las ecuaciones (3.2.6), (3.2.7) y (3.2.8) respectivamente. dependiendo del orden del polinomio será la cantidad de multiplicadores necesarios, lo que es un aspecto muy importante que debe ser considerado al momento de la implementación en FPGA.

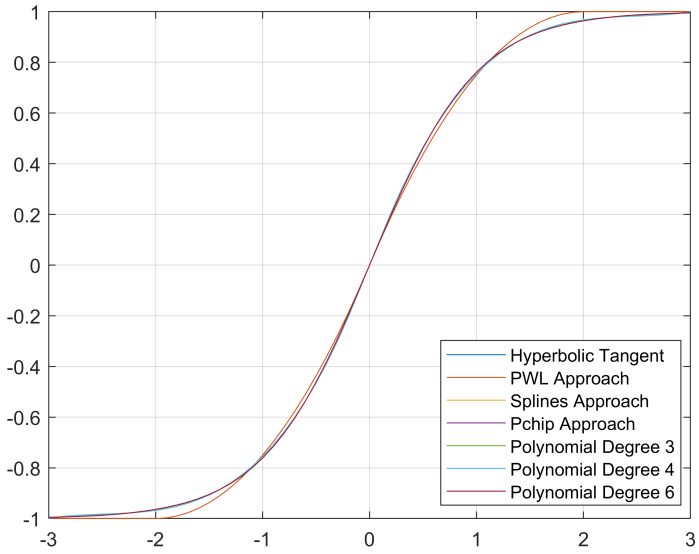
$$H(z) = 0.06151z^3 - 0.4597z^2 + 1.161z - 0.007825 \tag{3.2.6}$$

$$H(z) = 0.0003316z^4 + 0.05952z^3 - 0.456z^2 + 1.159z - 0.007577 \tag{3.2.7}$$

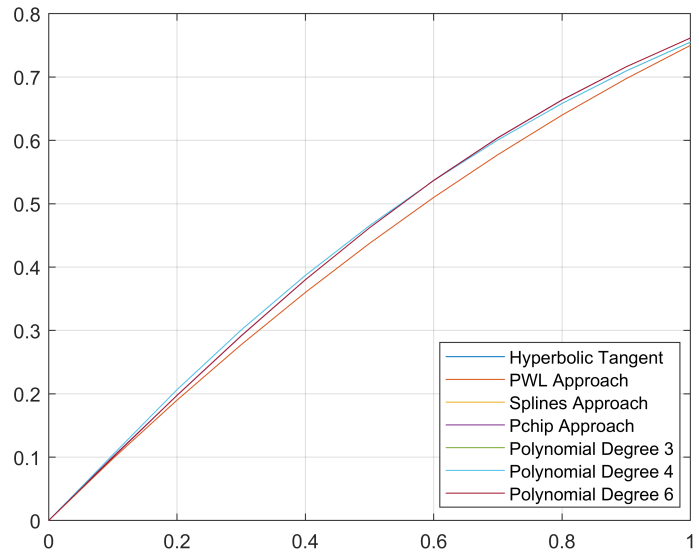
$$H(z) = 0.006923z^6 - 0.07478z^5 + 0.3044z^4 - 0.515z^3 + 0.0415z^2 + 0.998z - 0.0001633 \tag{3.2.8}$$

Para realizar todas las aproximaciones propuestas de la función tangente hiperbólica, utilizamos el software MATLAB R2022b, el cual nos permite encontrar todos los coeficientes de los Splines y PCHIP, así como realizar las aproximaciones polinomiales. La figura 3.7 muestra los resultados de la simulación de la función original y las aproximaciones.

La tabla 3.4, muestra el error RMSE entre la función tangente hiperbólica y las diferentes aproximaciones propuestas, podemos notar que la aproximación PWL que es la más sencilla y de menos costo computacional es la que presenta mayor error, el



(a)



(b)

Figura 3.7: Aproximaciones de la Función Tangente Hiperbólica. (a): Rango $[-3, 3]$, (b): Zoom.

menor error es obtenido con la aproximación PCHIP, sin embargo, para su implementación es necesario utilizar multiplexores y memorias para almacenar las constantes, la siguiente aproximación con los mejores resultados es el polinomio de grado 6, pero

esta es la que implica un mayor costo computacional.

Tabla 3.4: Resultados de simulación de las diferentes aproximaciones de la función tangente hiperbólica.

Método	**RMSE**
PWL	2.3223×10^{-2}
Splines 3	8.9577×10^{-4}
Pchip	6.8125×10^{-7}
Polinomio G. 3	4.4882×10^{-3}
Polinomio G. 4	4.4885×10^{-3}
Polinomio G. 6	4.3519×10^{-4}

Capítulo 4

Predicción de Series temporales Caóticas usando el Modelo Híbrido LSTM-ESN

En este capítulo abordaremos la predicción de series temporales caóticas con forzamiento del maestro. Las redes Long Short-Term Memory (LSTM) son ampliamente utilizadas en esta tarea, sin embargo, tienen un alto costo computacional debido al entrenamiento de todas las matrices que componen sus celdas, en este capítulo veremos una propuesta de una red LSTM combinada con redes de estado echo (LSTM-ESN) dentro de sus celdas, buscando reducir el costo computacional debido al entrenamiento. Adicionalmente, añadimos un método de gradiente descendiente de orden fraccionario para entrenar la matriz de salida del modelo propuesto, para aprovechar la propiedad de memoria implícita en los métodos numéricos utilizados para resolver derivadas de orden fraccionario.

En la predicción de series temporales caóticas con forzamiento del maestro, el

método de ventanas deslizantes es introducido. Este método consiste en dividir los datos en ventanas de longitud M, las cuales sirven de entrada a los métodos de aprendizaje automático. Cada ventana puede tener 1 o D puntos de salida, representando la cantidad de pasos hacia adelante que pueden ser predichos. Este concepto es ilustrado en la figura 4.1, donde el tamaño de la ventana es fijado en 6, y la salida esta conformada por 3 datos, esto significa que el sistema o método de aprendizaje automático tendrá 6 entradas y podrá predecir 3 pasos hacia adelante [80].

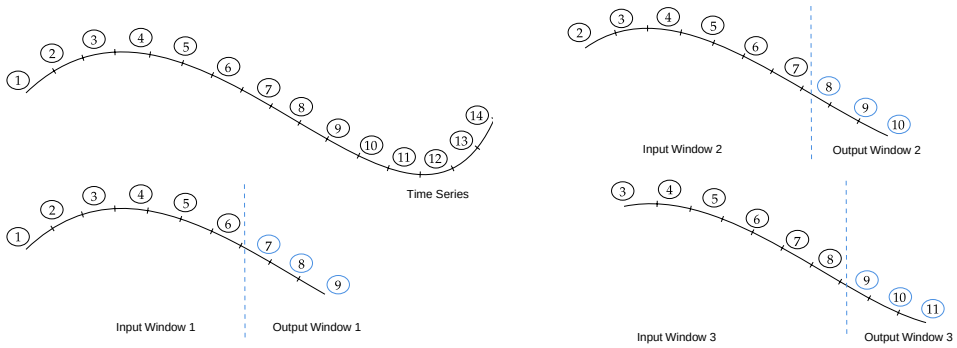


Figura 4.1: Método de ventanas deslizantes con múltiples salidas.

4.1. Long Short-Term Memory (LSTM)

Las redes LSTM (Long Short-Term Memory) son una categoría especial de las redes neuronales recurrentes, una de las principales características de las LSTM es que están conformadas por bloques de memoria, el bloque de memoria es una subred conectada recurrentemente que contiene módulos funcionales llamados la celda de memoria y compuertas [81]. La celda de memoria es la encargada de recordar el estado temporal de la red neuronal y las compuertas son las encargadas de controlar el patrón del flujo de información. Las compuertas están divididas en tres, la compuerta de entrada, encargada de controlar cuánta información nueva fluye hacia la celda de memoria, la compuerta de olvido que se encarga de controlar cuánta información

de la celda de memoria permanece en la actual celda de memoria, y la compuerta de salida que determina cuánta información es usada para calcular la activación de salida del bloque de memoria [41]. La figura 4.2 muestra la celda de memoria de la LSTM con sus respectivas compuertas.

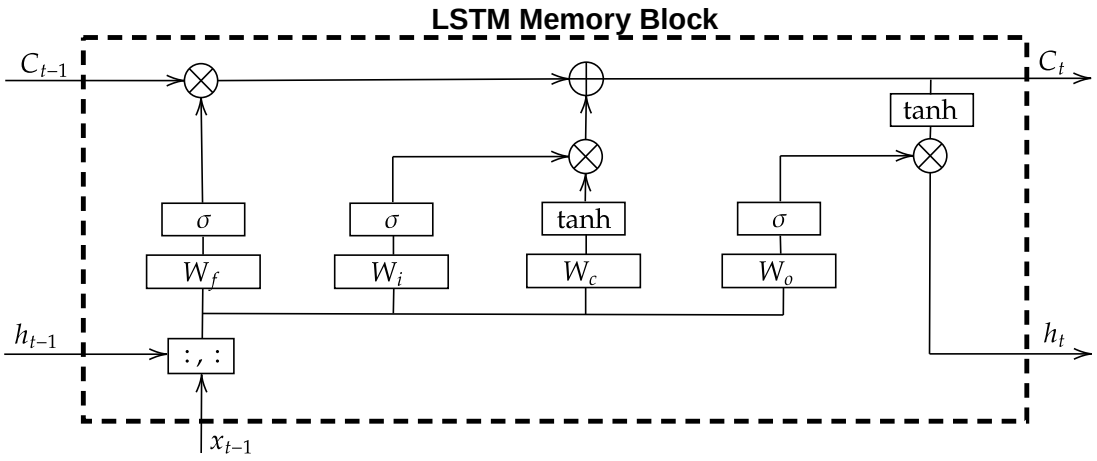


Figura 4.2: Bloque de memoria de una LSTM.

La celda de memoria de una red LSTM esta descrita por la ecuación (4.1.1), donde: σ es la función sigmoide, $w_{xi}, w_{xf}, w_{xo}, w_{xc} \in R^{d_h \times d_i}$, $w_{hi}, w_{hf}, w_{ho}, w_{hc} \in R^{d_h \times d_h}$ son los pesos entrenables, $b_i, b_f, b_o, b_c \in R^{d_h}$ son los sesgos, $c, h \in R^{d_h}$ son el estado de la celda y el estado oculto, $g_t^f, g_t^i, g_t^o \in R^{d \times (d_h + d_i)}$ representan las compuertas de olvido, entrada y salida respectivamente [36].

$$
\begin{aligned}
i_t &= \sigma(w_{xi}x_t + w_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(w_{xf}x_t + w_{hf}h_{t-1} + b_f) \\
o_t &= \sigma(w_{xo}x_t + w_{ho}h_{t-1} + b_o) \\
c_t &= f_t c_{t-1} + i_t \tanh(w_{xc}x_t + w_{hc}h_{t-1} + b_c) \\
h_t &= o_t \tanh(c_t)
\end{aligned}
\tag{4.1.1}
$$

La ecuación (4.1.2) describe la capa de salida de una LSTM donde $\mathbf{w_{hN}}$ son pesos

entrenables. En total, para una red LSTM se deben entrenar 12 matrices por cada celda que conforme la red y la matriz de salida. Generalmente, para entrenar los pesos de los parámetros de una LSTM se utiliza algún método de gradiente descendiente y se definen una cantidad de épocas en las que se realiza el entrenamiento, esto hace que las LSTM sean de alto costo computacional en su etapa de entrenamiento.

$$y_t = w_{h^N} h_t^N \tag{4.1.2}$$

4.2. Modelo Híbrido de una Red LSTM con ESN

Se propone una LSTM combinada con una ESN (LSTM-ESN) para reducir la cantidad de parámetros que deben ser entrenados y por consecuente el costo total del entrenamiento de la red. La figura 4.3 muestra la celda LSTM-ESN propuesta, donde las compuertas de entrada, salida y olvido son reemplazadas por redes de estado echo, así como el estado de la celda, donde la principal ventaja es que las matrices de la ESN son seleccionadas aleatoriamente, y las conexiones internas del depósito pueden estar escasamente conectadas, y por tanto, ya no es necesario entrenar los parámetros internos de la celda LSTM-ESN, reduciendo el costo computación debido al entrenamiento de dichas conexiones.

La ecuación (4.2.1) describe las ecuaciones de la celda LSTM-ESN propuesta, en la cual todas las matrices de pesos se generan aleatoriamente, como en una ESN, con una distribución uniforme en el intervalo $[-1, 1]$. Además, dichas matrices se reescalan de acuerdo con la ecuación (3.1.2), asegurando un radio espectral menor o igual a 1. Este enfoque permite reducir el costo computacional, ya que, a diferencia de la LSTM tradicional, donde todas las matrices de la celda suelen entrenarse me-

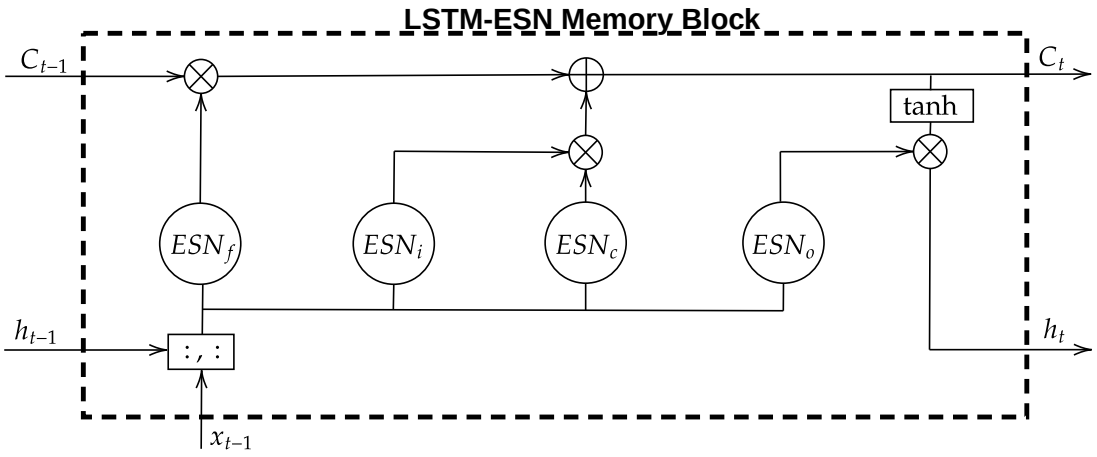


Figura 4.3: Bloque de memoria de una LSTM-ESN

diante un método de gradiente descendente, aquí no se requiere dicho entrenamiento. Finalmente, la función de activación $f(\cdot)$ corresponde a la tangente hiperbólica.

$$
\begin{aligned}
i_t &= (1-a)h_{t-1} + af(\mathbf{w_{xi}}x_t + \mathbf{w_{hi}}h_{t-1} + b_i) \\
f_t &= (1-a)h_{t-1} + af(\mathbf{w_{xf}}x_t + \mathbf{w_{hf}}h_{t-1} + b_f) \\
o_t &= (1-a)h_{t-1} + af(\mathbf{w_{xo}}x_t + \mathbf{w_{ho}}h_{t-1} + b_o) \\
c_t &= f_t c_{t-1} + i_t \tanh(\mathbf{w_{xc}}x_t + \mathbf{w_{hc}}h_{t-1} + b_c) \\
h_t &= o_t \tanh(c_t)
\end{aligned}
\tag{4.2.1}
$$

En la ecuación (4.2.1), las matrices $\mathbf{w_{xi}}, \mathbf{w_{hi}}, \mathbf{w_{xf}}, \mathbf{w_{hf}}, \mathbf{w_{xo}}, \mathbf{w_{ho}}, \mathbf{w_{xc}}, \mathbf{w_{hc}}$ se calculan como se describió anteriormente. Este proceso se realiza solo al inicio del entrenamiento y sus valores permanecen constantes a lo largo del mismo. Los valores de los sesgos (b_i, b_f, b_o, b_c) se fijaron en 1, y a representa la tasa de aprendizaje.

Además de la cantidad de celdas, es importante considerar la cantidad de neuronas requeridas en las redes ESN. Sin embargo, como se mostró en la sección anterior, las redes ESN pueden lograr buenos resultados en la predicción de series temporales caóticas utilizando un número reducido de neuronas en el depósito, lo que contribuye a mantener bajo el costo computacional del modelo propuesto.

En cuanto a la conexión de las celdas, la figura 4.4 ilustra su interconexión en el escenario de predicción con forzamiento del maestro. La cantidad de entradas corresponde al tamaño de la ventana deslizante, mientras que la cantidad de salidas está relacionada con el número de pasos hacia adelante que se pueden predecir. Además, existen otros tipos de interconexiones para las celdas LSTM documentadas en la literatura, que podrían ser exploradas en trabajos futuros.

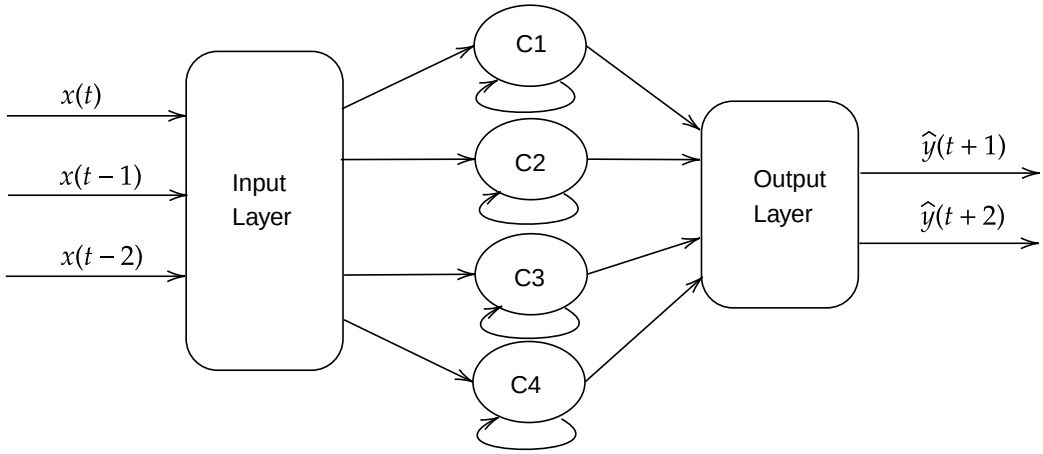


Figura 4.4: Conexión de las celdas de una red LSTM-ESN con forzamiento del maestro multi-salidas.

4.2.1. Entrenamiento de la matriz de salida

La red LSTM-ESN propuesta tiene un único parámetro entrenable: la matriz de salida, mostrada en la ecuación (4.1.2). Para entrenarla, se proponen dos métodos de actualización: el método de Gradiente Descendente y el método ADAM. Ambos requieren la definición de una función de costo y su derivada con respecto al parámetro que se desea optimizar [36]. En este caso, la función de costo seleccionada es el Error Cuadrático Medio (MSE), descrito en la ecuación (4.2.2), donde y_{target} es el valor objetivo, y_i es el valor predicho y N es el número total de datos.

$$f_c = \frac{1}{N}\sum_{i=1}^{N}(y_{target} - y_i)^2 \tag{4.2.2}$$

Al derivar la ecuación (4.2.2) con respecto a la matriz de salida $\mathbf{w_{hN}}$, se obtiene la ecuación (4.2.3), donde h_i^N representa los estados ocultos finales de las celdas LSTM-ESN.

$$\begin{aligned} g_i &= \tfrac{2}{N}\sum_{i=1}^{N}\dot{y}_i(y_i - y_{target}) \\ \dot{y}_i &= h_i^N \end{aligned} \tag{4.2.3}$$

El método de Gradiente Descendente actualiza los pesos de la matriz de salida según la ecuación (4.2.4), donde λ es la tasa de aprendizaje que controla el tamaño de los pasos dados hacia el mínimo de la función de costo.

$$w_{hN} = w_{hN} - \lambda g_i \tag{4.2.4}$$

El método ADAM, que combina las ideas de las técnicas de Gradiente Descendente con momento y RMSProp, ajusta los pesos de manera más eficiente en problemas de alta dimensionalidad y datos ruidosos. La función de actualización correspondiente está dada en la ecuación (4.2.5), donde $\beta_1 = 0.9$ y $\beta_2 = 0.99$ son los factores de decaimiento exponencial para las medias móviles de los primeros y segundos momentos del gradiente, respectivamente, y ε es una pequeña constante para evitar divisiones por cero [82].

$$\begin{aligned} m &= \beta_1 m + (1 - \beta_1) g_i \\ v &= \beta 2v + (1 - \beta_2) {g_i}^2 \\ whN &= whN - \tfrac{\alpha m}{\sqrt{v+\varepsilon}} \end{aligned} \tag{4.2.5}$$

ADAM ajusta dinámicamente la tasa de aprendizaje de cada parámetro, lo que suele resultar en una convergencia más rápida y estable en comparación con el Gradiente Descendente tradicional. Este método es especialmente útil para redes profundas y modelos de series temporales como la LSTM-ESN, ya que permite manejar eficientemente gradientes pequeños o variables.

4.3. Modelo Híbrido de una Red LSTM con ESN y Gradiente Descendiente de Orden Fraccionario

El cálculo fraccionario es una extensión natural de los operadores de integración y diferenciación tradicionales hacia órdenes no enteros, representado por el operador generalizado ${}_{\alpha}D_{a}^{t}$, donde a y t son los límites de la operación y $\alpha \in \mathbb{R}$ es el orden fraccionario. A diferencia del cálculo convencional, el cálculo fraccionario ofrece mayor flexibilidad al modelar sistemas que exhiben efectos de larga duración, lo que ha sido especialmente útil en campos como la física, control de sistemas y, más recientemente, en el aprendizaje automático.

En particular, han surgido recientemente enfoques basados en el gradiente descendente fraccionario para entrenar modelos de aprendizaje automático [83]. Estos métodos aprovechan las propiedades de memoria inherentes a las aproximaciones numéricas del cálculo fraccionario, lo que puede mejorar el proceso de aprendizaje. Basados en estas ventajas, se propone implementar un algoritmo de gradiente descendente de orden fraccionario [84] para entrenar la matriz de salida, como se muestra en la ecuación (4.3.1), con el objetivo de desarrollar una variante fraccionaria de la red LSTM-ESN, denominada LSTM-ESN FR. En esta ecuación, η_{α} es el tamaño de paso fraccionario y D^{α} denota la derivada fraccionaria de orden α.

$$W_{hN} = W_{hN} - \eta_\alpha D^\alpha g_i \tag{4.3.1}$$

El cálculo fraccionario se fundamenta en varias definiciones clave de diferenciación e integración fraccionaria, entre las cuales destacan las de Grünwald-Letnikov, Riemann-Liouville y Caputo [85, 86, 87]. En este trabajo, se ha decidido utilizar la definición de Grünwald-Letnikov, debido a su simplicidad numérica y el principio de *Memoria Corta*, que optimiza el costo computacional al limitar el número de pasos históricos utilizados en los cálculos.

La definición de Grünwald-Letnikov está dada por la ecuación (4.3.2), donde α es el orden fraccionario y h es el tamaño del paso.

$${}^{GL}D_t^\alpha f(t) = \lim_{h\to 0} \frac{1}{h^\alpha} \sum_{j=0}^{\infty} (-1)^j \begin{pmatrix} \alpha \\ j \end{pmatrix} f(t - jh) \tag{4.3.2}$$

La expresión (4.3.2) contiene coeficientes binomiales, los cuales pueden calcularse de forma recurrente mediante la ecuación (4.3.3). Estos coeficientes permiten reescribir la derivada fraccionaria en la forma más eficiente mostrada en la ecuación (4.3.4).

$$C_0^{(\alpha)} = 1 \qquad C_j^{(\alpha)} = (1 - \frac{1+\alpha}{j}) C_{j-1}^{(\alpha)} \tag{4.3.3}$$

$${}^{GL}D_t^\alpha f(t) = \frac{1}{h^\alpha} \sum_{j=0}^{\infty} C_j{}^{(\alpha)} f(t - jh) \tag{4.3.4}$$

Hasta este punto, se ha presentado la derivada de Grünwald-Letnikov en tiempo continuo, pero para utilizarla en un algoritmo de entrenamiento como el gradiente descendente fraccionario, es necesario convertirla a su versión discreta. El resultado

de esta discretización está dado en la ecuación (4.3.5) [88], que muestra cómo se aplica en el contexto de datos discretos.

$$ {}^{GL}D_t^{\alpha} f(t_k) = \frac{1}{h^{\alpha}}[f(t_{k-1}) + \sum_{j=1}^{k} C_j^{(\alpha)} f(t_{k-j})] \tag{4.3.5} $$

Como se mencionó anteriormente, una de las propiedades más importantes del cálculo fraccionario es el *principio de memoria*, que puede ser de dos tipos: memoria infinita y memoria corta. En la memoria infinita, todos los valores previos influyen en el cálculo del valor actual, lo que puede ser computacionalmente costoso. Por otro lado, la memoria corta utiliza solo una cantidad finita de valores históricos, lo que reduce significativamente el costo computacional sin sacrificar en gran medida la precisión.

El método de *memoria corta* de Grünwald-Letnikov se muestra en la ecuación (4.3.6), donde primero se define la longitud de la memoria L_m. Luego, el valor de v se calcula de acuerdo con la fórmula: $v = 1$ si $k < (\frac{L_m}{h})$ o $v = k - \frac{L_m}{h}$ si $k \geq (\frac{L_m}{h})$ [89]. Este enfoque es ideal para aplicaciones en aprendizaje automático, donde se busca un equilibrio entre rendimiento y eficiencia computacional.

$$ {}^{GL}D_t^{\alpha} f(t_k) = \frac{1}{h^{\alpha}}[f(t_{k-1}) + \sum_{j=v}^{k} C_j^{(\alpha)} f(t_{k-j})] \tag{4.3.6} $$

4.4. Complejidad Espacial

La complejidad espacial de las redes LSTM está dada por $O(12C)$, donde C es la cantidad de celdas de la red. Este valor refleja que el entrenamiento de una red LSTM tradicional implica ajustar 12 matrices de pesos por cada celda, lo que genera un aumento significativo en el costo computacional conforme crece el número de celdas.

Por otro lado, la complejidad espacial del modelo híbrido LSTM-ESN propuesto es constante, $O(1)$, ya que, independientemente del número de celdas, solo se entrena la matriz de salida.

En el modelo LSTM-ESN, las matrices internas no requieren entrenamiento, ya que se generan de forma aleatoria y se reescalan según los principios de las redes de estado eco (ESN). Esto representa una ventaja significativa en términos de eficiencia computacional, especialmente en aplicaciones donde el número de celdas aumenta considerablemente.

La figura 4.5 ilustra cómo, a medida que se incrementa la cantidad de celdas en una red LSTM, el número de parámetros entrenables y, por tanto, el costo computacional crece de forma proporcional. En contraste, en el modelo LSTM-ESN, el costo se mantiene constante.

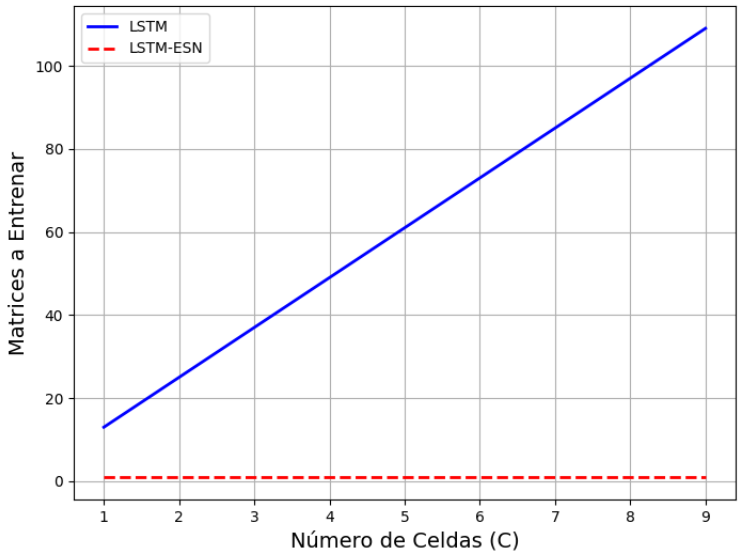


Figura 4.5: Complejidad espacial de una red LSTM vs el modelo híbrido LSTM-ESN.

Capítulo 5

Resultados Experimentales

En este capítulo se presentan los resultados obtenidos en la predicción de series temporales caóticas, tanto con como sin forzamiento del maestro. En la primera parte, se exponen los resultados de las diferentes estrategias utilizadas para aumentar el horizonte de predicción sin forzamiento del maestro. Gracias a la decimación de los datos y a la optimización de los hiperparámetros de la ESN, se logró incrementar el horizonte de predicción de la serie temporal del sistema de Lorenz en comparación con el estado del arte[1]. Posteriormente, se presentan los diagramas diseñados para la implementación en FPGA de la ESN, donde se destacan los buenos resultados obtenidos en la predicción utilizando una topología escasamente conectada, como es la topología en anillo.

En la segunda parte, se aborda la predicción con forzamiento del maestro. La principal contribución destacable es que los modelos híbridos propuestos, LSTM-ESN y LSTM-ESN FR, muestran mejores resultados en la predicción de un paso adelante de la serie temporal del sistema de Lorenz y del Producto Interno Bruto de México, en comparación con una red LSTM tradicional. Sin embargo, en la predicción de

[1] Ver tabla 5.5.

múltiples pasos hacia adelante, el modelo propuesto no logra superar los resultados de una red LSTM convencional; no obstante, se aproxima a los resultados reportados en el estado del arte, pero con un menor costo computacional[2].

5.1. Predicción sin Forzamiento del Maestro

La selección de los hiperparámetros de una red neuronal es un proceso crucial que define la precisión de la tarea a realizar. Por esta razón, se utiliza el método de búsqueda de rejilla para seleccionar el conjunto de hiperparámetros de la ESN que permitan obtener el error más pequeño en la predicción de series temporales caóticas. La selección de estos valores cambia con cada conjunto de datos, sin embargo, este método dará una luz del rango de los hiperparámetros que brindan los mejores resultados.

La figura 5.1 presenta los resultados del método de la rejilla en la predicción de la serie temporal de Lorenz, los parámetros de búsqueda son el radio espectral, y la tasa de aprendizaje, la medida utilizada es el RMSE entre los datos objetivo y los datos predichos. Para este experimento se utilizaron 10000 datos de la serie temporal caótica para el entrenamiento, según Jaeger [61] el número de neuronas en el depósito recomendado debe oscilar entre 1000 y 5000 ($\frac{T}{10} \leq N \leq \frac{T}{2}$, siendo T la cantidad de datos utilizados), sin embargo, uno de nuestros objetivos es aumentar el horizonte de predicción con la menor cantidad de neuronas posibles, así que el método de la rejilla fue realizado con solo 100 neuronas en el depósito.

Para la predicción de las series temporales caóticas utilizando decimación y basados en la búsqueda en rejilla, se seleccionaron los siguientes hiperparámetros: una tasa de aprendizaje de 0.5 y un radio espectral de 0.41. El impacto de la decima-

[2] Ver tabla 5.16.

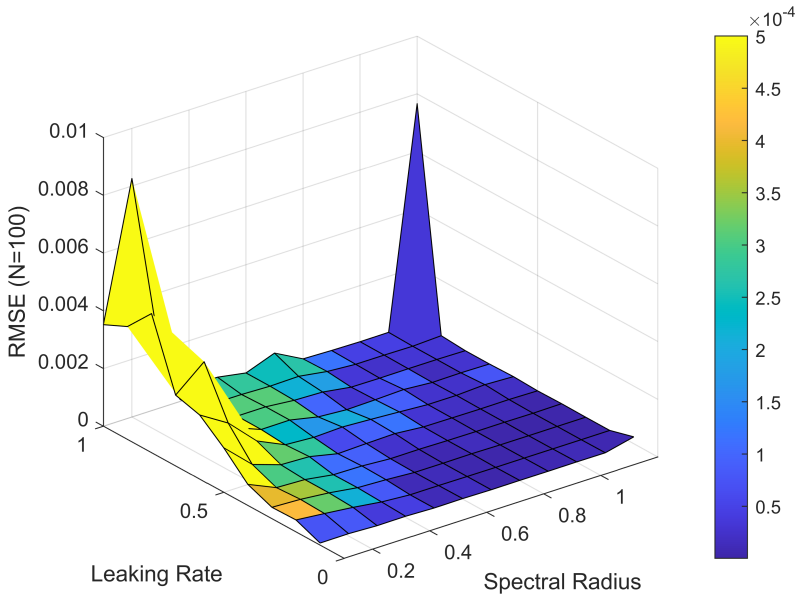


Figura 5.1: Búsqueda de rejilla para la selección de hiperparámetros de la ESN.

ción en las series temporales caóticas se muestra en las tablas 5.1 y 5.2, donde D representa el valor de la decimación. Los pasos totales predichos equivalen a los pasos predichos multiplicados por el factor de decimación, y la predicción se refiere a la longitud del conjunto de datos de prueba. En la tabla 5.1, se observa que una decimación de 50 de la serie temporal de Lorenz permite alcanzar un horizonte de predicción equivalente a 1500 pasos, en contraste con los 250 pasos que se pueden lograr sin decimación.

Tabla 5.1: Predicción de la serie temporal de Lorenz usando ESN para diferentes valores de decimación.

D	Pasos predichos	Pasos totales predichos	Predicción	RMSE
0	250	250	5000	$4,6020 \times 10^{-6}$
10	80	800	5040	$2,3270 \times 10^{-6}$
25	60	1500	5040	$6,4784 \times 10^{-6}$
50	30	1500	5010	$1,4494 \times 10^{-6}$

Por otro lado, en la tabla 5.2 se presentan los resultados de la predicción de la serie temporal de la neurona Hindmarsh-Rose (HR). A pesar de que la decimación

por 50 mantiene las propiedades de la serie temporal caótica, no necesariamente permite aumentar el horizonte de predicción en comparación con la serie temporal sin decimación. En este caso, la decimación por 25 es la que logra alcanzar el máximo horizonte de predicción.

Tabla 5.2: Predicción de la serie temporal de la neurona HR usando ESN para diferentes valores de decimación.

D	Pasos predichos	Pasos totales predichos	Predicción	RMSE
0	350	350	10150	$4,5872 \times 10^{-5}$
10	100	1000	10000	$4,1105 \times 10^{-7}$
25	45	1125	10035	$1,5395 \times 10^{-5}$
50	7	350	10000	$1,4543 \times 10^{-4}$

Para la predicción de series temporales caóticas utilizando ESN, las matrices $\mathbf{W_{in}}$ y $\mathbf{W}$ se generan de manera aleatoria. Por esta razón, se propone optimizar alguna de ellas para aumentar el horizonte de predicción. La matriz de pesos $\mathbf{W}$ del depósito debe ser re-escalada a una matriz esparcida con un radio espectral menor o igual a 1 para mantener la propiedad de estado de eco [59]. Esta condición complica la optimización de la matriz $\mathbf{W}$; por lo tanto, se decidió optimizar la matriz $\mathbf{W_{in}}$.

La población inicial del algoritmo PSO corresponde a los valores de la matriz $\mathbf{W_{in}}$ utilizada en los resultados previos para aumentar la predicción. La función de aptitud es el RMSE entre los valores objetivo (y_{target}) y los valores predichos (y_i) de la serie temporal caótica. Este procedimiento se realiza tanto para la serie temporal del sistema de Lorenz y de la neurona HR.

A partir del proceso de optimización de la ESN, seleccionamos los dos casos más optimistas de la tabla 5.1, que corresponden a la serie temporal caótica con decimación de 25 y 50. El algoritmo PSO se ejecuta con una población de 30 partículas y 10 generaciones, y se realizo una optimización escalada. Por ejemplo, cuando $D = 25$, la optimización se realiza para 100 pasos adelante, y la matriz $\mathbf{W_{in}}$ de la población inicial se utiliza para la siguiente optimización de 150 pasos adelante, y así sucesiva-

mente, hasta que sea imposible aumentar los pasos adelante manteniendo un error pequeño. La Tabla 5.3 muestra los resultados de diferentes ejecuciones de optimización para dos decimaciones, 25 y 50. La Figura 5.2 muestra la predicción para $D = 50$, ya que este es el caso en el que se obtiene el mayor horizonte de predicción con un error pequeño.

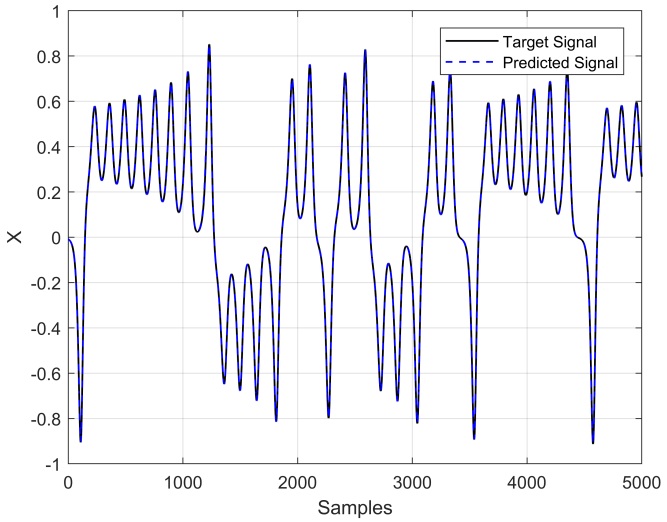
Tabla 5.3: Predicción de la serie temporal de Lorenz utilizando una red ESN optimizada.

Ejecución	**D**	**Pasos Adelante**	**Pasos Predichos Totales**	**Pasos Predichos**	**RMSE**
1	25	250	6250	5000	3.3315×10^{-5}
2					7.7769×10^{-6}
3					6.1854×10^{-5}
4					1.9145×10^{-5}
5					1.2868×10^{-5}
6					4.6597×10^{-5}
1	50	200	10000	5000	0.5899×10^{-5}
2					8.8973×10^{-6}
3					3.6936×10^{-5}
4					5.1623×10^{-5}
5					1.3510×10^{-5}
6					7.2846×10^{-5}

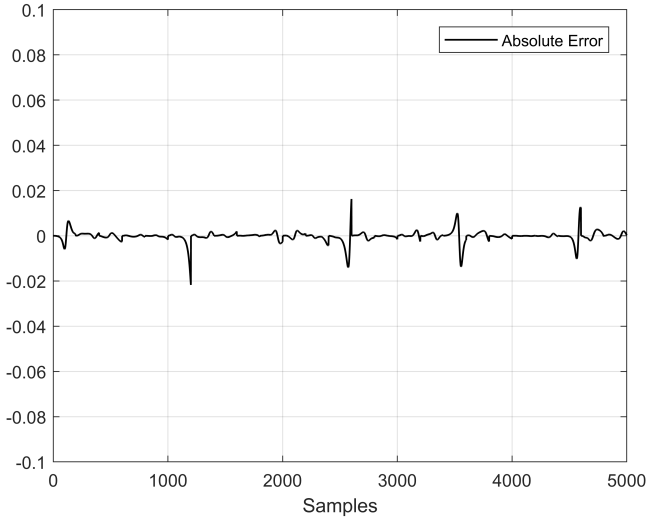
Para la neurona HR, se realizo el mismo procedimiento de optimización escalada y se seleccionaron los casos cuando $D = 10$ y $D = 25$, los cuales obtuvieron los mejores resultados según la tabla 5.2. La tabla 5.4 muestra los resultados de la optimización.

A partir de la tabla 5.4, se puede concluir que no hay una gran diferencia en la cantidad total de pasos adelante predichos, y tampoco existe una diferencia significativa en cuanto al error. Sin embargo, cuando $D = 25$, se puede observar una mayor dinámica en el gráfico a pesar de tener una ventana de 5000 pasos como se puede observar en la figura 5.3.

Hasta este punto, se ha mantenido el enfoque en dos objetivos principales: el primero se centró en cómo aumentar el horizonte de predicción de series temporales caóticas, y el segundo fue mantener el depósito con un tamaño pequeño. La tabla



(a)



(b)

Figura 5.2: Predicción de la serie temporal del sistema de Lorenz utilizando una ESN optimizada y decimación de 50. (a): Serie temporal predicha y serie original, (b): Error.

5.5 muestra los resultados obtenidos de la predicción utilizando la ESN optimizada con casos de decimación para las series temporales de Lorenz y de la neurona HR, y estos se comparan con trabajos similares. Como se puede apreciar, nuestros resultados de predicción son bastante competitivos. Por ejemplo, para el sistema de Lorenz sin decimación, la predicción fue de 250 pasos adelante, un resultado similar

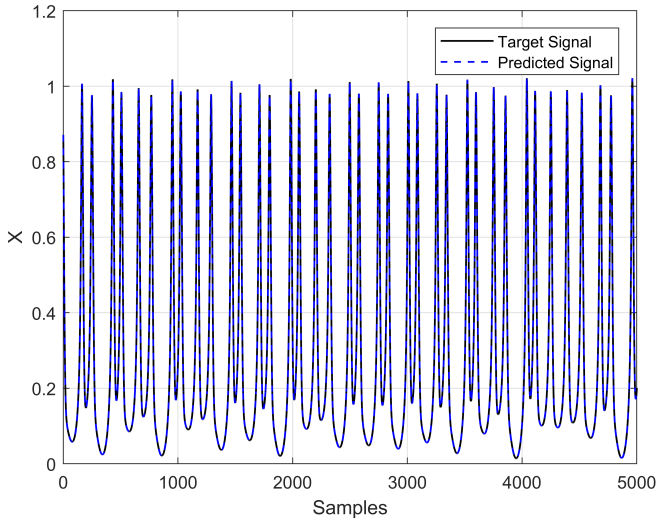
Tabla 5.4: Predicción de la serie temporal de la neurona HR utilizando una red ESN optimizada.

Ejecución	**D**	**Pasos Adelante**	**Pasos Predichos Totales**	**Pasos Predichos**	**RMSE**
1	10	700	7000	5600	2.5441×10^{-6}
2					1.3352×10^{-5}
3					6.2291×10^{-6}
4					1.2683×10^{-5}
5					1.5212×10^{-5}
6					3.2412×10^{-6}
1	25	300	7500	5100	3.4141×10^{-6}
2					3.8000×10^{-6}
3					6.8645×10^{-7}
4					1.0267×10^{-6}
5					1.8325×10^{-6}
6					1.3844×10^{-5}

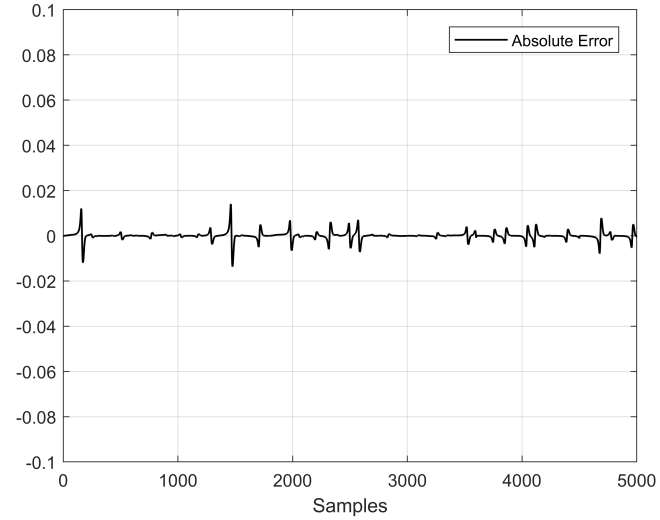
al reportado en [28], donde los autores utilizan tres veces el número de neuronas en el depósito. En [6], los autores predicen 460 pasos utilizando 5000 neuronas en el depósito, una diferencia significativa con respecto a nuestro trabajo, en el cual, en todos los casos, la ESN consta de solo 100 neuronas en el depósito. Por otro lado, podemos destacar cómo el proceso de decimación incrementa el número de pasos adelante predichos; solo con la decimación de 10 mejoramos los resultados presentados en [39] y la decimación por 50 con optimización mejora significativamente el horizonte de predicción con respecto a los demás trabajos.

Tabla 5.5: Resultados de predicción utilizando ESN optimizado sin/con decimación (D) y comparación con trabajos relacionados.

Método	Lorenz				NHR			
	Pasos Predichos	RMSE	Neuronas	W_{elem}	Pasos Predichos	RMSE	Neuronas	W_{elem}
ESN	250	4.6020×10^{-6}	100	1×10^4	350	4.5872×10^{-5}	100	1×10^4
ESN + D10	800	2.3270×10^{-6}	100	1×10^4	1000	4.1105×10^{-7}	100	1×10^4
ESN + D25	1500	6.4784×10^{-6}	100	1×10^4	1125	1.5395×10^{-5}	100	1×10^4
ESN + D50	1500	1.4494×10^{-6}	100	1×10^4	350	1.4543×10^{-4}	100	1×10^4
ESN + PSO + D10	—	—	—	—	7000	2.5441×10^{-6}	100	1×10^4
ESN + PSO + D25	6250	3.3315×10^{-6}	100	1×10^4	7500	3.4141×10^{-6}	100	1×10^4
ESN + PSO + D50	10000	5.8987×10^{-6}	100	1×10^4	—	—	—	—
ESN [28]	300	—	300	9×10^4	—	—	—	—
ESN [6]	460	—	5000	5×10^6	—	—	—	—
RNN-LSTM [6]	180	—	5000	5×10^6	—	—	—	—
ANN [6]	120	—	5000	5×10^6	—	—	—	—
ESN [39]	700	—	300	9×10^4	—	—	—	—
RESN [23]	500	0.2238	200	4×10^4	—	—	—	—



(a)



(b)

Figura 5.3: Predicción de la serie temporal de la neurona HR utilizando una ESN optimizada y decimación de 25. (a): Serie temporal predicha y serie original, (b): Error.

Otro resultado interesante es que la optimización de la matriz de entrada $\mathbf{W_{in}}$ ayuda a aumentar en más de un orden de magnitud el horizonte de predicción en el mejor de los casos. La predicción de series temporales caóticas de algunos comportamientos neuronales, como la neurona HR, no se ha realizado antes[3]. Este tipo de

[3]Hasta la fecha de la publicación del articulo en 2022.

neurona tiene un comportamiento lento que puede complicar la tarea de predicción en ventanas de tiempo prolongadas. También se pueden utilizar otros modelos de neuronas, como las Redes Neuronales Celulares [90], la Neurona de Hopfield [91], la Neurona de Huber Braun [92], así como otros sistemas relacionados con la modelización de la salud [4].

5.1.1. Implementación en FPGA

Las topologías mencionadas en la sección 3.2.1, en la pagina 31, se utilizan para realizar la predicción de la serie temporal del sistema de Lorenz, con el fin de determinar cómo se comporta el error al realizar la predicción con cada una de ellas.

Para la primera topología, se va a variar el valor de k, que se refiere al número de entradas de cada neurona. La tabla 5.6 muestra los resultados de la predicción de 20 pasos adelante de la serie temporal del sistema de Lorenz. Se utilizaron 10000 datos de series temporales para el entrenamiento, 1000 datos para la prueba, y 50 neuronas (aunque según las recomendaciones generales el número de neuronas debe estar entre $(\frac{T}{10} \leq N \leq \frac{T}{2})$, usamos una cantidad menor para poder realizar una implementación en FPGA). La tasa de aprendizaje se establece en 0.5, el coeficiente de regularización en 1×10^{-5}, el sesgo de salida es 1, las matrices $\mathbf{W}$ y $\mathbf{W_{in}}$ se generaron aleatoriamente como es la recomendación general, y la matriz $\mathbf{W}$ se escala de acuerdo con la ecuación (3.1.2) para cada uno de los valores del radio espectral (ρ_r) utilizados.

Tabla 5.6: Predicción de la serie temporal del sistema de Lorenz con la Topología 1.

k	$\rho_r = 0.5$		$\rho_r = 0.75$		$\rho_r = 1$	
	MSE	R^2	MSE	R^2	MSE	R^2
10	1.7808×10^{-5}	0.9999	$4,5818 \times 10^{-6}$	1.0000	8.3589×10^{-6}	0.9999
20	1.2654×10^{-5}	0.9999	$6,1909 \times 10^{-6}$	1.0000	1.9669×10^{-5}	0.9999
30	1.0133×10^{-5}	0.9999	4.0856×10^{-6}	1.0000	2.2327×10^{-6}	1.0000
40	1.5284×10^{-5}	0.9999	2.0224×10^{-6}	1.0000	1.2226×10^{-5}	0.9999
50	1.4309×10^{-5}	0.9999	2.0067×10^{-6}	1.0000	5.9343×10^{-6}	1.0000

En la topología 1, cuando $k = 50$ la matriz $\mathbf{W}$ esta completamente llena, se podría intuir que entre mayor sea la cantidad de valores no cero de la matriz de conexiones $\mathbf{W}$, se debe obtener un menor MSE, sin embargo, no es directamente proporcional, y esto es por la naturaleza aleatoria de las conexiones y los pesos. La tabla 5.7 muestra los resultados de las predicciones de la serie temporal del sistema de Lorenz realizadas con las topologías 2, 3 y 4, en este caso solamente cambia el radio espectral, ya que la cantidad de conexiones en la matriz $\mathbf{W}$ es constante en cada topología.

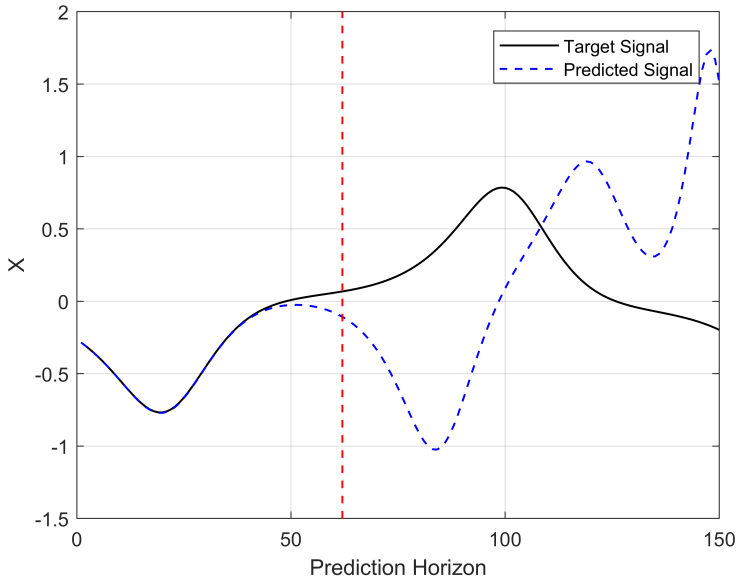
Tabla 5.7: Predicción de la serie temporal del sistema de Lorenz con las topologías 2, 3 y 4.

Topología	$\rho_r = 0.5$		$\rho_r = 0.75$		$\rho_r = 1$	
	MSE	R^2	MSE	R^2	MSE	R^2
2	8.1602×10^{-6}	0.9999	7.0037×10^{-6}	0.9999	9.8323×10^{-6}	0.9999
3	3.1532×10^{-6}	1.0000	4.1465×10^{-6}	1.0000	2.7020×10^{-6}	1.0000
4	1.6635×10^{-5}	0.9999	9.3734×10^{-6}	0.9999	3.4963×10^{-6}	1.0000

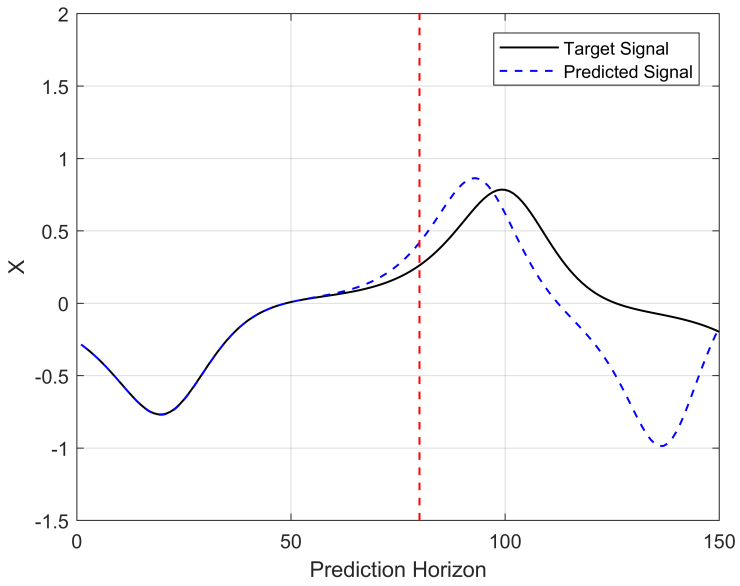
La tabla 5.6 muestra que el menor error MSE obtenido con la topología 1 es de 2.0067×10^{-6} con un valor de $k = 50$, según la tabla 3.3, esto corresponde a 2500 conexiones en la matriz $\mathbf{W}$. En la tabla 5.7, el menor error MSE es 2.7020×10^{-6} y obtenido con la topología 3, según la tabla 3.3, esta topología solo tiene 50 conexiones en la matriz $\mathbf{W}$. Es interesante resaltar que la diferencia de errores entre las dos topologías no es significativa, sin embargo, la topología 3 utiliza el 2 % de las conexiones de la topología 1. En general podemos ver que las topologías 2, 3 y 4 presentan resultados muy competitivos respecto a la topología 1 utilizando menos conexiones internas.

Luego de seleccionar las dos topologías con los mejores resultados, se busca el máximo horizonte de predicción que se puede obtener con cada una de ellas.

La figura 5.4, muestra el horizonte de predicción alcanzado en la predicción de la serie temporal de Lorenz con la topología 1 y la topología 3, 60 y 80 pasos respectivamente, esto implica que, aunque la topología 3 tiene menos conexiones internas



(a)



(b)

Figura 5.4: Máximo horizonte de predicción obtenido con cada una de las topologías en la predicción de la serie temporal de Lorenz. (a): Topología 1. (b): Topología 3.

obtuvo un mayor horizonte de predicción.

Para la implementación en FPGA de una ESN para la predicción de la serie tem-

poral del sistema de Lorenz, primero, se debe diseñar un esquema general donde se representen las ecuaciones de la red (3.1.1) y (3.1.3) usando bloques que indican la interconexión entre las entradas y salidas de cada uno de ellos. La figura 5.5, muestra el esquema general de la ESN, compuesta por tres memorias ROM donde se almacenan las matrices $\mathbf{W}$, $\mathbf{W_{in}}$ y $\mathbf{W_{out}}$, dos memorias RAM donde se almacenan los estados internos que se deben actualizar cada que se hace una nueva predicción y en la tercera se almacenan los datos predichos. Previamente, se hizo enfoque en que las neuronas deben tener la misma cantidad de entradas, con el fin de poder reutilizar el bloque diseñado para una sola neurona N veces (siendo N el número de neuronas en el depósito). También es necesario diseñar una máquina de estados general que controla la lectura y escritura de memorias, otra máquina de estados encargada del flujo de procesamiento de la función tangente hiperbólica. Estas máquinas de estado se sincronizan para proporcionar una bandera en el momento que se ha predicho un nuevo dato, esto es necesario ya que se utilizan varios ciclos de reloj para realizar la predicción.

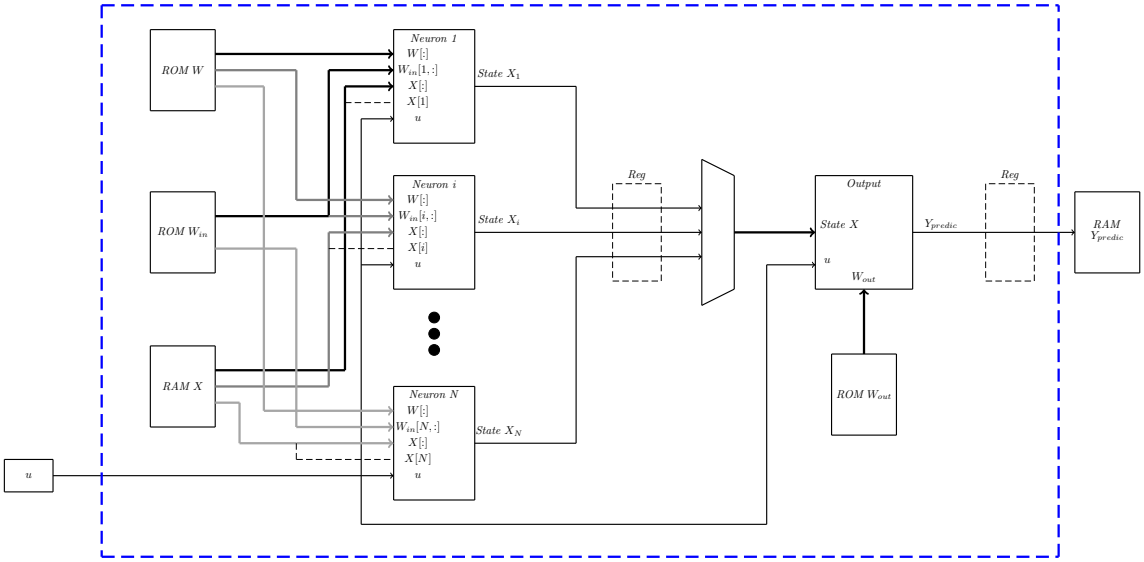


Figura 5.5: Diagrama de bloques para la implementación de la red ESN.

La figura 5.6 representa el diagrama de bloques de una neurona, el sub-bloque

Topology, es de gran importancia debido a que allí es donde se ve reflejada la importancia de las topologías propuestas en este trabajo. Otro bloque importante, es la tangente hiperbólica, ya que es todo un reto implementar esta función no lineal en FPGA debido a las aproximaciones que se deben utilizar y las implicaciones en recursos y error introducidos por esta. Los parámetros que se multiplican por la tasa de aprendizaje (λ), encerrados en cuadros punteados se cambian por un corrimiento a la derecha, este corrimiento es equivalente a realizar una multiplicación por 0.5 (el cual es el valor de la tasa de aprendizaje), lo que disminuye el costo computacional, al requerir menos recursos para realizar un corrimiento en comparación a una multiplicación de 9-bits.

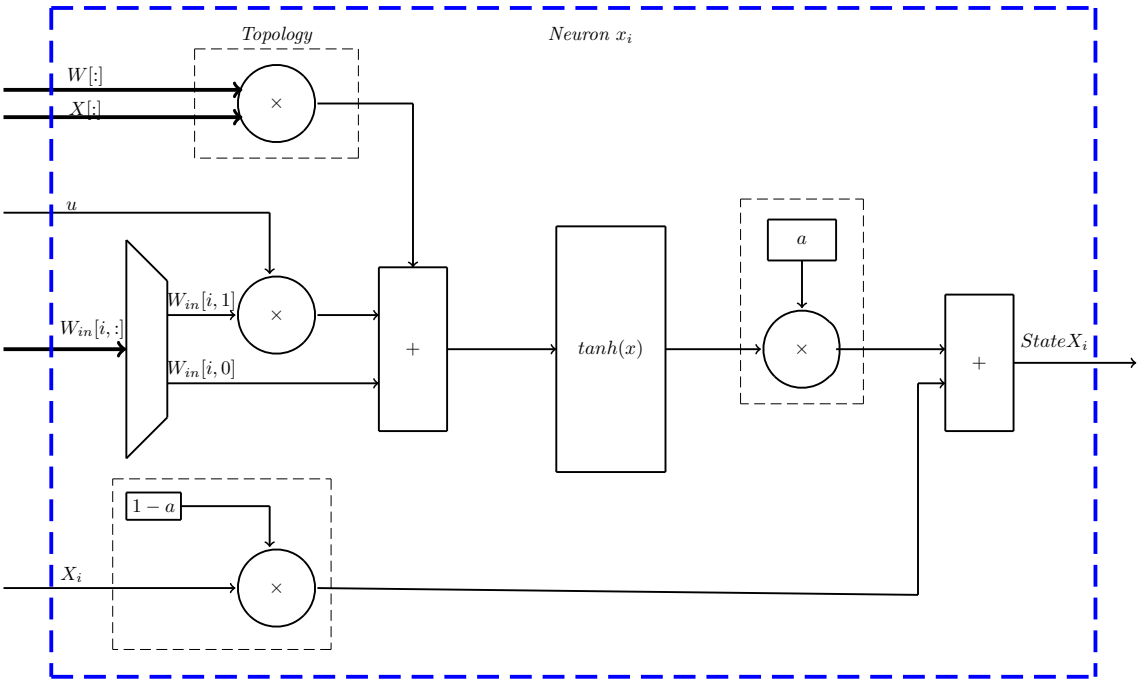


Figura 5.6: Diagrama de bloques para la implementación en FPGA de la Neurona i.

La topología 1 en su configuración completamente conectada (la más común) utiliza una cantidad de multiplicadores igual al número de neuronas como se muestra en la figura 5.7, a diferencia de la topología 3 (topología anillo) que solo utiliza un multiplicador correspondiente al valor no cero de la matriz **W** , sin importar la cantidad de neuronas que contenga el depósito. En este caso, para la topología 1,

cada neurona necesita 50 multiplicadores, a diferencia de la topología 3 que solo necesita un multiplicador, lo que permite reducir en más del 90 % la cantidad de multiplicadores en cada neurona.

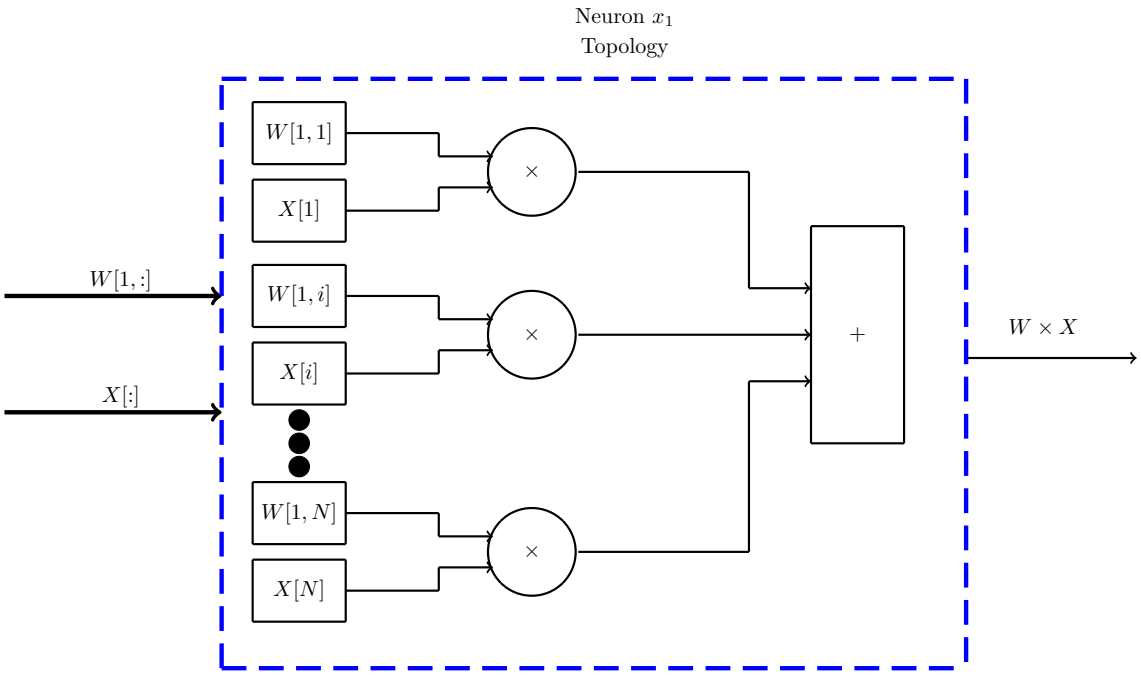


Figura 5.7: Diagrama de bloques de la Topología 1.

La figura 5.8, muestra el diagrama del bloque de salida, este bloque es relativamente sencillo, solo hay que garantizar que se hayan actualizado los estados de todas las neuronas antes de realizar las operaciones.

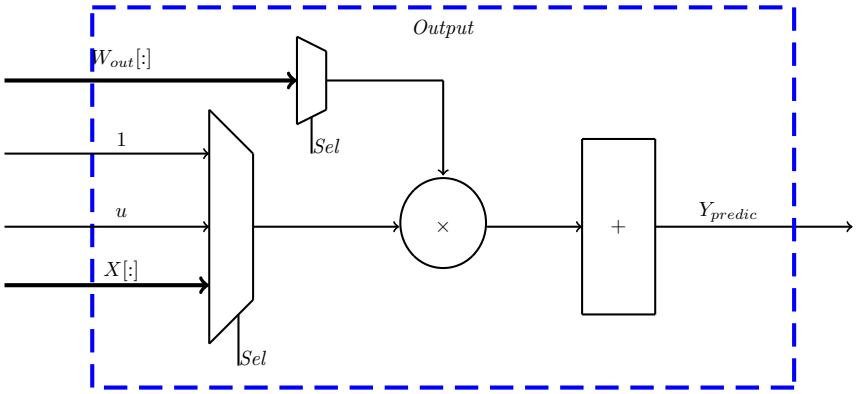


Figura 5.8: Diagrama para calcular la salida

De acuerdo a los resultados de las simulaciones realizadas de la función tanh, las aproximaciones PCHIP y Polinomio de grado 6 son los que presentan menor error de aproximación. Para la aproximación polinomial de grado 6 se va a implementar la ecuación (3.2.8), que como se ha mencionado requiere un mayor costo computacional; para la aproximación PCHIP se va a implementar la ecuación (3.2.5) reescrita como se muestra en la ecuación (5.1.1), donde $x = x - x_1$, con el objetivo de reducir los recursos de la implementación, la figura 5.9 muestra el diagrama propuesto para implementarla y está controlada por una máquina de estados, que a su vez es controlada por la máquina de estados general de la ESN, esto implica que aunque reducimos el costo computacional de la implementación se aumenta la latencia.

$$H(x) = x(x(ax + b) + c) + d \tag{5.1.1}$$

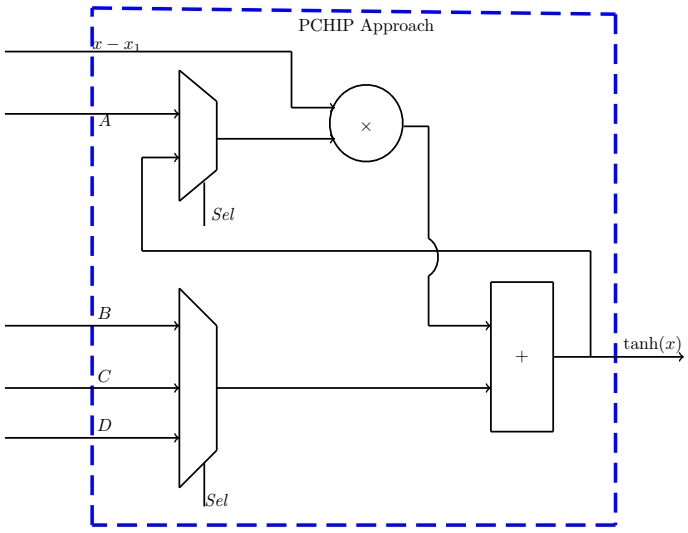


Figura 5.9: Diagrama de bloques de la aproximación PCHIP de la función tangente hiperbólica.

5.1.1.1. Recursos y Topologías

Como ya se mencionó, para la implementación de la función tanh se van a utilizar dos aproximaciones (un polinomio de grado 6 y PCHIP), la tabla 5.8 muestra los sumadores, multiplicadores y latencia de cada una de las aproximaciones, así como de las topologías 1 y 3, donde se puede destacar que la topología 3 utiliza solo el 2 % de los multiplicadores que la topología 1.

Tabla 5.8: Recursos utilizados por 1 neurona con topología 1 (K = 50) y topología 3.

Recursos	tanh (Polinomio G. 6)	tanh (PCHIP)	Topología 1	Topología 3
Multiplicadores	5	3	50	1
Sumadores	1	2	1	0
Latencia (ciclos de reloj)	2	3	2	1

Solo se va a implementar la topología 3, ya que la topología 1 implica un mayor costo computacional y no hay una diferencia significativa en cuanto el error y el horizonte de predicción entre estas dos topologías, pero si con respecto al costo computacional. Para la implementación en FPGA hay que definir el formato de punto fijo que se utilizará, para esto se analizó el rango dinámico de la serie temporal del sistema de Lorenz y de las operaciones realizadas por la ESN, y se eligió un tamaño de palabra de 32 bits con un formato de 6.26, 6 bits para la parte entera incluyendo el signo y 26 bits para la parte flotante, la síntesis se realizo en *Quartus II para la cyclone IV GX EP4CGX150DF31C7*. En la tabla 5.9 se muestran los recursos asignados por Quartus en la implementación de la topología 3 con las dos aproximaciones de la tangente mencionados previamente.

Para la implementación, se eligió la topología 3 debido a que consume menos recursos y ofrece mejores resultados. En cuanto a la aproximación de la tangente hiperbólica, se considerarán dos opciones: PCHIP y un polinomio de grado 6. La selección entre estas alternativas dependerá de los recursos disponibles y de la tole-

Tabla 5.9: Recursos utilizados por la topología 3 con N neuronas usando las aproximaciones PCHIP y Polinomio G. 6 para la función tanh.

Recursos	**Polinomio G. 6**	**PCHIP**
Multiplicadores	$6N + 1$	$2N + 1$
Sumadores	$3N + 1$	$3N + 1$
Total Elementos Lógicos	913395	120931
Total Registros	1655	6621
Elementos de multiplicador embebido de 9-bits	720	720
Latencia (Ciclos de Reloj)	8	9

rancia al error aceptable. La figura 5.10 presenta los resultados de la predicción de la serie temporal de Lorenz utilizando la topología 3 con la aproximación polinómica de grado 6 para la función tangente hiperbólica. Por su parte, la figura 5.11 muestra los resultados de la predicción empleando la aproximación PCHIP. En ambos casos, se realizó una predicción de 5 pasos hacia adelante, obteniendo un total de 400 puntos de prueba.

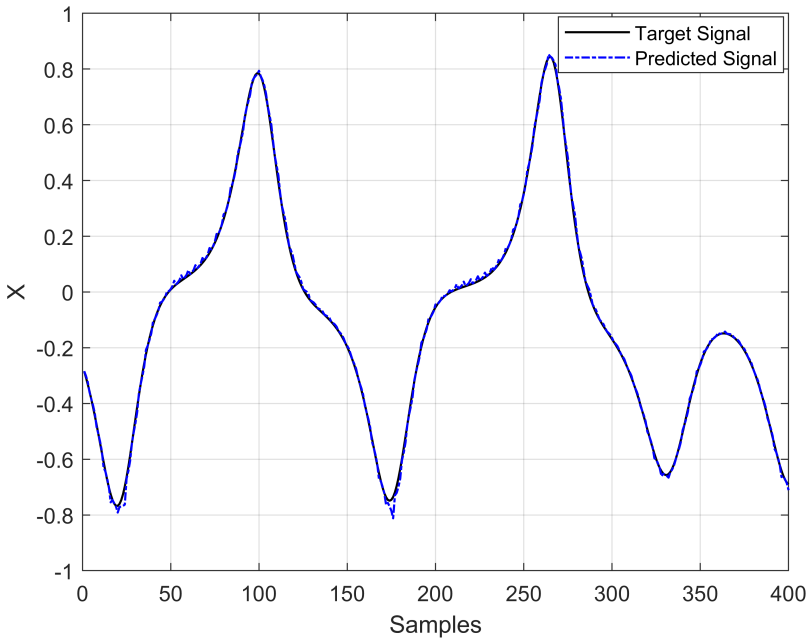


Figura 5.10: Predicción de 5 pasos adelante de la serie temporal de Lorenz con la topología 3 y la aproximación polinómica de grado 6 para la función tanh.

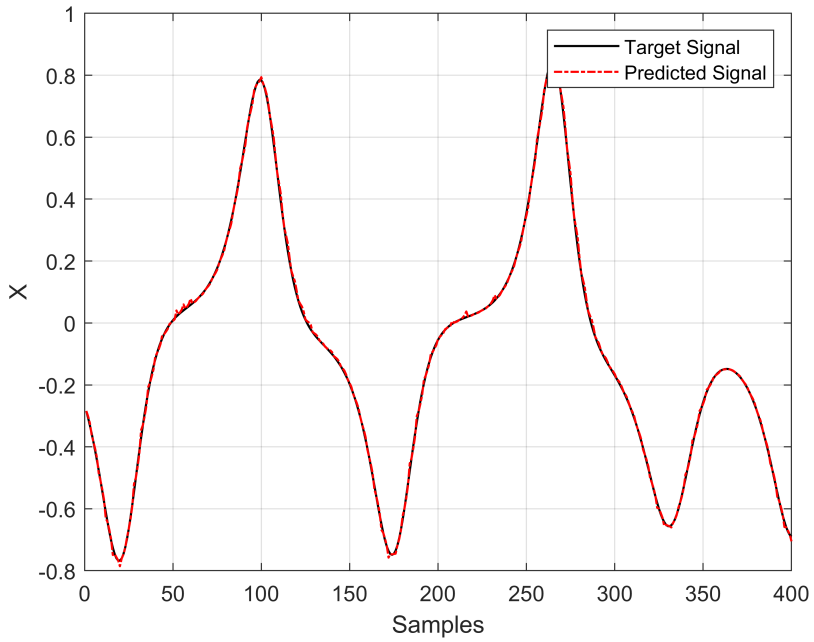


Figura 5.11: Predicción de 5 pasos adelante de la serie temporal de Lorenz con la topología 3 y la aproximación PCHIP para la función tanh.

Con la aproximación PCHIP se obtuvo un error MSE de 6.3054×10^{-5} en 400 puntos de prueba, mientras que con la aproximación polinómica el error MSE fue de 1.2222×10^{-4}. Esto concuerda con los resultados de la simulación, donde la aproximación de la tangente hiperbólica que tuvo menos error también presentó menor error en la predicción.

5.2. Predicción con Forzamiento del Maestro

La red LSTM se simula utilizando la biblioteca Keras de Python, mientras que para las redes LSTM-ESN y LSTM-ESN FR se implementan las ecuaciones de cada una de sus celdas en Python. El experimento inicial tiene como objetivo predecir un paso hacia adelante de las series temporales de los sistemas de Lorenz y Chen utilizando una única celda. Esta configuración representa el mínimo necesario para estos

métodos de aprendizaje automático. Según las ecuaciones descritas en la sección 4.1, los cálculos indican que, para una celda de una red LSTM, se deben entrenar 12 matrices más la matriz de la capa de salida. En contraste, las redes LSTM-ESN y LSTM-ESN FR requieren entrenar únicamente la matriz de salida, independientemente del número de celdas utilizadas. La configuración de la ventana deslizante es de 1×1, lo que implica que se dispone de un dato de entrada y un dato de salida (o predicción). La división de los datos de las series temporales consiste en 1000 datos para entrenamiento y 1000 datos para test. En el caso de la red LSTM-ESN FR, se estableció una longitud de memoria corta del 5 % y una derivada de orden fraccionario de 0.6. Los resultados de la predicción de las series temporales de los sistemas de Lorenz y Chen se presentan en las figuras 5.12 y 5.13, respectivamente.

En las figuras 5.12 y 5.13, se resalta la mejora en la predicción de las series temporales con los modelos propuestos LSTM-ESN y LSTM-ESN FR, en comparación con el método tradicional LSTM. Las figuras 5.14 y 5.15 muestran los datos predichos frente a los datos objetivos, representados en color azul. La línea diagonal negra representa el resultado ideal, lo que corrobora los resultados anteriores, evidenciando que los modelos propuestos logran mejores resultados en la predicción de las series temporales.

Los resultados cuantitativos (MSE y R^2) de la predicción de las series temporales de los sistemas de Lorenz y Chen con una red LSTM y los modelos propuestos LSTM-ESN y LSTM-ESN FR son mostrados en la tabla 5.10.

Tabla 5.10: Error en la predicción de 1 paso adelante de las series temporales de los sistemas de Lorenz y Chen usando los modelos LSTM, LSTM-ESN y LSTM-ESN FR con una sola celda.

Serie	**LSTM**		**LSTM-ESN DG**		**LSTM-ESN FR**	
Temporal	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$
Lorenz	0.0515	0.6554	0.0277	0.8142	0.0103	0.9307
Chen	0.0163	0.8566	0.0044	0.9607	0.0036	0.9682

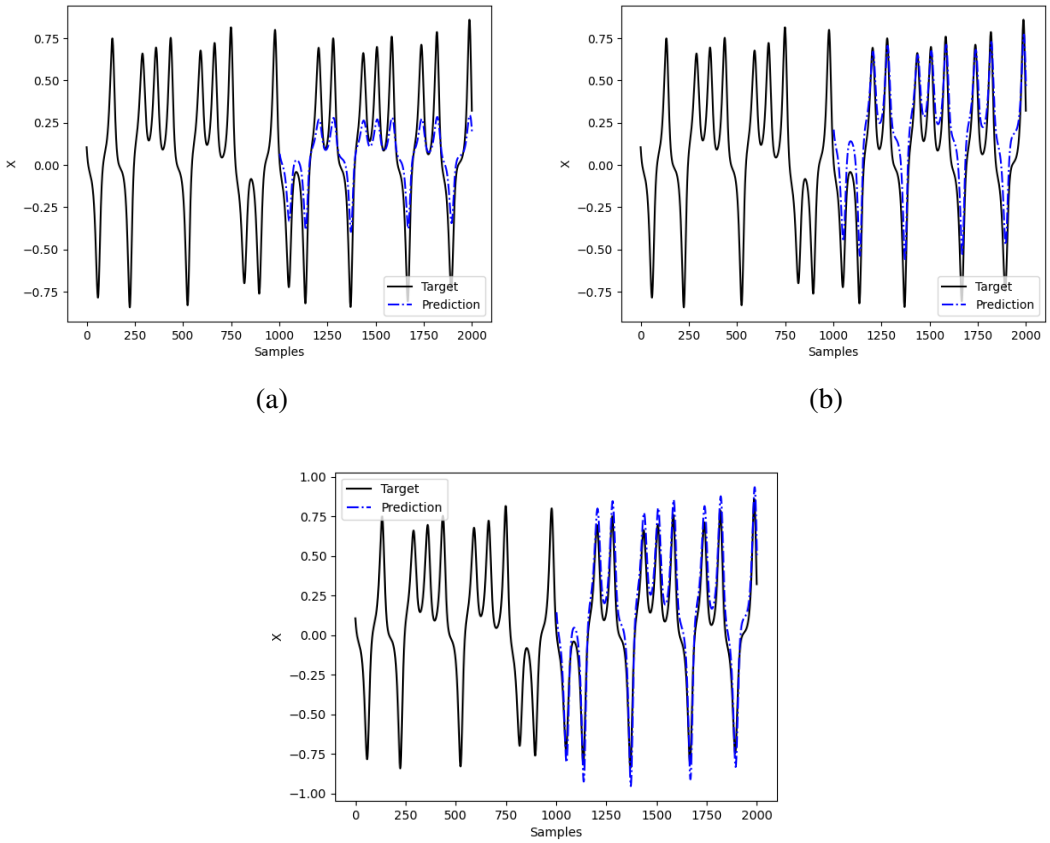


Figura 5.12: Predicción de 1 paso adelante de la serie temporal del sistema de Lorenz. (a): LSTM, (b): LSTM-ESN. (c): LSTM-ESN FR.

Con el objetivo de expandir el alcance de los experimentos, se decidió aumentar el número de celdas utilizadas y el tamaño de la ventana deslizante. Específicamente, se definió un tamaño de 20 celdas para todos los modelos empleados. Es importante destacar que la cantidad de matrices que requieren entrenamiento en una red LSTM depende del número de celdas utilizadas, lo que resulta en una formulación de $12 \times n_{\text{cells}} + 1 = 241$ matrices en este caso. En contraste, los modelos propuestos LSTM-ESN y LSTM-ESN FR solo requieren el entrenamiento de la matriz de salida, sin depender del número de celdas. Esto representa una ventaja significativa en términos de costo computacional durante la etapa de entrenamiento.

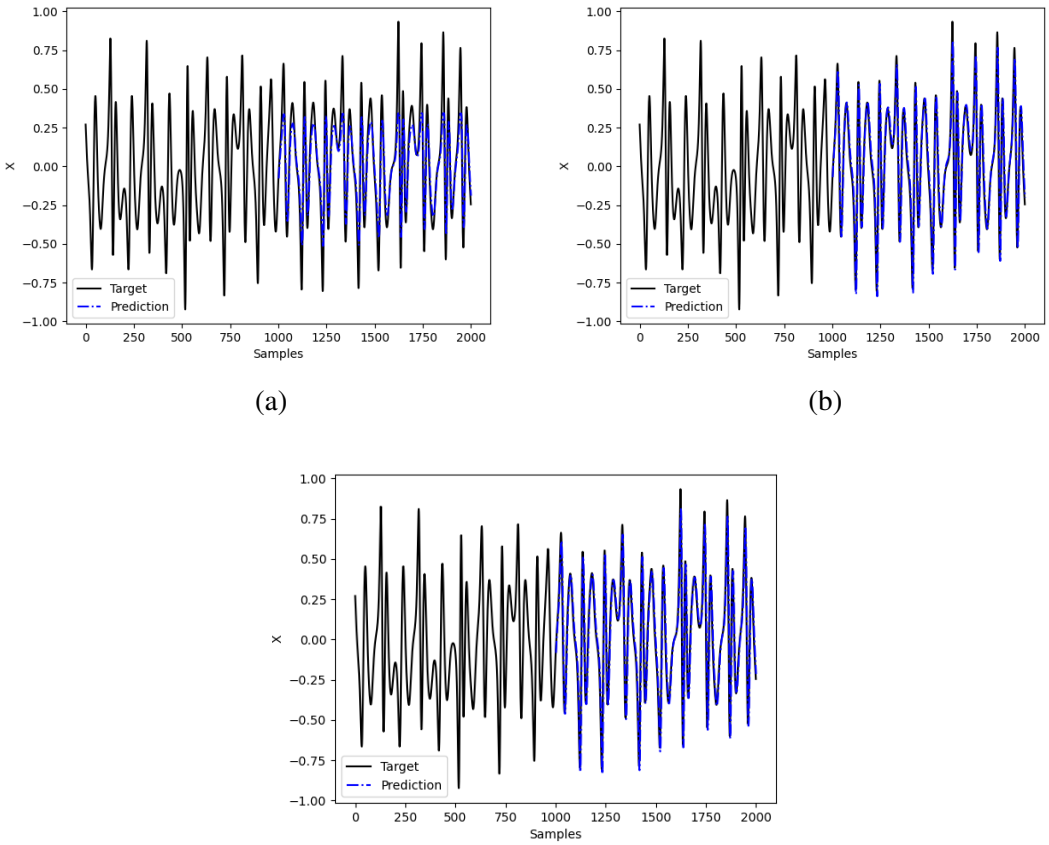
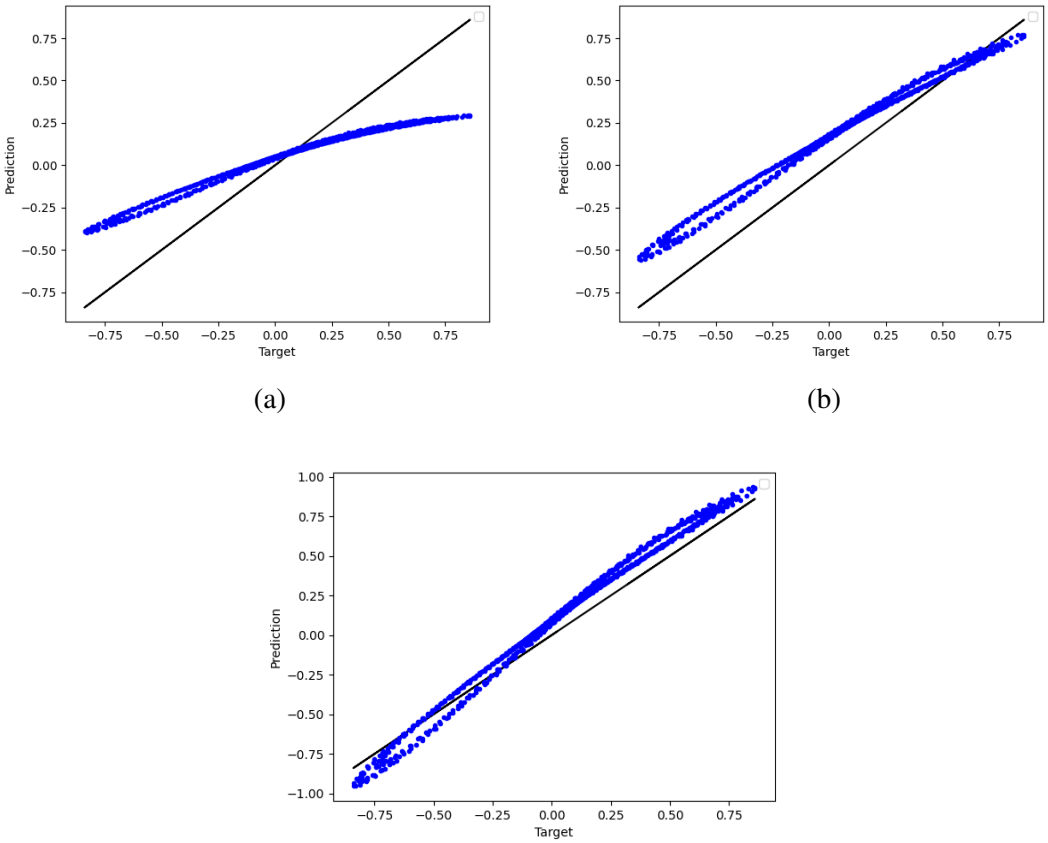


Figura 5.13: Predicción de 1 paso adelante de la serie temporal del sistema de Chen. (a): LSTM, (b): LSTM-ESN. (c): LSTM-ESN FR.

Las tablas 5.11 y 5.12 presentan los resultados obtenidos de la predicción de un paso adelante de las series temporales de Lorenz y Chen, respectivamente. Para estos experimentos, se utilizaron 1000 datos para el entrenamiento, 500 datos para las pruebas y 20 celdas para la red LSTM, y los modelos propuestos LSTM-ESN (con dos métodos diferentes para el ajuste de la matriz de salida, gradiente descendiente y ADAM) y LSTM-ESN FR (los parámetros del gradiente descendiente son los mismos utilizados en el experimento de una sola celda).

Como se mencionó previamente, existe una gran diferencia entre la cantidad de parámetros entrenables de una red LSTM y los modelos propuestos LSTM-ESN.



(a) (b)

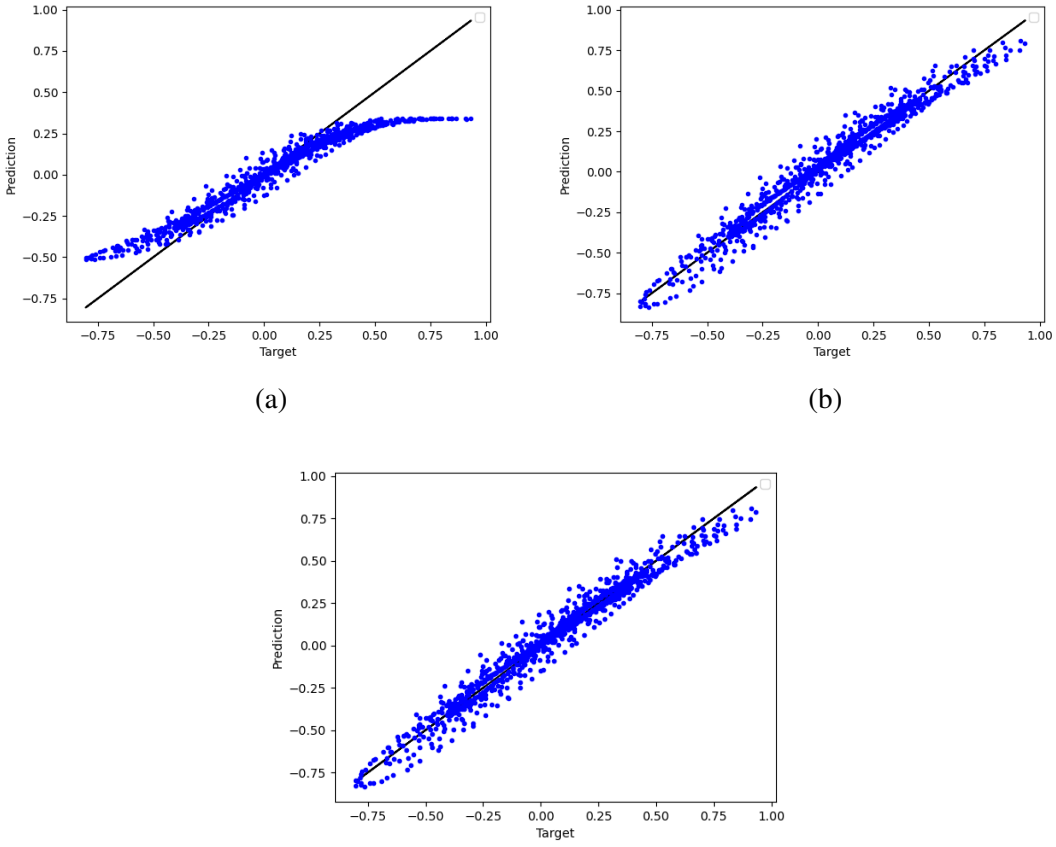
(c)

Figura 5.14: Datos objetivo vs Predicción de 1 paso adelante de la serie temporal del sistema de Lorenz (a): LSTM, (b): LSTM-ESN. (c): LSTM-ESN FR.

Tabla 5.11: Error en la predicción de un paso adelante de la serie temporal de Lorenz usando los modelos LSTM, LSTM-ESN y LSTM-ESN FR.

Ventana	**LSTM**		**LSTM-ESN DG**		**LSTM-ESN ADAM**		**LSTM-ESN FR**	
Deslizante	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$
1	0.0707	0.5235	0.0362	0.7555	0.0283	0.8090	0.0023	0.9846
2	0.0018	0.9874	0.0123	0.9169	0.0036	0.9754	0.0026	0.9823
4	0.0040	0.9728	0.0068	0.9539	0.0102	0.9315	0.0063	0.9581
10	0.0027	0.9821	0.0229	0.8478	0.0212	0.8596	0.0047	0.9689

Específicamente, los modelos propuestos reducen el número de matrices entrenadas en mas del 90 %, sin embargo, después de revisar los resultados presentados en las tablas 5.11 y 5.12, es notable que la reducción en los parámetros entrenables no nece-



(a) (b)

(c)

Figura 5.15: Datos objetivo vs Predicción de la serie temporal del sistema de Chen. 5.15a: LSTM, 5.15b: LSTM-ESN. 5.15c: LSTM-ESN FR.

Tabla 5.12: Error en la predicción de 1 paso adelante de la serie temporal de Chen usando los modelos LSTM, LSTM-ESN y LSTM-ESN FR.

Ventana	LSTM		LSTM-ESN DG		LSTM-ESN ADAM		LSTM-ESN FR	
Deslizante	MSE	R^2	MSE	R^2	MSE	R^2	MSE	R^2
1	0.0282	0.7463	0.0048	0.9567	0.0117	0.8945	0.0049	0.9556
2	0.0083	0.9255	0.0122	0.8904	0.0208	0.8133	0.0072	0.9364
4	0.0128	0.8851	0.0249	0.7778	0.0308	0.7251	0.0010	0.9907
10	0.0090	0.9203	0.0085	0.9246	0.0311	0.7258	0.0028	0.9752

sariamente implica un mayor error en la predicción de las series temporales. Por otro lado, la red LSTM presenta mejores resultados a medida que incrementa el tamaño de la ventana deslizante, a diferencia de los modelos propuestos, los cuales presentan mejores resultados con tamaños de ventana más pequeños. Esta característica constituye una ventaja para los modelos propuestos, ya que el tamaño de la ventana corresponde directamente al número de entradas de la red, lo que en consecuencia conlleva a reducir el costo computacional. También podemos destacar de estos experimentos que no hay una diferencia significativa entre los diferentes métodos de entrenamiento de la matriz de salida del modelo LSTM-ESN.

En la tabla 5.13, se presenta la comparación de los resultados obtenidos en la predicción de un paso hacia adelante con otros trabajos reportados en la literatura. A pesar de que las propuestas de esta tesis no logran mejorar el error en la predicción, se mantienen bastante cerca de los resultados obtenidos por otros autores. Además, nuestras propuestas utilizan una cantidad menor de celdas para la predicción, específicamente un 50 % menos de celdas en comparación con lo reportado en [93].

Si bien los modelos propuestos de LSTM-ESN se destacan por su buen desempeño con un tamaño de ventana pequeña, surge una implicación notable cuando se busca aumentar la cantidad de pasos predichos. La necesidad de expandir el tamaño de la ventana podría comprometer potencialmente la precisión de la predicción. Para

Tabla 5.13: Comparación de la predicción de 1 paso adelante utilizando diferentes métodos de aprendizaje automático.

Método	Sistema Caótico	Pasos Predichos	Neuronas/Celdas	MSE
AESN [54]	Lorenz		500	6×10^{-3}
AESN [54]	Rössler		500	1.65×10^{-2}
ALM-ESN [44]	Lorenz		200	2.21×10^{-5}
IPBO-FNN [94]	Lorenz	500		4.5×10^{-2}
MCSA-WNN [95]	Lorenz	500		8.20×10^{-3}
LSTM [93]	Lorenz	800	40	2.22×10^{-2}
LSTM-ESN FR	Lorenz	500	20	4.79×10^{-2}
LSTM-ESN FR	Chen	500	20	3.16×10^{-2}

explorar esta hipótesis, se realizan experimentos que involucran un aumento en la cantidad de pasos predichos (h-pasos adelante) y la ventana deslizante. Sin embargo, es un desafío determinar el tamaño de ventana óptimo para la predicción de h-pasos adelante. Para abordar este problema, se plantea realizar experimentos con diferentes tamaños de ventana (2h, 4h, 8h y 10h) para predecir h-pasos adelante. Las tablas 5.14 y 5.15 presentan los resultados de estas predicciones para los sistemas Lorenz y Chen, respectivamente. Estos resultados permiten elegir el mejor tamaño de ventana que permita tener una mayor precisión en la predicción dependiendo de la serie temporal y el método de aprendizaje automático utilizado.

Del análisis presentado en las tablas 5.14 y 5.15, se hace evidente que el modelo LSTM-ESN propuesto exhibe limitaciones para predecir un número significativo de pasos por delante (h-pasos adelante). Esta limitación puede surgir debido al diseño inherente de las redes de Estado de Eco, las cuales están diseñadas principalmente para predicciones de un paso adelante o de h pasos adelante sin forzamiento maestro, las ESN no se emplean convencionalmente para múltiples salidas y esta característica puede influir en las capacidades predictivas del modelo propuesto.

Como ya se mencionó, la principal característica del modelo propuesto, es que solo es necesario entrenar la matriz de salida, reduciendo el costo computacional en el

Tabla 5.14: Error en la predicción de h-pasos adelante de la serie temporal de Lorenz usando los modelos LSTM, LSTM-ESN y LSTM-ESN FR.

SW		**LSTM**		**LSTM-ESN DG**		**LSTM-ESN ADAM**	
Entrada	**Salida**	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$
4	2	0.0053	0.9640	0.0091	0.9382	0.0108	0.9272
8	2	0.0075	0.9500	0.0194	0.8699	0.0171	0.8853
16	2	0.0033	0.9780	0.0128	0.9152	0.0298	0.8037
20	2	0.0039	0.9741	0.0171	0.8884	0.0342	0.7764
10	5	0.0153	0.8985	0.0402	0.7319	0.0329	0.7803
20	5	0.0065	0.9571	0.0190	0.8753	0.0369	0.7580
40	5	0.0083	0.9460	0.0494	0.6791	0.0866	0.4379
50	5	0.0081	0.9445	0.0489	0.6665	0.0628	0.5713
20	10	0.0187	0.8784	0.0261	0.8290	0.0465	0.6957
40	10	0.0165	0.8932	0.0590	0.6180	0.0967	0.3738
80	10	0.0150	0.8939	0.1096	0.2242	0.1893	0.3390

Tabla 5.15: Error en la predicción de h-pasos adelante de la serie temporal de Chen usando los modelos LSTM, LSTM-ESN y LSTM-ESN FR.

Ventana Deslizante		**LSTM**		**LSTM-ESN DG**		**LSTM-ESN ADAM**	
Entrada	**Salida**	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$
4	2	0.0166	0.8515	0.0212	0.8185	0.0299	0.7338
8	2	0.0059	0.9469	0.0107	0.9403	0.0071	0.9365
16	2	0.0035	0.9686	0.0119	0.8973	0.0157	0.8629
20	2	0.0028	0.9749	0.0266	0.7656	0.0109	0.9049
10	5	0.0178	0.8424	0.0241	0.8140	0.0183	0.8402
20	5	0.0105	0.9085	0.0271	0.7666	0.0169	0.8539
40	5	0.0269	0.7583	0.0726	0.3565	0.0856	0.2413
20	10	0.0621	0.4594	0.0472	0.5971	0.0399	0.6598
40	10	0.0636	0.4306	0.0892	0.2176	0.0833	0.2694

entrenamiento, sin embargo, de ve afectada la capacidad de aprendizaje de la red, especialmente, cuando se busca realizar la predicción de h-pasos adelante.

Además, la eficacia de los resultados de las predicciones depende del sistema. Por ejemplo, el error de la predicción varió entre las series temporales de Lorenz y Chen, y la serie temporal de Chen arrojó resultados comparativamente inferiores. Es esencial destacar que incluso la red LSTM tuvo dificultades para realizar predicciones precisas

para 10 pasos adelante de la serie temporal de Chen. Esta variabilidad dependiente del sistema subraya la importancia de considerar la dinámica específica del sistema al evaluar los modelos predictivos, ya que pueden surgir disparidades en el rendimiento en función de las características inherentes de las series temporales.

En la tabla 5.16, se presenta una comparación de la predicción de h pasos hacia adelante con resultados de estudios previos en el estado del arte. En este contexto, nuestra propuesta demuestra un rendimiento comparable a los resultados reportados, a pesar de contar con un costo computacional significativamente menor en el proceso de entrenamiento.

Tabla 5.16: Comparación de la predicción de h pasos adelante utilizando diferentes métodos de aprendizaje automático.

Método	**Sistema Caótico**	**Pasos Predichos**	**Neuronas/Celdas**	**MSE**
LSTM [93]	Lorenz	2	10	0.0033
BD-LSTM [93]	Lorenz	2	10	0.0054
CNN [93]	Lorenz	2		0.0067
RNN [93]	Lorenz	2	20	0.0129
LSTM-ESN	Lorenz	2	20	0.0091

5.2.1. Series Económicas

El conjunto de datos del Producto Interno Bruto de México (PIB) comprende 174 registros, que abarcan desde 1980 hasta el segundo trimestre de 2023. Dado que el PIB se reporta trimestralmente, la predicción de un paso adelante permite estimar el PIB del siguiente trimestre, lo cual es un dato relevante en el ámbito económico. Del total de datos obtenidos del INEGI, se destina el 70 % para el entrenamiento y el restante 30 % para las pruebas, lo que equivale a 120 y 54 datos, respectivamente. La escasez de datos presenta un desafío significativo, ya que limita la información disponible para el entrenamiento. Además, para incrementar el número de datos predichos, es necesario ampliar la ventana de predicción, lo cual no es viable debido

a la limitada cantidad de datos. Por estas razones, solo se lleva a cabo la predicción de un paso hacia adelante, variando el tamaño de la ventana.

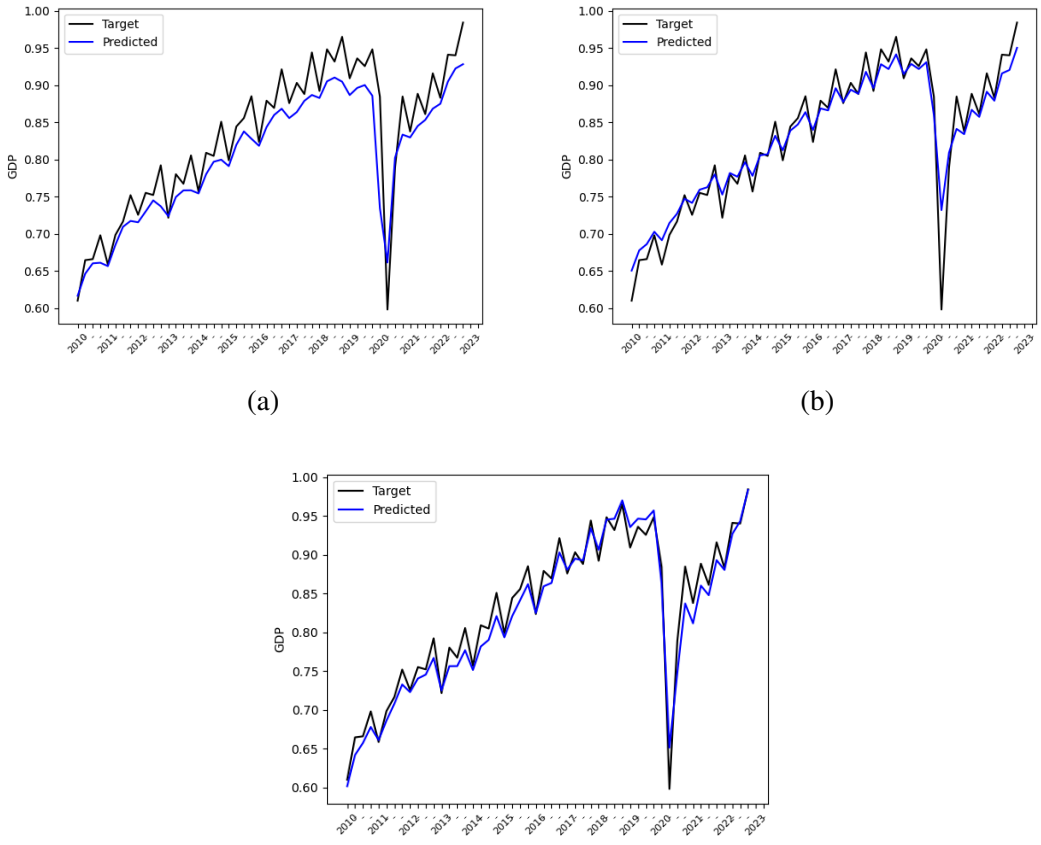
La tabla 5.17 muestra los resultados de la predicción del PIB de México utilizando la red LSTM y los modelos propuestos LSTM-ESN, empleando dos métodos de actualización para la matriz de salida: Gradiente Descendiente y ADAM, así como el modelo LSTM-ESN FR.

Tabla 5.17: Error en la predicción de 1 paso adelante del Producto Interno Bruto de México usando los modelos LSTM, LSTM-ESN y LSTM-ESN FR.

Ventana	**LSTM**		**LSTM-ESN GD**		**LSTM-ESN ADAM**		**LSTM-ESN FR**	
Deslizante	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$	**MSE**	$\mathbf{R^2}$
1	0.0012	0.8629	0.0012	0.8510	0.0004	0.9477	0.0003	**0.9622**
2	0.0014	0.8423	0.0006	0.9245	0.0007	0.9253	0.0004	**0.9596**
3	0.0023	0.7376	0.0018	0.7954	0.0007	0.9191	0.0004	**0.9581**

La figura 5.16 muestra los resultados de la predicción del PIB de México, lo que permite contrastar los resultados cuantitativos de la tabla 5.17.

Cabe destacar que las series temporales caóticas derivadas de fenómenos del mundo real suelen presentar comportamientos abruptos y complejos, como sucede con el PIB. A pesar de los desafíos inherentes asociados con los conjuntos de datos pequeños que hacen que el entrenamiento de redes neuronales sea una tarea compleja, se observan en la tabla 5.17 y la figura 5.16 resultados prometedores con los modelos LSTM-ESN y LSTM-ESN FR propuestos, presentando mejores resultados en comparación al método LSTM, a pesar de contar con un conjunto de datos pequeño.



(a)

(b)

(c)

Figura 5.16: Predicción de un paso adelante del Producto Interno Bruto de México. (a): LSTM, (b): LSTM-ESN, (c): LSTM-ESN FR

Capítulo 6

Conclusiones

Las conclusiones de esta tesis están divididas en dos partes, en la primera abordamos los resultados más relevantes de las diferentes estrategias utilizadas para la predicción de series temporales sin forzamiento del maestro, en la segunda parte se presentan las principales contribuciones en la predicción de series temporales caóticas con forzamiento del maestro.

Principales conclusiones de la predicción de series temporales caóticas sin forzamiento del maestro:

- Esta primera parte tuvo como objetivo aumentar el horizonte de predicción en series temporales caóticas utilizando ESN optimizadas e implementando la decimación en las series generadas a partir de los sistemas de Lorenz y la neurona HR. La primera contribución es la estrategia de decimación de las series temporales. En este caso es particularmente útil para la serie temporal de la neurona HR ya que presenta un comportamiento lento, esta neurona tiene la particularidad de perder el caos fácilmente, así que si se modifica el tamaño de paso utilizado en el método numérico para cambiar la dinámica lenta de la serie

temporal, se puede perder el caos. Los resultados de predicción presentados en la tabla 5.5 muestran que nuestro método propuesto para optimizar la ESN e incluir la decimación incrementa considerablemente los pasos predichos sin afectar la dinámica de la serie temporal.

- La segunda contribución fue la optimización de la matriz de entrada $\mathbf{W_{in}}$ de la ESN mediante una estrategia escalada y el algoritmo PSO. Este proceso consistió en optimizar la matriz para un número determinado de pasos hacia adelante y, posteriormente, usar esa matriz optimizada para inicializar la población en pruebas subsiguientes. Este enfoque mejoró la predicción hasta alcanzar un valor de error mínimo establecido, extendiendo el horizonte predicho. Con esta estrategia se incremento al menos en un orden de magnitud el horizonte de predicción.

- Otro hallazgo clave es que el método propuesto permitió obtener horizontes de predicción más amplios utilizando solo 100 neuronas. Esto representa una reducción significativa en el tamaño del depósito en comparación con estudios previos, logrando incrementar el horizonte de predicción de la serie temporal de Lorenz, pero con un costo computacional considerablemente menor.

- Este trabajo también investigó diferentes topologías para las ESN. Se compararon topologías con conexiones dispersas, que tienen grandes pesos de conexión, y topologías con conexiones densas, que presentan pesos de conexión pequeños, pero ambas respetan la propiedad del estado de eco. Aunque en simulación esta diferencia puede ser irrelevante, en la implementación en FPGA la densidad de conectividad es un aspecto crítico. Los resultados mostraron que una topología en anillo, que utiliza solo el 2 % de las conexiones de una topología completamente conectada (para este caso, con 50 neuronas en el depósito), ofrece un horizonte de predicción más largo sin sacrificar la precisión. Además, la topología en anillo reduce en más del 90 % el número de multiplicadores por

neurona, lo que se traduce en un ahorro significativo de recursos.

- Un problema observado fue la discrepancia entre el número de pasos predichos en la simulación y la implementación en FPGA. Esta diferencia puede ser debida al error en la aproximación de la tangente y al formato de punto fijo empleado en la FPGA. Para trabajos futuros, se recomienda realizar las simulaciones utilizando la función tangente aproximada y truncar los datos según el formato de punto fijo, para evitar estas discrepancias entre la simulación y la implementación.

Esta tesis ha logrado ampliar significativamente el horizonte de predicción en series temporales caóticas mediante la optimización de ESN, incluyendo la decimación y la optimización de la matriz de entrada. Además, se ha demostrado que es posible reducir considerablemente los recursos computacionales al emplear topologías eficientes como la de anillo, especialmente para implementaciones en FPGA. La combinación de estos enfoques ha permitido obtener resultados de predicción competitivos con un costo computacional menor, lo que representa una contribución significativa en el ámbito de la predicción de series temporales caóticas.

A continuación se presentan las principales conclusiones de la predicción de series temporales caóticas con forzamiento del maestro:

- En esta tesis, se propusieron los modelos híbridos LSTM-ESN y LSTM-ESN FR para predecir series temporales caóticas utilizando una configuración de forzamiento maestro, con el objetivo de reducir el costo computacional asociado al entrenamiento de las matrices del modelo. En la configuración más eficiente, con una sola celda, el modelo LSTM-ESN superó al LSTM tradicional. Cuando se amplió el análisis a un mayor número de celdas, ambos modelos presentaron errores similares; sin embargo, el LSTM-ESN ofreció una ventaja

clara al requerir únicamente el entrenamiento de la matriz de salida, lo que supone menos del 90 % de los parámetros en comparación con la red LSTM.

- Un hallazgo importante fue el rendimiento superior de nuestro modelo propuesto con tamaños de ventana más pequeños, en contraste con la práctica convencional en LSTM, donde ventanas más grandes suelen ofrecer mejores resultados. En nuestro enfoque, una ventana más pequeña reduce las operaciones y, por ende, el costo computacional. No obstante, se observa que el error de predicción de la LSTM-ESN aumenta con el número de pasos predichos, probablemente debido a que este modelo produce mejores resultados con ventanas pequeñas. A medida que se incrementa el tamaño de la ventana, la precisión de la predicción disminuye, un fenómeno también observado en el LSTM tradicional, lo que resalta la importancia de ajustar cuidadosamente el tamaño de la ventana para optimizar la predicción.

- Exploramos dos métodos distintos para actualizar la matriz de salida del modelo propuesto LSTM-ESN: gradiente descendiente y el método ADAM, ambos ampliamente utilizados en el entrenamiento de redes neuronales. Los resultados experimentales indicaron que no había diferencias significativas en los errores de predicción entre ambos métodos al aplicarlos a series temporales caóticas.

- Sin embargo, cuando se entreno la matriz de salida utilizando un gradiente descendiente de orden fraccionario, se observaron mejoras en los resultados para la predicción de un paso adelante. Esto puede deberse al principio de memoria corta implementado. Aunque el uso de derivadas de orden fraccionario incrementa el costo computacional del entrenamiento, este aumento no es significativo en comparación con la cantidad de matrices que deben entrenarse en la LSTM convencional.

- Las redes LSTM-ESN y LSTM-ESN FR también mostraron un rendimiento superior en la predicción del producto interno bruto (PIB) de México en

comparación con la red LSTM, a pesar de las limitaciones impuestas por una base de datos pequeña. Nos centramos en predecir un paso adelante (el cambio trimestre a trimestre en el PIB), dado que predecir varios pasos resulta complicado por la necesidad de incrementar el tamaño de la ventana, lo que dificulta el entrenamiento.

- Para investigaciones futuras, se pueden emplear métodos de optimización para determinar el tamaño óptimo de ventana, minimizando así los errores de predicción. También se puede dar mayor énfasis a la selección cuidadosa de hiperparámetros para extender los horizontes de predicción, especialmente en escenarios con bases de datos limitadas como el PIB. Además, se puede explorar la predicción de series temporales caóticas contaminadas con ruido, o conjuntos de datos que presenten ruido intrínseco o inducido durante la medición.

Finalmente, esta tesis ha demostrado que las redes LSTM-ESN y LSTM-ESN FR propuestas, en configuración con forzamiento maestro, ofrecen ventajas significativas sobre la LSTM tradicional, tanto en términos de costo computacional como de rendimiento predictivo. El uso de tamaños de ventana más pequeños permitió reducir las operaciones computacionales sin sacrificar la precisión en las predicciones, y el entrenamiento de la matriz de salida mediante un gradiente descendiente de orden fraccionario mejoró aún más los resultados. A pesar de los desafíos presentados por bases de datos pequeñas, como en el caso del PIB de México, estos modelos mostraron ser efectivos, lo que sugiere un amplio potencial para futuras aplicaciones en la predicción de series temporales caóticas.

6.1. Contribuciones

La principal contribución de este trabajo doctoral es la propuesta de los modelos híbridos LSTM-ESN y LSTM-ESN FR para la predicción de series temporales caóticas, sin embargo, estos resultados aun no están publicados.

Artículos publicados como primer autor:

1. A. M. González-Zapata, E. Tlelo-Cuautle, B. Ovilla-Martínez, I. Cruz-Vega, and L. G. De la Fraga, "Optimizing Echo State Networks for Enhancing Large Prediction Horizons of Chaotic Time Series", *Mathematics*, vol. 10, p. 3886, Jan. 2022. Number: 20 Publisher: Multidisciplinary Digital Publishing Institute [30].

2. A.M. González-Zapata, E. Tlelo-Cuautle, and I.Cruz-Vega, "On the Optimization of Machine Learning Techniques for Chaotic Time Series Prediction", *Electronics*, vol. 11, p. 3612, Jan. 2022. Number: 21 Publisher: Multidisciplinary Digital Publishing Institute [19].

3. A. M. González-Zapata, L. G. de la Fraga, B. Ovilla-Martínez, E. Tlelo-Cuautle, and I- Cruz-Vega, "Enhanced FPGA implementation of Echo State Networks for chaotic time series prediction", *Integration*, vol. 92, pp. 48-57, Sept. 2023 [96].

Colaboración en artículos:

1. E. Tlelo-Cuautle, J. D. Díaz-Muñoz, A. M. González-Zapata, R. Li, W. D. León-Salas, F. V. Fernández, O. Guillén-Fernández, and I. Cruz-Vega, "Optimization of fractional-order chaotic cellular neural networks by metaheuristics", *The European Physical Journal Special Topics*, vol. 231 pp. 2037-2043, Aug. 2022 [90].

2. J. A. Martínez-Garcia, A. M. Gonzalez-Zapata, E. J. Rechy-Ramirez, and E. Tlelo-Cuautle, "On the Prediction of Chaotic Time Series using Neural Networks", *Chaos Theory and Applications*, vol. 4, pp. 94-103, July 2022 [93].

3. M. A. Valencia-Ponce, A. M. González-Zapata, L. G. de la Fraga, C. Sanchez-Lopez, and E. Tlelo-Cuautle, "Integrated Circuit Design of Fractional-Order Chaotic Systems Optimized by Metaheuristics", *Electronics*, vol. 12, p. 413, Jan. 2023 [97].

capítulo de libro:

1. L. G. de la Fraga, A. M. González-Zapata, and A. C. Ramírez, "Echo State Networks to Solve Classification Tasks", in *Machine Learning for Complex and Unmanned Systems*, CRC Press, 2024. Num Pages: 11 [98].

Anexo A

Medidas en la Predicción de Series Temporales Caóticas

A.1. Tiempos de Lyapunov

Uno de los objetivos al momento de realizar la predicción de series temporales caóticas es poder predecir la mayor cantidad de datos posibles, en estos casos los autores suelen presentar sus resultados en *n pasos predichos hacia adelante* (*n steps-ahead*) o *n tiempos de Lyapunov* ($n\lambda^{-1}$), El tiempo de Lyapunov es el inverso del máximo exponente de Lyapunov del sistema, y no es recomendada ya que existen diferentes formas de encontrar los exponentes de Lyapunov, ya sea a partir del modelo o a partir de los datos usando software como TISEAN, donde parámetros como el tamaño de paso afectan el resultado de los exponentes de Lyapunov y por tanto es difícil realizar una comparación con otros trabajos.

A.2. Horizonte de Predicción

Otra medida importante en la predicción de series temporales caóticas, es el horizonte de predicción, el cual esta definido como el intervalo de tiempo durante el cual el error normalizado es menor que un umbral k_n [77, 22], la ecuación (A.2.1) describe como calcular el error normalizado [25], donde HP es el horizonte de predicción, y_{target} son los datos objetivo, y son los datos predichos.

$$\frac{\|y_{target}(i) - y(i)\|}{\sqrt{\frac{1}{HP}\sum_{j=1}^{HP}\|y(j)\|^2}} \leq k_n \tag{A.2.1}$$

A.3. Raíz del Error Cuadrático Medio (RMSE) y Error Cuadrático Medio (MSE)

Representa la desviación estándar de la muestra de las diferencias entre los valores pronosticados y los valores observados, el resultado ideal de este error es cero, lo que significa que los valores pronosticados son iguales a los valores observados, el RMSE es calculado a partir de la ecuación (A.3.1).

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}} \tag{A.3.1}$$

Donde $\hat{y}$ es el valor de la predicción, y es el valor observado y n es el número de muestras [99].

El MSE se calcula de forma similar a la ecuación (A.3.1), solo es necesario elevar al cuadrado la ecuación, o no calcular la raíz cuadrada.

A.4. Error Absoluto Promedio (MAE)

Se utiliza para medir que tan cerca están los valores pronosticados de los valores observados, el resultado ideal de este error es cero, lo que implica que los valores pronosticados son iguales a los resultados finales, para calcular el MAE se utiliza la ecuación (A.4.1).

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i| \tag{A.4.1}$$

Donde $\hat{y}$ es el valor de la predicción, y es el valor observado, y n es el número de muestras [99].

A.5. Coeficiente de determinación (R^2)

El coeficiente determina la calidad del modelo para replicar los resultados, y la proporción de variación de los resultados que puede explicarse por el modelo, cuando el resultado de este error es igual a 1 significa que el modelo replica perfectamente los resultados, es decir, los valores pronosticados son iguales a los valores observados, el R^2 es calculado utilizando la ecuación (A.5.1)

$$R^2 = \frac{{\sigma^2}_{XY}}{{\sigma^2}_X {\sigma^2}_Y} \tag{A.5.1}$$

Donde ${\sigma^2}_{XY}$ es la covarianza de (X, Y), ${\sigma^2}_X$ es la desviación típica de la variable X y ${\sigma^2}_Y$ es la desviación típica de la variable Y [100].

Bibliografía

[1] B. T. Nadiga, "Reservoir Computing as a Tool for Climate Predictability Studies," *Journal of Advances in Modeling Earth Systems*, vol. 13, no. 4, p. e2020MS002290, 2021. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2020MS002290.

[2] P. D. Dueben and P. Bauer, "Challenges and design choices for global weather and climate models based on machine learning," *Geoscientific Model Development*, vol. 11, pp. 3999–4009, Oct. 2018. Publisher: Copernicus GmbH.

[3] S. Scher, "Toward Data-Driven Weather and Climate Forecasting: Approximating a Simple General Circulation Model With Deep Learning," *Geophysical Research Letters*, vol. 45, no. 22, pp. 12,616–12,622, 2018. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2018GL080704.

[4] S. Shahi, C. D. Marcotte, C. J. Herndon, F. H. Fenton, Y. Shiferaw, and E. M. Cherry, "Long-Time Prediction of Arrhythmic Cardiac Action Potentials Using Recurrent Neural Networks and Reservoir Computing," *Frontiers in Physiology*, vol. 12, Sept. 2021. Publisher: Frontiers.

[5] K.-j. Kim, "Financial time series forecasting using support vector machines," *Neurocomputing*, vol. 55, pp. 307–319, Sept. 2003.

[6] A. Chattopadhyay, P. Hassanzadeh, and D. Subramanian, "Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods:

reservoir computing, artificial neural network, and long short-term memory network," *Nonlinear Processes in Geophysics*, vol. 27, pp. 373–389, July 2020. Publisher: Copernicus GmbH.

[7] F. A. Munir, M. Zia, and H. Mahmood, "Designing multi-dimensional logistic map with fixed-point finite precision," *Nonlinear Dynamics*, vol. 97, pp. 2147–2158, Sept. 2019.

[8] M. A. Valencia-Ponce, E. Tlelo-Cuautle, and L. G. de la Fraga, "Estimating the Highest Time-Step in Numerical Methods to Enhance the Optimization of Chaotic Oscillators," *Mathematics*, vol. 9, p. 1938, Jan. 2021. Number: 16 Publisher: Multidisciplinary Digital Publishing Institute.

[9] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach," *Physical Review Letters*, vol. 120, p. 024102, Jan. 2018. Publisher: American Physical Society.

[10] U. Thissen, R. van Brakel, A. P. de Weijer, W. J. Melssen, and L. M. C. Buydens, "Using support vector machines for time series prediction," *Chemometrics and Intelligent Laboratory Systems*, vol. 69, pp. 35–49, Nov. 2003.

[11] S. Scher and G. Messori, "Generalization properties of feed-forward neural networks trained on Lorenz systems," *Nonlinear Processes in Geophysics*, vol. 26, pp. 381–399, Nov. 2019. Publisher: Copernicus GmbH.

[12] P. Amil, M. C. Soriano, and C. Masoller, "Machine learning algorithms for predicting the amplitude of chaotic laser pulses," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, p. 113111, Nov. 2019.

[13] P. Antonik, M. Gulina, J. Pauwels, and S. Massar, "Using a reservoir computer to learn chaotic attractors, with applications to chaos synchronization and

cryptography," *Physical Review E*, vol. 98, p. 012215, July 2018. Publisher: American Physical Society.

[14] P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics," *Neural Networks*, vol. 126, pp. 191–217, June 2020.

[15] Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 28, p. 061104, June 2018.

[16] T. Arcomano, I. Szunyogh, J. Pathak, A. Wikner, B. R. Hunt, and E. Ott, "A Machine Learning-Based Global Atmospheric Forecast Model," *Geophysical Research Letters*, vol. 47, no. 9, p. e2020GL087776, 2020. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2020GL087776.

[17] L. M. Smith, J. Z. Kim, Z. Lu, and D. S. Bassett, "Learning continuous chaotic attractors with a reservoir computer," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 32, p. 011101, Jan. 2022.

[18] Z. Tian, "Echo state network based on improved fruit fly optimization algorithm for chaotic time series prediction," *Journal of Ambient Intelligence and Humanized Computing*, vol. 13, pp. 3483–3502, July 2022.

[19] A. M. González-Zapata, E. Tlelo-Cuautle, and I. Cruz-Vega, "On the Optimization of Machine Learning Techniques for Chaotic Time Series Prediction," *Electronics*, vol. 11, p. 3612, Jan. 2022. Number: 21 Publisher: Multidisciplinary Digital Publishing Institute.

[20] N. Chouikhi, B. Ammar, N. Rokbani, and A. M. Alimi, "PSO-based analysis of Echo State Network parameters for time series forecasting," *Applied Soft Computing*, vol. 55, pp. 211–225, June 2017.

[21] A. M. Schäfer and H.-G. Zimmermann, "Recurrent neural networks are universal approximators," *International Journal of Neural Systems*, vol. 17, pp. 253–263, Aug. 2007. Publisher: World Scientific Publishing Co.

[22] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 28, p. 041101, Apr. 2018.

[23] D. Li, M. Han, and J. Wang, "Chaotic Time Series Prediction Based on a Novel Robust Echo State Network," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, pp. 787–799, May 2012. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[24] A. Griffith, A. Pomerance, and D. J. Gauthier, "Forecasting chaotic systems with very low connectivity reservoir computers," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 29, p. 123108, Dec. 2019.

[25] A. Racca and L. Magri, "Robust Optimization and Validation of Echo State Networks for learning chaotic dynamics," *Neural Networks*, vol. 142, pp. 252–268, Oct. 2021.

[26] Z. Wang, Y.-R. Zeng, S. Wang, and L. Wang, "Optimizing echo state network with backtracking search optimization algorithm for time series forecasting," *Engineering Applications of Artificial Intelligence*, vol. 81, pp. 117–132, May 2019.

[27] H.-C. Chen and D.-Q. Wei, "Chaotic time series prediction using echo state network based on selective opposition grey wolf optimizer," *Nonlinear Dynamics*, vol. 104, pp. 3925–3935, June 2021.

[28] J. Pathak, Z. Lu, B. R. Hunt, M. Girvan, and E. Ott, "Using machine learning to replicate chaotic attractors and calculate Lyapunov exponents from data,"

Chaos: An Interdisciplinary Journal of Nonlinear Science, vol. 27, p. 121102, Dec. 2017. Publisher: American Institute of Physics.

[29] X. Li, F. Bi, X. Yang, and X. Bi, "An Echo State Network With Improved Topology for Time Series Prediction," *IEEE Sensors Journal*, vol. 22, pp. 5869–5878, Mar. 2022. Conference Name: IEEE Sensors Journal.

[30] A. M. González-Zapata, E. Tlelo-Cuautle, B. Ovilla-Martinez, I. Cruz-Vega, and L. G. De la Fraga, "Optimizing Echo State Networks for Enhancing Large Prediction Horizons of Chaotic Time Series," *Mathematics*, vol. 10, p. 3886, Jan. 2022. Number: 20 Publisher: Multidisciplinary Digital Publishing Institute.

[31] L. Shen, J. Chen, Z. Zeng, J. Yang, and J. Jin, "A novel echo state network for multivariate and nonlinear time series prediction," *Applied Soft Computing*, vol. 62, pp. 524–535, Jan. 2018.

[32] D. Kleyko, E. P. Frady, M. Kheffache, and E. Osipov, "Integer Echo State Networks: Efficient Reservoir Computing for Digital Hardware," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, pp. 1688–1701, Apr. 2022. Conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[33] A. D. Pano-Azucena, E. Tlelo-Cuautle, B. Ovilla-Martinez, L. G. de la Fraga, and R. Li, "Pipeline FPGA-Based Implementations of ANNs for the Prediction of up to 600-Steps-Ahead of Chaotic Time Series," *Journal of Circuits, Systems and Computers*, vol. 30, p. 2150164, July 2021. Publisher: World Scientific Publishing Co.

[34] M. Dale, S. O'Keefe, A. Sebald, S. Stepney, and M. A. Trefzer, "Reservoir computing quality: connectivity and topology," *Natural Computing*, vol. 20, pp. 205–216, June 2021.

[35] R. Chandra, S. Goyal, and R. Gupta, "Evaluation of Deep Learning Models for Multi-Step Ahead Time Series Prediction," *IEEE Access*, vol. 9, pp. 83105–83123, 2021. Conference Name: IEEE Access.

[36] Y. Lei, J. Hu, and J. Ding, "A hybrid model based on deep LSTM for predicting high-dimensional chaotic systems," Jan. 2020. arXiv:2002.00799 [cs, eess].

[37] M. Sangiorgio, F. Dercole, and G. Guariso, "Forecasting of noisy chaotic systems with deep neural networks," *Chaos, Solitons & Fractals*, vol. 153, p. 111570, Dec. 2021.

[38] M. Xu, M. Han, T. Qiu, and H. Lin, "Hybrid Regularized Echo State Network for Multivariate Chaotic Time Series Prediction," *IEEE Transactions on Cybernetics*, vol. 49, pp. 2305–2315, June 2019. Conference Name: IEEE Transactions on Cybernetics.

[39] S. Bompas, B. Georgeot, and D. Guéry-Odelin, "Accuracy of neural networks for the simulation of chaotic dynamics: Precision of training data vs precision of the algorithm," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 30, p. 113118, Nov. 2020. Publisher: American Institute of Physics.

[40] Y.-C. Bo, P. Wang, and X. Zhang, "An asynchronously deep reservoir computing for predicting chaotic time series," *Applied Soft Computing*, vol. 95, p. 106530, Oct. 2020.

[41] Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep Learning with Long Short-Term Memory for Time Series Prediction," *IEEE Communications Magazine*, vol. 57, pp. 114–119, June 2019. Conference Name: IEEE Communications Magazine.

[42] X. Yao and Z. Wang, "Fractional Order Echo State Network for Time Series Prediction," *Neural Processing Letters*, vol. 52, pp. 603–614, Aug. 2020.

[43] L. G. de la Fraga, B. Ovilla-Martínez, and E. Tlelo-Cuautle, "Echo state network implementation for chaotic time series prediction," *Microprocessors and Microsystems*, vol. 103, p. 104950, Nov. 2023.

[44] J. Qiao, L. Wang, C. Yang, and K. Gu, "Adaptive Levenberg-Marquardt Algorithm Based Echo State Network for Chaotic Time Series Prediction," *IEEE Access*, vol. 6, pp. 10720–10732, 2018. Conference Name: IEEE Access.

[45] M. Sangiorgio and F. Dercole, "Robustness of LSTM neural networks for multi-step forecasting of chaotic time series," *Chaos, Solitons & Fractals*, vol. 139, p. 110045, Oct. 2020.

[46] S. Strogatz, "Nonlinear dynamics and chaos (studies in nonlinearity)," 1994.

[47] K. Tsaneva-Atanasova, H. M. Osinga, T. Rieß, and A. Sherman, "Full system bifurcation analysis of endocrine bursting models," *Journal of Theoretical Biology*, vol. 264, pp. 1133–1146, June 2010.

[48] P. Augustová and Z. Beran, "Characteristics of the Chen Attractor," in *Nostradamus 2013: Prediction, Modeling and Analysis of Complex Systems* (I. Zelinka, G. Chen, O. E. Rössler, V. Snasel, and A. Abraham, eds.), Advances in Intelligent Systems and Computing, (Heidelberg), pp. 305–312, Springer International Publishing, 2013.

[49] "Instituto Nacional de Estadística y Geografía (INEGI)."

[50] G. J. Dolecek, *Advances in multirate systems*. Springer, 2017.

[51] "Nonlinear Time Series Routines."

[52] K. E. Chlouverakis and J. C. Sprott, "A comparison of correlation and Lyapunov dimensions," *Physica D: Nonlinear Phenomena*, vol. 200, pp. 156–164, Jan. 2005.

[53] G. Yanan, C. Xiaoqun, L. Bainian, and P. Kecheng, "Chaotic Time Series Prediction Using LSTM with CEEMDAN," *Journal of Physics: Conference Series*, vol. 1617, p. 012094, Aug. 2020. Publisher: IOP Publishing.

[54] M. Xu and M. Han, "Adaptive Elastic Echo State Network for Multivariate Time Series Prediction," *IEEE Transactions on Cybernetics*, vol. 46, pp. 2173–2183, Oct. 2016. Conference Name: IEEE Transactions on Cybernetics.

[55] X. Guo, Y. Sun, and J. Ren, "Low dimensional mid-term chaotic time series prediction by delay parameterized method," *Information Sciences*, vol. 516, pp. 1–19, Apr. 2020.

[56] M. N. Alemu, "A Fuzzy Model for Chaotic Time Series Prediction," 2018.

[57] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky- integrator neurons," *Neural Networks*, vol. 20, pp. 335–352, Apr. 2007.

[58] I. B. Yildiz, H. Jaeger, and S. J. Kiebel, "Re-visiting the echo state property," *Neural Networks*, vol. 35, pp. 1–9, Nov. 2012.

[59] M.-H. Yusoff, J. Chrol-Cannon, and Y. Jin, "Modeling neural plasticity in echo state networks for classification and regression," *Information Sciences*, vol. 364-365, pp. 184–196, Oct. 2016.

[60] M. Lukoševičius and H. Jaeger, "Reservoir computing approaches to recurrent neural network training," *Computer Science Review*, vol. 3, pp. 127–149, Aug. 2009.

[61] H. Jaeger, "Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach," *GMD-Forschungszentrum Informationstechnik, 2002.*, vol. 5, Jan. 2002.

[62] S. Wang, X.-J. Yang, and C.-J. Wei, "Harnessing Non-linearity by Sigmoid-wavelet Hybrid Echo State Networks (SWHESN)," in *2006 6th World Congress on Intelligent Control and Automation*, vol. 1, pp. 3014–3018, June 2006.

[63] J. S. Verducci, X. Shen, and J. Lafferty, *Prediction and Discovery: AMS-IMS-SIAM Joint Summer Research Conference, Machine and Statistical Learning : Prediction and Discovery, June 25-29, 2006, Snowbird, Utah.* American Mathematical Soc., 2007. Google-Books-ID: 0xK9AwAAQBAJ.

[64] A. Bala, I. Ismail, R. Ibrahim, and S. M. Sait, "Applications of Metaheuristics in Reservoir Computing Techniques: A Review," *IEEE Access*, vol. 6, pp. 58012–58029, 2018. Conference Name: IEEE Access.

[65] Y. Zhang, Y. Yu, and D. Liu, "The Application of modified ESN in chaotic time series prediction," in *2013 25th Chinese Control and Decision Conference (CCDC)*, pp. 2213–2218, May 2013. ISSN: 1948-9447.

[66] D. Xu, J. Lan, and J. Principe, "Direct adaptive control: an echo state network and genetic algorithm approach," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 3, pp. 1483–1486 vol. 3, July 2005. ISSN: 2161-4407.

[67] A. Bala, I. Ismail, and R. Ibrahim, "Cuckoo Search Based Optimization of Echo State Network for Time Series Prediction," in *2018 International Conference on Intelligent and Advanced System (ICIAS)*, pp. 1–6, Aug. 2018.

[68] M. Zhang, B. Wang, Y. Zhou, and H. Sun, "WOA-Based Echo State Network for Chaotic Time Series Prediction," *Journal of the Korean Physical Society*, vol. 76, pp. 384–391, Mar. 2020.

[69] N. Chouikhi, R. Fdhila, B. Ammar, N. Rokbani, and A. M. Alimi, "Single- and multi-objective particle swarm optimization of reservoir structure in Echo

State Network," in *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 440–447, July 2016. ISSN: 2161-4407.

[70] J. Liu, T. Sun, Y. Luo, S. Yang, Y. Cao, and J. Zhai, "Echo state network optimization using binary grey wolf algorithm," *Neurocomputing*, vol. 385, pp. 310–318, Apr. 2020.

[71] C. Yang, J. Qiao, and L. Wang, "A novel echo state network design method based on differential evolution algorithm," in *2017 36th Chinese Control Conference (CCC)*, pp. 3977–3982, July 2017. ISSN: 1934-1768.

[72] N. Chouikhi, B. Ammar, N. Rokbani, A. M. Alimi, and A. Abraham, "A Hybrid Approach Based on Particle Swarm Optimization for Echo State Network Initialization," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2896–2901, Oct. 2015.

[73] S. Otte, M. V. Butz, D. Koryakin, F. Becker, M. Liwicki, and A. Zell, "Optimizing recurrent reservoirs with neuro-evolution," *Neurocomputing*, vol. 192, pp. 128–138, June 2016.

[74] X. Na, M. Han, W. Ren, and K. Zhong, "Modified BBO-Based Multivariate Time-Series Prediction System With Feature Subset Selection and Model Parameter Optimization," *IEEE Transactions on Cybernetics*, vol. 52, pp. 2163–2173, Apr. 2022. Conference Name: IEEE Transactions on Cybernetics.

[75] R. Eberhart and Y. Shi, "Guest Editorial Special Issue on Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 201–203, June 2004. Conference Name: IEEE Transactions on Evolutionary Computation.

[76] Q. Bai, "Analysis of particle swarm optimization algorithm," *Computer and information science*, vol. 3, no. 1, p. 180, 2010.

[77] N. A. K. Doan, W. Polifke, and L. Magri, "Physics-Informed Echo State Networks for Chaotic Systems Forecasting," in *Computational Science – ICCS 2019* (J. M. F. Rodrigues, P. J. S. Cardoso, J. Monteiro, R. Lam, V. V. Krzhizhanovskaya, M. H. Lees, J. J. Dongarra, and P. M. Sloot, eds.), Lecture Notes in Computer Science, (Cham), pp. 192–198, Springer International Publishing, 2019.

[78] E. Tlelo-Cuautle, J. Rangel-Magdaleno, and L. G. De La Fraga, *Engineering Applications of FPGAs*. Cham: Springer International Publishing, 2016.

[79] Mathworks, "Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) - MATLAB pchip - MathWorks América Latina." https://la.mathworks.com/help/matlab/ref/pchip.html, 2022.

[80] P. Lara-Benítez, M. Carranza-García, and J. C. Riquelme, "An Experimental Review on Deep Learning Architectures for Time Series Forecasting," *International Journal of Neural Systems*, vol. 31, p. 2130001, Mar. 2021. Publisher: World Scientific Publishing Co.

[81] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997. Conference Name: Neural Computation.

[82] M. Gooneratne, K. C. Sim, P. Zadrazil, A. Kabel, F. Beaufays, and G. Motta, "Low-Rank Gradient Approximation for Memory-Efficient on-Device Training of Deep Neural Network," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3017–3021, May 2020. ISSN: 2379-190X.

[83] Y. Wei, Y. Kang, W. Yin, and Y. Wang, "Generalization of the gradient method with fractional order gradient direction," *Journal of the Franklin Institute*, vol. 357, pp. 2514–2532, Mar. 2020.

[84] M. Joshi, S. Bhosale, and V. A. Vyawahare, "A survey of fractional calculus applications in artificial neural networks," *Artificial Intelligence Review*, vol. 56, pp. 13897–13950, Nov. 2023.

[85] K. Oldham and J. Spanier, *The fractional calculus theory and applications of differentiation and integration to arbitrary order*. Elsevier, 1974.

[86] K. S. Miller and B. Ross, "An introduction to the fractional calculus and fractional differential equations," *(No Title)*, 1993.

[87] I. Podlubny, *Fractional differential equations: an introduction to fractional derivatives, fractional differential equations, to methods of their solution and some of their applications*. elsevier, 1998.

[88] N. J. Ford and A. C. Simpson, "The numerical solution of fractional differential equations: Speed versus accuracy," *Numerical Algorithms*, vol. 26, pp. 333–346, Apr. 2001.

[89] L. Dorcak, J. Prokop, and I. Kostial, "Investigation of the properties of fractional-order dynamical systems," in *Proc. 11th Int. Conf. ProcessControl*, pp. 19–20, 1994.

[90] E. Tlelo-Cuautle, A. M. González-Zapata, J. D. Díaz-Muñoz, L. G. de la Fraga, and I. Cruz-Vega, "Optimization of fractional-order chaotic cellular neural networks by metaheuristics," *The European Physical Journal Special Topics*, vol. 231, pp. 2037–2043, Aug. 2022.

[91] E. Tlelo-Cuautle, J. D. Díaz-Muñoz, A. M. González-Zapata, R. Li, W. D. León-Salas, F. V. Fernández, O. Guillén-Fernández, and I. Cruz-Vega, "Chaotic Image Encryption Using Hopfield and Hindmarsh–Rose Neurons Implemented on FPGA," *Sensors*, vol. 20, p. 1326, Jan. 2020. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[92] A. M. González-Zapata, E. Tlelo-Cuautle, I. Cruz-Vega, and W. D. León-Salas, "Synchronization of chaotic artificial neurons and its application to secure image transmission under MQTT for IoT protocol," *Nonlinear Dynamics*, vol. 104, pp. 4581–4600, June 2021.

[93] J. A. Martinez-garcia, A. M. Gonzalez-zapata, E. J. Rechy-ramirez, and E. Tlelo-cuautle, "On the Prediction of Chaotic Time Series using Neural Networks," *Chaos Theory and Applications*, vol. 4, pp. 94–103, July 2022. Number: 2 Publisher: Akif AKGÜL.

[94] Z. Cao, "Evolutionary optimization of artificial neural network using an interactive phase-based optimization algorithm for chaotic time series prediction," *Soft Computing*, vol. 24, pp. 17093–17109, Nov. 2020.

[95] P. Ong and Z. Zainuddin, "Optimizing wavelet neural networks using modified cuckoo search for multi-step ahead chaotic time series prediction," *Applied Soft Computing*, vol. 80, pp. 374–386, July 2019.

[96] A. M. Gonzalez-Zapata, L. G. de la Fraga, B. Ovilla-Martinez, E. Tlelo-Cuautle, and I. Cruz-Vega, "Enhanced FPGA implementation of Echo State Networks for chaotic time series prediction," *Integration*, vol. 92, pp. 48–57, Sept. 2023.

[97] M. A. Valencia-Ponce, A. M. González-Zapata, L. G. de la Fraga, C. Sanchez-Lopez, and E. Tlelo-Cuautle, "Integrated Circuit Design of Fractional-Order Chaotic Systems Optimized by Metaheuristics," *Electronics*, vol. 12, p. 413, Jan. 2023. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[98] L. G. d. l. Fraga, A. M. Gonzalez-Zapata, and A. C. Ramírez, "Echo State Networks to Solve Classification Tasks," in *Machine Learning for Complex and Unmanned Systems*, CRC Press, 2024. Num Pages: 11.

[99] C. J. Willmott and K. Matsuura, "On the use of dimensioned measures of error to evaluate the performance of spatial interpolators," *International Journal of Geographical Information Science*, vol. 20, pp. 89–102, Jan. 2006. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/13658810500286976.

[100] R. G. Pontius, O. Thontteh, and H. Chen, "Components of information for multiple resolution comparison between maps that share a real variable," *Environmental and Ecological Statistics*, vol. 15, pp. 111–142, June 2008.