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

DISEÑO DE INTERVALOS DE PREDICCIÓN NEURONALES BASADOS EN *DEEP LEARNING* Y *JOINT SUPERVISION*

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL ELÉCTRICO

SEBASTIÁN ALFONSO IVÁN PARRA FLORES

PROFESORA GUÍA:
DORIS SÁEZ HUEICHAPAN

MIEMBROS DE LA COMISIÓN:
PABLO ESTÉVEZ VALENCIA
DANIEL SBARBARO HOFER

SANTIAGO DE CHILE
2021

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
Y AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: SEBASTIÁN ALFONSO IVÁN PARRA FLORES
FECHA: 2021
PROF. GUÍA: DORIS SÁEZ HUEICHAPAN

DISEÑO DE INTERVALOS DE PREDICCIÓN NEURONALES BASADOS EN *DEEP LEARNING* Y *JOINT SUPERVISION*

En este trabajo se formulan nuevas metodologías para la construcción de modelos de intervalo, los cuales tienen como objetivo estimar la incertidumbre de un modelo predictivo por medio del cálculo de un rango de valores donde se espera encontrar el valor real de la señal. En particular, se propone una nueva función de costo para el entrenamiento de modelos de intervalo neuronales, denominanda *Joint Supervision* Selectivo, que presenta un bajo costo computacional y no posee problemas de convergencia en espacios de búsqueda complejos, con el objetivo de ser compatible con arquitecturas de *Deep Learning* y aprendizaje en línea. Adicionalmente, se propone una extensión de un modelo de intervalo neuronal existente para compatibilizarlo con sistemas difusos Takagi-Sugeno.

Ambas propuestas fueron probadas con simulaciones y comparadas con modelos del estado del arte, midiendo su error predictivo, porcentaje de cobertura y ancho de los intervalos, y el tiempo de entrenamiento. Los resultados mostraron que las arquitecturas de *Deep Learning* obtuvieron un menor error predictivo e intervalos más delgados sólo cuando fueron utilizadas en aplicaciones con espacios de búsqueda sencillos. También se apreció que el método neuronal propuesto logró menores tiempos de entrenamiento en implementaciones paramétricamente densas y demostró una mejor calidad de intervalo en aplicaciones con espacios de búsqueda complejos.

A esas oncecitas que me escucharon y ayudaron a que esto fuese posible

Agradecimientos

Me gustaría partir agradeciendo a mi familia, que ha estado apoyándome y brindándome amor desde que tengo memoria y que sin duda me han ayudado a ser la persona que soy ahora. En particular, quiero agradecer a mi padre Hector por siempre estar pendiente de mi bienestar y siempre creer en mis capacidades, a mi hermana Pierreangely (dejémoslo en Piña mejor) por apoyarme en mis momentos más difíciles, y a mi abuela Elizabeth, quien siempre ha demostrado interés en mis avances y estoy seguro ahora debe tener una orgullosa sonrisa en su rostro al leer este párrafo.

Agradezco profundamente a mis amigos que me han acompañado durante estos (largos) años de mi vida universitaria: En primer lugar agradecer inmensamente a Mariana por su inconmensurable apoyo, cariño, y compañía tanto en los días de emoción como en los de estrés y ansiedad, que fue especialmente valioso durante estos últimos años de caos y encierro. A mis amigos de inducción/plan común Caro, Vicente y Vale, donde si bien cada uno terminó siguiendo sus propios caminos, fueron un grupo de amigos fundamental durante mis primeros años de carrera, por lo que los recuerdo con mucho afecto. Al grupo de LA SAULAAAA (y el subgrupo de los BRLDWS), una amistad que se viene cultivando desde mis años escolares y ha superado la prueba del tiempo, a quienes les agradezco por todas esas risas que traen con cada junta (aunque sean una vez a las miles!) y por haberme escuchado en los momentos donde más los necesitaba. A las amistades que hice en el Coro FCFM por presentarme a un grupo humano muy bonito que me ayudó a crecer como persona. Por último, quiero darles unas enormes gracias al grupo de las Oncecitas por todas esas juntas, tanto planificadas como espontáneas, donde a través de un simple pancito con palta-tomate y un tecito mis problemas parecían solucionables y mis días se llenaban de cariño y felicidad.

También quiero agradecer al Instituto de Sistemas Complejos de Ingeniería ANID PIA/-BASAL AFB180003, al Solar Energy Research Center SERC-Chile ANID/FONDAP/15110019, y al CONICYT/FONDECYT 1170683 "Robust Distributed Predictive Control Strategies for the Coordination of Hybrid AC and DC Microgrids" por apoyar este trabajo. Finalmente, le doy las gracias a todo el equipo del Laboratorio de Control Avanzado del DIE por siempre haber presentado una buena disposición a ayudarme a refinar los contenidos de este trabajo, y en particular a mi profesora guía Doris Sáez Hueichapán por preocuparse de mi trabajo y apoyarme durante estos años de estallidos y pandemias.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Antecedentes generales

1.1.1. Fundamentos del modelamiento de intervalos

Para comprender la utilidad y aplicabilidad de los modelos de intervalo, es necesario definir algunos conceptos fundamentales primero. En cuanto al modelamiento de incerteza en modelos predictivos, la teoría asume que todas las observaciones son generadas por una función generadora de datos $f(x)$ combinada con un ruido aditivo [1, 2, 3].

$$y = f(x) + \varepsilon, \tag{1.1}$$

donde ε es una variable aleatoria con media cero, también conocida como ruido de los datos. Esta variable es responsable de introducir incertidumbre en los modelos predictivos en la forma de incerteza aleatoria, que corresponde a toda incerteza originada por la exclusión de variables complejas del modelo que no pueden ser estimadas con suficiente precisión, o bien debido a la presencia de procesos inherentemente estocásticos en el sistema observado.

Basándose en esta formulación, el objetivo de los modelos predictivos se reduce a intentar producir un estimador $\hat{f}(x)$ de la función generadora de datos original, para así calcular predicciones del valor esperado del sistema (ya que no toman en cuenta la incerteza aleatoria aportada por el término ε). Este procedimiento introduce un tipo adicional de incerteza, conocida como incerteza epistémica, producto de que el estimador $\hat{f}(x)$ es sólo una aproximación de la función generadora de datos. Asumiendo que ambos tipos de incerteza son independientes, la varianza total del error en la predicción puede ser expresada como

$$\sigma^2_{total} = \sigma^2_{model} + \sigma^2_{data}, \tag{1.2}$$

donde σ^2_{model} se atribuye a la incerteza epistémica y σ^2_{data} a la incerteza aleatoria.

Debido la diversidad de factores que pueden contribuir a la incerteza epistémica, existen autores en la literatura [4] que han propuesto una subclasificación para este término:

1. **Error de especificación o sesgo:** Incerteza determinada por la precisión del estimador $\hat{f}(x)$ para aproximar la función generadora de datos $f(x)$ bajo condiciones óptimas de parámetros y datos.

2. **Incerteza de los datos:** Incerteza relacionada con la representatividad de los datos de entrenamiento con respecto a la distribución total de entradas y la sensibilidad del modelo a muestras nunca antes vistas.

3. **Incerteza paramétrica:** Incerteza relacionada con el proceso de optimización paramétrica del modelo. Puede ser producida por fenómenos como estancamiento debido a mínimos locales, o una finalización prematura del algoritmo.

Es importante notar que, si bien la clasificación de los tipos de incerteza es simple, no ocurre lo mismo al intentar estimar su valor. Esto último se debe a que la incerteza puede provenir de fuentes muy diversas actuando simultáneamente, por lo que la expresión total de este término suele ser altamente compleja.

Otro punto que es necesario aclarar dentro del contexto del modelamiento de intervalos es la diferenciación entre intervalos de predicción e intervalos de confianza. Estrictamente hablando, el estimador $\hat{f}(x)$ genera predicciones de la función generadora $f(x)$, que corresponde al valor esperado real de los datos y, como se puede apreciar en (1.1). Un intervalo de confianza (IC) calcula, a partir de los datos observados, un rango estimado de valores donde se podría encontrar el valor esperado real $f(x)$, según una cierta probabilidad. En este sentido, el IC sólo es capaz de cuantificar la incerteza de $\hat{f}(x)$ como estimador de $f(x)$, es decir, la incerteza epistémica. Por otro lado, un intervalo de predicción (IP) cuantifica la incerteza de utilizar $\hat{f}(x)$ para predecir y, por lo que define un rango de valores en donde se podrían encontrar las observaciones y, según una cierta probabilidad. Debido a esto, el IP debe tomar en cuenta tanto la incerteza epistémica como la aleatoria.

Según los conceptos definidos en el párrafo anterior, se podría representar a un IC como una estructura basada en la probabilidad de encontrar el valor esperado de una observación dentro de la vecindad de un estimador $P(f(x)|\hat{f}(x))$, mientras que un IP se encuentra basado en la probabilidad de encontrar el valor real de una observación dentro de este rango $P(y|\hat{f}(x))$. A partir de esta formulación, resulta sencillo notar que los IP necesariamente deben ser igual o más anchos que los IC. Teniendo esto en mente, este trabajo se ha enfocado en el desarrollo de intervalos de predicción debido a que son la estructura que mejor soluciona el problema de cuantificación de incerteza para las predicciones a futuro de un modelo predictivo.

En cuanto a la razón detrás de la utilización de modelos de intervalo en vez de otras técnicas de cuantificación de incerteza, esto se debe a que los IP son una herramienta práctica en el sentido de que pueden expresar la incerteza de un modelo predictivo usando el mínimo de información, sólo requiriendo tres valores: Un límite superior, un límite inferior, y la probabilidad de cobertura. Esto contrasta con otras técnicas como el uso de funciones de densidad de probabilidad (FDP), las cuales si bien contienen información exacta sobre la incerteza, sólo una pequeña familia de distribuciones pueden ser descritas con pocos parámetros. Para casos generales, las distribuciones de probabilidad deben ser aproximadas utilizando una muestra grande de puntos de la función. Es por esto que los IP fueron elegidos como el mecanismo de cuantificación de incerteza más entendible, que a la vez minimizaba el uso de parámetros.

Debido al efecto de acoplamiento de las distintas fuentes de incerteza de un modelo, es usual que el ancho promedio de los intervalos de predicción aumente a medida que aumenta el horizonte de predicción. El correcto modelamiento de este incremento de la zona de incerteza

es crucial en algunas aplicaciones, tales como la aplicación de estrategias de control predictivo o sistemas de detección de fallas, donde las decisiones del modelo son tomadas basándose en las predicciones a futuro. Debido a esto, el principal objetivo de los modelos de intervalo es maximizar su informatividad. Esto se logra mediante la obtención de un intervalo lo más delgado posible, pero que a la vez no contenga menos de un cierto porcentaje (previamente fijado) de los datos.

En este contexto, en la literatura se han utilizado un conjunto de métricas específicas para validar y comparar la efectividad de modelos de intervalo. Por un lado, la raíz del error cuadrático medio (RMSE por sus siglas en inglés) se ha empleado para medir la precisión del modelo predictivo para el cálculo de predicciones puntuales. Esta métrica se puede calcular según

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y(k) - \hat{y}(z_k))^2}, \tag{1.3}$$

donde N representa el número total de datos, $y(k)$ es la k-ésima muestra de la base de datos, y $\hat{y}(z_k)$ es la predicción del modelo para la muestra de entrada z_k. Por el otro lado, en [5, 3] se proponen las métricas *Prediction Interval Coverage Probability* (PICP) y *Prediction Interval Normailzed Average Width* (PINAW) para medir el desempeño del porcentaje de cobertura de datos y ancho promedio de modelos de intervalo, respectivamente. Estos indicadores pueden ser calculados según

$$\text{PINAW} = \frac{1}{NR}\sum_{k=1}^{N}\left(\overline{y}(z_k) - \underline{y}(z_k)\right), \tag{1.4}$$

donde

$$R = \max_k\{y(k)\} - \min_k\{y(k)\} \tag{1.5}$$

es el rango de valores del conjunto de datos de entrenamiento, y

$$\text{PICP} = \frac{1}{N}\sum_{k=1}^{N} c_k, \tag{1.6}$$

donde

$$c_k = \begin{cases} 1, & \text{if } \underline{y}(z_k) \leq y_k \leq \overline{y}(z_k) \\ 0, & \text{otherwise} \end{cases} \tag{1.7}$$

indica si el intervalo dado por los límites $[\underline{y}(z_k), \overline{y}(z_k)]$ contiene la observación $y(k)$.

1.1.2. Redes neuronales y *Deep Learning* para el modelamiento de sistemas dinámicos

El uso de arquitecturas basadas redes neuronales para el modelamiento de sistemas y series de tiempo ha proliferado en la literatura recientemente. En parte, esta popularidad se debe a la propiedad de estos algoritmos de ser aproximadores universales [6], la cual hace a las redes neuronales una solución atractiva para aplicaciones que presentan dinámicas no

lineales [7, 8, 9] (suponiendo que se cuenta con una base de datos representativa del sistema). Entre estas aplicaciones, existen trabajos en la literatura que reportan que los modelos predictivos estadísticos, entre los cuales se encuentran las redes neuronales, presentan un mejor desempeño en algunas aplicaciones de pronóstico de variables climáticas, en comparación a las predicciones hechas por modelos fenomenológicos [10].

Históricamente, el modelamiento neuronal es un área que se ha investigado desde al menos la década de los 60s [11]. A partir de este periodo, se han desarrollado continuamente nuevas arquitecturas para resolver distintos tipos de problemas. En el contexto del modelamiento de sistemas dinámicos, el modelo con mayor trayectoria corresponde al de la red neuronal feedforward con una capa oculta. Matemáticamente, su salida puede ser calculada según

$$\hat{y}(z_k) = \sum_{j=1}^{N_h} w_j \left(f\left(\sum_{\mathrm{i}=1}^{n_z} w_{j\mathrm{i}} z_k^{(\mathrm{i})} + b_j \right) \right) + b, \tag{1.8}$$

donde $\hat{y}(z_k)$ es la predicción del modelo para la muestra de entrada z_k, N_h corresponde al número de neuronas en la capa oculta de la red, w_j es el peso que conecta la j-ésima neurona de la capa oculta con la salida del modelo, $f(\cdot)$ corresponde a una función de activación no lineal (usualmente se ocupa la función tangente hiperbólica tanh o la función sigmoide), n_z corresponde al tamaño del vector de entradas del modelo, $w_{j\mathrm{i}}$ es el peso que conecta la i-ésima neurona de la capa de entrada con la j-ésima neurona en la capa oculta, b_j es el sesgo asociado a la j-ésima neurona en la capa oculta, b es el sesgo asociado a la salida del modelo, y $z_k = [z_k^{(1)}, ..., z_k^{(n_z)}]$ es el vector de entradas del modelo, el cual está compuesto por observaciones pasadas del sistema $y(k)$ y, en algunos casos, de variables exógenas $u(k)$

$$\begin{aligned} z_k = \; & [y(k-1), y(k-2), \ldots, y(k-n_y), \\ & u(k-1), u(k-2), \ldots, u(k-n_u)]^T . \end{aligned} \tag{1.9}$$

Posterior al desarrollo de esta arquitectura, la introducción de las redes recurrentes en la década de los 80s [12] permitió obtener grandes avances en la capacidad predictiva de los modelos neuronales. Esto se debe a que las redes recurrentes corresponden a una variante especial en donde las neuronas se encuentran reorganizadas para formar capas secuenciales, de manera que la red pueda simular tener un comportamiento dinámico en el tiempo.

Matemáticamente, esta reestructuración significa que las arquitecturas recurrentes no requieren el uso de entradas en la forma de una lista de regresores del sistema a predecir. En cambio, estos modelos sólo reciben como entrada el valor inmediatamente anterior en la secuencia. Para lograr que el modelo guarde un registro o memoria del comportamiento histórico de esta, las redes recurrentes utilizan un vector de parámetros oculto que contiene una codificación del estado actual de la secuencia. A partir de lo anterior, es posible obtener predicciones a partir del valor inmediatamente anterior en la secuencia y el valor actual del vector de estado, el cual entrega información del contexto en el cual fue recibida la entrada. De igual manera, una vez que estos modelos calculan una predicción, es necesario actualizar el valor del vector de estado para realizar predicciones adicionales. A modo de ejemplo, en la figura 1.1 se muestra un diagrama de la arquitectura general de una red neuronal recurrente,

donde x_i corresponde a entrada del modelo en el instante i-ésimo, h_i corresponde al vector de estado en el instante i-ésimo, e y_i corresponde a la predicción del modelo en el instante i-ésimo.

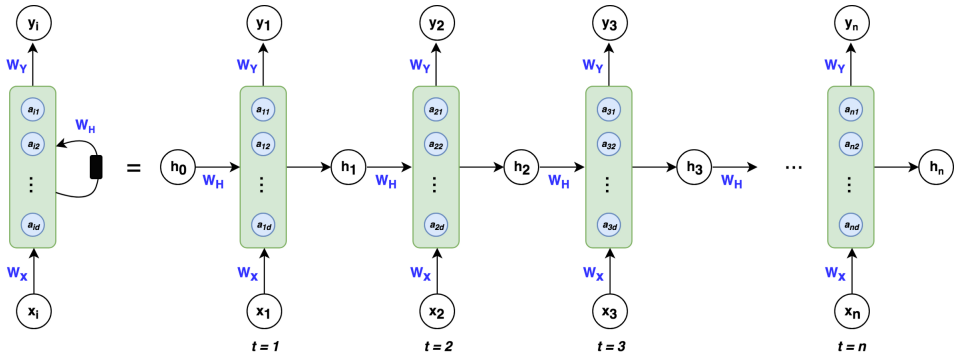


Figura 1.1: Diagrama de la arquitectura general de una red neuronal recurrente [13]

La principal ventaja de la utilización de arquitecturas recurrentes en comparación a las redes *feedforward* radica en que, debido al uso de los vectores de estado, las primeras no requieren de la ejecución de algoritmos de optimización estructural para seleccionar el número de regresores a utilizar como entrada para el modelo, sino que este proceso se encuentra integrado dentro del entrenamiento de la red. Al eliminar el factor de la intervención humana de este procedimiento, es posible obtener modelos con un menor error de predicción que las arquitecturas *feedforward* tradicionales, suponiendo que se cuenta con una base de datos suficientemente grande y representativa del sistema a predecir. Esto último resulta crucial al intentar aplicar herramientas de *Deep Learning*.

El *Deep Learning* consiste en una serie de técnicas utilizadas en el modelamiento neuronal para aumentar la capacidad de cómputo de estas arquitecturas. La principal diferencia entre estos modelos y aquellos basados en redes *feedforward* tradicionales corresponde a que los algoritmos de *Deep Learning* se encuentran diseñados para permitir la utilización de modelos con un mayor número de capas y parámetros, como se puede apreciar en la figura 1.2. En los últimos años, estos modelos han crecido exponencialmente en popularidad debido a su capacidad de obtener resultados que previamente se creían imposibles para una computadora, permitiendo el desarrollo de tecnologías como la visión computacional aplicada a la robótica y vehículos autónomos, el procesamiento de lenguaje natural para el desarrollo de asistentes virtuales, e incluso la generación artificial de fotografías y videos realistas. Sin embargo, debido al gran número de parámetros de las arquitecturas utilizadas, las ventajas del *Deep Learning* sólo son aprovechables en aplicaciones donde se disponga de una gran cantidad de datos de entrenamiento y recursos computacionales.

La necesidad de recurrir a técnicas especiales para crear modelos con más capas se debe a que, tras la realización de experimentos con redes neuronales tradicionales, se demostró que cuando estos modelos superan un cierto umbral de tamaño, el entrenamiento por *backpropagation* no logra converger correctamente debido a que el gradiente en las capas más profundas de la red desvanece a cero, o bien diverge al infinito. A estos fenómenos se les conoce como los problemas de desvanecimiento y explosión de gradiente, respectivamente [15]. Además, es importante mencionar que, debido al número de parámetros que poseen estos modelos, no es factible realizar su entrenamiento mediante optimizadores no lineales, como lo son los

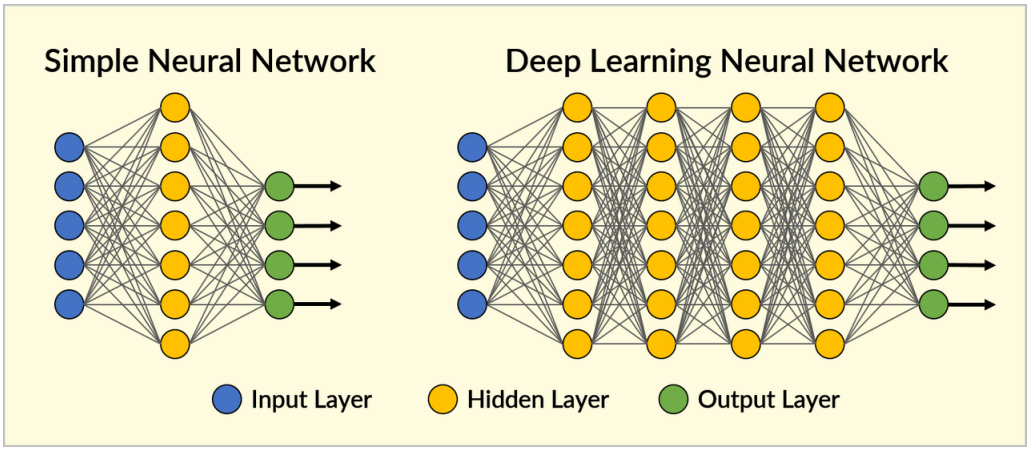


Figura 1.2: Diagrama comparativo de la arquitectura general de una red neuronal tradicional (izquierda) y una red *Deep Learning* (derecha) [14]

algoritmos evolutivos. Debido a lo anterior, el entrenamiento de modelos de *Deep Learning* requiere el uso de arquitecturas y optimizadores especializados.

Dentro de las arquitecturas compatibles con el *Deep Learning* se pueden encontrar aquellas que hacen un uso eficiente de sus parámetros y no requieren de la intervención humana para realizar procesos de selección de características, tales como las redes convolucionales para el procesamiento de imágenes y las redes recurrentes para el procesamiento de series de tiempo. Esto se debe a que los modelos a utilizar deben presentar costos computacionales bajos, de manera que no escalen en gran medida al aumentar el número de capas de la red. Además, dado que estas técnicas poseen una naturaleza de caja negra, es difícil descifrar su funcionamiento interno, por lo que el impacto de la intervención humana en los procesos de selección de características está sujeto fuertemente al conocimiento a priori que se tiene del sistema donde se desea aplicar el modelo. En cambio, al delegar esta tarea al modelo e incorporarla dentro de la metodología de identificación de parámetros es posible aumentar el campo de aplicabilidad de los modelos neuronales a procesos de los que no se posee tanto conocimiento sin impactar en grandes medidas el desempeño de estos algoritmos.

Con respecto a los optimizadores compatibles con el *Deep Learning*, en esta categoría se puede encontrar a todos aquellos algoritmos basados en métodos de gradiente descendente estocástico [16]. Esta familia de optimizadores es utilizada principalmente debido a que permite aprovechar el reducido costo computacional de los métodos basados en gradiente descendente, pero a la vez son suficientemente robustos como para poder evitar algunos mínimos locales y discontinuidades y así obtener resultados más cercanos a un óptimo global. Si bien existen varias variantes de este algoritmo, actualmente el método Adam (*Adaptive Moment Estimation*) [17] es uno de los más populares debido a su versatilidad.

Sumado a lo anterior, otra familia de técnicas también usada frecuentemente para mejorar el desempeño de los modelos predictivos neuronales corresponde a los métodos de conjunto. Estos permiten mejorar el desempeño de un modelo al entrenar varios modelos a la vez. Si bien existen muchas variantes, el método que resulta de mayor interés para el modelamiento de intervalos es la técnica del *bootstrapping* [18]. Este tipo de conjunto se construye mediante el entrenamiento de varios modelos neuronales en paralelo utilizando una submuestra

aleatoria de los datos totales como conjunto de entrenamiento. Luego, una vez entrenados los modelos, la predicción total del conjunto se puede obtener calculando el promedio de todas las predicciones individuales. Además, a diferencia de otros métodos de conjunto, el *bootstrapping* permite obtener una estimación de la incerteza de la predicción del conjunto mediante el cálculo de su desviación estándar, lo cual hace a estos modelos llamativos para el modelamiento de intervalos. Esto se puede evidenciar en que distintas variantes de *bootstrapping* se han utilizado en diversas aplicaciones en la literatura, tales como control de sistemas de transporte público [19], control de procesos industriales [20] y, de particular interés para este trabajo, el pronóstico de variables climáticas y demanda eléctrica [21, 22, 23]

1.1.3. Modelos de intervalos neuronales y Método de *Joint Supervision*

Si bien las redes neuronales presentan un desempeño adecuado en aplicaciones de predicción de sistemas dinámicos, los enfoques tradicionales no se preocupan de la cuantificación de la incerteza de la predicción. Esto último dificulta la utilización de estas arquitecturas para aplicaciones dedicadas a la toma de decisiones, puesto que para esto es imprescindible contar con información de la precisión de la salida del modelo.

En este contexto, los intervalos de predicción han sido propuestos para solucionar este problema. Estos intervalos proporcionan un rango alrededor de la salida del modelo, el cual representa numéricamente la incerteza presente en la predicción. De esta manera, los modelos de intervalo pueden proveer información que permita considerar múltiples escenarios dentro de un sistema de toma de decisiones.

Dentro de los distintos métodos de construcción de intervalos de predicción neuronales existentes en la literatura, el método de *Joint Supervision* destaca por su capacidad de obtener resultados del estado del arte tanto para el error de predicción como para el ancho promedio del intervalo, lo anterior sin la necesidad de realizar ningún supuesto sobre el comportamiento de la incerteza del sistema a predecir. Si bien el funcionamiento de este método se explica detalladamente en la sección 2.2.2, en resumen el modelo *Joint Supervision* consiste en una red neuronal con tres neuronas en la capa de salida, donde dos son responsables de predecir los límites superior e inferior del intervalo, y una de calcular las predicciones puntuales del modelo. Para lograr entrenar esta arquitectura, el método utiliza una metodología de identificación de parámetros basada en la optimización de una función de costos por medio del algoritmo de *backpropagation*, sumado a un proceso de ajuste hiperparamétrico en donde el modelo es entrenado reiteradas veces mediante una búsqueda logarítmica.

Actualmente, el método de *Joint Supervision* ha encontrado éxito en la literatura en aplicaciones de pronóstico de demanda eléctrica [24, 25]. Sin embargo, su compatibilidad con otros casos de estudio se encuentra afectada por sus dos principales desventajas. En primer lugar, la necesidad de reentrenar el modelo reiteradamente dentro de su proceso de ajuste hiperparamétrico perjudica su costo computacional, limitando su aplicabilidad para usos que requieran de un entrenamiento en línea. En segundo lugar, la ejecución de la búsqueda logarítmica también aporta problemas de convergencia al entrenamiento del método. Esto se debe a que el entrenamiento de redes neuronales no es un proceso determinístico, por lo que es posible que durante el ajuste hiperparamétrico el algoritmo converja a un resultado

subóptimo debido al ruido introducido por la estocacidad inherente del modelo base.

Es importante notar que el diseño de modelos de intervalo no se limita únicamente a modelos basados en redes neuronales, sino que estas estructuras pueden ser desarrolladas para prácticamente cualquier tipo de modelo predictivo. Sin embargo, dentro del nicho de los modelos basados en inteligencia computacional, destacan junto a las redes neuronales los modelos de intervalo basados en sistemas difusos Takagi-Sugeno. Esto se debe a que comparten gran parte de las ventajas y son capaces de construir intervalos de calidad similar o superior que los modelos neuronales, junto con tener la ventaja adicional de ser modelos más interpretables. Debido a esto, los modelos de intervalo difusos resultan un buen punto de comparación para los modelos de intervalo neuronales. En efecto, es posible corroborar en la literatura la existencia de metodologías de diseño de modelos de intervalo que son aplicables tanto para arquitecturas neuronales y difusas [26], aunque este no es el caso del método de *Joint Supervision*.

1.2. Motivación

A lo largo de los últimos años, las micro-redes han sido un tema de investigación cuya popularidad ha ido en constante aumento. Las principales razones de este fenómeno se pueden atribuir a que estos sistemas prometen ventajas económicas mediante una disminución de los costos de transmisión de energía [27], y además proponen una solución factible para la energización de comunidades rurales y/o aisladas mediante la utilización de recursos energéticos propios de la zona [28].

No obstante, a pesar de sus ventajas, la implementación de micro-redes presenta una serie de desafíos distintos a los de un sistema de distribución energético convencional. El más importante de estos corresponde al carácter altamente estocástico de los recursos y la demanda energética propio de las comunidades pequeñas. Debido a esto, la operación de estos sistemas se debe tratar como un problema de toma de decisiones, donde dependiendo de la disponibilidad esperada a futuro de los recursos energéticos (por ejemplo, radiación solar o velocidad del viento) y la predicción de demanda energética, el operador o algoritmo debe optar por utilizar fuentes de energía renovables, o bien fuentes alternativas basadas en diésel.

Como se mencionó en la sección anterior, esta familia de problemas no pueden ser resueltos por medio de modelos predictivos tradicionales, puesto que la toma de decisiones requiere, además de predicciones puntuales, de una cuantificación precisa de la incerteza de la predicción del modelo. A estos antecedentes se les suma que, para el caso de la predicción de recursos energéticos para la operación de micro-redes, existe una preferencia por la utilización de modelos estadísticos que sean compatibles con un entrenamiento en línea. Esto último se debe a que el comportamiento estacional del clima hace que la disponibilidad de los recursos energéticos manifieste una periodicidad larga, por lo que es difícil contar con una base de datos suficientemente representativa como para que el modelo pueda capturar estas dinámicas en el primer entrenamiento.

Es por esto que resulta de interés estudiar la posibilidad de diseñar modelos de intervalo precisos para la predicción climática y de demanda eléctrica por medio de modelos basados en redes neuronales, considerando que su factibilidad para la predicción de variables climáticas a

corto plazo ya se encuentra demostrada [10]. En este contexto, las técnicas de *Deep Learning* y *bootstrapping* surgen como herramientas llamativas, considerando que ambas consistentemente han permitido mejorar el desempeño de modelos neuronales tradicionales.

Sin embargo, la utilización de estas herramientas impone una serie de exigencias en las características del modelo: En primer lugar, el uso de técnicas de *Deep Learning* restringe el universo de modelos factibles a sólo aquellos que sean compatibles con optimizadores basados en métodos de gradiente, y que además no introduzcan muchos parámetros adicionales. En segundo lugar, la utilización de métodos de conjunto y aprendizaje en línea pone un límite máximo al costo computacional de entrenamiento. Tomando en cuenta estas restricciones, el método de *Joint Supervision* podría ser considerado un candidato factible para solucionar este problema, pero, como se mencionó en la sección anterior, sus desventajas en cuanto a costo computacional y convergencia del entrenamiento limitan su aplicabilidad. Por lo tanto, existe un interés por desarrollar modelos de intervalo basados en *Joint Supervision* que no compartan estas limitaciones.

1.3. Hipótesis

Para la realización del presente trabajo de investigación se han considerado las siguientes hipótesis:

- Es posible desarrollar modelos de intervalo basados en redes neuronales para la predicción de sistemas dinámicos, que además sean compatibles con métodos de optimización de gradiente, y posean un bajo costo computacional.
- Es posible mejorar la precisión de los modelos de intervalo neuronales por medio del *Deep Learning* y el *bootstrapping*
- Es posible diseñar métodos de intervalo neuronal basados en *Joint Supervision* con un costo computacional suficientemente pequeño como para implementar modelos basados en *Deep Learning*.

1.4. Objetivos

1.4.1. Objetivo general

Diseñar una metodología para la construcción de intervalos de predicción a partir de modelos neuronales y el método de *Joint Supervision*, que además realice la menor cantidad de supuestos posibles sobre el comportamiento de la incerteza y presente características que la hagan compatible con técnicas de *Deep Learning*, métodos de conjunto, y entrenamiento en línea.

1.4.2. Objetivos específicos

1. Realizar una revisión de literatura de los modelos de intervalo neuronal existentes y estudiar sus supuestos, costo computacional, número de parámetros, y compatibilidad con métodos de optimización de gradiente.

2. Diseñar un modelo de intervalo neuronal con *Deep Learning* basado en *Joint Supervision* que no realice supuestos sobre el comportamiento de la incerteza del sistema, sea compatible con métodos de optimización de gradiente, y posea un bajo costo computacional y número de parámetros
3. Diseñar una extensión del método de *Joint Supervision* basado en redes neuronales para hacerlo compatible con arquitecturas difusas Takagi-Sugeno
4. Validar y evaluar el impacto del modelo neuronal propuesto mediante simulaciones experimentales utilizando una base de datos experimental y una base de datos climáticos reales, comparando el costo computacional, el error de predicción, y el ancho y porcentaje de cobertura del intervalo entre el método de intervalo neuronal diseñado, el método de *Joint Supervision* basado en redes neuronales, y su extensión basada en sistemas difusos Takagi-Sugeno.

1.5. Publicaciones

1.5.1. Publicaciones de revistas

S. Parra, D. Sáez. "Deep Learning Prediction Intervals Based on Selective Joint Supervision". *En preparación para envío.*

O. Cartagena, **S. Parra**, D. Muñoz-Carpintero, L. G. Marín, D. Sáez. Review on Fuzzy and Neural Prediction Interval Modelling for Nonlinear Dynamical Systems", *IEEE Access*, vol. 9, pp. 23357-23384, 2021. https://doi.org/10.1109/ACCESS.2021.3056003

1.5.2. Publicaciones de conferencia

D. Muñoz-Carpintero, **S. Parra**, O. Cartagena, D. Sáez, L. G. Marín, I. Škrjanc. "Fuzzy Interval Modelling based on Joint Supervision", *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Glasgow, UK, 2020, pp. 1-8.
https://doi.org/10.1109/FUZZ48607.2020.9177779

1.6. Estructura del trabajo

La estructura del presente trabajo de tesis se encuentra dividida en 4 capítulos más una sección de conclusiones, descritos brevemente a continuación.

En primer lugar, en el Capítulo 1 se plantean los principales antecedentes que permiten contextualizar la propuesta que se desarrolla en este documento, junto con la motivación que justifica su relevancia, los objetivos generales y específicos del trabajo, y finalmente la metodología propuesta para su ejecución.

Posteriormente, en el Capítulo 2 se presenta una revisión de literatura en el tópico de modelamiento de intervalos basados en redes neuronales, especificando los fundamentos, variantes, y aplicaciones de cada método. Adicionalmente, este capítulo también incluye un análisis comparativo entre todos los métodos encontrados.

Luego, en el Capítulo 3 se procede a explicar las propuestas planteadas en este trabajo, describiendo detalladamente la fundamentación teórica y la metodología de entrenamiento de cada uno de los modelos.

A continuación, en el Capítulo 4 se presentan los montajes experimentales que fueron utilizados para validar el desempeño de los modelos propuestos, especificando las características de los datos y la configuración paramétrica utilizada en cada caso de estudio.

Finalmente, en la última sección de este documento se exponen las principales conclusiones del trabajo, destacando las contribuciones realizadas a la línea de investigación y proponiendo trabajo a futuro para continuar con su desarrollo.

Capítulo 2

Revisión de Literatura

Considerando que en la literatura existe una amplia variedad de métodos para construir intervalos de predicción a partir de modelos neuronales, existen autores [29]que han propuesto una clasificación que separa a estos métodos según cómo se entrenan.

Según este criterio, se pueden reconocer dos categorías: Métodos Secuenciales, y Métodos Directos. En primer lugar, los métodos Secuenciales son aquellos que utilizan como base un estimador $\hat{f}(x)$ previamente entrenado, y luego construyen un intervalo de predicción por medio de pasos adicionales. En segundo lugar, los métodos Directos corresponden a aquellos que realizan la identificación del intervalo en conjunto con el entrenamiento del modelo predictivo.

A continuación, se procederá a describir los métodos de construcción de intervalos existentes en la literatura para cada una de las categorías previamente mencionadas.

2.1. Métodos secuenciales

2.1.1. Método Delta

El método Delta propone una estrategia fundamentada en la teoría de regresión no-lineal clásica para obtener una aproximación de la incerteza de la predicción en redes neuronales *feedforward* con una capa oculta [30]. Para lograr esto, el método se basa en el cálculo de la expansión en serie de Taylor de la función de error cuadrático medio, la cual es usada como función de costo para entrenar a la red neuronal. Considerando esto, y asumiendo además que la varianza del error de predicción es homogénea y normalmente distribuida, la varianza de la incerteza total para cada predicción se puede calcular como

$$\sigma^2_{tot}(z_k) = \sigma^2_{\varepsilon}(1 + g_0^T(J^T(z_k)J(z_k))^{-1}g_0), \tag{2.1}$$

donde g_0 corresponde a la matriz de parámetros del modelo neuronal, $J(z_k)$ representa la matriz Jacobiana de la red evaluada en la muestra de entrada z_k, y σ_{ε} es la varianza del ruido

de los datos, la cual puede ser calculada a partir de

$$\sigma_{\varepsilon}^2 = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}(z_k))^2, \tag{2.2}$$

donde y_k representa la k-ésima muestra de entrenamiento, y $\hat{y}(z_k)$ es la predicción del modelo para la muestra de entrada z_k.

Después de calcular este valor, el intervalo de predicción de $(1-\alpha)\,\%$ de confianza puede construirse, asumiendo que es simétrico y Gaussiano, como

$$\overline{y}(z_k) = \hat{y}(z_k) + t_{n-p}^{1-\frac{\alpha}{2}}\sigma_{tot}(z_k) \tag{2.3}$$

$$\underline{y}(z_k) = \hat{y}(z_k) - t_{n-p}^{1-\frac{\alpha}{2}}\sigma_{tot}(z_k), \tag{2.4}$$

donde n es el número de muestras de entrada que fueron utilizadas para entrenar el modelo, p es el número de parámetros del modelo neuronal, y $t_{n-p}^{1-\frac{\alpha}{2}}$ representa el cuantil $\frac{\alpha}{2}$-ésimo de la función de distribución acumulada de una distribución t de Student con $n-p$ grados de libertad.

Debido a depender del supuesto de incerteza homogénea y Gaussiana, este método puede fallar en producir intervalos de calidad para aplicaciones donde el ruido de los datos sea variable en el tiempo. Además, debido a que este método requiere el cálculo de la matriz Jacobiana del modelo, su cálculo puede ser computacionalmente costoso para modelos que presenten una gran cantidad de parámetros.

Este método ha sido aplicado en la literatura en predicción de carga eléctrica [30], monitoreo del proceso de deposición de pasta de soldar [31], predicción de temperatura y humedad relativa [32], monitoreo de calidad de agua [33], y predicción de tiempos de viaje [34].

Finalmente, existen variantes de estos métodos que se han reportado en la literatura para mejorar su desempeño. En primer lugar, en [35], los autores proponen la inclusión de un factor de decaimiento de peso dentro de la función de costos de la red para prevenir el sobreajuste. Por otro lado, en [36] se propone un modelo de intervalo basado en este método, pero donde el entrenamiento de la red neuronal se basa en la optimización de las métricas PICP y PINAW definidas en las ecuaciones (1.4) y (1.6), respectivamente. Por último, en [37] se propone una variante con una nueva función de costos, la cual intenta minimizar el ancho del intervalo y a la vez mantener los parámetros del modelo lo más cerca posible a los valores que no violan los supuestos del método Delta. Sin embargo, a pesar de las mejoras en desempeño mostradas por estas variantes, las principales limitaciones de este método persisten.

2.1.2. Método Bayesiano

El método Bayesiano se basa en tratar el modelo predictivo neuronal como un estimador Bayesiano [38, 39, 40]. Esto implica que los parámetros de la red neuronal son modelados como variables aleatorias en vez de puntos fijos. De esta manera, la incerteza de la predicción, y consecuentemente los intervalos de predicción, puede ser derivada intuitivamente a partir de la distribución probabilística de los parámetros del modelo.

Para optimizar los pesos de la red, el método Bayesiano intenta estimar una distribución posterior de los parámetros w dado un conjunto de datos de entrenamiento z_k, representada por $p(w|z_k)$, por medio de la aplicación del Teorema de Bayes. Utilizando esta estrategia, es posible calcular una expresión que es proporcional a la distribución posterior deseada, a partir de la cual se pueden aplicar métodos Monte Carlo basados en cadenas de Markov (MCMC por sus siglas en inglés) para obtener un valor aproximado de la distribución posterior real.

Una vez que se logra determinar una aproximación a la distribución posterior, la varianza del error total de predicción puede ser calculada según

$$\sigma^2_{tot}(z_k) = \sigma^2_{\varepsilon} + \sigma^2_{w^{MP}}(z_k) \tag{2.5}$$

$$= \frac{1}{\beta} + \nabla \hat{y}^T_{w^{MP}}(z_k)(H^{MP})^{-1}\nabla \hat{y}_{w^{MP}}(z_k), \tag{2.6}$$

donde σ^2_{ε} es la varianza del ruido de los datos, la cual se puede obtener a partir de la ecuación (2.2), w^{MP} representa la estimación Máximo a Posteriori de los pesos de la red, $\nabla \hat{y}_{w^{MP}}(z_k)$ es el gradiente de la red con respecto a los parámetros w^{MP} evaluado en la muestra de entrada z_k, y H^{MP} es la matriz Hessiana del modelo.

Finamente, si se especifica un porcentaje de cobertura deseado, es posible construir un intervalo de predicción calculando los cuantiles correspondentes de la distribución posterior [40, 3]

$$\overline{y}(z_k) = \hat{y}(z_k) + q^{1-\frac{\alpha}{2}}\sigma_{tot}(z_k) \tag{2.7}$$

$$\underline{y}(z_k) = \hat{y}(z_k) - q^{1-\frac{\alpha}{2}}\sigma_{tot}(z_k), \tag{2.8}$$

donde $q^{1-\frac{\alpha}{2}}$ es el $1-\frac{\alpha}{2}$-ésimo cuantil de la distribución posterior normalizada.

A pesar de haber reportado una mejor calidad de intervalos que el Método Delta y contar con una fundamentación matemática respaldada por la estadística Bayesiana, la necesidad de calcular la matriz Hessiana tiene como consecuencia un aumento considerable del costo computacional, lo cual impide su aplicabilidad a problemas reales. Esto último se refleja en el bajo número de aplicaciones del método reportadas en literatura, las cuales se resumen principalmente en problemas de predicción de carga eléctrica [41], predicción de evapotranspiración potencial [42], y estimación de tiempos de viaje [34].

2.1.3. Método de Estimación de Media y Varianza

El método de Estimación de Media y Varianza, comúnmente abreviado en la literatura como MVEM por sus siglas en inglés, propone que un IP puede ser construido a partir del entrenamiento de dos modelos neuronales por separado: Uno encargado de estimar las predicciones puntuales (NN_y), y otro encargado de estimar la varianza del error de predicción (NN_σ) [43].

La metododología utilizada para entrenar estos modelos se separa en 2 etapas. En primer lugar, se debe entrenar la red NN_y utilizando una función de costos de error cuadrático medio, como se suele proceder con un modelo predictivo neuronal tradicional. Luego, la red NN_σ se entrena para estimar la varianza del error σ^2_k entre cada predicción $\hat{y}(z_k)$ y su respectiva

muestra de entrenamiento y_k. Para lograr esto, se considera una función de costos de log-verosimilitud

$$C(y, \hat{y}) = \frac{1}{2} \sum_{k=1}^{N} \left(\log(\hat{\sigma}^2(z_k)) + \frac{(y_k - \hat{y}(z_k))^2}{\hat{\sigma}^2(z_k)} \right) \tag{2.9}$$

Finalmente, luego de obtener los predictores $\hat{\sigma}^2(z_k)$ y $\hat{y}(z_k)$, es posible construir un IP utilizando la misma metodología que el Método Bayesiano:

$$\overline{y}(z_k) = \hat{y}(z_k) + q^{1-\frac{\alpha}{2}} \hat{\sigma}(z_k) \tag{2.10}$$

$$\underline{y}(z_k) = \hat{y}(z_k) - q^{1-\frac{\alpha}{2}} \hat{\sigma}(z_k) \tag{2.11}$$

Debido a que este método estima la incerteza de la predicción a partir de la salida de un modelo neuronal, tiene un menor costo computacional que los Métodos Delta (si es que el modelo es paramétricamente denso) y Bayesiano. A su vez, el MVEM también es capaz de adaptarse a incertezas de comportamiento complejo gracias a la capacidad aproximadora universal de las redes neuronales.

Sin embargo, la principal desventaja de este método radica en la suposición de que el modelo NN_y estima con perfección el verdadero valor esperado de los datos y_k. Esto se debe a que la función de costos de log-verosimilitud propuesta en la ecuación (2.9) asume que el error de predicción se encuentra normalmente distribuido con media en $\hat{y}(z_k)$, lo cual significa que el modelo NN_y está siendo tratando como un estimador insesgado del valor esperado de los datos reales y_k.

Este método ha sido aplicado en la literatura para resolver problemas de predicción de generación de potencia eólica [44], pronóstico de tipo de cambio [45], predicción de vida útil restante de componentes industriales [46], pronóstico de índices del mercado de valores [47], y estimación de desplazamientos por deslizamiento de tierra [47].

Finalmente, existen variantes de este método en la literatura que han intentado mejorar su desempeño. En primer lugar, en [44] se propone una nueva técnica para entrenar la red encargada de predecir la varianza del error de predicción NN_σ, la cual consiste en una sintonización fina de los pesos del modelo utilizando una función de costos *Coverage Width-based Criterion* (CWC). Este costo tiene el objetivo de incorporar el objetivo de la minimización del ancho del intervalo dentro del proceso de optimización, y puede ser calculado como

$$\text{CWC} = \text{PINAW} \left\{ 1 + \gamma(\text{PICP}) \; exp\left(\eta(\mu - \text{PICP})\right) \right\}, \tag{2.12}$$

donde las variables PICP y PINAW pueden ser calculadas a partir de las ecuaciones (1.4)-(1.7), y $\gamma(\text{PICP})$ es una función por tramos que puede ser representada por

$$\gamma = \begin{cases} 0 & \text{PICP} \geq \mu, \\ 1 & \text{PICP} < \mu. \end{cases} \tag{2.13}$$

En esta variante, los coeficientes η y μ son dos hiperparámetros que controlan el comportamiento del costo CWC, donde μ representa el porcentaje de cobertura nominal que es

deseado para el modelo de intervalo, mientras que η se utiliza para controlar la magnitud de la penalización para situaciones donde el porcentaje de cobertura del modelo no alcanza el valor deseado.

Por otro lado, en [47] se propone otra variante de este método en donde se reemplaza la arquitectura de modelo neuronal tradicional por la de una red de estado de eco, las cuales son un tipo de red neuronal recurrente. Dado que las arquitecturas recurrentes se encuentran especialmente diseñadas para el modelamiento dinámico, esta variación reportó mejores estimaciones de varianza en tareas de predicción de series de tiempo.

2.1.4. Método *Bootstrap*

El método *Bootstrap* [1] combina un conjunto de redes neuronales con técnicas estratégicas de muestreo para calcular tanto predicciones puntuales como la cuantificación de la incerteza con mayor precisión que la obtenida por métodos tradicionales basados en un solo modelo [18].

Como se explicó en la sección 1.1.2, en el área del modelamiento neuronal los modelos de conjunto consisten en una serie de metodologías que permiten la utilización de varias redes neuronales para resolver un único problema en conjunto [48]. Entre los distintos tipos de modelos de conjunto existentes en la literatura, el *bootstrapping* destaca por su habilidad de cuantificar la incerteza de la predicción del modelo [18]. Si bien existen diversas variantes de esta técnica, el algoritmo básico se puede describir de la siguiente forma:

1. Obtener S subconjuntos de datos a partir de un procedimiento de muestreo con reemplazo sobre el total de datos de entrenamiento.
2. Entrenar un modelo neuronal para cada uno de los subconjuntos muestreados (obteniendo un total de S modelos neuronales).
3. Una vez que se hayan entrenado todos los modelos, la salida total del conjunto puede ser calculada como

$$\hat{y}(z_k) = \frac{1}{S-1} \sum_{s=1}^{S} \hat{y}_s(z_k) \tag{2.14}$$

donde $\hat{y}(z_k)$ es la predicción total del conjunto para la muestra de entrada z_k y $\hat{y}_s(z_k)$ es la predicción del s-ésimo modelo neuronal para la muestra de entrada z_k

Luego de haber entrenado un modelo de conjunto, el Método *Bootstrap* propone que la incerteza espistémica $\sigma^2_{\hat{y}}$ puede ser estimada a partir de la varianza de las predicciones de los S modelos individuales:

$$\hat{\sigma}^2_{\hat{y}}(z_k) = \frac{1}{S} \sum_{s=1}^{S} (\hat{y}_s(z_k) - \hat{y}(z_k))^2 \tag{2.15}$$

Después de esto, la incerteza aleatoria es estimada por medio del entrenamiento de una nueva red neuronal $\hat{\sigma}^2_D(z_k)$ que intente estimar los residuales remanentes

$$\hat{r}^2(z_k) = \text{máx}\left(\left[y_k - \hat{y}_{validation}(z_k) \right]^2 - \hat{\sigma}^2_{\hat{y}}(z_k), 0 \right), \tag{2.16}$$

donde y_k es la k-ésima muestra de entrenamiento y

$$\hat{y}_{validation}(z_k) = \frac{\sum_{s=1}^{S} q_s(z_k)\hat{y}_s(z_k)}{\sum_{s=1}^{S} q_s(z_k)} \tag{2.17}$$

$$q_s(z_k) = \begin{cases} 1, & z_k \text{ está en el conjunto de validación del modelo } s \\ 0, & \text{si no lo está} \end{cases} \tag{2.18}$$

Este último paso de entrenamiento es logrado mediante el uso de una función de costos de log-verosimilitud similar a la mostrada en la ecuación (2.9):

$$C(\hat{r}^2) = \frac{1}{2}\sum_{k=1}^{N}\left(\log(\hat{\sigma}_D^2(z_k)) + \frac{\hat{r}^2(z_k)}{\hat{\sigma}_D^2(z_k)}\right) \tag{2.19}$$

Finalmente, una vez que ambas incertezas han sido estimadas, es posible construir un intervalo de predicción utilizando la misma metodología que los métodos secuenciales anteriores

$$\hat{\sigma}_{tot}^2(z_k) = \sigma_D^2 + \sigma_{\hat{y}}^2 \tag{2.20}$$

$$\overline{y}(z_k) = \hat{y}(z_k) + t_S^{1-\frac{\alpha}{2}}\hat{\sigma}_{tot}(z_k) \tag{2.21}$$

$$\underline{y}(z_k) = \hat{y}(z_k) - t_S^{1-\frac{\alpha}{2}}\hat{\sigma}_{tot}(z_k), \tag{2.22}$$

donde t_S representa el cuantil $\frac{\alpha}{2}$ de la función de distribución acumulada de una distribución t de Student con S grados de libertad.

Dado que el Método *Bootstrap* requiere el entrenamiento de varias redes neuronales, tiene un mayor costo computacional que el Método de Estimación de Media y Varianza. A pesar de esto, este problema sólo impacta en gran medida los costos computacionales de entrenamiento, por lo que el Método *Bootstrap* es una solución factible para aplicaciones que requieran rápidos tiempos de estimación. Además, considerando que este método potencialmente es capaz de obtener errores de predicción menores, su desempeño puede ser mejor que el del Método de Estimación de Media y Varianza, aunque esta hipótesis depende de la suposición de que cada uno de los S modelos neuronales del conjunto han sido entrenados con una suficiente cantidad de datos. Finalmente, es importante notar que, debido a la forma en que es entrenado, el Método Bootstrap es el único modelo de intervalo neuronal capaz de separar la incerteza de predicción en sus dos componentes, lo cual puede resultar de interés para ciertas aplicaciones.

Este método ha sido aplicado en la literatura para resolver problemas de pronóstico de descarga promedio por rebalse de olas en estructuras costeras [49], pronóstico de potencia eólica generada [21, 22], pronóstico de nivel de lagos [50], predicción de tiempos de viaje de buses [19], predicción de flujo de agua de alimentación para transitoria nuclear [20], estimación de degradación de componentes industriales sujetos a fatiga [51], pronóstico de precio de la electricidad [23], diseño de superaleación basada en níquel [52], y recuperación de profundidad óptica de aerosoles [53].

Además, en la literatura se han reportado algunas variantes de este método. En primer lugar, en [54] se propone un estimador más preciso para la incerteza epistémica mediante la división del conjunto en M sub-conjuntos más pequeños, de manera que para cada muestra de entrada z_k es posible obtener M predicciones distintas

$$\xi = \{\hat{y}^{\mathrm{i}}_{ens}(z_k)\}^{M}_{\mathrm{i}=1} \tag{2.23}$$

a partir de las cuales se puede obtener un conjunto de P re-muestras *bootstrap* según un proceso de muestreo con reposición

$$\Xi = \{\xi^*_j\}^{P}_{j=1} \tag{2.24}$$

$$\xi^*_j = \{\hat{y}^{j_1}_{ens}(z_k), ..., \hat{y}^{j_M}_{ens}(z_k)\} \tag{2.25}$$

de manera que la incerteza epistémica puede ser calculada como

$$\sigma^2_{\hat{y}}(z_k) = \frac{1}{P}\sum_{j=1}^{P} \sigma^{*^2}_j(z_k), \tag{2.26}$$

donde

$$\sigma^{*^2}_j(z_k) = \frac{1}{M}\sum_{\mathrm{i}=1}^{M} \left(\hat{y}^{j_{\mathrm{i}}}_{ens}(z_k) - \hat{y}^{j}(z_k)\right)^2 \tag{2.27}$$

$$\hat{y}^{j}(z_k) = \frac{1}{M}\sum_{\mathrm{i}=1}^{M} \hat{y}^{j_{\mathrm{i}}}_{ens}(z_k) \tag{2.28}$$

Por el otro lado, en [55] se propone un método diferente para la estimación de la incerteza aleatoria σ^2_D basado en el entrenamiento de una red neuronal utilizando la función de costos CWC mostrada en la ecuación (2.12) por medio de un algoritmo de recocido simulado.

2.1.5. Método de Covarianza

El método de Covarianza presenta una metodología similar a la del Método Delta, pero utiliza un enfoque estadístico basándose en los trabajos realizados por Škrjanc [56], y Sáez et al. [57] sobre modelos de intervalo basados en arquitecturas difusas.

Como este método es de tipo secuencial, es necesario contar con un modelo neuronal previamente entrenado antes de poder realizar los cálculos necesarios para construir el intervalo. Para el caso específico del Método de Covarianza, se requiere construir un modelo de perceptrón multicapa con una única capa oculta, representado matemáticamente en la ecuación (1.8). Usando esta notación, es posible definir el vector de salida de la capa oculta Z_k y el vector de parámetros del modelo θ como

$$Z_k = [Z_{1k}, ..., Z_{N_h k}, 1] \tag{2.29}$$

$$\theta = [w_1, ..., w_{N_h}, b], \tag{2.30}$$

donde

$$Z_{jk} = \tanh\left(\sum_{\mathrm{i}=1}^{N_{in}} w_{j\mathrm{i}} z_k^{(\mathrm{i})} + b_j\right) \tag{2.31}$$

$$\hat{y}(z_k) = Z_k^T \theta \tag{2.32}$$

Luego, asumiendo que el error de predicción es homogéneo y normalmente distribuido, se puede calcular una expresión para la varianza del error de predicción como

$$\sigma_k = \sigma_{\mathrm{e}}(1 + Z_k^T (Z^* Z^{*T})^{-1} Z_k)^{\frac{1}{2}}, \tag{2.33}$$

donde σ_{e} representa la varianza del ruido de los datos, la cual puede ser calculada por medio de la ecuación (2.2), Z_k^T se calcula utilizando los datos del conjunto de validación, y Z^* es la matriz de todas las salidas de la capa oculta Z_k del modelo, para cada muestra del conjunto de entrenamiento.

Finalmente, una vez que se obtiene este valor, es posible calcular un IP como

$$\overline{y}(z_k) = \hat{y}(z_k) + \alpha\sigma_k \tag{2.34}$$

$$\underline{y}(z_k) = \hat{y}(z_k) - \alpha\sigma_k \tag{2.35}$$

donde α es un hiperparámetro que requiere ser ajustado por medio de un algoritmo iterativo (como por ejemplo, una búsqueda logarítmica) para acomodar el intervalo al porcentaje de cobertura de datos deseado.

Como se puede notar a partir de las ecuaciones (2.1) y (2.33), el Método de Covarianza utiliza una metodología de cuantificación de incerteza similar a la utilizada por el Método Delta, donde la principal diferencia radica en que este último utiliza un enfoque basado en los parámetros, evidenciado en la necesidad de calcular la matriz Jacobiana del modelo predictivo, mientras que el Método de Covarianza utiliza una estimación basada en los datos. Estas similitudes traen como consecuencia que ambos métodos comparten las mismas limitaciones, principalmente en cuanto a la dependencia en la suposición de que el error de predicción es homogéneo y Gaussiano. Además, es importante notar que la incorporación de un proceso iterativo dentro del algoritmo de entrenamiento hace que este método sea, en general, más costoso computacionalmente que el Método Delta.

Este método ha sido aplicado en la literatura para resolver problemas de pronóstico de carga eléctrica [24].

2.1.6. Método de Números Difusos

El método de Números Difusos propone la construcción de un modelo de intervalo por medio del uso de una arquitectura de red neuronal modificada donde los parámetros son modelados como números difusos [26]. Esto significa que, basándose en la notación presentada en las ecuaciones (1.8) y (2.31), los pesos que conectan la capa oculta de la red con la capa de salida w_j son representados por su valor medio ($g^{(j)}$) más dos valores de *spread* ($\underline{s}^{(j)}, \overline{s}^{(j)}$) que modelan su rango de incerteza.

En cuanto a la metodología de entrenamiento, esta se puede dividir en dos rutinas de identificación de parámetros: Por un lado, el primer procedimiento de entrenamiento es responsable de identificar el valor medio $g^{(j)}$ de los parámetros, y consiste en una optimización por minimización del error cuadrático medio mediante el algoritmo de *Backpropagation*. Por otro lado, el segundo procedimiento se enfoca en entrenar el modelo para identificar los parámetros de los valores de *spread* $(\underline{s}^{(j)}, \overline{s}^{(j)})$, los cuales se obtienen resolviendo el problema de optimización sin restricciones

$$\min_{\overline{s}_i, \underline{s}_i} \quad J = \eta_1 \text{PINAW} + \text{e}xp\{-\eta_2 \left[\text{PICP} - (1-\alpha)\right]\}, \tag{2.36}$$

donde se puede apreciar que la función objetivo presenta una forma similar a la métrica CWC presentada en la ecuación (2.12).

Luego, una vez que se han identificado todos los parámetros del modelo de intervalo, los límites del intervalo de predicción $(\overline{y}, \underline{y})$ pueden ser calculados según

$$\overline{y}(z_k) = \sum_{j=1}^{N_h} g^{(j)} Z_k^{(j)} + g^{(0)} + \sum_{j=1}^{N_h} \overline{s}^{(j)} |Z_k^{(j)}| + \overline{s}^{(0)} \tag{2.37}$$

$$\underline{y}(z_k) = \sum_{j=1}^{N_h} g^{(j)} Z_k^{(j)} + g^{(0)} - \sum_{j=1}^{N_h} \underline{s}^{(j)} |Z_k^{(j)}| + \underline{s}^{(0)}, \tag{2.38}$$

donde $Z_k^{(j)}$ se encuentra definido de la misma forma que en la ecuación (2.31).

Debido a la posibilidad de usar parámetros de *spread* no simétricos y no requerir supuestos sobre el comportamiento de la incerteza a modelar, este método es capaz de obtener intervalos precisos en términos de ancho y porcentaje de cobertura , incluso cuando el sistema a predecir presenta un comportamiento complejo (tales como incertezas no Gaussianas, o sistemas que presenten heterocedasticidad). A pesar de esto, este incremento en la robustez del modelo trae consigo el costo computacional de tener que entrenar el triple de parámetros que un modelo neuronal tradicional.

Este método ha sido aplicado en la literatura principalmente en problemas de pronóstico de carga eléctrica [26]

2.2. Métodos directos

2.2.1. *Lower Upper Bound Estimation* (LUBE)

El método LUBE propone una solución para la construcción de intervalos de predicción donde un único modelo neuronal con una arquitectura modificada es entrenado para predecir directamente los límites superior e inferior del intervalo [5].

La red neuronal propuesta para este método consiste de un modelo similar a la arquitectura de modelos predictivos neuronales tradicionales, pero que utiliza dos neuronas en la capa de salida. Esto significa que algunos parámetros de la red serán compartidos entre ambas salidas, lo cual los hace incompatibles con una rutina tradicional de *Backpropagation*. Debido a esto,

para solucionar este problema, las arquitecturas con múltiples salidas calculan el gradiente de los pesos compartidos como el valor promedio entre los gradientes derivados de cada una de las salidas.

Luego, para entrenar este modelo de múltiples salidas, este método propone utilizar el criterio CWC mostrado en la ecuación (2.12) como función de costo.

Es importante notar que el método LUBE ha reportado consistentemente en la literatura un menor ancho promedio de intervalo y un menor costo computacional que los métodos secuenciales tradicionales, por lo cual la gran mayoría de los esfuerzos investigativos dedicado a modelos de intervalo neuronales se han enfocado en mejorar el procedimiento de optimización de este método. Debido a que la función de costo CWC es altamente no lineal, se han propuesto un amplio rango de algoritmos de optimización no convencionales para su solución, tales como algoritmos de recocido simulado [5], optimización por enjambre de partículas [58], algoritmos genéticos [59], e incluso algoritmos evolutivos multiobjetivo [60].

Este método ha sido aplicado en la literatura para resolver problemas de pronóstico de generación de potencia eólica [21, 58, 61], pronóstico de velocidad del viento [62, 59], pronóstico de generación de potencia solar [63, 60], pronóstico de carga eléctrica [58, 64, 65, 66], pronóstico de descarga de flujo de corriente [67], estimación de desplazamientos por deslizamiento de tierra [68], predicción de temperatura de reactores [69, 70], pronóstico de inundaciones [71, 72], y estimación de compensación de potencia reactiva para hornos de arco eléctrico [73].

En cuanto a las variantes de este método reportadas en la literatura, la mayoría de estas consisten en la inclusión de técnicas que no modifican la estructura fundamental del método, tales como procesos adicionales de selección de características [66], modelos de conjunto [66, 69], y optimización multiobjetivo [74]. Sin embargo, en [64] se propone una variante con cambios más profundos, donde se apunta a reemplazar el término PINAW dentro de la función de costos CWC (ver ecuación (2.12)) por una nueva métrica, denominada *Prediction Interval Normalized Root-mean-square Width* (PINRW):

$$\text{PINRW} = \frac{1}{R}\sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(\overline{y}(z_k) - \underline{y}(z_k)\right)^2}, \tag{2.39}$$

donde el coeficiente de normalización se calcula de igual manera que en la ecuación (1.5). Esta modificación fue propuesta con el objetivo de intentar acercar el comportamiento de la función de costo de intervalo al de la función de error cuadrático medio, de manera que al calcular el gradiente de la nueva métrica propuesta se imponga una penalización proporcional a la magnitud del error de predicción.

Adicionalmente, en [72] se propone un reemplazo diferente al término PINAW de la función de costos, el cual fue denominado *Prediction Interval Average Relative Width* (PIARW):

$$\text{PIARW} = \frac{1}{N}\sum_{k=1}^{N}\frac{\left(\overline{y}(z_k) - \underline{y}(z_k)\right)}{y_k}, \tag{2.40}$$

donde y_k es el k-ésimo dato de entrenamiento. Este trabajo también propone la inclusión de un término adicional dentro de la función de costos con el objetivo de optimizar la simetría

del intervalo, denominado *Prediction Interval Symmetry* (PIS)

$$\text{PIS} = \frac{1}{N}\sum_{k=1}^{N}\frac{\left|y_k - \frac{\overline{y}(z_k)+\underline{y}(z_k)}{2}\right|}{\overline{y}(z_k) - \underline{y}(z_k)}, \tag{2.41}$$

de manera que el nuevo costo total, denominado *Coverage Width Symmetry-based Criterion* (CWSC) en alusión a la inclusión del criterio de simetría del intervalo, puede ser calculado como

$$\begin{aligned}&\text{CWSC} = \\ &\gamma(\text{PIS})\text{e}^{\eta_3(\text{PIS}-\mu_2)} + \eta_2\text{PIARW} + \gamma(\text{PICP})\text{e}^{-\eta_1(\text{PICP}-\mu_1)}\end{aligned} \tag{2.42}$$

Estas modificaciones fueron propuestas en un intento de que el Método LUBE realice una optimización más informada al darle acceso a todos los datos de entrenamiento y_k a la función de costo, y al mismo tiempo sea capaz de darle importancia a la optimización de la predicción del valor esperado mediante la inclusión del componente de simetría PIS.

2.2.2. Método *Joint Supervision*

El método *Joint Supervision* , al igual que el Método LUBE, propone una metodología para construir modelos de intervalo neuronales a partir del entrenamiento de un único modelo basado en una arquitectura modificada de red neuronal [24]. A diferencia de la propuesta de LUBE, el Método *Joint Supervision* utiliza un modelo neuronal con 3 neuronas en la capa de salida, donde dos salidas son usadas para predecir los límites superior e inferior del intervalo, mientras que la tercera salida cumple el rol de predecir el valor esperado (también conocido como la predicción puntual) de la salida.

Para entrenar esta arquitectura, el Método *Joint Supervision* propone la utilización de una función de costos distinta para cada una de las salidas del modelo $(\overline{y}, \hat{y}, \underline{y})$. En primer lugar, todas las salidas comparten una componente de costo de error cuadrático medio para minimizar el error de predicción. Sin embargo, una componente de costo adicional es aplicada sólo a las salidas responsables de la predicción de los límites del intervalo. Estas componentes consisten en una penalización de error cuadrático medio complementaria que sólo es aplicada en puntos donde los datos de entrenamiento quedan fuera del intervalo predicho por el modelo. En particular, la penalización se aplicará a la salida responsable de predecir el límite superior del intervalo si el dato se encuentra sobre el rango predicho, o bien se aplicará a la salida responsable de calcular el límite inferior si el dato se encuentra por debajo del rango predicho. Matemáticamente, las 3 funciones de costo pueden expresarse como

$$\overline{L} = \frac{1}{N}\sum_{k=1}^{N}(y_k - \overline{y}(z_k))^2 + \lambda\frac{1}{N}\sum_{k=1}^{N}ReLU^2(y_k - \overline{y}(z_k)) \tag{2.43}$$

$$\hat{L} = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}(z_k))^2 \tag{2.44}$$

$$\underline{L} = \frac{1}{N}\sum_{k=1}^{N}(y_k - \underline{y}(z_k))^2 + \lambda\frac{1}{N}\sum_{k=1}^{N}ReLU^2(\underline{y}(z_k) - y_k), \tag{2.45}$$

donde

$$ReLU(x) = \begin{cases} 0, & \text{if } x < 0 \\ x, & \text{if } x \geq 0 \end{cases} \tag{2.46}$$

y λ es un hiperparámetro que es directamente proporcional al ancho del intervalo y debe ser ajustado para cada experimento individual a través de un algoritmo de búsqueda iterativa (como por ejemplo, una búsqueda logarítmica). En cada iteración del proceso de ajuste hiperparamétrico, es necesario entrenar un nuevo modelo *Joint Supervision* con un valor de λ determinado, evaluar su porcentaje de cobertura, y, en caso de que este sea distinto al deseado, proponer un nuevo valor de λ que disminuya o aumente el ancho del intervalo, según sea necesario.

La principal ventaja de este método por sobre otros métodos directos radica en que la simplicidad de su función de costo permite la utilización de algoritmos de Gradiente Descendente Estocástico para su optimización, los cuales han sido específicamente optimizados para el entrenamiento de modelos neuronales, resultando en menores tiempos de entrenamiento y evaluación. No obstante, a pesar de esta disminución del costo computacional, la inclusión del hiperparámetro λ en las funciones de costo del modelo (2.43) y (2.45), junto con la necesidad de ajustarlo mediante un proceso de búsqueda iterativa donde en cada iteración es necesario volver a entrenar una nueva red neuronal, introduce la necesidad de reentrenar el modelo varias veces para poder obtener una solución óptima. Esto perjudica directamente el costo de entrenamiento del modelo, lo cual trae como consecuencia que el Método de *Joint Supervision* no pueda aprovechar por completo la simplicidad de su función de costo.

Este método ha sido aplicado en la literatura principalmente para resolver problemas de pronóstico de carga eléctrica [24, 25]

2.2.3. Método *Quality Driven* (QD)

El método *Quality Driven* se inspira en los fundamentos del Método LUBE y propone un modelo de intervalo que utiliza la misma arquitectura de red neuronal con dos salidas, pero introduce una función de costos nueva y simplificada [4]:

$$L_{QD} = \text{MPIW}_{capt} + \lambda \frac{N}{\alpha(1-\alpha)} \max(0, (1-\alpha) - \text{PICP})^2, \tag{2.47}$$

donde λ es un hiperparámetro que requiere un único ajuste universal para todo experimento, N es la cantidad total de datos de entrenamiento, $(1-\alpha)$ es el porcentaje de cobertura deseado para el modelo de intervalo, y $MPIW_{capt}$ consiste en una nueva métrica denominada *Captured Mean Prediction Interval Width*, cuyo valor se puede obtener a partir de

$$\text{MPIW}_{capt} = \frac{\sum_{k=1}^{N} (\overline{y}(z_k) - \underline{y}(z_k)) \mathrm{i}_k}{\sum_{k=1}^{N} \mathrm{i}_k}, \tag{2.48}$$

donde

$$\mathrm{i}_k = \begin{cases} 1, & \text{if } \underline{y}(z_k) \leq y_k \leq \overline{y}(z_k) \\ 0, & \text{otherwise} \end{cases} \tag{2.49}$$

Basándose en esta propuesta, el Método *Quality Driven* en teoría es capaz de construir intervalos de predicción utilizando optimización por Gradiente Descendente Estocástico, convirtiéndolo en una alternativa menos costosa computacionalmente que Método LUBE, pero que a la vez mantiene gran parte de su precisión en cuanto a cobertura y ancho del intervalo. Además, es importante notar que la inclusión del coeficiente α, el cual representa el porcentaje de cobertura deseado para el modelo de intervalo, dentro de la función de costos permite que este método pueda considerar explícitamente la restricción de porcentaje de cobertura dentro del proceso de entrenamiento de la red. Como consecuencia de esto, el Método *Quality Driven* no requiere de procesos de búsqueda iterativa adicionales para ajustar sus hiperparámetros, a diferencia del Método *Joint Supervision*. A pesar de lo anterior, el desempeño de este método se encuentra limitado debido a que las simplificaciones propuestas para modificar el costo utilizado en LUBE se basan en el supuesto de que los datos de entrenamiento son independientes e idénticamente distribuidos, lo cual no es compatible con el modelamiento de sistemas dinámicos.

A pesar de las ventajas reportadas para este método, actualmente sólo ha sido aplicado en bases de datos artificiales y no existen aplicaciones en la literatura que puedan corroborar estos resultados, aunque esto se debe principalmente a lo reciente de este método.

2.2.4. Método *Bayes by Backprop* (BBB)

El método *Bayes by Backprop* propone un algoritmo que puede ser aplicado para entrenar redes neuronales Bayesianas (las cuales, como se mencionó en el Método Bayesiano, modelan sus parámetros como distribuciones de probabilidad) sin tener que recurrir a métodos Monte Carlo basados en cadenas de Markov (MCMC) o el cálculo de matrices Hessianas. Para lograr esto, el método utiliza un algoritmo de *backpropagation* modificado que es compatible con redes Bayesianas y a la vez puede converger a una distribución posterior variacional de la distribución probabilística de los pesos del modelo [75].

El algoritmo BBB consiste en aplicar una técnica conocida en la literatura como el truco de reparametrización local [76] para alterar levemente la media y desviación estándar de los parámetros en cada iteración de entrenamiento

$$\theta = (\mu, \sigma^2) \tag{2.50}$$

$$\varepsilon \sim \mathcal{N}(0, 1) \tag{2.51}$$

$$f(\varepsilon) = w = \mu + \sigma \cdot \varepsilon, \tag{2.52}$$

donde θ es el vector de pesos de la red, μ es el vector de medias de los parámetros, y σ es el vector de desviaciones estándar de los parámetros. Usando esta técnica, es posible calcular un gradiente, y consecuentemente actualizar los pesos de la red, como

$$\nabla\mu = \frac{\partial f}{\partial w} + \frac{\partial f}{\partial \mu} \tag{2.53}$$

$$\nabla\sigma = \frac{\partial f}{\partial w}\frac{\varepsilon}{\sigma} + \frac{\partial f}{\partial \sigma} \tag{2.54}$$

$$\mu \leftarrow (\mu - \alpha\nabla\mu) \tag{2.55}$$

$$\sigma \leftarrow (\sigma - \alpha\nabla\sigma) \tag{2.56}$$

Una vez que se obtiene una distribución posterior final, la incerteza de la predicción del modelo puede ser calculada mediante la realización de varias predicciones para cada muestra de entrada. Dado que los pesos de la red neuronal actúan como variables aleatorias, sus valores son recalculados constantemente para cada nueva predicción, según la ecuación (2.52). De esta manera, es posible calcular estadísticos para la media y la desviación estándar de cada predicción. Finalmente, el IP se puede construir mediante el cómputo del cuantil correspondiente de la distribución probabilística generada por las predicciones del modelo.

A pesar de que este método ha reportado resultados satisfactorios en bases de datos artificiales, es necesaria la realización de experimentos adicionales para verificar cómo se compara su desempeño con respecto a otros métodos existentes.

2.2.5. Método *Randomized Prior Functions*

El método *Randomized Prior Functions* [77] introduce un enfoque Bayesiano para la estimación de incerteza que no requiere el modelamiento de los parámetros del modelo como variables aleatorias. En cambio, el Método *Randomized Prior Functions* propone la utilización de una arquitectura modificada de red neuronal donde las predicciones totales son obtenidas por medio de la suma del resultado de dos redes neuronales por separado: NN_{reg}, que consiste en un modelo neuronal tradicional, y NN_{prior}, que consiste en un modelo no entrenable cuyos parámetros son inicializados aleatoriamente. Usando esta notación, la predicción total del modelo se calcula como

$$\hat{y}_{tot}(z_k) = \hat{y}_{reg}(z_k) + \hat{y}_{prior}(z_k), \tag{2.57}$$

donde la red NN_{prior} juega un rol similar al de la distribución a priori definida en la inferencia Bayesiana, dado que controla el comportamiento del modelo en regiones con mayor incerteza. Esta intuición es estudiada a mayor profundidad en la propuesta original [77], donde se logra demostrar la existencia de una conexión entre la generación de predicciones usando esta arquitectura y la extracción de muestras de una distribución posterior.

De esta manera, el *Método Randomized Prior Functions* puede estimar su incerteza a través de la generación repetitiva de predicciones utilizando distintas redes a priori NN_{prior}, para cada muestra de entrada. Luego, para construir un IP sólo basta con calcular el cuantil correspondiente de la distribución de probabilidad resultante para las predicciones del modelo.

A pesar de que este método ha reportado resultados satisfactorios en bases de datos artificiales, es necesaria la realización de experimentos adicionales para verificar cómo se compara su desempeño con respecto a otros métodos existentes.

2.3. Discusión

Para comparar y discutir el desempeño y aplicabilidad de los métodos presentados en la sección anterior, es necesario observar las características principales que distinguen a cada modelo, tales como la metodología utilizada para la generación del intervalo, el número de parámetros de la red, el procedimiento de identificación, y los supuestos realizados sobre el comportamiento de la incerteza. Para facilitar este análisis, las tablas 2.1 y 2.2 muestran un resumen de estas características para los métodos secuenciales y directos, respectivamente.

Es importante notar que, al especificar el número de parámetros de cada método, se utilizó la notación presentada en las ecuaciones (1.9) y (1.8).

Debido a las diferentes metodologías utilizadas por cada modelo para cuantificar su costo computacional, la inclusión de esta variable fue manejada indirectamente en las tablas según tres criterios: El número de parámetros adicionales introducidos por el método para construir el intervalo, el tipo de método de optimización que se requiere utilizar para la identificación de parámetros (métodos de gradiente tienen un menor costo computacional que otras alternativas no lineales), y el número de veces que se debe entrenar un modelo para converger a una solución óptima (el costo computacional aumenta con el número de modelos y el número de veces que se debe entrenar un modelo).

Basándose en la información entregada por estas tablas, es posible notar que los métodos directos tienden a poseer un mayor costo computacional y número de parámetros que los métodos secuenciales. Esta tendencia puede ser explicada debido a que los métodos secuenciales dependen en gran parte de un modelo base previamente entrenado, por lo que el entrenamiento está enfocado en sólo obtener aquellos parámetros que están directamente involucrados en el cálculo del intervalo de predicción. Por otro lado, los métodos directos incorporan tanto el entrenamiento del modelo predictivo como el del modelo de intervalo en un único procedimiento, lo cual puede aumentar la complejidad de estos modelos (como por ejemplo se puede evidenciar en el método LUBE, el cual no puede ser entrenado utilizando algoritmos de optimización de redes neuronales tradicionales). Sin embargo, también se puede notar que existen excepciones a esta tendencia, principalmente por parte de métodos secuenciales que no realizan supuestos sobre el comportamiento de la incerteza, como es el caso del método Bayesiano y el método de Números Difusos.

Entre los modelos de intervalo reportados en esta revisión, el método *Bootstrap* destaca por sus características específicas. Esto se debe a dos causas principales: En primer lugar, este método es el único capaz de estimar la incerteza epistémica y aleatoria por separado sin realizar ningún supuesto de homocedasticidad, lo cual permite la obtención de estimadores insesgados para estas variables. En segundo lugar, el método *Bootstrap* puede ser utilizado en conjunción con otros modelos de intervalo para mejorar su desempeño y/o cuantificar rangos de confianza para los parámetros del intervalo.

Adicionalmente, existen también otros modelos de intervalo notables que encuentran utilidad en casos específicos. Por ejemplo, si una aplicación requiere tiempos de entrenamiento extremadamente cortos y la incerteza del modelo puede ser razonablemente aproximada como homocedástica, entonces el método Delta muy probablemente sea la solución más adecuada. Por el otro lado, si una aplicación realiza el entrenamiento de manera *offline* y presenta una incerteza compleja y difícil de modelar, métodos tales como el de Números Difusos, *Joint Supervision*, *Quality-Driven*, o incluso *Bootstrap* (especialmente si se desea una cuantificación por separado de la incerteza aleatoria y epistémica) pueden ser las soluciones más apropiadas, dado que realizan pocos o ningún supuesto sobre el comportamiento del sistema. En particular, se realizan los siguientes comentarios sobre la aplicabilidad de cada uno de los métodos reportados:

1. **Método Delta:** Adecuado para aplicaciones que requieren modelos con muy bajo costo computacional y número de parámetros. Adicionalmente, debido a los supuestos

Tabla 2.1: Características de los modelos de intervalo neuronal basados en métodos secuenciales

Tipo de intervalo	Características
Método Delta	- Método basado en el cálculo de la matriz Jacobiana del modelo y la varianza del ruido de los datos. - Intervalo definido por 2 parámetros ($\sigma_{tot}, t_{n-p}^{1-\frac{\alpha}{2}}$). - Entrenamiento del modelo realizado por métodos de optimización lineales. - Asume incerteza homogénea y Gaussiana.
Método Bayesiano	- Método basado en modelar los parámetros de la red como distribuciones de probabilidad, representadas por una media y una varianza. - Intervalo definido por $2(N_h(n_y + n_u + 1) + (N_h + 1))$ parámetros. - Entrenamiento del modelo realizado por algoritmos MCMC y el cálculo de la matriz Hessiana. - No realiza supuestos sobre el comportamiento de la incerteza.
Método de Estimación de Media y Varianza	- Método basado en entrenar dos redes neuronales secuencialmente (una para predicciones puntuales, y otra para la varianza del error). - Intervalo definido por $(N_h(n_y + n_u + 1) + (N_h + 1))$ parámetros. - Entrenamiento del modelo realizado con algoritmo de *Backpropagation*. - Asume incerteza epistémica nula.
Método *Bootstrap*	- Método basado en entrenar un conjunto de redes neuronales. - Intervalo definido por $(B + 1)(N_h(n_y + n_u + 1) + (N_h + 1))$ parámetros, donde B es el tamaño del conjunto. - Entrenamiento del modelo realizado con algoritmo de *Backpropagation*. - Asume error de predicción simétrico.
Método de Covarianza	- Método basado en el cálculo de matrices de regresores, varianza del ruido de los datos, y un procedimiento de ajuste hiperparamétrico utilizando un algoritmo de búsqueda logarítmica. - Intervalo definido por 2 parámetros (α, σ_k). - Entrenamiento del modelo ealizado por método de optimización lineales. - Asume incerteza homogénea y Gaussiana.
Método de Números Difusos	- Método basado en modelar los parámetros de la red como números difusos con una media y dos valores de *spread*. - Intervalo definido por $2(N_h + 1)$ parámetros. - Entrenamiento del modeo realizado por métodos de optimización no lineales. - No realiza supuestos sobre el comportamiento de la incerteza.

realizados por este método, la incerteza del modelo debe ser estudiada para determinar si puede ser razonablemente aproximada como homogénea y Gaussiana.

2. **Método Bayesiano:** El cálculo de la matriz Hessiana hace que este método sea más adecuado para aplicaciones donde los altos costos computacionales y tiempos de entrenamiento no son un problema. Además, es importante notar que la baja cantidad de aplicaciones reportadas en la literatura para este método (en parte debido a los problemas de tiempos de ejecución mencionados anteriormente) hace que su aplicabilidad sea difícil de determinar.
3. **Método de Estimación de Media y Varianza:** El desempeño de este método se ve fuertemente afectado por el supuesto de incerteza epistémica nula. Actualmente existen otros métodos con un costo computacional similar, como por ejemplo el método *Quality-Driven*, que son capaces de construir intervalos de mayor calidad (más delgados), dado que no se sustentan de un supuesto tan fuerte.
4. **Método *Bootstrap*:** Adecuado para aplicaciones donde largos tiempos de entrenamiento no son una limitante y se desea tener una cuantificación separada de la incerteza epistémica y aleatoria.
5. **Método de Covarianza:** Si bien este método tiene un desempeño similar al del método Delta, la necesidad de ejecutar un algoritmo de búsqueda logarítmica para la optimización hiperparamétrica hace que el método de Covarianza sea más costoso computacionalmente.
6. **Método de Números Difusos:** Dado que los parámetros de *spread* sólo son ocupados

Tabla 2.2: Características de los modelos de intervalo neuronal basados en métodos directos

Tipo de intervalo	Características
Método LUBE	- Método basado en entrenar una única red neuronal con dos salidas. - Intervalo definido por $N_h(n_y + n_u + 1) + 2(N_h + 1)$ parámetros. - Entrenamiento del modelo realizado por métodos de optimización no lineales. - Asume error de predicción simétrico.
Método *Joint Supervision*	- Método basado en entrenar una única red neuronal con tres salidas, y un procedimiento de ajuste hiperparamétrico utilizando un algoritmo de búsqueda logarítmica. - Intervalo definido por $N_h(n_y + n_u + 1) + 3(N_h + 1)$ parámetros. - Entrenamiento del modelo realizado con algoritmo de *Backpropagation*. - No realiza supuestos sobre el comportamiento de la incerteza.
Método *Quality-Driven*	- Método basado en entrenar una única red neuronal con dos salidas. - Intervalo definido por $N_h(n_y + n_u + 1) + 2(N_h + 1)$ parámetros. - Entrenamiento del modelo realizado con algoritmo de *Backpropagation*. - Asume error de predicción simétrico.
Método *Bayes by Backprop*	- Método basado en modelar los parámetros de la red como distribuciones de probabilidad, representadas por una media y una varianza. - Intervalo definido por $2(N_h(n_y + n_u + 1) + (N_h + 1))$ parámetros. - Entrenamiento del modelo realizado con algoritmo de *Backpropagation* modificado. - Asume error de predicción simétrico.
Método *Randomized Prior Functions*	- Método basado en entrenar un conjunto de redes neuronales más una red prior adicional no entrenable. - Intervalo definido por $B(N_h(n_y + n_u + 1) + (N_h + 1))$ parámetros, donde B es el tamaño del conjunto. - Entrenamiento del modelo realizado con algoritmo *Backpropagation*. - Asume error de predicción simétrico.

en la capa final de la red, en general este método posee un desempeño inferior a otros método con un costo computacional similar, como por ejemplo el método LUBE.

7. **Método LUBE:** Adecuado para aplicaciones que posean una incerteza con un comportamiento complejo, donde además el número de parámetros del modelo y el costo computacional no sean un problema. Adicionalmente, antes de utilizar este método se recomienda realizar un análisis comparativo con los métodos *Joint Supervision* y *Quality-Driven*. En general, el método LUBE posee un menor tiempo de entrenamiento que el método *Joint Supervision*, y a la vez no realiza supuestos sobre el comportamiento de la incerteza del sistema.
8. **Método Joint Supervision:** Adecuado para aplicaciones que posean una incerteza con un comportamiento complejo, donde además el número de parámetros del modelo y el costo computacional no sean un problema. Adicionalmente, antes de utilizar este método se recomienda realizar un análisis comparativo con los métodos LUBE y *Quality-Driven*. En general, el método *Joint Supervision* posee el mayor costo computacional entre las 3 alternativas, pero tiene la ventaja de ser capaz de construir intervalos asimétricos.
9. **Método Quality-Driven:** Adecuado para aplicaciones que posean una incerteza con un comportamiento complejo, donde además el número de parámetros del modelo y el costo computacional no sean un problema. Adicionalmente, antes de utilizar este método se recomienda realizar un análisis comparativo con los métodos *Joint Supervision* y LUBE. En general, el método *Quality-Driven* posee el menor costo computacional entre las 3 alternativas, pero depende fuertemente del supuesto de datos i.i.d., lo cual puede perjudicar la calidad del intervalo en sistemas dinámicos.
10. **Método Bayes by Backprop:** Dado que en la literatura no se han realizado experimentos comparativos con otros métodos del estado del arte, este método es más adecuado para propósitos de investigación.

Tabla 2.3: Aplicaciones de intervalos de predicción neuronales

Tipo de Modelo de Intervalo	Aplicaciones			
	Algoritmos de detección de fallas	Microrredes y sistemas de energías renovables	Series de tiempo caóticas y sistemas no lineales	Otros tipos de control robusto
Método Delta	[31]	[30]	[34]	[32, 33]
Método Bayesiano		[41]	[34]	[42]
Método de Estimación de Media y Varianza		[44, 46]	[45, 47]	[47]
Método *Bootstrap*	[51]	[21, 22, 23]	[19]	[49, 50, 20] [52, 53]
Método de Covarianza	—	—	—	—
Método de Números Difusos		[26]		
Método LUBE	[73]	[21, 58, 61, 62, 63] [60, 58, 64, 65, 66]		[67, 68, 69] [70, 71, 72]
Método *Joint Supervision*		[24, 25]		
Método *Quality-Driven*	—	—	—	—
Método *Bayes by Backprop*	—	—	—	—
Método *Randomized Prior Functions*	—	—	—	—

11. **Método Randomized Prior Functions:** Dado que en la literatura no se han realizado experimentos comparativos con otros métodos del estado del arte, este método es más adecuado para propósitos de investigación.

Otro aspecto que puede resultar de utilidad para un análisis comparativo es el tipo de aplicaciones reportadas en la literatura para cada uno de los métodos. Para facilitar este estudio, en la tabla 2.3 se muestra un resumen de las aplicaciones reportadas para cada uno de los modelos de intervalo neuronal reportados en esta sección. Como se puede apreciar en la tabla, los modelos de intervalo neuronal han sido utilizados principalmente para aplicaciones de control robusto, especialmente en tópicos relacionados con sistemas de energías renovables y operación robusta de microrredes, donde estos modelos han sido consistentemente usados para tareas como pronóstico de velocidad del viento, predicción de generación de potencia solar, y pronóstico de demanda eléctrica.

Capítulo 3

Metodología

En esta sección se procederá a describir el procedimiento planteado para lograr el cumplimiento de los objetivos. A modo de resumen, en la figura 3.1 se presenta un diagrama de la estructura general de la metodología adoptada para el presente trabajo de investigación.

Como se puede apreciar en la figura 3.1, la estrategia propuesta comienza con la realización de una revisión de literatura en el tópico de modelamiento de intervalos basado en redes neuronales. A partir de esta, se estudió arquitecturas para la generación de intervalos de predicción que posean características deseables de desempeño, costo computacional, número de parámetros y/o metodología de entrenamiento, considerando los requisitos planteados en los objetivos de esta tesis.

Posteriormente, basándose en uno de los métodos del estado del arte se procedió a diseñar una nueva propuesta de modelo de intervalo neuronal que presenta un bajo costo computacional, que es compatible con optimizadores basados en métodos de gradiente, y no realiza supuestos sobre el comportamiento de la incerteza. Adicionalmente, con el objetivo de validar el modelo de intervalo propuesto, en paralelo a este paso también se propuso una extensión del método utilizado como base para compatibilizarlo con arquitecturas difusas Takagi-Sugeno.

Una vez diseñada la propuesta neuronal, se procedió a generar dos variantes de este método. Por un lado, un modelo de intervalo basado en redes neuronales tradicionales, y por el otro, un modelo de intervalo basado en arquitecturas de *Deep Learning*. Luego, con el objetivo de verificar el desempeño de estos modelos, se diseñaron pruebas experimentales evaluando dos casos de estudio: Una base de datos generada artificialmente, y una correspondiente a datos reales de potencia solar.

El procedimiento utilizado para el desarrollo de los experimentos consistió primeramente en un proceso de búsqueda exhaustiva con el objetivo de optimizar los hiperparámetros de los modelos, tales como la tasa de aprendizaje, el tipo de optimizador, el límite a la magnitud del gradiente, y el número de lotes por época. Después de haber obtenido una configuración paramétrica satisfactoria, se procedió a entrenar ambos modelos neuronales por medio del algoritmo de *backpropagation*. Adicionalmente, en paralelo a este proceso, también se ejecutó un algoritmo de identificación de parámetros para entrenar la propuesta basada en sistemas

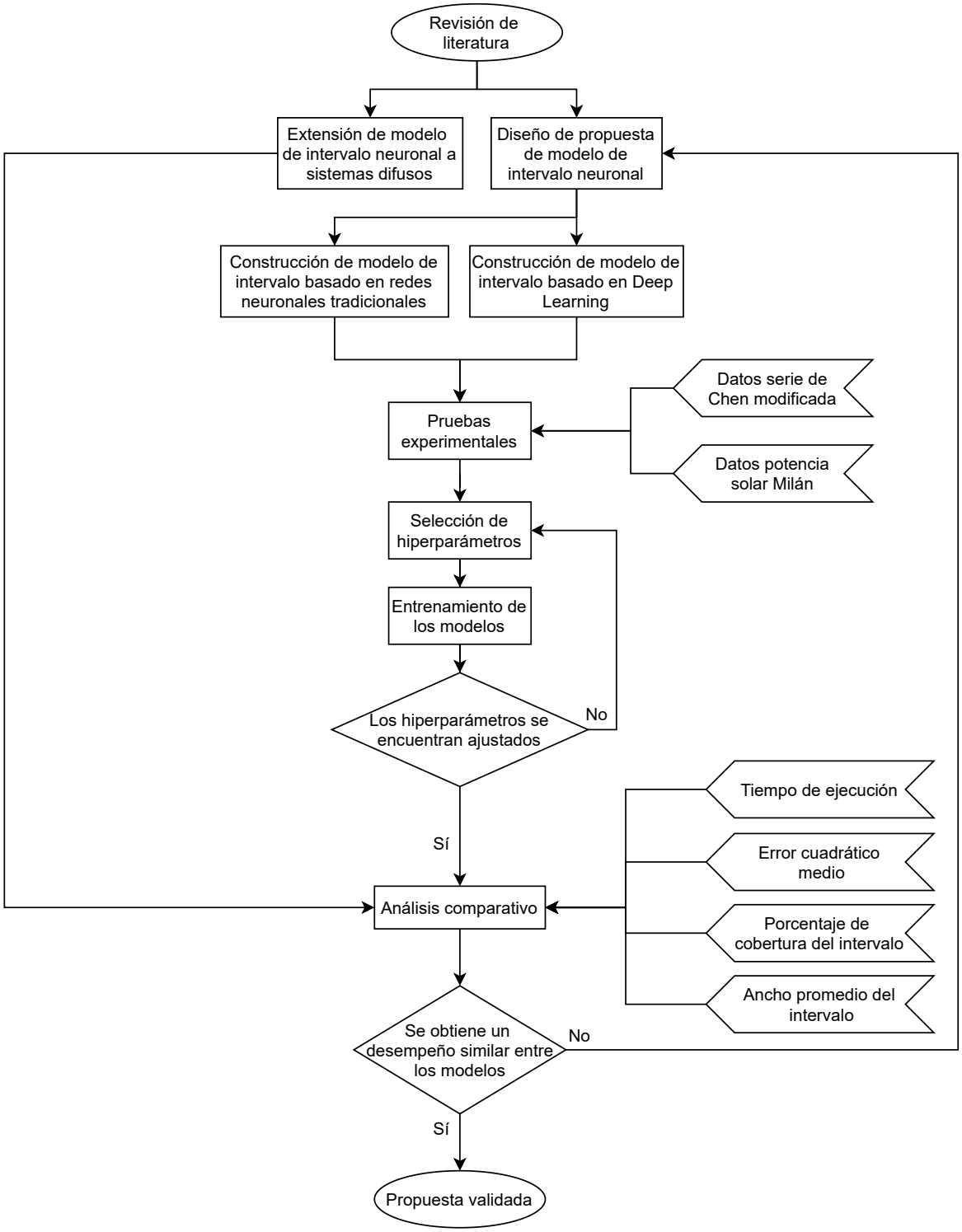


Figura 3.1: Estructura de la metodología de trabajo

difusos.

Finalmente, una vez entrenados los modelos para cada caso de estudio, se realizó un análisis comparativo entre las variantes de la propuesta neuronal, la propuesta difusa, y el método del estado del arte que fue utilizado como base para el diseño de estas. En este paso, se estudió

el desempeño de los modelos en los ámbitos de costo computacional, error de predicción, y ancho y porcentaje de cobertura del intervalo. De esta manera, según esta metodología las propuestas planteadas en este trabajo serían validadas si los resultados de precisión obtenidos por los modelos logran ser similares al estado del arte, pero a la vez presentan un menor costo computacional que la alternativa existente.

Capítulo 4

Modelos Propuestos

En este capítulo se procede a describir detalladamente las dos principales propuestas planteadas para cumplir los objetivos de este trabajo de Tesis. Estas consisten en una nueva metodología que permite construir modelos de intervalo *Joint Supervision* a partir de sistemas difusos Takagi-Sugeno, junto con el diseño de un nuevo método de construcción de intervalos de predicción neuronales basado en la implementación de una nueva función de costo inspirada en el método de *Joint Supervision*, denominado *Joint Supervision* Selectivo.

Dentro de las descripciones de cada método se puede encontrar una explicación de su fundamentación teórica, donde se especifican aspectos como la arquitectura de los modelos predictivos, la formulación matemática e implementación computacional de la función de costos, y la metodología de identificación de parámetros. Además, debido a la complejidad del planteamiento matemático del método de *Joint Supervision* Selectivo, esta propuesta contiene una subsección adicional donde se estudia su aplicabilidad en casos de estudio reales.

4.1. Modelo de intervalo difuso basado en *Joint Supervision*

Con el propósito de corroborar la aplicabilidad del método de Joint Supervision descrito en la sección 2.2.2 a otros sistemas basados en inteligencia computacional, y a la vez disponer de un punto de comparación adicional al realizar pruebas experimentales, en este trabajo se propone una metodología para adaptar la estructura de modelos difusos Takagi-Sugeno con el objetivo de implementar el método de Joint Supervision [78].

En primer lugar, se propuso una nueva formulación multi-salida para modelos Takagi-Sugeno con el propósito de imitar la habilidad de las redes neuronales de compartir sus pesos entre sus salidas. Suponiendo que un modelo Takagi-Sugeno se encuentra formado por un sistema de reglas del tipo

$$R^j : \text{Si } z_k^{(1)} \text{ es } A_j^{(1)} \text{ y ... y } z_k^{(n)} \text{ es } A_j^{(n)} \text{ entonces} \quad \hat{y}_j(z_k) = \theta_j^{(0)} + \theta_j^{(1)} z_k^{(1)} + ... + \theta_j^{(n)} z_k^{(n)}, \tag{4.1}$$

donde $A_j^{(\mathrm{i})}$ corresponde al conjunto difuso asociado a la i-ésima componente del vector de entradas para la j-ésima regla, $\hat{y}_j(z_k)$ es la salida local de la j-ésima regla del modelo, $\theta_j = [\theta_j^{(0)}, \theta_j^{(1)}, ..., \theta_j^{(n)}]$ es el vector de parámetros de las consecuencias de la j-ésima regla del modelo, n es el número total de componentes del vector de entrada, y $z_k = [z_k^{(1)}, z_k^{(2)}, ..., z_k^{(n)}]$ es el vector de entradas en el instante k-ésimo, el cual está compuesto de una serie de auto-regresores y, en algunos casos, variables exógenas, según se muestra en 1.9, la salida total del modelo puede calcularse como

$$\hat{y}(z_k) = \sum_{j=1}^{M} \beta_j(z_k) \hat{y}_j(z_k), \tag{4.2}$$

donde $\hat{y}(z_k)$ es la salida predicha para el vector de entradas z_k y $\beta_j(z_k)$ es el grado de activación normalizado de la regla j-ésima para el vector de entradas z_k.

Basándose en esta notación, para el caso de un modelo de Joint Supervision difuso se define una arquitectura Takagi-Sugeno con 3 salidas donde todas provienen de un mismo conjunto de premisas:

$$\overline{y}(z_k) = \sum_{j=1}^{M} \beta_j(z_k) \overline{y}_j(z_k), \tag{4.3}$$

$$\hat{y}(z_k) = \sum_{j=1}^{M} \beta_j(z_k) \hat{y}_j(z_k), \tag{4.4}$$

$$\underline{y}(z_k) = \sum_{j=1}^{M} \beta_j(z_k) \underline{y}_j(z_k), \tag{4.5}$$

donde

$$\overline{y}_j(z_k) = [1 \ z_k^T] \overline{\theta}_j, \tag{4.6}$$

$$\hat{y}_j(z_k) = [1 \ z_k^T] \hat{\theta}_j, \tag{4.7}$$

$$\underline{y}_j(z_k) = [1 \ z_k^T] \underline{\theta}_j, \tag{4.8}$$

y los vectores $\{\overline{\theta}_j, \hat{\theta}_j, \underline{\theta}_j\}$ representan los vectores de parámetros de las consecuencias de la j-ésima regla asociados a cada una de las 3 salidas del modelo, donde dos están encargadas

del cálculo de los límites del intervalo $(\overline{y}(z_k), \underline{y}(z_K))$, y una se dedica a la predicción del valor esperado de la señal $(\hat{y}(z_k))$.

Es importante notar que, al igual que con la variante neuronal del método de *Joint Supervision*, esta formulación multi salida le entrega al modelo difuso la capacidad de compartir parte de sus parámetros a lo largo de sus salidas. Esta propiedad permite que las arquitecturas usadas para este método de intervalo puedan aprovechar la información compartida a lo largo de sus salidas para ser más eficientes en cuanto a su densidad paramétrica.

Adicionalmente, la estructura difusa propuesta en las ecuaciones (4.3)-(4.8) también tiene un sentido práctico. Esto se debe a que, según la teoría de modelamiento difuso, los conjuntos y reglas difusas definidas en (4.1) representan las zonas de operación del sistema para las cuales cada salida lineal local del modelo es válida. Lo anterior significa que el diseñar una formulación multi salida donde se comparten las premisas permite seguir la intuición de que todas las salidas del modelo obedecen a las mismas zonas de operación en los datos.

Con respecto a la metodología de identificación de parámetros, como se puede apreciar en la figura 4.1, el algoritmo consiste primeramente en ejecutar un análisis de sensibilidad con los datos de entrenamiento para determinar el conjunto óptimo de regresores a utilizar como entrada del modelo, basándose en el procedimiento planteado en [79]. Luego, de acuerdo a lo planteado por Sugeno y Yasukawa [80], el número óptimo de reglas difusas junto con las funciones de membresía de cada conjunto difuso son estimadas mediante un algoritmo de clusterización difusa en el mismo conjunto de datos. Posteriormente, los vectores de parámetros de las consecuencias $\{\overline{\theta}_j, \hat{\theta}_j, \underline{\theta}_j\}$ son calculados por medio de la optimización de las funciones de costo *Joint Supervision*:

$$\overline{L} = \frac{1}{N}\sum_{k=1}^{N}(y_k - \overline{y}(z_k))^2 + \lambda\frac{1}{N}\sum_{k=1}^{N} ReLU^2(y_k - \overline{y}(z_k)) \tag{4.9}$$

$$\hat{L} = \frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}(z_k))^2 \tag{4.10}$$

$$\underline{L} = \frac{1}{N}\sum_{k=1}^{N}(y_k - \underline{y}(z_k))^2 + \lambda\frac{1}{N}\sum_{k=1}^{N} ReLU^2(\underline{y}(z_k) - y_k), \tag{4.11}$$

Finalmente, al igual que con la variante neuronal de este método, es necesario realizar un proceso iterativo de ajuste hiperparamétrico. En esta implementación se ejecuta un algoritmo de búsqueda logarítmica donde en cada iteración se entrena un nuevo modelo con un valor determinado para el hiperparámetro λ por medio de optimización por enjambre de partículas. Una vez entrenado el modelo, el algoritmo procede a calcular su porcentaje de cobertura y lo compara con el valor deseado por el usuario. Si la métrica obtenida es de un valor suficientemente cercano al deseado, la búsqueda finaliza. Si no, el valor del hiperparámetro

es actualizado según la expresión

$$\lambda^{(\mathrm{i}+1)} = \begin{cases} 2\lambda^{(\mathrm{i})} & \text{si } \lambda_{up} = 0 \wedge \mathrm{PICP} < (1-\alpha) \\ \frac{\lambda^{(\mathrm{i})}+\lambda_{low}}{2} & \text{si PICP} > (1-\alpha) \\ \frac{\lambda^{(\mathrm{i})}+\lambda_{up}}{2} & \text{si PICP} < (1-\alpha), \end{cases} \tag{4.12}$$

donde $\lambda^{(\mathrm{i})}$ corresponde al valor del hiperparámetro en la iteración i-ésima de la búsqueda logarítmica, PICP corresponde al porcentaje de cobertura del modelo calculado según la ecuación (1.6), $(1-\alpha)$ corresponde al porcentaje de cobertura deseado por el usuario, y $\{\lambda_{up}, \lambda_{low}\}$ corresponden a valores pivote que son inicializados en cero y almacenan el menor valor histórico de λ que excede el valor deseado $(1-\alpha)$ y el mayor valor histórico que subestima el límite deseado, respectivamente. Una vez finalizada este búsqueda, el proceso finaliza con una fase adicional de reentrenamiento (permaneciendo constante el valor del hiperparámetro), en donde se vuelve a ejecutar la identificación de parámetros un cierto número de veces y se conserva el modelo que obtenga el intervalo más delgado. Este último paso se realiza con el objetivo de disminuir la incerteza asociada a la inicialización de los parámetros, dado que el entrenamiento de un modelo difuso es un proceso no determinístico.

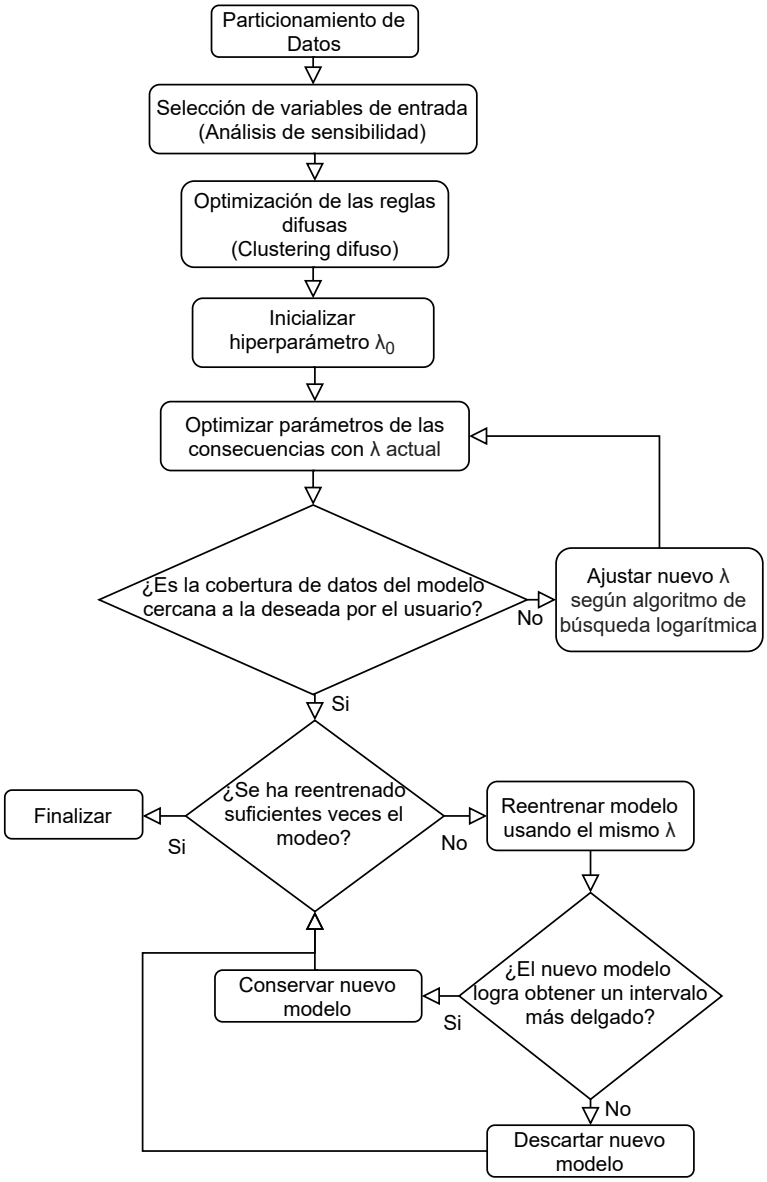


Figura 4.1: Rutina de identificación de parámetros para el método de *Joint Supervision* basado en modelos difusos

4.2. Modelo de intervalo neuronal basado en *Joint Supervision* Selectivo y *Deep Learning*

4.2.1. Descripción del modelo

En este trabajo de tesis se propone una nueva metodología basada en *Joint Supervision* para la obtención de intervalos de predicción a partir de modelos neuronales, que además es compatible con herramientas de *Deep Learning*. Este método consiste en el entrenamiento de una arquitectura de red neuronal *Joint Supervision* (es decir, con tres neuronas en la capa de salida), cuya identificación de parámetros es realizada por medio de la optimización de nuevas funciones de costo para las salidas responsables de calcular el intervalo:

$$\overline{L}_{sel}(\alpha,\overline{y},\underline{y},y)=\frac{1}{N}\sum_{k=1}^{N}(y_k-\overline{y}(z_k))^2+\beta\frac{1}{N}\sum_{k=1}^{N}ReLU^2(y_k-\overline{y}(z_k))\cdot f_S(\alpha,\overline{y},\underline{y},y_k) \tag{4.13}$$

$$\hat{L}(\hat{y},y)=\frac{1}{N}\sum_{k=1}^{N}(y_k-\hat{y}(z_k))^2 \tag{4.14}$$

$$\underline{L}_{sel}(\alpha,\overline{y},\underline{y},y)=\frac{1}{N}\sum_{k=1}^{N}(y_k-\underline{y}(z_k))^2+\beta\frac{1}{N}\sum_{k=1}^{N}ReLU^2(\underline{y}(z_k)-y_k)\cdot f_S(\alpha,\overline{y},\underline{y},y_k), \tag{4.15}$$

donde $(1-\alpha)$ es el porcentaje deseado de datos capturados por el intervalo, β es un hiperparámetro que no requiere ser ajustado iterativamente, $f_S(\cdot)$ es una función indicatriz de la forma

$$f_S(\alpha,\overline{y},\underline{y},y_k)=\begin{cases}1, & y_k\in S_{N_p}(\overline{y},\underline{y})\\ 0, & \text{otherwise}\end{cases} \tag{4.16}$$

$$N_p=N([1-\alpha]-\text{PICP}), \tag{4.17}$$

$S_{N_p}(\overline{y},\underline{y})$ es el conjunto de los N_p puntos no capturados más cercanos al intervalo dado por los arreglos $(\overline{y},\underline{y})$

$$S_{N_p}(\overline{y},\underline{y})=\{y_k|0<\mathrm{d}_{int}(y_k,\overline{y},\underline{y})\leq\mathrm{d}_{int}(y_{sort}(N_p),\overline{y},\underline{y})\} \tag{4.18}$$

$$S_{N_p}(\overline{y},\underline{y})=\emptyset,\ N_p\leq 0, \tag{4.19}$$

$\mathrm{d}_{int}(y_k,\overline{y},\underline{y})$ es la distancia del dato y_k al intervalo formado por los arreglos $(\overline{y},\underline{y})$

$$\mathrm{d}_{int}(y_k,\overline{y},\underline{y})=\text{máx}(0,[y_k-\overline{y}(z_k)],[\underline{y}(z_k)-y_k]), \tag{4.20}$$

y y_{sort} es el vector de los puntos no capturados ordenados según su distancia al intervalo $(\overline{y},\underline{y})$, de manera que

$$\mathrm{d}_{int}(y_{sort}(k),\overline{y},\underline{y})\leq\mathrm{d}_{int}(y_{sort}(k+1),\overline{y},\underline{y}) \tag{4.21}$$

Al comparar estas funciones de costo con las utilizadas por el método de *Joint Supervision* tradicional mostradas en (2.43) - (2.45), se puede apreciar que el nuevo costo de intervalo propuesto, denominado costo de intervalo selectivo, tiene una forma muy parecida a su variante original, salvo la inclusión de la función indicatriz $f_S(\cdot)$. Este componente representa

la diferencia crucial que permite mitigar las desventajas del método de *Joint Supervision* tradicional.

Como se explicó en la sección 2.2.2, la variante tradicional de este método utiliza dos componentes de costo: Una cuyo objetivo es la disminución del ancho del intervalo y del error de predicción por medio de la aplicación de una penalización por error cuadrático medio a todos los puntos de entrenamiento, y otra con el objetivo de aumentar el ancho del intervalo por medio de la aplicación de una penalización adicional por error cuadrático medio sólo a aquellos puntos que no son capturados por el modelo. Luego, para ajustar el porcentaje de cobertura del intervalo, se aplica un ponderador al segundo componente, cuyo valor se determina mediante una búsqueda iterativa.

Sin embargo, la inclusión de la función $f_S(\cdot)$ en el método propuesto implica que la penalización complementaria aplicada a las salidas responsables de la predicción del intervalo ya no tomará en cuenta a todos los puntos que se encuentren fuera de este. En cambio, la componente de costo propuesta sólo considera las $N([1-\alpha]-\text{PICP})$ muestras más cercanas al intervalo, donde $N([1-\alpha]-\text{PICP})$ corresponde a la cantidad de puntos necesarios para cumplir con la restricción de porcentaje de cobertura deseado para el modelo. Esto significa que tanto la optimización del modelo predictivo como el ajuste de la cobertura de datos del intervalo ocurren dentro de un mismo proceso de optimización, eliminando la necesidad de procedimientos iterativos adicionales.

En cuanto a la implementación de este método, si bien el planteamiento matemático del nuevo costo presentado en las ecuaciones (4.13) - (4.21) puede parecer complejo, su implementación computacional es más sencilla. A continuación, en el algoritmo 1 se muestran

los pasos a seguir para el cálculo computacional de la nueva componente de costo de intervalo.

Algorithm 1: Cálculo de la nueva componente de costo de intervalo selectivo.

Result: Componente de costo de intervalo selectivo $\overline{C}_{sel}$,$\underline{C}_{sel}$
Entradas: Datos de entrenamiento y, intervalo predicho $\{\overline{y}, \underline{y}\}$, porcentaje de cobertura deseado $(1-\alpha)$;
Calcular porcentaje de cobertura actual del intervalo según ecuación (1.6);
if $PICP \geq (1-\alpha)$ **then**
  $\overline{C}_{sel} = \underline{C}_{sel} = 0$;
else
  Calcular cantidad de puntos necesarios para cumplir con la restricción de porcentaje de cobertura $N_p = N([1-\alpha] - \text{PICP})$;
  Identificar todos los puntos que no están siendo capturados por el intervalo y calcular su distancia a este según la ecuación (4.20);
  Ordenar los puntos no capturados según su distancia al intervalo, de menor a mayor, de manera que se cumpla la relación mostrada en (4.21);
  for i $\leftarrow 0$ **to** N_p **do**
    if y_{sort}(i) *está sobre el intervalo* **then**
      Se penaliza la muestra correspondiente de la salida $\overline{y}$ con un error cuadrático medio adicional
    else
      Se penaliza la muestra correspondiente de la salida $\underline{y}$ con un error cuadrático medio adicional
    end
  end
end

Con respecto a la metodología de identificación de parámetros, en la figura 4.2 se puede apreciar un diagrama resumen de los pasos a ejecutar para entrenar modelos de *Joint Supervision* Selectivo basados en redes neuronales *feedforward*. Si bien existen varias semejanzas con la mitad superior de la metodología utilizada por las variantes tradicional y difusa de este método, mostrada en la figura 4.1, es en la última mitad del diagrama donde se puede notar el ahorro de tiempo de entrenamiento que podría aportar esta propuesta al eliminar los pasos relacionados con el procedimiento de búsqueda iterativa.

Por otro lado, si se desea diseñar un modelo de *Joint Supervision* Selectivo basado en redes neuronales recurrentes, tales como las redes LSTM, y *Deep Learning*, el procedimiento de identificación se debe realizar como se muestra en la figura 4.3. Al observar el diagrama se puede notar que la metodología utilizada es similar a la presentada en la figura 4.2, con la excepción de que no se incluye el paso correspondiente a la selección de variables de entrada. Esto se debe a que, como se explicó en la sección 1.1.2, las arquitecturas neuronales recurrentes incorporan este proceso dentro del entrenamiento de modelo.

4.2.2. Análisis de la propuesta

Si bien en el apartado anterior se explicó en detalle el fundamento matemático y las bases de la implementación computacional del método de *Joint Supervision* Selectivo, también es

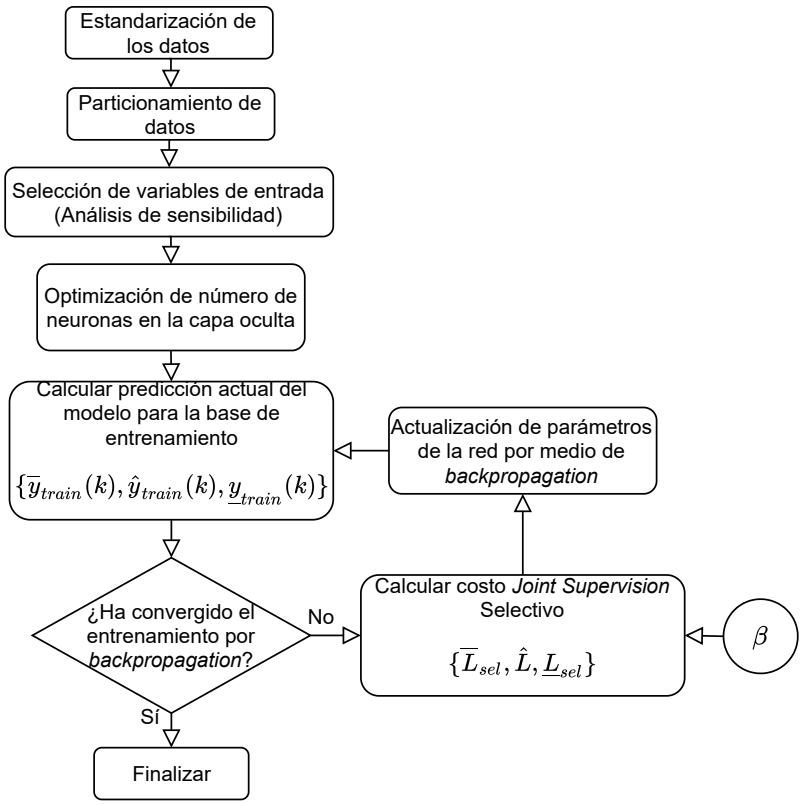


Figura 4.2: Rutina de identificación de parámetros para el método de *Joint Supervision* Selectivo basado en redes neuronales *feedforward*

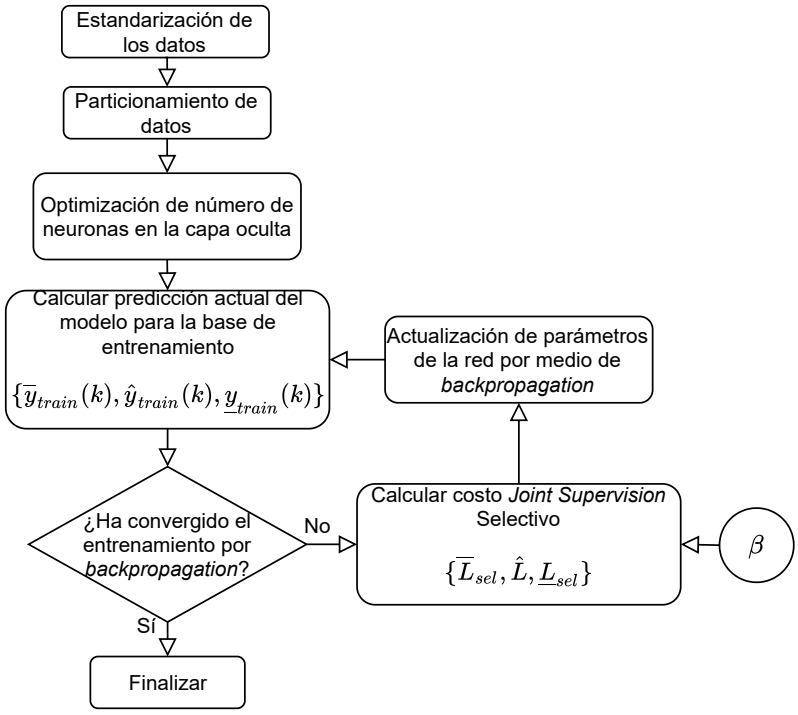


Figura 4.3: Rutina de identificación de parámetros para el método de *Joint Supervision* Selectivo basado en redes neuronales recurrentes

necesario estudiar y explicar ciertos aspectos relevantes de esta propuesta para comprender su funcionamiento a cabalidad. Debido a esto, en esta sección se procede a analizar aquellas componentes de este método que requieran de una examinación más profunda.

En primer lugar, es necesario estudiar la diferenciabilidad de la nueva función de costos. Esto se debe a que un requisito fundamental para la utilización de técnicas de *Deep Learning* es el uso de costos que sean compatibles con optimizadores basados en descenso por gradiente, y dada la expresión matemática del costo de intervalo selectivo presentado en (4.13) y (4.15), es posible que existan regiones con discontinuidades que prohíban la aplicación de estos algoritmos. Sin embargo, si se observa el desarrollo

$$\frac{\mathrm{d}\overline{C}_{sel}}{\mathrm{d}\overline{y}} = \frac{\mathrm{d}}{\mathrm{d}\overline{y}}\left(\frac{1}{N}\sum_{k=1}^{N} ReLU(y(k)-\overline{y}(z_k))^2 \cdot f_S(\alpha,\overline{y},\underline{y},y(k))\right) \tag{4.22}$$

$$= \frac{\mathrm{d}}{\mathrm{d}\overline{y}}\left(\frac{1}{N}\sum_{k=1}^{N} ReLU(y(k)-\overline{y}(z_k))^2\right) \cdot f_S(\alpha,\overline{y},\underline{y},y(k)) \tag{4.23}$$

$$+ \frac{1}{N}\sum_{k=1}^{N} ReLU(y(k)-\overline{y}(z_k))^2 \cdot \frac{\mathrm{d}}{\mathrm{d}\overline{y}}\left(f_S(\alpha,\overline{y},\underline{y},y(k))\right) \tag{4.24}$$

$$= D_1 + D_2, \tag{4.25}$$

donde

$$D_1 = \frac{\mathrm{d}}{\mathrm{d}\overline{y}}\left(\frac{1}{N}\sum_{k=1}^{N} ReLU(y(k)-\overline{y}(z_k))^2\right) \cdot f_S(\alpha,\overline{y},\underline{y},y(k)) \tag{4.26}$$

$$D_2 = \frac{1}{N}\sum_{k=1}^{N} ReLU(y(k)-\overline{y}(z_k))^2 \cdot \frac{\mathrm{d}}{\mathrm{d}\overline{y}}\left(f_S(\alpha,\overline{y},\underline{y},y(k))\right) \tag{4.27}$$

a partir de lo cual se puede notar que

$$D_1 = \begin{cases} 2(y(k)-\overline{y}(z_k)), & (y(k)-\overline{y}(z_k)) > 0 \wedge y(k) \in S_{N([1-\alpha]-\mathrm{PICP})}(\overline{y},\underline{y}) \\ 0, & \text{otherwise,} \end{cases} \tag{4.28}$$

se puede apreciar que, a pesar de la complejidad del conjunto S_{N_p}, el cálculo de D_2 se reduce al cálculo de la derivada de una función indicatriz. Esto significa que el término sólo puede tomar un valor igual a cero (en regiones donde la función indicatriz sea constante) o indefinido (en las regiones puntuales donde la función cambia de valor). A pesar de que estos últimos puntos violan el principio de diferenciabilidad en teoría, es común en las implementaciones computacionales tomar una de las derivadas laterales de la función en caso de caer en una de estas zonas puntuales.

De esta manera, continuando con el desarrollo planteado en (4.22)-(4.28), el cálculo computacional del gradiente total de la componente superior de costo de intervalo selectivo $\overline{C}_{sel}$ resulta

$$\frac{\mathrm{d}\overline{C}_{sel}}{\mathrm{d}\overline{y}} = \frac{\mathrm{d}}{\mathrm{d}\overline{y}}\left(\frac{1}{N}\sum_{k=1}^{N} ReLU(y(k)-\overline{y}(z_k))^2\right) \cdot f_S(\alpha,\overline{y},\underline{y},y(k)), \tag{4.29}$$

la cual es una función compatible con optimizadores basados en métodos de gradiente.

Por otro lado, también es necesario estudiar la inclusión del ponderador β dentro de las funciones de costo propuestas en (4.13) y (4.15). Esto se debe a que, mientras que en el método de *Joint Supervision* tradicional se incluye un ponderador equivalente para ajustar el porcentaje de cobertura del intervalo, este no debiese el caso de la variante propuesta, puesto que es la componente de costo de intervalo selectivo quien debiese estar a cargo de realizar este ajuste.

En el caso del método de *Joint Supervision* Selectivo, el ponderador β es incluido para garantizar su convergencia. Esto se puede explicar al analizar el comportamiento de las nuevas funciones de costo de intervalo en escenarios donde el modelo se encuentre cerca de cumplir con la restricción de porcentaje de cobertura. En estas situaciones ocurrirá que, a medida que el ancho del intervalo predicho por el modelo crece para intentar cumplir esta condición, la componente de costo de intervalo selectivo C_{sel} tenderá a disminuir su valor. A su vez, mientras esto sucede, la componente de costo que aplica una penalización de error cuadrático medio a todos los puntos aumentará a medida que el crecimiento del intervalo lo aleje de los puntos que ya se encuentran capturados, favoreciendo la disminución de su ancho. Lo anterior trae como consecuencia que el intervalo no sea capaz de converger al porcentaje de cobertura deseado, puesto que, antes de que esto ocurra, la componente de la función de costo responsable de disminuir su ancho comienza a preponderar, resultando en un modelo que sólo logra oscilar al límite de la convergencia deseada.

De esta manera, para solucionar este problema se incluye el ponderador β en las funciones de costo de intervalo selectivo mostradas en las ecuaciones (4.13) y (4.15). En cuanto al procedimiento utilizado para obtener el valor de este coeficiente, es necesario notar que existe una dependencia entre la dispersión de la base de datos de entrenamiento y el valor de β, donde una mayor dispersión significará un mayor valor del ponderador. Sin embargo, como el escenario de tener que ajustar esta variable para cada caso de estudio no es deseable, se propone la aplicación de una estandarización de los datos como un paso de preprocesamiento adicional, de manera que sea posible obtener empíricamente un único valor de uso general para β.

Un último aspecto que resulta de interés analizar para corroborar la aplicabilidad del método propuesto es el comportamiento de la función de costos en casos de borde, principalmente en los casos en que el intervalo tiene un ancho nulo (o sea, las tres salidas de la red neuronal son iguales), y en los casos donde el intervalo supera el porcentaje de cobertura deseado.

En el primer caso de borde, se puede apreciar que el comportamiento de la función de costos es similar al de un modelo *Joint Supervision* tradicional: Si bien las tres salidas de la red neuronal comparten la misma componente de costo que penaliza todos los datos según su error cuadrático, tendrán una distinta componente de costo de intervalo selectivo C_{sel}. Esto se debe a que la salida responsable de calcular el límite superior del intervalo sólo recibirá una penalización adicional sobre aquellos puntos que se encuentren sobre el intervalo, mientras que la salida responsable de calcular el límite inferior sólo recibe una penalización adicional sobre los puntos que se encuentran bajo el intervalo. Esto se traduce en que el ancho del intervalo comenzará a aumentar, de manera que su porcentaje de cobertura aumente.

En el segundo caso de borde, cuando el intervalo presenta un porcentaje de cobertura

mayor al deseado, se puede apreciar que se cumplirá la desigualdad

$$\text{PICP} \geq (1 - \alpha) \tag{4.30}$$

por lo que

$$N([1 - \alpha] - \text{PICP}) \leq 0 \tag{4.31}$$

lo que se traduce en que, por definición,

$$S_{N([1-\alpha]-PICP)} = \emptyset. \tag{4.32}$$

Esto último significa que

$$f_s(\alpha, \overline{y}, \underline{y}, y(k)) = 0 \ \forall k \implies \overline{C}_{sel} = \underline{C}_{sel} = 0, \tag{4.33}$$

por lo que el costo total de entrenamiento cuando el intervalo presenta un mayor porcentaje de cobertura al deseado es

$$\overline{L}_{sel}(\alpha, \overline{y}, \underline{y}, y) = \frac{1}{N} \sum_{k=1}^{N} (y_k - \overline{y}(z_k))^2 \tag{4.34}$$

$$\hat{L}(\hat{y}, y) = \frac{1}{N} \sum_{k=1}^{N} (y_k - \hat{y}(z_k))^2 \tag{4.35}$$

$$\underline{L}_{sel}(\alpha, \overline{y}, \underline{y}, y) = \frac{1}{N} \sum_{k=1}^{N} (y_k - \underline{y}(z_k))^2, \tag{4.36}$$

Como consecuencia de esto, mientras el intervalo predicho por el modelo tenga un porcentaje de cobertura mayor al deseado, sólo actuará la componente de costo que penaliza a todos los puntos según su error cuadrático medio. En otras palabras, el intervalo intenta converger hacia el valor esperado de la señal para disminuir su ancho. Este fenómeno se mantendrá hasta que el porcentaje de cobertura del modelo vuelva a ser menor al deseado, momento en el cual volverá a actuar la componente de costo de intervalo selectivo para nuevamente intentar de aumentar el ancho del intervalo.

4.3. Discusión

En el presente capítulo se mostraron los dos principales aportes de este trabajo de tesis, describiendo detalladamente su funcionamiento y justificando su aplicabilidad en caso de ser necesario. En primer lugar, se planteó una nueva metodología para compatibilizar el método de *Joint Supervision* neuronal con sistemas difusos Takagi-Sugeno multi salida, con el objetivo de generar modelos de intervalo que aprovechen la interpretabilidad de estas arquitecturas, junto con además poder contar con un punto de referencia adicional al momento de realizar pruebas comparativas. Por el otro lado, también se propuso un nuevo método de construcción de intervalos de predicción neuronales, denominado *Joint Supervision* Selectivo, el cual se basa en la utilización de una nueva función de costo inspirada en el método de *Joint Supervision*, pero que intenta solucionar los problemas de convergencia y costo computacional de su variante original.

Sin embargo, para comprobar las hipótesis planteadas para este último método es necesario diseñar pruebas experimentales que midan su desempeño en distintas aplicaciones, comparando la capacidad predictiva (por medio de la medición del error de predicción), calidad del intervalo (por medio de la medición de su porcentaje de cobertura y ancho promedio), y costo computacional de este modelo con respecto a sus versiones original y difusa.

Capítulo 5

Simulaciones y Resultados

Con el objetivo de estudiar el desempeño de los modelos propuestos, se implementaron dos simulaciones experimentales: La predicción de muestras de una base de datos artificial basada en la serie de Chen modificada [81], y el pronóstico de generación de potencia solar de una instalación ubicada en el Politecnico di Milano, Milán, Italia [82]. En ambos casos, se entrenaron modelos de intervalo para obtener una cobertura del 90 % de los datos ante distintos horizontes de predicción. Adicionalmente, con el objetivo de tener un punto de comparación para validar las propuestas diseñadas, también se entrenaron modelos de intervalo basados en *Joint Supervision* tradicional, el cual presenta un desempeño comparable a otros métodos de construcción de intervalo neuronal del estado del arte [83]. Finalmente, es importante mencionar que para el caso de los métodos basados en redes neuronales, estos fueron probados con tres tipos de arquitecturas basales distintas: Una basada en una única red neuronal *feedforward*, otra basada en una única red neuronal LSTM entrenada con *gradient clipping* para mejorar la convergencia de su entrenamiento, y una basada en un conjunto *bootstrap* de 10 redes neuronales *feedforward*. Por lo tanto, si bien en este trabajo sólo se comparan 3 métodos de intervalo distintos, la cantidad total de modelos a comparar es de 7 (ambos métodos neuronales fueron probados con 3 tipos de modelos base distintos).

Para comparar el desempeño de los modelos implementados, en cada uno de estos experimentos se utilizaron 4 métricas distintas. Estas consistieron en el error cuadrático medio, calculado según la ecuación (1.3) para comparar la capacidad predictiva de los modelos, el porcentaje de cobertura del intervalo, calculado según la ecuación (1.6)-(1.7) para determinar el grado de cumplimiento de la condición de cobertura de datos, el ancho promedio del intervalo, calculado según (1.4)-(1.5) para estimar el grado de informatividad del intervalo (mientras más delgado mejor), y el tiempo total de entrenamiento, con el objetivo de estimar el costo computacional de cada método. A continuación, se presentan los resultados obtenidos en los casos de estudio propuestos para este trabajo.

5.1. Experimento 1: Serie de Chen modificada

Para este experimento se utilizaron datos generados artificialmente a partir de la serie de Chen modificada, la cual tiene la forma

$$
\begin{aligned}
y(k) = \quad & [0{,}8 - 0{,}5 \exp\left(-y^2(k-1)\right)]\, y(k-1) - \\
& [0{,}3 - 0{,}9 \exp\left(-y^2(k-1)\right)]\, y(k-2) + \\
& u(k-1) + 0{,}2\ u(k-2) + \\
& 0{,}1\ u(k-1)u(k-2) + \mathrm{e}(k),
\end{aligned}
\tag{5.1}
$$

donde $\mathrm{e}(k)$ representa una componente de ruido que depende del estado anterior de la señal

$$
\mathrm{e}(k) = 0{,}5 \exp\left(-y(k-1)^2\right) \beta(k), \tag{5.2}
$$

y β es una señal de ruido blanco.

Como se puede apreciar en la ecuación (5.1), la serie de Chen modificada se caracteriza por ser un sistema dinámico con variable exógena, que además presenta un comportamiento altamente no-lineal. Esto la hace un candidato ideal para estudiar la capacidad predictiva de modelos con la capacidad de ser aproximadores universales, como lo son las redes neuronales y los sistemas difusos.

A partir de la simulación de esta señal, se generaron un total de 10.000 muestras para este estudio, de las cuales el primer 60 % fue utilizado para el entrenamiento de los modelos (conjunto de entrenamiento), el siguiente 20 % fue usado para la optimización estructural (conjunto de prueba), y el último 20 % fue destinado a la validación de los modelos obtenidos (conjunto de validación). Utilizando estos conjuntos, se entrenaron los tres distintos métodos de intervalo para generar predicciones a 1 y 16 pasos a futuro.

En las tablas 5.1 y 5.2 se pueden apreciar las configuraciones hiperparamétricas finales obtenidas para el modelo difuso y cada uno de los 6 modelos neuronales implementados en este trabajo, respectivamente. Como se puede apreciar en la tabla 5.2, la gran mayoría de los hiperparámetros se mantuvieron constantes a lo largo de los modelos. Esto se debe a que para el experimento se intentó contar con configuraciones lo más parecidas posibles, con el objetivo de que sus diferencias en desempeño puedan ser principalmente atribuibles al tipo de modelo y método de construcción de intervalo utilizado. Sin embargo, es importante destacar las diferencias en el número de entradas usadas para los modelos LSTM en comparación al resto de los modelos: Debido a que las redes recurrentes siempre ocupan como entrada el valor inmediatamente anterior de la serie a predecir, el número de entradas de estos modelos siempre será igual al número de series utilizadas como entrada. En cambio, las implementaciones basadas en redes *feedforward* utilizan un análisis de sensibilidad para determinar el número óptimo de regresores a usar como entrada del modelo.

Los resultados obtenidos en este experimento se encuentran desglosados en los siguientes gráficos: En primer lugar, la figura 5.1 muestra el intervalo de predicción de 16 pasos obtenido por el modelo difuso implementado. Posteriormente, en la figura 5.2 se pueden apreciar los intervalos de predicción de 16 pasos obtenidos por los modelos neuronales implementados. Luego, la figura 5.3 muestra las métricas de desempeño (RMSE, PICP y PINAW) obtenidas por cada uno de los métodos. Finalmente, en la figura 5.4 se pueden apreciar los tiempos de entrenamiento medidos. La información detallada de todos los resultados obtenidos para este experimento puede ser encontrada en el apéndice A. A continuación, se procederá a discutir y analizar las observaciones más importantes que se pudo notar a partir de estos resultados.

Tabla 5.1: Configuración hiperparamétrica final obtenida para el modelo de *Joint Supervision* basado en sistemas difusos en el experimento de la serie de Chen modificada

Hiperparámetro	Valor
Número de entradas	2 (autoregresores) 2 (exógenas)
Número de reglas	3
Algoritmo de clustering difuso	Gustafson Kessel
Funciones de membresía	Gaussianas
Tipo de consecuencias	Lineales

Al analizar los intervalos mostrados en las figuras 5.1 y 5.2 es posible notar que tanto los modelos basados en redes *feedforward*, ya sea utilizando el método de *Joint Supervision* tradicional o el Selectivo, como el basado en sistemas difusos presentan un comportamiento similar. En estos gráficos también es posible identificar claramente la capacidad adaptativa del método de *Joint Supervision* al ajustar dinámicamente el ancho del intervalo según la magnitud que tome la componente de ruido de la señal. Además, también es importante destacar que en zonas cercanas a los cambios bruscos de valor en la señal los datos tienden a presentar una dispersión asimétrica, por lo que en estas zonas es posible aprovechar la asimetría del intervalo *Joint Supervision* para capturar de una manera más eficiente este comportamiento. Por el otro lado, si se observa los intervalos mostrados en las figuras 5.2c y 5.2d, se puede apreciar que las implementaciones basadas en redes LSTM logran obtener intervalos con una apariencia mucho más delgada que el resto de los modelos, lo cual es consistente con lo planteado en la teoría con respecto a la mejora en capacidad predictiva entregada por los modelos recurrentes y las técnicas de *Deep Learning*.

Adicionalmente, a partir de los resultados mostrados en la figura 5.3 es posible observar que todos los métodos probados en este experimento fueron capaces de cumplir con la restricción impuesta de 90 % de porcentaje de cobertura para el intervalo. A su vez, también se puede ver que las implementaciones basadas en redes LSTM destacan por obtener valores levemente inferiores de error cuadrático medio y ancho promedio del intervalo, indicando que poseen una mejor capacidad predictiva y una mayor informatividad del intervalo, lo cual corrobora el análisis preliminar realizado al observar las figuras 5.2c y 5.2d. Por el otro lado, al estudiar el desempeño de los modelos basados en conjuntos *bootstrap*, en este experimento no se pudo apreciar ninguna diferencia significativa al utilizar esta variante.

En cuanto al costo computacional del entrenamiento, en la figura 5.4 se puede apreciar que, si bien no existen mayores diferencias entre los modelos basados en redes *feedforward*, existe un margen sustancial al comparar las variantes del método *Joint Supervision* cuando son implementadas en modelos con mayor densidad paramétrica, como ocurre con las redes LSTM y los conjuntos *bootstrap*. Esto último indicaría que la propuesta de *Joint Supervision* Selectivo estaría cumpliendo con su objetivo de compatibilizar este método de construcción de intervalos con la utilización de herramientas de *Deep Learning* mediante la disminución de

Tabla 5.2: Configuración hiperparamétrica final obtenida para los modelos de intervalo basados en redes neuronales en el experimento de la serie de Chen modificada

Modelos Neuronales	Hiperparámetro			
	Número de entradas	Número de neuronas en la capa oculta	Optimizador	Tasa de aprendizaje
Joint Supervision (feedforward)	2 (autoregresores) 2 (exógenas)	14	ADAM	0.001
Joint Supervision (LSTM)	2	40	ADAM	0.001
Joint Supervision (bootstrap)	2 (autoregresores) 2 (exógenas)	14	ADAM	0.001
Joint Supervision Selectivo (feedforward)	2 (autoregresores) 2 (exógenas)	14	ADAM	0.001
Joint Supervision Selectivo (LSTM)	2	40	ADAM	0.001
Joint Supervision Selectivo (bootstrap)	2 (autoregresores) 2 (exógenas)	14	ADAM	0.001

su costo computacional. Por otro lado, las similitudes en el costo computacional observadas entre las implementaciones basadas en redes *feedforward* es atribuible a que el tiempo de entrenamiento del método de *Joint Supervision* tradicional depende fuertemente de la cantidad de iteraciones ejecutadas por el algoritmo de búsqueda logarítmica. De esta manera, es altamente probable que en este experimento particular la búsqueda haya podido converger con un número reducido de iteraciones.

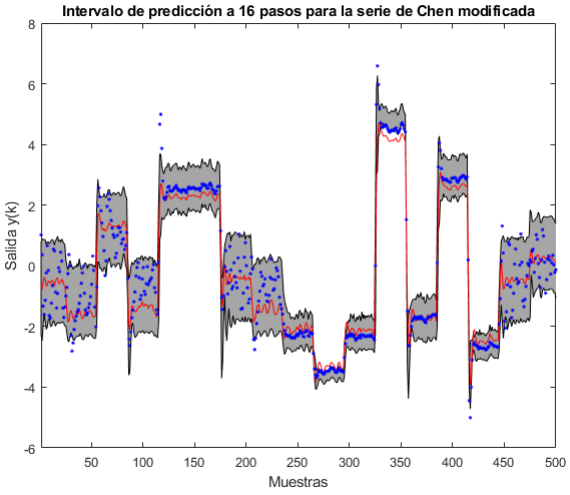
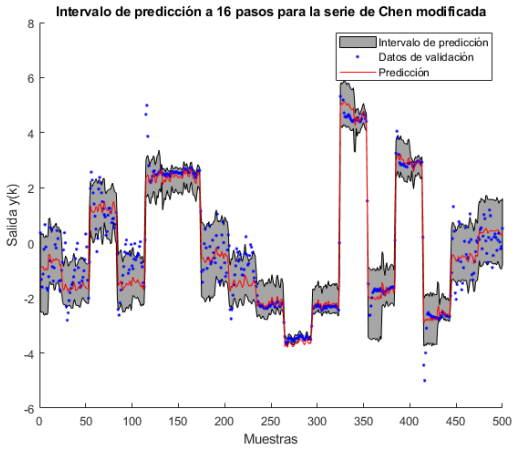
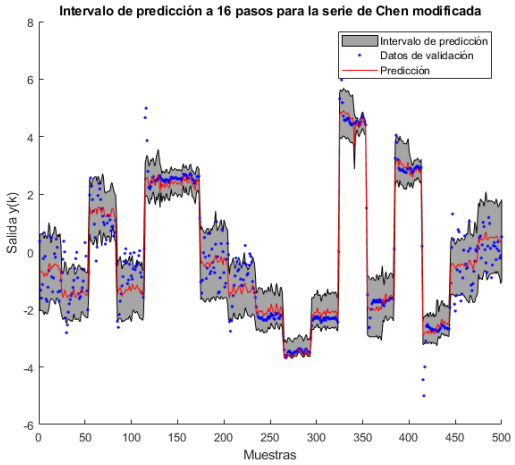


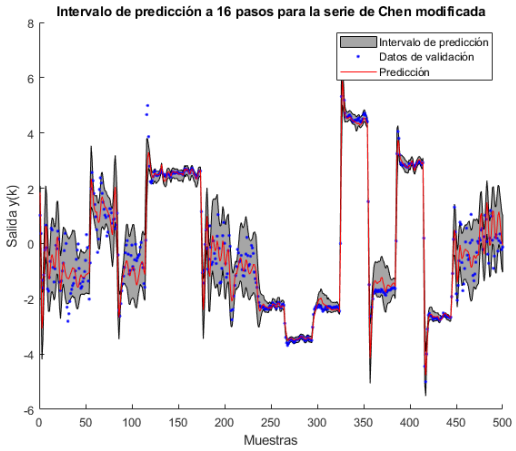
Figura 5.1: Intervalo de predicción obtenido con el método de *Joint Supervision* basado en sistemas difusos para la predicción a 16 pasos de la serie de Chen modificada



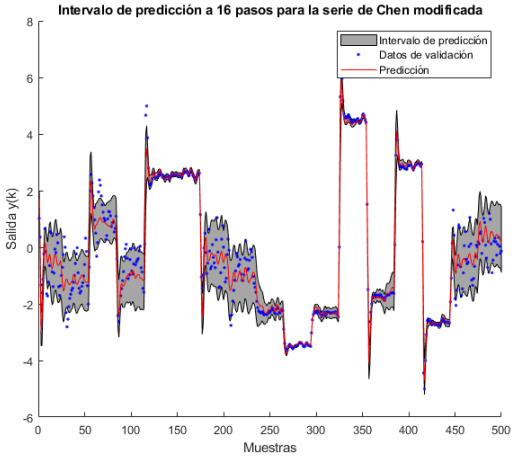
(a) *Joint Supervision* basado en redes *feedforward*



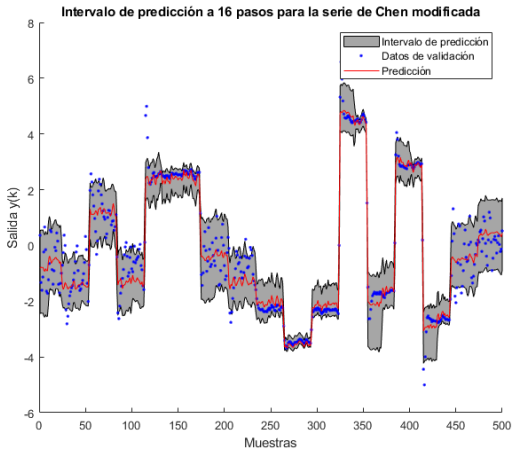
(b) *Joint Supervision* Selectivo basado en redes *feedforward*



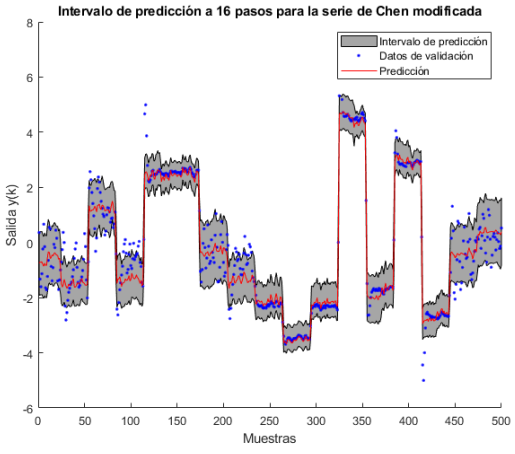
(c) *Joint Supervision* basado en redes LSTM



(d) *Joint Supervision* Selectivo basado en redes LSTM



(e) *Joint Supervision* basado en conjunto *bootstrap*



(f) *Joint Supervision* Selectivo basado en conjuntos *bootstrap*

Figura 5.2: Intervalos de predicción obtenidos con los métodos basados en redes neuronales para la predicción a 16 pasos de la serie de Chen modificada

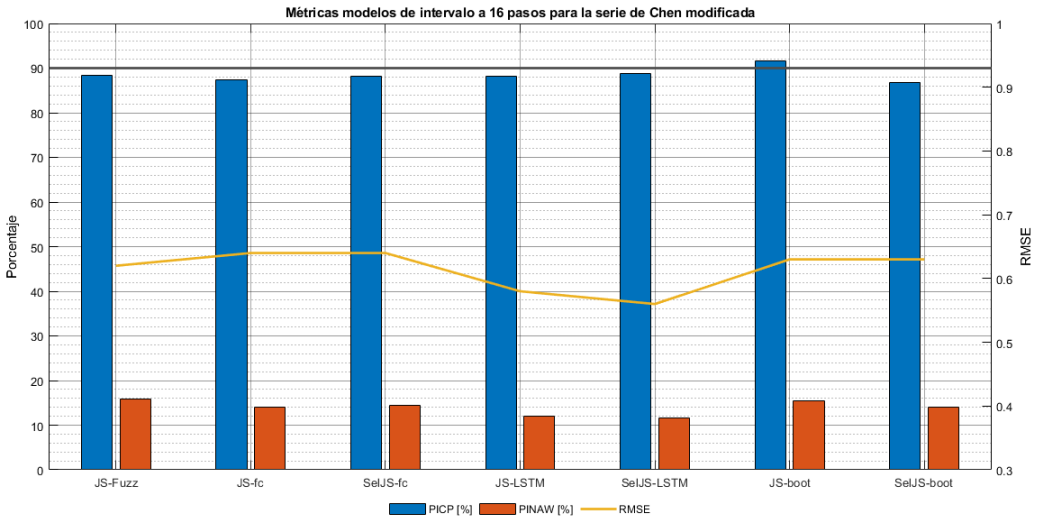


Figura 5.3: Métricas de desempeño obtenidas por los métodos de intervalo implementados para la predicción a 16 pasos de la serie de Chen modificada. De izquierda a derecha, los métodos mostrados son: *Joint Supervision* difuso (*JS-fuzz*), *Joint Supervision* basado en redes *feedforward* (*JS-fc*), *Joint Supervision* Selectivo basado en redes *feedforward* (*SelJS-fc*), *Joint Supervision* basado en redes LSTM (*JS-LSTM*), *Joint Supervision* Selectivo basado en redes LSTM (*SelJS-LSTM*), *Joint Supervision* basado en conjunto *bootstrap* (*JS-boot*), y *Joint Supervision* Selectivo basado en conjunto *bootstrap* (*SelJS-boot*)

.

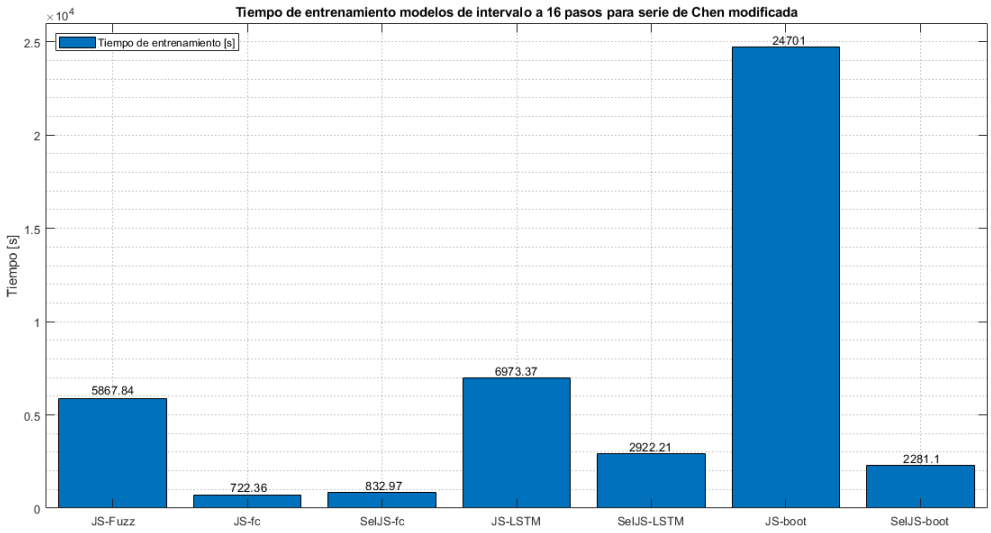


Figura 5.4: Tiempos de entrenamiento obtenidos por los métodos de intervalo implementados para la predicción a 16 pasos de la serie de Chen modificada. De izquierda a derecha, los métodos mostrados son: *Joint Supervision* difuso (*JS-fuzz*), *Joint Supervision* basado en redes *feedforward* (*JS-fc*), *Joint Supervision* Selectivo basado en redes *feedforward* (*SelJS-fc*), *Joint Supervision* basado en redes LSTM (*JS-LSTM*), *Joint Supervision* Selectivo basado en redes LSTM (*SelJS-LSTM*), *Joint Supervision* basado en conjunto *bootstrap* (*JS-boot*), y *Joint Supervision* Selectivo basado en conjunto *bootstrap* (*SelJS-boot*)

.

5.2. Experimento 2: Pronóstico de generación de potencia solar

Para la segunda prueba experimental se utilizaron datos de generación de potencia solar obtenidos en el *Multi-Good Microgrid Laboratory* (MG^2_{lab}) del Politécnico di Milano, Milán, Italia [82]. La base de datos consiste de un total de 11.688 mediciones tomadas entre los años 2017 y 2018, con un tiempo de muestreo de una hora. Al igual que con el experimento anterior, el primer 60 % fue utilizado para el entrenamiento de los modelos (conjunto de entrenamiento), el siguiente 20 % fue usado para la optimización estructural (conjunto de prueba), y el último 20 % fue destinado a la validación de los modelos obtenidos (conjunto de validación). A partir de estos conjuntos, se entrenaron las 7 variantes de modelos para generar predicciones a 1 y 12 pasos a futuro.

Es importante destacar que existen características específicas de este experimento que dificultan el entrenamiento de modelos de intervalo neuronal. En primer lugar, se tiene que los datos de entrenamiento son siempre mayores o iguales a cero (dado que no es posible obtener una generación negativa de potencia solar). Esto dificulta el entrenamiento debido a que, producto del carácter estocástico de estos algoritmos, existe la posibilidad de que el modelo calcule predicciones negativas en ciertos puntos, lo cual podría generar intervalos de ancho excesivo al dispararse la incertidumbre como consecuencia de la inexistencia de registros históricos de datos tomando estos valores. Sin embargo, este problema puede ser en parte solucionado al aplicar una saturación en la salida de la red como una etapa de postprocesamiento adicional. En segundo lugar, como la señal a predecir está relacionada con la disponibilidad de potencia solar, esta base de datos presenta una marcada periodicidad que sigue el comportamiento cíclico día/noche de esta variable. Como consecuencia de esto, existe un gran porcentaje (cercano al 50 %) de los datos que consisten en regiones constantes e igual a cero. Esto produce un espacio de búsqueda de soluciones del optimizador más complejo, generando una mayor cantidad de mínimos locales que en otros experimentos.

En las tablas 5.3 y 5.4 se pueden apreciar las configuraciones hiperparamétricas finales obtenidas para el modelo difuso y cada uno de los 6 modelos neuronales implementados en este trabajo, respectivamente. A diferencia de las configuraciones ocupadas para el experimento de la serie de Chen modificada, en este caso también se optó por utilizar tasas de aprendizaje diferentes en las implementaciones basadas en redes *feedforward* con respecto a aquellas basadas en redes LSTM. Esta decisión se tomó principalmente para garantizar una mejor convergencia de los modelos, ya que las redes LSTM presentaron problemas de convergencia con una alta probabilidad de estancarse en soluciones subóptimas al usar tasas de aprendizaje elevadas. Por otro lado, en este caso de estudio también se puede apreciar la diferencia en el número de entradas que se evidenció en la simulación anterior, la cual nuevamente se debe a las diferencias estructurales entre los tipos de redes implementadas.

Los resultados obtenidos en este experimento se encuentran desglosados en los siguientes gráficos: En primer lugar, la figura 5.5 muestra el intervalo de predicción de 12 pasos obtenido por el modelo difuso implementado. Posteriormente, en la figura 5.6 se pueden apreciar los intervalos de predicción de 12 pasos obtenidos por los modelos neuronales implementados. Luego, la figura 5.7 muestra las métricas de desempeño (RMSE, PICP y PINAW) obtenidas por cada uno de los métodos. Finalmente, en la figura 5.8 se pueden apreciar los tiempos de

Tabla 5.3: Configuración hiperparamétrica final obtenida para el modelo de *Joint Supervision* basado en sistemas difusos en el experimento de pronóstico de generación de potencia solar

Hiperparámetro	Valor
Número de entradas	23
Número de reglas	2
Algoritmo de clustering difuso	Fuzzy C-Means
Funciones de membresía	Gaussianas
Tipo de consecuencias	Lineales

entrenamiento medidos. La información detallada de todos los resultados obtenidos para este experimento puede ser encontrada en el apéndice B. A continuación, se procederá a discutir y analizar las observaciones más importantes que se pudo notar a partir de estos resultados.

Al analizar los intervalos mostrados en las figuras 5.5 y 5.6 es posible notar que los modelos basados en el método *Joint Supervision* tradicional (tanto en su variante neuronal como difusa) presentan un comportamiento similar, donde el intervalo presenta un ancho reducido en las regiones donde la potencia generada es nula, mientras que este valor se dispara en las mediciones tomadas durante el día, especialmente el límite inferior del intervalo. Por el otro lado, los modelos basados en *Joint Supervision* Selectivo presentan un comportamiento inverso, donde el intervalo tiende a presentar un mayor ancho en las muestras correspondientes a los periodos nocturnos, mientras que en los valores diurnos se obtiene un ancho menor. Estas diferencias en el comportamiento de los modelos permite evidenciar los problemas de convergencia del proceso de identificación de parámetros debido a las características de la señal a predecir. Además, es importante destacar que, si se observan las figuras 5.6c y 5.6d, no es posible apreciar grandes mejoras en la capacidad predictiva ni en la calidad del intervalo de las implementaciones basadas en redes LSTM. Es más, se puede notar que el uso de estas arquitecturas introdujo una zona de mayor error en la predicción, ubicada en las primeras muestras del conjunto de validación.

A partir de los resultados mostrados en la figura 5.7, es posible notar que los métodos neuronales basados en *Joint Superivsion* tradicional presentaron problemas al ajustar su porcentaje de cobertura, mostrando una tendencia a sobrepasar en creces el valor deseado. Entre estos modelos destaca la implementación basada en redes neuronales *feedforward*, la cual registró un porcentaje de cobertura mayor a un 97 %. Esto resulta perjudicial para el modelo debido a que un sobreajuste de la cobertura del intervalo indica que este aumentó innecesariamente su ancho para cubrir puntos que eran prescindibles. A su vez, también se puede apreciar que, a diferencia del experimento anterior, las implementaciones basadas en redes LSTM no presentaron una mejora significativa en la calidad del intervalo, e incluso obtuvieron una peor capacidad predictiva que el resto de los modelos. Ambas observaciones anteriores pueden ser explicadas como consecuencia de la complejidad del espacio de búsqueda de soluciones. En efecto, como se mencionó anteriormente en la sección 1.1.3, la búsqueda

Tabla 5.4: Configuración hiperparamétrica final obtenida para los modelos de intervalo basados en redes neuronales en el experimento de pronóstico de generación de potencia solar

Modelos Neuronales	Hiperparámetro			
	Número de entradas	Número de neuronas en la capa oculta	Optimizador	Tasa de aprendizaje
Joint Supervision (feedforward)	31	33	ADAM	0.001
Joint Supervision (LSTM)	1	81	ADAM	0.0005
Joint Supervision (bootstrap)	31	33	ADAM	0.001
Joint Supervision Selectivo (feedforward)	31	33	ADAM	0.001
Joint Supervision Selectivo (LSTM)	1	81	ADAM	0.0005
Joint Supervision Selectivo (bootstrap)	31	33	ADAM	0.001

iterativa del método de *Joint Supervision* tradicional introduce problemas de convergencia en la rutina de entrenamiento del modelo, por lo que en aplicaciones donde la optimización es más compleja es posible obtener resultados anormales. Un proceso similar ocurre con las redes LSTM, donde la convergencia del entrenamiento se dificulta debido a la mayor densidad paramétrica de la arquitectura. Por otro lado, al estudiar el desempeño de las implementaciones basadas en conjuntos *bootstrap*, se puede notar que, si bien existe una leve mejoría en la capacidad predictiva, la calidad del intervalo se ve levemente perjudicada, aunque la magnitud de las variaciones no permite realizar conclusiones definitivas al respecto ya que pueden ser atribuibles a la estocacidad del entrenamiento de estos modelos. Finalmente, si se observan los porcentajes de cobertura obtenidos por el método de *Joint Supervision* Selectivo y se comparan con los obtenidos en el experimento anterior (ver figura 5.3), es posible apreciar que, a pesar de que para ambos casos de estudio se utilizó el mismo valor para el hiperparámetro β, los modelos fueron capaces de cumplir con la restricción impuesta para esta métrica. Esto último permite demostrar la aplicabilidad del hiperparámetro como una variable única de uso general que no es necesario ajustar para cada experimento.

En cuanto al costo computacional del entrenamiento, en la figura 5.8 se puede apreciar que, a pesar de que no existen mayores diferencias entre los modelos basados en redes *feedforward*, nuevamente existe una gran diferencia al comparar las variantes neuronales del método *Joint Supervision* cuando son implementadas en modelos basados en redes LSTM y conjuntos *bootstrap*, la cual es incluso más grande que la observada en el experimento anterior. Esto último vuelve a comprobar el cumplimiento del objetivo de la propuesta de *Joint Supervision* Selectivo con respecto a la compatibilización de herramientas de *Deep Learning* mediante la disminución del costo computacional. Finalmente, es importante destacar que, a pesar de la reducción del costo computacional aportado por la propuesta planteada en este trabajo, aún

existe un margen considerable entre los tiempos obtenidos con redes *feedforward* y aquellos obtenidos con redes LSTM. Este fenómeno ocurre debido a que el uso de arquitecturas recurrentes implica un aumento considerable en el número de parámetros del modelo, por lo que siempre presentarán un mayor costo computacional que otras redes más tradicionales. Sin embargo, este problema es, en parte, solucionable si se utilizan recursos computacionales específicos, tales como las GPUs, para el entrenamiento de estos modelos.

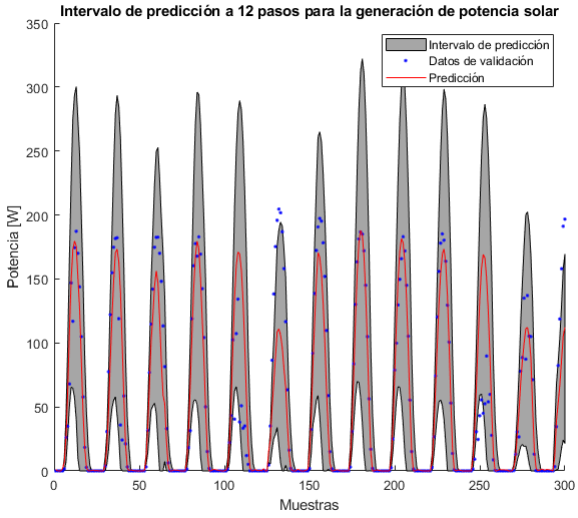
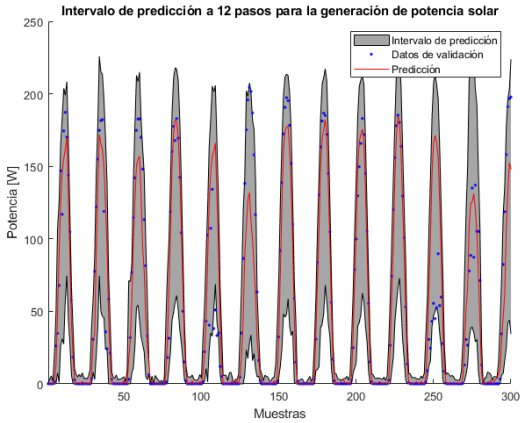
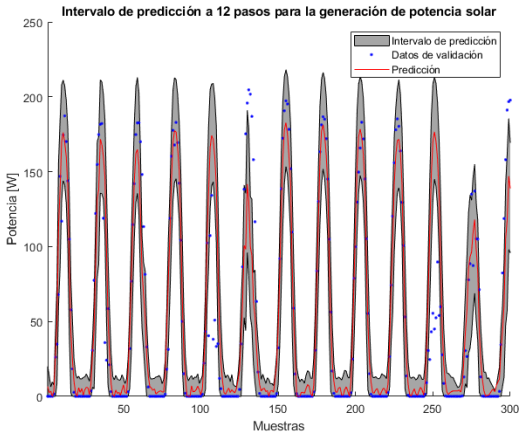


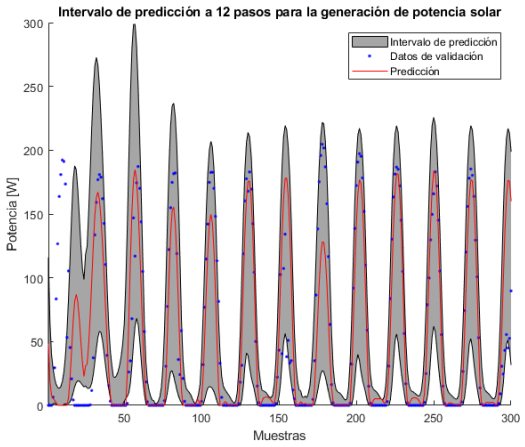
Figura 5.5: Intervalo de predicción obtenido con el método de *Joint Supervision* basado en sistemas difusos para la predicción a 12 pasos de la potencia solar generada



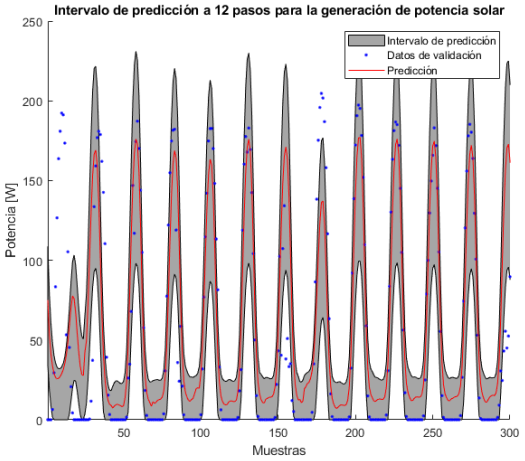
(a) *Joint Supervision* basado en redes *feedforward*



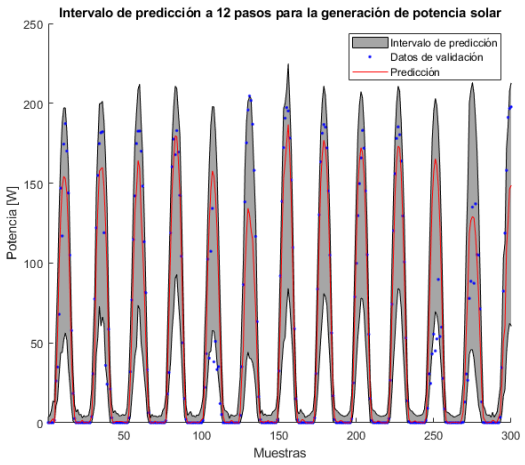
(b) *Joint Supervision* Selectivo basado en redes *feedforward*



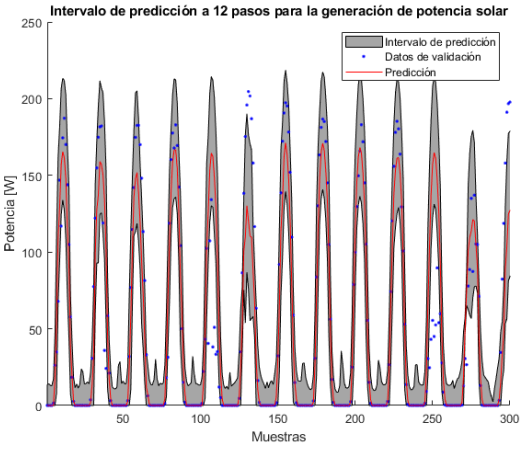
(c) *Joint Supervision* basado en redes LSTM



(d) *Joint Supervision* Selectivo basado en redes LSTM



(e) *Joint Supervision* basado en conjunto *bootstrap*



(f) *Joint Supervision* Selectivo basado en conjuntos *bootstrap*

Figura 5.6: Intervalos de predicción obtenidos con los métodos basados en redes neuronales para la predicción a 12 pasos de la potencia solar generada

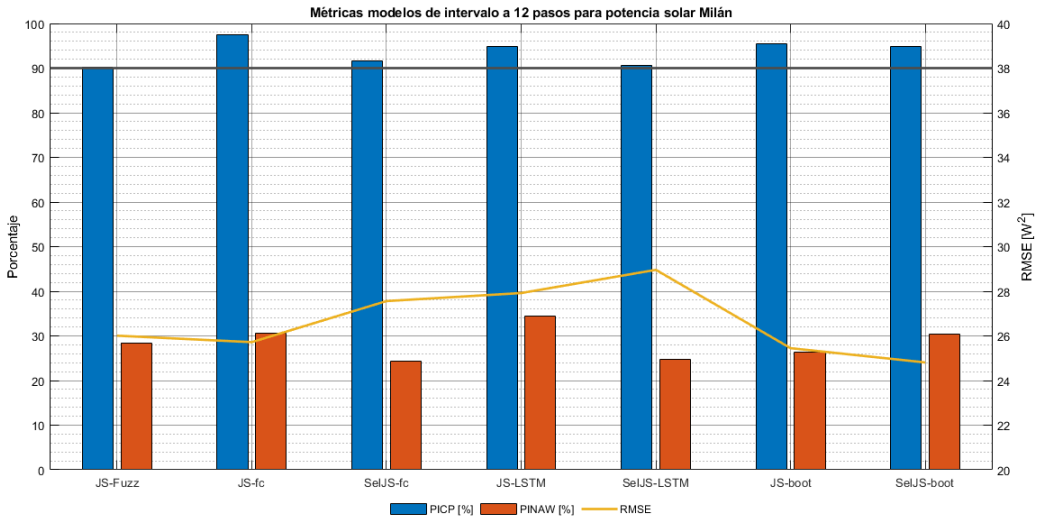


Figura 5.7: Métricas de desempeño obtenidas por los métodos de intervalo implementados para la predicción a 12 pasos de la potencia solar generada. De izquierda a derecha, los métodos mostrados son: *Joint Supervision* difuso (*JS-fuzz*), *Joint Supervision* basado en redes *feedforward* (*JS-fc*), *Joint Supervision* Selectivo basado en redes *feedforward* (*SelJS-fc*), *Joint Supervision* basado en redes LSTM (*JS-LSTM*), *Joint Supervision* Selectivo basado en redes LSTM (*SelJS-LSTM*), *Joint Supervision* basado en conjunto *bootstrap* (*JS-boot*), y *Joint Supervision* Selectivo basado en conjunto *bootstrap* (*SelJS-boot*)

.

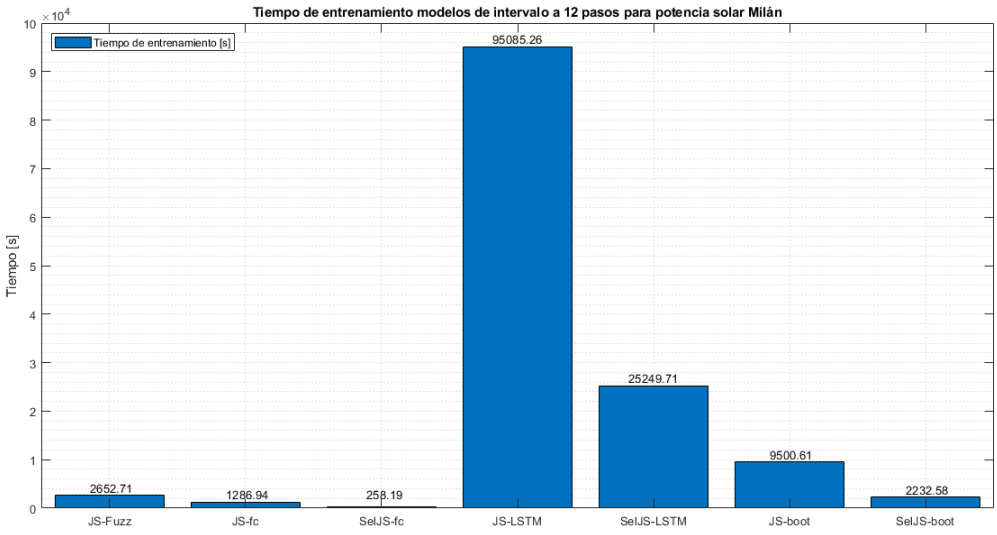


Figura 5.8: Tiempos de entrenamiento obtenidos por los métodos de intervalo implementados para la predicción a 12 pasos de la potencia solar generada. De izquierda a derecha, los métodos mostrados son: *Joint Supervision* difuso (*JS-fuzz*), *Joint Supervision* basado en redes *feedforward* (*JS-fc*), *Joint Supervision* Selectivo basado en redes *feedforward* (*SelJS-fc*), *Joint Supervision* basado en redes LSTM (*JS-LSTM*), *Joint Supervision* Selectivo basado en redes LSTM (*SelJS-LSTM*), *Joint Supervision* basado en conjunto *bootstrap* (*JS-boot*), y *Joint Supervision* Selectivo basado en conjunto *bootstrap* (*SelJS-boot*)

.

Conclusiones

En este trabajo se propuso una nueva metodología para la construcción de intervalos de predicción basada en modelos neuronales y una función de costo de *Joint Supervision* Selectivo. Este modelo tiene como objetivo intentar conservar las ventajas del método de *Joint Supervision* tradicional, en particular con respecto a su robustez y compatibilidad con optimizadores basados en métodos de gradiente, pero a la vez mejorar la convergencia del algoritmo de entrenamiento y disminuir el costo computacional al utilizar herramientas de *Deep Learning* y métodos de conjunto.

Adicionalmente, con el propósito de corroborar la aplicabilidad del método de Joint Supervision a otros sistemas basados en inteligencia computacional, y a la vez disponer de un punto de comparación adicional al realizar pruebas experimentales, también se diseñó una extensión del método de *Joint Supervision* tradicional compatible con sistemas difusos Takagi-Sugeno. Esta propuesta tiene la ventaja de poder aprovechar el buen desempeño de la variante neuronal de este método en cuanto al ancho del intervalo generado, y a la vez poder sacar provecho de la mayor interpretabilidad de los sistemas difusos.

Para comparar el desempeño de los modelos propuestos en cuanto a costo computacional, capacidad predictiva, y calidad del intervalo, se diseñaron dos pruebas experimentales en donde se entrenaron modelos de intervalo basados en los métodos propuestos, junto a modelos basados en *Joint Supervision* tradicional. Los casos de estudio evaluados consistieron en una base de datos artificial de un sistema dinámico altamente no lineal, y una aplicación real basada en la predicción de generación de potencia solar. Con el objetivo de obtener comparaciones para distintas arquitecturas neuronales, en cada experimento se evaluaron un total de 3 modelos para cada método neuronal: Un modelo basado en redes *feedforward*, uno basado en redes LSTM, y finalmente un modelo basado en un conjunto bootstrap de redes *feedforward*.

A partir de los resultados obtenidos se pudo notar que las implementaciones basadas en modelos difusos presentaron una capacidad predictiva y ancho de intervalo similar al de los modelos basados en redes *feedforward* y conjuntos *bootstrap*, donde además las variantes *feedforward* obtuvieron los menores tiempos de entrenamiento. A partir de esto se puede concluir que no fue posible apreciar una mejora notoria en el desempeño de los modelos neuronales al aplicar métodos de conjunto, mientras que sí se pudo comprobar un aumento considerable en el costo computacional, refutando de esta manera la segunda hipótesis considerada para este trabajo con respecto al *bootstrapping*. Sin embargo, es importante mencionar que los experimentos realizados en este trabajo no son suficientes para desechar completamente

esta técnica, puesto que la calidad de los resultados pudo haber sido perjudicada debido a la elección de un tamaño de conjunto demasiado pequeño. Finalmente, también se debe notar que en los casos de estudio evaluados no se implementaron conjuntos de redes LSTM, por lo que es necesario realizar más experimentos para evaluar la factibilidad de esta variante.

Con respecto a los resultados obtenidos por los modelos basados en redes LSTM, se puede apreciar que en el experimento de los datos generados artificialmente a partir de la serie de Chen modificada, estas arquitecturas lograron una notable mejora en la capacidad predictiva del modelo y el ancho promedio del intervalo, mientras que en el experimento de predicción de generación de potencia solar la capacidad predictiva fue peor a la del resto de los modelos y no se pudo observar diferencias en la informatividad del intervalo. A partir de esto se concluye que el desempeño de estas redes muy probablemente depende de las características específicas del sistema a predecir y la complejidad del espacio de búsqueda de soluciones. No obstante, es necesario realizar más experimentos con datos reales para corroborar estas observaciones preliminares, por lo que no es posible concluir sobre la comprobación de la segunda a hipótesis de este trabajo con los resultados obtenidos. Adicionalmente, se pudo notar que las redes LSTM presentaron consistentemente un elevado costo computacional, lo cual, si bien no es un problema en la mayoría de las aplicaciones, es un aspecto que se debe tener en mente si se desea aplicar técnicas de entrenamiento en línea, donde será necesario verificar que el tiempo de entrenamiento sea menor al tiempo de muestreo del sistema.

Por otro lado, a partir de los resultados experimentales también se pudo notar que el método de *Joint Supervision* Selectivo propuesto en este trabajo obtuvo métricas de desempeño similares o mejores que su variante tradicional. Entre estos resultados fue posible apreciar que en aplicaciones con espacios de búsqueda complejos, sólo se veía afectada la cobertura de los modelos *Joint Supervision* clásicos, mientras que la versión propuesta no sufría estos problemas. Adicionalmente, los experimentos también demostraron que la variante Selectiva de este método consistentemente logró un menor costo computacional al utilizar arquitecturas paramétricamente densas como lo son las redes profundas y los métodos de conjunto, comprobando la primera y tercera hipótesis que se consideraron en este trabajo. Estas dos observaciones también permiten concluir que el método de *Joint Supervision* Selectivo presenta una mejor convergencia y un menor costo computacional que su variante original, corroborando cumplimiento de los objetivos propuestos para este trabajo.

Como trabajo a futuro se propone diseñar simulaciones de laboratorio en donde se apliquen modelos basados en *Joint Supervision* Selectivo con estrategias de entrenamiento en línea, con el objetivo de comprobar la compatibilidad de la metodología propuesta con estas herramientas. Finalmente, también se propone construir pruebas que permitan estudiar el comportamiento de modelos neuronales de intervalo con múltiples entradas y salidas en aplicaciones dirigidas hacia la predicción de variables climáticas. Esto último se debe a que, en general, este tipo de señales no pueden ser modeladas adecuadamente como sistemas autoregresivos, puesto que siempre existe una dependencia de otros indicadores meteorológicos (por ejemplo, la radiación solar puede verse influenciada por la temperatura ambiental y la humedad del aire, ya que estas variables tienen relación con la aparición de nubes). Como consecuencia de esto, los modelos neuronales autoregresivos tienden a manifestar grandes errores de estimación a mediano y largo plazo, por lo que su aplicabilidad está restringida a horizontes de predicción pequeños. Sin embargo, la incorporación de señales adicionales

que puedan aportar una mayor información contextual al modelo predictivo podría hacer factibles a estas arquitecturas para el cálculo de predicciones climáticas a un mayor número de pasos, ampliando así la aplicabilidad de este trabajo.

Bibliografía

[1] Tom Heskes. Practical confidence and prediction intervals. In *Advances in neural information processing systems*, pages 176–182, 1997.

[2] Georgios Papadopoulos, Peter J Edwards, and Alan F Murray. Confidence estimation methods for neural networks: A practical comparison. *IEEE transactions on neural networks*, 12(6):1278–1287, 2001.

[3] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F Atiya. Comprehensive review of neural network-based prediction intervals and new advances. *IEEE Transactions on neural networks*, 22(9):1341–1356, 2011.

[4] Tim Pearce, Alexandra Brintrup, Mohamed Zaki, and Andy Neely. High-quality prediction intervals for deep learning: A distribution-free, ensembled approach. In *International Conference on Machine Learning*, pages 4075–4084, 2018.

[5] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Amir F Atiya. Lower upper bound estimation method for construction of neural network-based prediction intervals. *IEEE transactions on neural networks*, 22(3):337–346, 2010.

[6] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[7] SABS Chen and SA Billings. Neural networks for nonlinear dynamic system modelling and identification. *International journal of control*, 56(2):319–346, 1992.

[8] G Peter Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

[9] Jin-Quan Huang and Frank L Lewis. Neural-network predictive control for nonlinear dynamic systems with time-delay. *IEEE Transactions on Neural Networks*, 14(2):377–389, 2003.

[10] Ma Lei, Luan Shiyan, Jiang Chuanwen, Liu Hongling, and Zhang Yan. A review on the forecasting of wind speed and generated power. *Renewable and Sustainable Energy Reviews*, 13(4):915–920, 2009.

[11] Bernard Widrow and Marcian E Hoff. Adaptive switching circuits. Technical report, Stanford Univ Ca Stanford Electronics Labs, 1960.

[12] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[13] Ben Khuong. The basics of recurrent neural networks (rnns). Medium, Jun 2020. https://medium.com/towards-artificial-intelligence/whirlwind-tour-of-rnns-a11effb7808f.

[14] Ray Bernard. Deep learning to the rescue. SecurityInfoWatch, Mar 2019. https://www.securityinfowatch.com/video-surveillance/video-analytics/article/21069937/deep-learning-to-the-rescue.

[15] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[16] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[18] Gerhard Paass. Assessing and improving neural network predictions by the bootstrap algorithm. In *Advances in Neural Information Processing Systems*, pages 196–203, 1993.

[19] Ehsan Mazloumi, Geoff Rose, Graham Currie, and Sara Moridpour. Prediction intervals to account for uncertainties in neural network predictions: Methodology and application in bus travel time prediction. *Engineering Applications of Artificial Intelligence*, 24(3):534–542, 2011.

[20] Enrico Zio. A study of the bootstrap method for estimating the accuracy of artificial neural networks in predicting nuclear transient processes. *IEEE Transactions on Nuclear Science*, 53(3):1460–1478, 2006.

[21] Abbas Khosravi, Saeid Nahavandi, and Doug Creighton. Prediction intervals for short-term wind farm power generation forecasts. *IEEE Transactions on sustainable energy*, 4(3):602–610, 2013.

[22] Pierre Pinson and Georges Kariniotakis. On-line assessment of prediction risk for wind power production forecasts. *Wind Energy: An International Journal for Progress and Applications in Wind Power Conversion Technology*, 7(2):119–132, 2004.

[23] Abbas Khosravi, Saeid Nahavandi, and Doug Creighton. Quantifying uncertainties of neural network-based electricity price forecasts. *Applied energy*, 112:120–129, 2013.

[24] Nicolás Cruz, Luis G Marín, and Doris Sáez. Neural network prediction interval based on joint supervision. In *2018 International Joint Conference on Neural Networks (IJCNN)*,

pages 1–8. IEEE, 2018.

[25] Nicolás Cruz, Luis G Marín, and Doris Sáez. Prediction intervals with lstm networks trained by joint supervision. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[26] Luis G Marín, Nicolás Cruz, Doris Sáez, Mark Sumner, and Alfredo Núñez. Prediction interval methodology based on fuzzy numbers and its extension to fuzzy systems and neural networks. *Expert Systems with Applications*, 119:128–141, 2019.

[27] Alexandre Oudalov and Antonio Fidigatti. Adaptive network protection in microgrids. *International Journal of Distributed Energy Resources*, 5(3):201–226, 2009.

[28] Kaitlyn Bunker, Kate Hawley, and Jesse Morris. *Renewable Microgrids: Profiles from Islands and Remote Communities Across the Globe*. Rocky Mountain Institute, Nov 2015.

[29] HM Dipu Kabir, Abbas Khosravi, Mohammad Anwar Hosen, and Saeid Nahavandi. Neural network-based uncertainty quantification: A survey of methodologies and applications. *IEEE access*, 6:36218–36234, 2018.

[30] JT Gene Hwang and A Adam Ding. Prediction intervals for artificial neural networks. *Journal of the American Statistical Association*, 92(438):748–757, 1997.

[31] SL Ho, M Xie, LC Tang, K Xu, and TN Goh. Neural network modeling with confidence bounds: a case study on the solder paste deposition process. *IEEE Transactions on Electronics Packaging Manufacturing*, 24(4):323–332, 2001.

[32] Tao Lu and Martti Viljanen. Prediction of indoor temperature and relative humidity using neural network models: model comparison. *Neural Computing and Applications*, 18(4):345, 2009.

[33] Yi-Ming Kuo, Chen-Wuing Liu, and Kao-Hung Lin. Evaluation of the ability of an artificial neural network model to assess the variation of groundwater quality in an area of blackfoot disease in taiwan. *Water research*, 38(1):148–158, 2004.

[34] Abbas Khosravi, Ehsan Mazloumi, Saeid Nahavandi, Doug Creighton, and JWC Van Lint. Prediction intervals to account for uncertainties in travel time prediction. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):537–547, 2011.

[35] Richard D De VlEAUX, Jennifer Schumi, Jason Schweinsberg, and Lyle H Ungar. Prediction intervals for neural networks via nonlinear regression. *Technometrics*, 40(4):273–282, 1998.

[36] Abbas Khosravi, Saeid Nahavandi, and Doug Creighton. A prediction interval-based approach to determine optimal structures of neural network metamodels. *Expert systems with applications*, 37(3):2377–2387, 2010.

[37] Abbas Khosravi, Saeid Nahavandi, and Doug Creighton. Construction of optimal pre-

diction intervals for load forecasting problems. *IEEE Transactions on Power Systems*, 25(3):1496–1503, 2010.

[38] John S Denker and Yann LeCun. Transforming neural-net output levels to probability distributions. In *Advances in neural information processing systems*, pages 853–859, 1991.

[39] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[40] Lyle H Ungar, Richard D De Veaux, and Evelyn Rosengarten. Estimating prediction intervals for artificial neural networks. In *Proc. of the 9th Yale Workshop on Adaptive and Learning Systems*, 1996.

[41] Abbas Khosravi, Saeid Nahavandi, and Doug Creighton. Load forecasting and neural networks: A prediction interval-based perspective. In *Computational intelligence in power engineering*, pages 131–150. Springer, 2010.

[42] Alfonso F Torres, Wynn R Walker, and Mac McKee. Forecasting daily potential evapotranspiration using machine learning and limited climatic data. *Agricultural Water Management*, 98(4):553–562, 2011.

[43] David A Nix and Andreas S Weigend. Estimating the mean and variance of the target probability distribution. In *Proceedings of 1994 ieee international conference on neural networks (ICNN'94)*, volume 1, pages 55–60. IEEE, 1994.

[44] Abbas Khosravi and Saeid Nahavandi. An optimized mean variance estimation method for uncertainty quantification of wind power forecasts. *International Journal of Electrical Power & Energy Systems*, 61:446–454, 2014.

[45] Jenq-Neng Hwang and Erik Little. Real time recurrent neural networks for time series prediction and confidence estimation. In *Proceedings of International Conference on Neural Networks (ICNN'96)*, volume 4, pages 1889–1894. IEEE, 1996.

[46] Marco Rigamonti, Piero Baraldi, Enrico Zio, Indranil Roychoudhury, Kai Goebel, and Scott Poll. Ensemble of optimized echo state networks for remaining useful life prediction. *Neurocomputing*, 281:121–138, 2018.

[47] Wei Yao, Zhigang Zeng, and Cheng Lian. Generating probabilistic predictions using mean-variance estimation and echo state network. *Neurocomputing*, 219:536–547, 2017.

[48] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.

[49] Marcel RA van Gent, Henk FP van den Boogaard, Beatriz Pozueta, and Josep R Medina. Neural network modelling of wave overtopping at coastal structures. *Coastal engineering*, 54(8):586–593, 2007.

[50] Mansour Talebizadeh and Ali Moridnejad. Uncertainty analysis for the forecast of la-

ke level fluctuations using ensembles of ann and anfis models. *Expert Systems with applications*, 38(4):4126–4135, 2011.

[51] Piero Baraldi, Michele Compare, Sergio Sauco, and Enrico Zio. Ensemble neural network-based particle filtering for prognostics. *Mechanical Systems and Signal Processing*, 41(1-2):288–300, 2013.

[52] BD Conduit, Nick G Jones, Howard J Stone, and Gareth John Conduit. Design of a nickel-base superalloy using a neural network. *Materials & Design*, 131:358–365, 2017.

[53] Kosta Ristovski, Slobodan Vucetic, and Zoran Obradovic. Uncertainty analysis of neural-network-based aerosol retrieval. *IEEE transactions on geoscience and remote sensing*, 50(2):409–414, 2011.

[54] John G Carney, Pádraig Cunningham, and Umesh Bhagwan. Confidence and prediction intervals for neural network ensembles. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 2, pages 1215–1218. IEEE, 1999.

[55] Abbas Khosravi, Saeid Nahavandi, Doug Creighton, and Dipti Srinivasan. Optimizing the quality of bootstrap-based prediction intervals. In *The 2011 International Joint Conference on Neural Networks*, pages 3072–3078. IEEE, 2011.

[56] Igor Škrjanc. Fuzzy confidence interval for ph titration curve. *Applied mathematical modelling*, 35(8):4083–4090, 2011.

[57] Doris Sáez, Fernand Ávila, Daniel Olivares, Claudio Cañizares, and Luis Marín. Fuzzy prediction interval models for forecasting renewable resources and loads in microgrids. *IEEE Transactions on Smart Grid*, 6(2):548–556, 2014.

[58] Hao Quan, Dipti Srinivasan, and Abbas Khosravi. Short-term load and wind power forecasting using neural network-based prediction intervals. *IEEE transactions on neural networks and learning systems*, 25(2):303–315, 2013.

[59] Ronay Ak, YANFU LI, and Enrico Zio. Estimation of prediction intervals of neural network models by a multi-objective genetic algorithm. In *Uncertainty Modeling in Knowledge Engineering and Decision Making*, pages 1036–1041. World Scientific, 2012.

[60] Inés M Galván, José M Valls, Alejandro Cervantes, and Ricardo Aler. Multi-objective evolutionary optimization of prediction intervals for solar energy forecasting with neural networks. *Information Sciences*, 418:363–382, 2017.

[61] Zhichao Shi, Hao Liang, and Venkata Dinavahi. Direct interval forecast of uncertain wind power based on recurrent neural networks. *IEEE Transactions on Sustainable Energy*, 9(3):1177–1187, 2017.

[62] Ranran Li and Yu Jin. A wind speed interval prediction system based on multi-objective optimization for machine learning method. *Applied energy*, 228:2207–2220, 2018.

[63] Qiang Ni, Shengxian Zhuang, Hanming Sheng, Gaoqiang Kang, and Jian Xiao. An ensemble prediction intervals approach for short-term pv power forecasting. *Solar Energy*, 155:1072–1083, 2017.

[64] Hao Quan, Dipti Srinivasan, and Abbas Khosravi. Uncertainty handling using neural network-based prediction intervals for electrical load forecasting. *Energy*, 73:916–925, 2014.

[65] Jiyang Wang, Yuyang Gao, and Xuejun Chen. A novel hybrid interval prediction approach based on modified lower upper bound estimation in combination with multiobjective salp swarm algorithm for short-term load forecasting. *Energies*, 11(6):1561, 2018.

[66] Mashud Rana, Irena Koprinska, Abbas Khosravi, and Vassilios G Agelidis. Prediction intervals for electricity load forecasting using neural networks. In *The 2013 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2013.

[67] Riccardo Taormina and Kwok-Wing Chau. Ann-based interval forecasting of streamflow discharges using the lube method and mofips. *Engineering Applications of Artificial Intelligence*, 45:429–440, 2015.

[68] Cheng Lian, Zhigang Zeng, Wei Yao, Huiming Tang, and Chun Lung Philip Chen. Landslide displacement prediction with uncertainty based on neural networks with random hidden weights. *IEEE transactions on neural networks and learning systems*, 27(12):2683–2695, 2016.

[69] Mohammad Anwar Hosen, Abbas Khosravi, Saeid Nahavandi, and Douglas Creighton. Improving the quality of prediction intervals through optimal aggregation. *IEEE Transactions on Industrial Electronics*, 62(7):4420–4429, 2014.

[70] Mohammad Anwar Hosen, Abbas Khosravi, Saeid Nahavandi, and Douglas Creighton. Prediction interval-based neural network modelling of polystyrene polymerization reactor–a new perspective of data-based modelling. *Chemical Engineering Research and Design*, 92(11):2041–2051, 2014.

[71] Xiao-Yun Chen and Kwok-Wing Chau. Uncertainty analysis on hybrid double feedforward neural network model for sediment load estimation with lube method. *Water Resources Management*, 33(10):3563–3577, 2019.

[72] Hairong Zhang, Jianzhong Zhou, Lei Ye, Xiaofan Zeng, and Yufan Chen. Lower upper bound estimation method considering symmetry for construction of prediction intervals in flood forecasting. *Water Resources Management*, 29(15):5505–5519, 2015.

[73] Abdollah Kavousi-Fard, Abbas Khosravi, and Saeid Nahavandi. Reactive power compensation in electric arc furnaces using prediction intervals. *IEEE Transactions on Industrial Electronics*, 64(7):5295–5304, 2017.

[74] Lei Ye, Jianzhong Zhou, Hoshin V Gupta, Hairong Zhang, Xiaofan Zeng, and Lu Chen. Efficient estimation of flood forecast prediction intervals via single-and multi-objective

versions of the lube method. *Hydrological Processes*, 30(15):2703–2716, 2016.

[75] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, pages 1613–1622, 2015.

[76] Durk P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in neural information processing systems*, pages 2575–2583, 2015.

[77] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629, 2018.

[78] Diego Munoz-Carpintero, Sebastián Parra, Oscar Cartagena, Doris Sáez, Luis G Marín, and Igor Škrjanc. Fuzzy interval modelling based on joint supervision. In *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2020.

[79] Doris Sáez and Aldo Cipriano. A new method for structure identification of fuzzy models and its application to a combined cycle power plant. *Engineering Intelligent Systems for Electrical Engineering and Communications*, 9(2):101–107, 2001.

[80] Michio Sugeno and Takahiro Yasukawa. A fuzzy-logic-based approach to qualitative modeling. *IEEE Transactions on fuzzy systems*, 1(1):7–31, 1993.

[81] S. Chen, S. A. Billings, and P. M. Grant. Non-linear system identification using neural networks. *International Journal of Control*, 51(6):1191–1214, 1990.

[82] Alfredo Nespoli, Marco Mussetta, Emanuele Ogliari, Sonia Leva, Luis Fernández-Ramírez, and Pablo García-Triviño. Robust 24 hours ahead forecast in a microgrid: A real case study. *Electronics*, 8(12):1434, 2019.

[83] Oscar Cartagena, Sebastián Parra, Diego Muñoz-Carpintero, Luis G Marín, and Doris Sáez. Review on fuzzy and neural prediction interval modelling for nonlinear dynamical systems. *IEEE Access*, 9:23357–23384, 2021.

Apéndice A

Resultados detallados experimento serie de Chen modificada

Tabla A.1: Métricas de desempeño y tiempos de entrenamiento obtenidas para los modelos de intervalo implementados para el experimento de la serie de Chen modificada

		RMSE	PICP [%]	PINAW [%]	Tiempo de Entrenamiento [s]
1 Paso adelante	Joint Supervision Difuso	0.33	90.79	8.3	**41.28**
	Joint Supervision (feedforward)	**0.24**	91.14	4.8	502.15
	Joint Supervision Selectivo (feedforward)	0.26	87.23	4.99	152.14
	Joint Supervision (LSTM)	0.27	**90.09**	5.16	6560.59
	Joint Supervision Selectivo (LSTM)	0.27	87.54	4.71	9901.72
	Joint Supervision (bootstrap)	**0.24**	92.64	4.7	33519.16
	Joint Supervision Selectivo (bootstrap)	0.25	86.58	**4.2**	4359.96
16 Pasos adelante	Joint Supervision Difuso	0.62	88.45	15.88	5867.84
	Joint Supervision (feedforward)	0.64	87.33	14.08	**722.36**
	Joint Supervision Selectivo (feedforward)	0.64	88.24	14.37	832.97
	Joint Supervision (LSTM)	0.58	88.09	11.99	6973.37
	Joint Supervision Selectivo (LSTM)	**0.56**	**88.7**	**11.53**	2922.21
	Joint Supervision (bootstrap)	0.63	91.52	15.48	24701
	Joint Supervision Selectivo (bootstrap)	0.63	86.83	14.1	2281.1

Apéndice B

Resultados detallados experimento potencia solar Milán

Tabla B.1: Métricas de desempeño y tiempos de entrenamiento obtenidas para los modelos de intervalo implementados para el experimento de pronóstico de generación de potencia solar

		RMSE [W^2]	PICP [%]	PINAW [%]	Tiempo de Entrenamiento [s]
1 Paso adelante	Joint Supervision Difuso	13.18	**90.35**	11.19	2659.84
	Joint Supervision (feedforward)	12.81	93.97	13.44	614.13
	Joint Supervision Selectivo (feedforward)	13.09	91.31	10.02	**262.55**
	Joint Supervision (LSTM)	15.46	92.77	16.74	154227.14
	Joint Supervision Selectivo (LSTM)	12.94	91.05	**8.65**	31611.56
	Joint Supervision (bootstrap)	**12.54**	96.24	13.26	6281.71
	Joint Supervision Selectivo (bootstrap)	13.15	92.79	12.23	2565.92
12 Pasos adelante	Joint Supervision Difuso	26.01	**90.17**	28.45	2652.71
	Joint Supervision (feedforward)	25.72	97.5	30.61	1286.24
	Joint Supervision Selectivo (feedforward)	27.56	91.7	**24.27**	**258.19**
	Joint Supervision (LSTM)	27.92	94.75	34.38	95085.26
	Joint Supervision Selectivo (LSTM)	28.96	90.67	24.7	25249.71
	Joint Supervision (bootstrap)	25.45	95.44	26.32	9500.61
	Joint Supervision Selectivo (bootstrap)	**24.81**	94.82	30.37	2232.58