Comparative Study of Methods to Obtain the Number of Hidden Neurons of an Auto-encoder in a High-dimensionality Context

Héctor R. Vega-Gutiérrez, Carlos Castorena, Roberto Alejo and Everardo E. Granda-Gutiérrez

***Abstract*—Fourteen formulas from the state-of-art were used in this paper to find the optimal number of neurons in the hidden layer of an autoencoder neural network. The latter is employed to reduce the dataset dimension on high-dimensionality scenarios with not significant reduction in classification accuracy in comparison to the use of the whole dataset. A Deep Learning neural network was employed to analyze the effectiveness of the studied formulas in classification terms (accuracy). Eight high-dimensional datasets were processed in an experimental set in order to assess this proposal. Results presented in this work show that formula 13 (used to find the number of hidden neurons of the auto-encoder) is effective to reduce the data dimensionality without a statistically significant reduction of the classification performance, as it was confirmed by the Freidman test and the Holm's post-hoc test.**

***Index Terms*—High-dimensionality data, Neural Network, Auto-encoder**

I. Introducción

En la actualidad, muchos de los problemas reales se encuentran inmersos en un contexto de *big data*, debido a que los datos que se generan de ellos provienen de múltiples fuentes, a gran velocidad y son de gran volumen [1], [2]. Por ejemplo, en el campo de la atención médica [3], en el desarrollo de sistemas de recomendación en los negocios [4], o incluso en la detección de conductas violentas en sistemas de video vigilancia [5], sin olvidar los recientes avances en el estudio del ADN (Ácido Desoxirribonucleico) [6], por mencionar algunas aplicaciones.

En escenarios de *big data*, el aprendizaje automático ha tomado mucha fuerza gracias a su capacidad de extraer patrones cada vez más complejos [7] desde grandes volúmenes de datos de forma automatizada [1]. Las redes neuronales artificiales (RNA) son uno de los clasificadores de aprendizaje automático más comunes en ambientes *big data*; en particular, las redes artificiales de aprendizaje profundo (RNAP), y las redes auto codificadoras (RNAC) o *auto-encoder*. Las RNAP tienen la característica principal de tener por lo menos dos capas ocultas y ser entrenadas por métodos híbridos de aprendizaje [8]. Las RNAC se están usando para pre-entrenar RNAP o como reductoras de dimensionalidad. No obstante, ambas, traen consigo un considerable incremento en el costo computacional asociado a su correcto funcionamiento, pero gracias a los avances Tecnológicos en hardware (por ejemplo, el uso de unidades gráficas GPU para una ejecución en paralelo) y a su abaratamiento se han podido llevar a la vida real estos modelos computacionales. Asimismo, la disponibilidad de plataformas de trabajo de acceso libre como *SPARK* o *TensorFlow* [9], [10], han permitido que muchos de los problemas actuales en ambientes *big data* sean abordados. El estudio del ADN [6] ha llevado a problemas relacionados a la clasificación de bases de datos de alta dimensionalidad y con muy pocas muestras, e inclusive altamente desbalanceadas. Las bases de datos bio-médicas de microarrays de expresión genética se caracterizan por ser de gran dimensionalidad y tener pocas muestras [11], por lo que el diseño y construcción de clasificadores efectivos para tratarlas es un reto vigente en el aprendizaje automático y el profundo.

Actualmente, la amplia dimensionalidad de los datos puede ser enfrentada por medio de métodos de selección, extracción y/o construcción de características [12], los cuales se han desarrollado desde diversos enfoques como: ingeniería de características [13], la cual se basa en la selección de características originales o en construcción de nuevas; el aprendizaje de representaciones [14] se refiere al uso de redes neuronales para automatizar completamente el proceso de generación de características. La meta de la extracción de características es encontrar una mejor representación de datos para el algoritmo de aprendizaje automático que se desea usar, dado que la representación original podría no ser la mejor; los métodos de extracción de características se enfocan en combinar variables para remover información redundante e irrelevante [15]. Múltiples algoritmos de aprendizaje, especialmente aquellos basados en redes neuronales como las redes neuronales auto codificadoras (RNAC) o $autoencoder$, caen en esta categoría. En términos generales, estas metodologías buscan seleccionar, transformar y/o construir de manera automática el mejor grupo de características para el entrenamiento, descartando aquellas que no aporten ningún tipo de información [16]. El uso de las RNAC para la reducción de la dimensionalidad de los datos ha sido un tema previamente estudiado; trabajos como [17] mencionan el uso de una RNAC para filtrar la información antes de entrar a un clasificador, obteniendo resultados positivos en la clasificación global de los datos. Una de las aplicaciones

Héctor R. Vega-Gutiérrez, Tecnológico Nacional de México / IT Toluca, Metepec, Estado de México, C.P. 52149, México, e-mail: hvegag@toluca.tecnm.mx.

Carlos Castorena, Tecnológico Nacional de México / IT Toluca, Metepec, Estado de México, C.P. 52149, México, e-mail: ccastorenal@toluca.tecnm.mx.

Roberto Alejo, Tecnológico Nacional de México / IT Toluca, Metepec, Estado de México, C.P. 52149, México, e-mail: ralejoe@toluca.tecnm.mx. *Autor para correspondencia: Roberto Alejo.*

Everardo E. Granda-Gutiérrez, Universidad Autónoma del Estado de México / CU UAEM Atlacomulco, Estado de México, C.P. 50400, México, e-mail: eegrandag@uaemex.mx.

más usuales de este tipo de redes es la detección de datos anómalos, así como el filtrar la información, sin embargo, la aplicación de una RNAC no se limita a filtrar los datos de entrada; por ejemplo, la Ref. [16] menciona distintas formas de aprovechar una RNAC para la compresión de los datos y así reducir el tamaño de los vectores de características de las muestras.

No obstante, existe un problema generalizado en todas las RNA (incluyendo las RNAC), el cual consiste en identificar el número apropiado de neuronas ocultas en la red. En este sentido, se han desarrollado numerosos trabajos encaminados a identificar o determinar automáticamente el número de neuronas ocultas (por ejemplo, véase las Ref. [18]–[26]). Este trabajo se centra en el estudio de las posibilidades de algunas de las propuestas más relevantes del estado del arte para identificar de manera directa el número óptimo de neuronas ocultas en una red neuronal y así enfrentar uno de los problemas de mayor relevancia en el estudio de las bases de datos bio-médicas de microarrays de expresión genética: su amplia dimensionalidad. La identificación del número de neuronas ocultas de la RNAC se realizó mediante las propuestas presentadas en [18]–[25], [36]–[39], aprovechando una RNAC para reducir la dimensionalidad de la base de datos. Finalmente, se comprobó la efectividad de estos métodos al utilizar las características resultantes del proceso de la RNAC como entrada a una RNAP y comparar su efectividad con respecto al uso de la base datos original.

II. Trabajos Relacionados

Son muchos los trabajos que exploran la reducción dimensional por medio de diversas técnicas [27], [28], mostrando que el uso de una red autocodificadora genera buenos resultados, incluso mejores que algoritmos como PCA (Análisis de Componentes Principales), que es uno de los métodos más usados para reducir la dimensionalidad de las bases de datos [29]. Meriwani [30] realiza un trabajo con una red autoencoder con múltiples capas intermedias, utilizando la salida del encoder en una red neuronal de aprendizaje profundo; sus experimentos se basan en 3 bases de datos médicas con alta dimensionalidad, mostrando una tendencia favorable en aquellas bases de datos donde se usa una entrada codificada sin tomar en cuenta el costo de procesamiento extra que se requiere para poder tener un beneficio en el AC (*Accuracy*) y en el e (*Error Cuadrático Medio*) [34]. Asimismo, en ese documento no se propone una configuración general para la arquitectura de la red autoencoder, la cual pueda mostrar los mejores resultados. Algunos trabajos actuales centran su estudio en el uso de una red convolucional autoencoder para reducir la dimensionalidad de las bases de datos y filtrar la información en $\mathbb{R}^2$, el principio de reducción es el mismo, pero estos estudios limitan su aplicación a clasificación de imágenes [31]. Vujicic y colaboradores [32] realizan un estudio comparativo de seis fórmulas desarrolladas por otros investigadores para determinar el número óptimo de neuronas en la capa oculta de una RNA; dichas fórmulas son utilizadas como referencia en este trabajo para ser puestas a prueba en una RNAC. La conclusión de dicho trabajo es que los resultados serán distintos en cada base de datos, señalando que no hay una fórmula general para atacar a todos los problemas. Un estudio similar fue realizado por Lunt [33] utilizando 14 fórmulas , de igual manera aplicadas a una RNAC, llegan a la conclusión que para bases de datos con pocas muestras es mejor utilizar un método heurístico. Ninguno de estos trabajos se centra en el estudio de bases de datos con gran dimensionalidad como las bases que se utilizan en este trabajo, por otro lado, las aplicaciones de estas fórmulas están limitadas a una RNAC y no son probadas en otras redes como RNAC.

III. Red Neuronal Auto-Codificadora (RNAC) y de Aprendizaje Profundo (RNAP)

Una Red Neuronal Auto-Codificadora (RNAC) o $auto-encoder$ es un modelo de aprendizaje automático no supervisado, en el cual se establece como valores objetivo los mismos valores de entrada, i.e., para un conjunto de datos de entrenamiento $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_q\}$, se espera que la salida de la RNAC para la entrada $\mathbf{x}_i$ sea el vector $\hat{\mathbf{x}}_i$, donde $\mathbf{x}_i \approx \hat{\mathbf{x}}_i$. La RNAC es un modelo neuronal de propagación hacia adelante y consta de una capa oculta y generalmente es entrenado por el método de $back-propagation$ [34]. En la capa oculta se genera un vector codificado $\mathbf{r}_i$, el cual corresponde a la entrada $\mathbf{x}_i$. Este tipo de algoritmo puede ser utilizado como filtro al existir una transformación de los datos (codificación) $C : \mathbf{x} \rightarrow \mathbf{r}$ y (decodificación) $D : \mathbf{r} \rightarrow \hat{\mathbf{x}}$, o como un reductor de dimensionalidad ya que $\mathbf{x} \in \mathbb{R}^m$ y $\mathbf{r} \in \mathbb{R}^n$ y m suele ser mayor a n [16].

Por otro lado, las redes neuronales de aprendizaje profundo (RNAP) o *deep learning neural networks* se caracterizan por tener múltiples capas ocultas que ayudan a abstraer problemas más complejos utilizando un número reducido de nodos o neuronas ocultas en su arquitectura en comparación a una red neuronal tradicional [8]. El objetivo de estos modelos es encontrar una relación entre los vectores entrada $\mathbf{x}_i$ con la salida $\mathbf{y}_i$. Esta salida $\mathbf{y}$ es una función de $\mathbf{x}$ y de $\mathbf{w}$ (pesos neuronales o sinápticos). El ajuste de $\mathbf{w}$ minimiza la diferencia o error que hay entre $\mathbf{y}$ y el valor real esperado $\mathbf{d}$. A los ajustes previamente mencionados, convencionalmente, se les llama entrenamiento, y este se realiza usando variantes o el algoritmo *back-propagation* [34]. El método Adam, el cual es una modificación del *back-propagation*, es uno de los métodos más comunes en el aprendizaje profundo [35]. Las funciones de activación más usadas en los RNAP son la función ReLU [8] (*Rectifies Linear Unit*) $f(z) = max(0, z)$ y la función *Sigmoide* [34] $f(z) = 1/(1 + e^{-\alpha z})$. La función ReLU es la más apropiada en ambientes de aprendizaje profundo debido a que al existir múltiples capas ocultas es menos propensa a la saturación a diferencia de la sigmoide. En la actualidad, existen diferentes modelos de RNAP como las convolucionales, o las recurrentes [8], sin embargo, en este trabajo se emplea una arquitectura de propagación hacia adelante del tipo Perceptron Multicapa o MLP (por sus siglas en inglés *Multilayer Perceptron*) con más de dos capas ocultas, la cual es considerada como una RNAP.

IV. Fórmulas (F) para Determinar el Número Óptimo de Neuronas en la Capa Intermedia de una Red Neuronal

Uno de los problemas que aún permanecen sin resolver en el campo de las RNA, es la identificación automática del número óptimo de neuronas en las capas ocultas. Durante décadas se han propuesto diferentes métodos para abordar este problema [26], y se han enfocado principalmente a redes de propagación hacia adelante con una capa oculta, en particular el Perceptron Multicapa. En este trabajo, se exploran las posibilidades de 14 propuestas obtenidas de los trabajos [18]–[25], [36]–[39] para la determinación del número apropiado de neuronas en la capa oculta de un MLP, y así encontrar el valor más acertado de neuronas ocultas en una RNAC, la cual es una variante de un MLP [34]. En la Tabla I se resumen las propuestas estudiadas para calcular el número de neuronas óptimo (N_h) de la capa oculta en un MLP. En ella se observan las expresiones para el cálculo de las neuronas ocultas, donde N_i es el número de neuronas en la capa de entrada o características, N_o es el número de neuronas en la capa de salida, que en el caso de un RNAC $N_i = N_o$, y, por último, N_{tr} corresponde al número de muestras en el conjunto de datos de entrenamiento. Asimismo, se muestran los resultados más relevantes obtenidos en los trabajos originales, los cuales debido a su naturaleza y propósito fueron medidos, principalmente, en términos de error cuadrático medio (e) [34] y exactitud en la clasificación (AC, Eq. 1). Los trabajos de las Ref. [18], [20], [25], [36], [38], [39] son estudios teóricos (E.T.), donde se muestran teoremas y demostraciones, que permiten establecer condiciones bajo las cuales se puede obtener el número óptimo de neuronas ocultas en una RNA, y no presentan resultados experimentales.

TABLA I
FÓRMULAS PARA EL CÁLCULO DEL NÚMERO (N_h) ÓPTIMO DE NEURONAS EN LA CAPA OCULTA EN UN MLP.

No.	Fórmula	Ref.	Año	Resultado
F1	$N_h \geq 2N_i + 1$	[18]	1993	E.T.
F2	$N_h \leq \frac{N_{tr}}{(N_i+1)}$	[19]	1994	$0.52 \leq AC \leq 1$
F3	$N_h = \sqrt{N_{tr}}$	[20]	1988	E.T.
F4	$N_h = log\,(N_{tr})$	[21]	1998	$0.76 \lesssim AC \lesssim 0.85$
F5	$N_h = \sqrt{N_i N_o}$	[22]	2009	$0.001 \leq e \leq 0.01$
F6	$N_h = \frac{(2N_i+1)}{(N_i-2)}$	[23]	2013	$e < 2.5 \times 10^{-8}$
F7	$N_h = C\left(\frac{N_{tr}}{N_i \log N_{tr}}\right)^{\frac{1}{2}}$	[24]	2008	$e < 0.0553$
F8	$N_h = \frac{N_{tr}}{N_i}$	[20]	1988	E.T.
F9	$N_h = \frac{N_{tr}}{(N_i+N_o)}$	[20], [25]	1988	E.T.
F10	$N_h = 2N_i$	[36]	1987	E.T.
F11	$N_h > N_o$	[37]	1988	$AC = 0.85$
F12	$N_h \leq N_{tr} - 1$	[39]	1991	E.T.
F13	$N_h = \frac{N_i}{N_{tr}}$	[20], [39]	1988	E.T.
F14	$N_h \geq \frac{N_i}{3}$	[38]	1993	E.T.

V. Método de Estudio Propuesto

En este trabajo se utiliza una RNAC para reducir la dimensionalidad de conjuntos de datos bio-médicos de microarrays de expresión genética, el cual es un reto vigente, no solo en el aprendizaje automático, sino también en el profundo. El propósito es obtener mejores o iguales resultados de clasificación a si se usa el conjunto de datos entero. El principal aporte de este trabajo es el estudio de las posibilidades de 14 fórmulas (previamente propuestas en el estado del arte [18]–[25], [36]–[39]), para la identificación automática del número de neuronas ocultas en una RNAC. El método empleado para este estudio, consiste de los siguientes pasos:

1) Identificar el número de neuronas ocultas (N_h) utilizando los métodos presentados en la sección IV.
2) Entrenar la RNAC con la configuración obtenida en el paso 1.
3) Usar los vectores codificados $\mathbf{r}_q$ (de dimensión N_h), como entrada a una RNAP (sec. III).
4) Comparar la precisión obtenida de la RNAP al utilizar los vectores codificados $\mathbf{r}_q$, respecto, a la generada por la RNAP con el conjunto de datos completo.

La Fig. 1 esquematiza el proceso anterior, en ella se observa a una RNAC utilizada para reducir el número de características de conjuntos de datos de alta dimensionalidad y posteriormente muestra, como los datos de la capa codificadora son empleados como entrada en un clasificador (en este caso a una RNAP), lo cual permite evaluar la efectividad de la reducción de la RNAC.

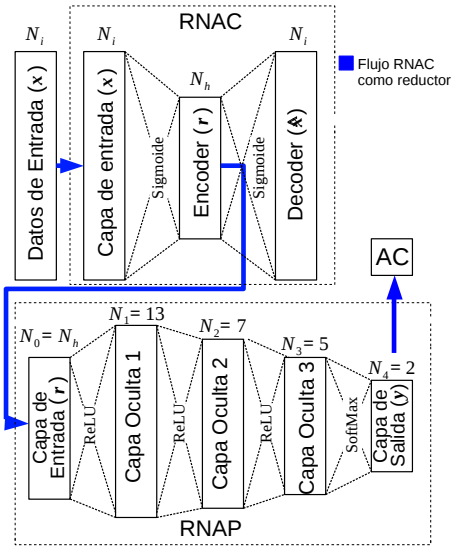

Fig. 1. Esquema del método de estudio seguido, para evaluar las posibilidades de14 fórmulas del estado del arte para obtener el número de neuronas ocultas (N_h) en una RNAC.

VI. Configuración de los Experimentos

Las bases de datos utilizadas para este trabajo corresponden a datos biomédicos de alta dimensión que incluyen información genética, datos de perfiles de proteínas y datos de secuencias genómica, extraídas del sitio Kent Ridge Biomedical DataSet Repository [40] (http://leo.ugr.es/elvira/DBCRepository/). La Tabla II muestra la codificación de los nombres de las bases de datos que constan de dos letras para facilitar su identificación, así como el número de atributos (N_i), número de muestras para el entrenamiento (N_{tr}) y número total de muestras (N_t). Los conjuntos de datos fueron normalizados utilizando la siguiente ecuación $x_j^i = x_j^i/|x_j^{max}|$, donde x_j^i es la característica o atributo j

de la muestra i y x_j^{max} corresponde al valor máximo del atributo j. La evaluación o prueba de la RNAC y de la

TABLA II
DESCRIPCIÓN DE LOS CONJUNTOS DE DATOS

Identificador	N_i	N_{tr}	N_t
BC	24481	68	97
CN	7129	42	60
CT	2000	43	62
DL	4026	33	47
LC	12533	127	181
LM	7129	67	96
OT	15154	177	253
PT	12600	95	136

RNAP se realizó utilizando el método *Hold-out* [34], el cual separa los archivos en dos partes mutuamente excluyentes, 70% para entrenamiento (EN), y 30% para la evaluación o prueba (EV), i.e., $EN \cap EV = \emptyset$, y el total de los datos $= EN \cup EV$. Se utilizó una RNAC con una capa oculta con N_h neuronas, calculadas por los métodos de la Tabla I. El entrenamiento es realizado con el algoritmo *back-propagation* [34] con una razón de aprendizaje $\eta = 0.01$ y 500 repeticiones o iteraciones. Para la RNAP se usa una configuración con 3 capas ocultas con 13, 7 y 5 neuronas respectivamente, con una función de activación ReLU [8]. La configuración fue obtenida mediante el método de prueba y error, para ello se probaron S configuraciones distintas y se adoptó la que produjo mejores resultados de clasificación en la mayoría de los conjuntos de datos. Para este proceso se utilizó únicamente el conjunto de entrenamiento (EN). EN se dividió en dos subconjuntos mutuamente excluyentes E (50%) y V (50%), el primero para entrenar la RNAP y el segundo para la validación de la configuración s del modelo, en otras palabras, $EN = E \cup V$, $E \cap V = \emptyset$, y $s = 1, 2, 3, ..., S$, donde $S = 15$. La capa de salida cuenta con dos neuronas y una función de activación *Softmax*. El entrenamiento es realizado con el método Adam [8] y una razón de aprendizaje $\eta = 0.001$ y 500 repeticiones. La medida de clasificación utilizada para la RNAP es la exactitud o acierto (AC), el cual es calculado por medio de la Ec. 1, donde NA corresponde al número de aciertos en la etapa de clasificación y T el total de muestras clasificadas.

$$AC = \frac{NA}{T}. \tag{1}$$

Debido a que el proceso de entrenamiento de las redes neuronales estudiadas en este trabajo es estocástico [34], cada uno de ellos genera una respuesta diferente por cada inicialización de la red neuronal, por esta razón y para dar solidez a los resultados presentados, el proceso de entrenamiento y clasificación es repetido 10 veces, tomando el mejor resultado de las diferentes inicializaciones de la RNAC, y de la RNAP. Esta es una práctica común en el contexto de las RNA, con diferentes variaciones, por ejemplo, véase la Ref. [41]. La RNAC y la RNAP fueron desarrolladas en la plataforma Keras 2.3.1 con Tensorflow 2.0 y el lenguaje de programación Python 3.7.5. El código fuente esta disponible en el sitio web: https://github.com/ccastore/NeruonasOcultasRedAutoencoder.

La prueba estadística de Friedman [42] es un Análisis estadístico no paramétrico en el cual se asigna un valor de clasificación (*ranking*) a los diferentes métodos, donde el valor 1 se le atribuye al mejor método, el 2 al segundo y así sucesivamente, en caso de existir un empate se calcula el promedio de la clasificación de ambos métodos. La prueba de Friedman utiliza los *ranking* promedio para calcular la estadística de Friedman, que puede ser calculada de la siguiente manera: $\chi_F^2 = \frac{12N}{K(K+1)}(\sum_j R_j^2 - \frac{K(K+1)^2}{4})$, donde K es el número de métodos a comparar, N el número de conjuntos de datos y R_j el rango promedio del método j en todos conjuntos de datos. No obstante, Iman y Davenport [43] demostraron que χ_F^2 tiene un comportamiento conservador, por esa razón, propusieron otro estadístico, el cual supera esta deficiencia y sigue una distribución F con $K-1$ y $(K-1)(N-1)$ grados de libertad, donde $F_F = \frac{(N-1)\chi_F^2}{N(K-1)-\chi_F^2}$. Por otro lado, si existen diferencias significativas en los métodos estudiados, es de utilidad saber cuál de esos métodos son los que muestran esas diferencias. Para ello se usa la prueba estadística post-hoc de Holm [44], la cual al igual que Friedman e Iman–Davenport, es una prueba estadística no paramétrica, que indica cuáles pares de métodos presentan diferencias significativas. La prueba de Holm rechaza todas las hipótesis nulas H_i una a la vez, hasta que ya no existen hipótesis por rechazar. Para esto se ordenan los valores p desde los más pequeños a los más grandes, i.e., $p_1 \leq p_2 \leq ... \leq p_i \leq ... \leq p_{K-1}$, los cuales corresponden a la secuencia de hipótesis $H_1, H_2, ..., H_{(K-1)}$. Posteriormente, el procedimiento rechaza desde H_1 hasta $H_{(i-1)}$, siempre y cuando i sea el menor entero tal que $p_i \leq \alpha/(K-i)$; en otras palabras, este procedimiento inicia con el valor de p más significante, donde $i = 1, 2, 3, ..., K-1$; α es el nivel de significancia estadística o confianza con un valor establecido en 0.05, y K es el número de métodos a comparar. En este trabajo, se usaron las pruebas de Friedman e Iman–Davenport, y cuando se observaron diferencias significativas en los resultados de los métodos estudiados, se aplicó la prueba de Holm. Se empleo el software KEEL [45], el cual es de acceso libre. El mecanismo considerado para para la medición de costo computacional al utilizar una RNAP como clasificador y una RNAC como reductor de la dimensión en conjuntos de datos de alta dimensionalidad, se muestra a continuación. Se propone utilizar como referencia el número de pesos neuronales o sinápticos que se tienen que ajustar para llevar acabo el proceso de clasificación o reducción. El número de parámetros para la RNAP es $P_{RNAP} = N_i N_1 + \sum_{j=2}^{L}(N_{(l-1)}N_l)$, i.e., la suma de todas las conexiones que hay entre las neuronas de cada capa, donde N_l es el número de neuronas de la capa l, y L es el número total de capas tomando en cuenta la capa de salida. Por otro lado, el número total de parámetros por calcular en la RNAC se obtiene como sigue: $P_{RNAC} = 2(N_i N_h)$. Finalmente, la ganancia opérdida en costo computacional (Ec. 2) esta dada por la relación que existe entre P_{RNAP} (usando la base de datos completa) y $P_{RNAC} + P_{RNAP}^r$, donde P_{RNAP}^r corresponde a la RNAP entrenada con la base de datos reducida o codificada, resultado del uso de la RNAC. Cuando $C < 0$ (Ec. 2) se habla de que hubo una reducción en el total de pesos neuronales, por lo tanto, existe una mejora en el rendimiento o reducción en el costo computacional, incluyendo el costo de

pre-procesamiento (P_{RNAC}).

$$C = \frac{P_{RNAC} + P^r_{RNAP}}{P_{RNAP}} - 1. \quad (2)$$

VII. Resultados y Discusión

En esta sección se muestran los principales resultados experimentales obtenidos en esta investigación. Primero, se muestran los resultados de aplicar las fórmulas de la sección IV a las bases de datos estudiadas (sección VI), y de esta forma obtener el número óptimo de neuronas ocultas (N_h) que será utilizado por la RNAC. Luego, se realiza un Análisis comparativo del uso de la base de datos completa y la codificada por la RNAC. Los resultados se miden en términos de efectividad en la clasificación (exactitud o acierto (AC, Ec. 1) y costo computacional (basado en el número de pesos sinápticos, Ec. 2). Finalmente, se incluye un Análisis estadístico no paramétrico, con el propósito de dar solidez a las conclusiones presentadas.

TABLA III
Valores óptimos de neuronas N_h calculados para una red autoencoder

	BC	CN	CT	DL	LC	LM	OT	PT
Orig.	24481	7129	2000	4026	12533	7129	15154	12600
F6	2	2	2	2	2	2	2	2
F4	6	5	5	5	7	6	7	7
F3	8	6	7	6	11	8	13	10
F13	360	170	47	122	104	106	86	258
F14	8160	2376	667	1342	4178	2376	505	8160
F2	0	0	0	0	0	0	0	0
F7	0	0	0	0	0	0	0	0
F8	0	0	0	0	0	0	0	0
F9	0	0	0	0	0	0	0	0
F12	1	1	1	1	1	1	1	1
F5	24481	7129	2000	4026	12533	7129	15154	24481
F11	24482	7130	2001	4027	12534	7130	15155	24482
F10	48962	14258	400	8052	25066	14258	30308	48962
F1	48963	14259	4001	8053	25067	14259	30309	48963

Los valores de N_h obtenidos por los métodos estudiados en la sección IV, se muestran en la Tabla III, y se consideran únicamente aquellas fórmulas que cumplen la condición $1 < N_h < N_i$, debido a que en este trabajo se busca reducir automáticamente la dimensionalidad de los conjuntos de datos bio-médicos de microarrays de expresión genética, el cual es un reto vigente en el aprendizaje automático, inclusive en el profundo. Las fórmulas F2, F7, F8 y F9, usan el número de características (N_i) como divisor (Tabla I), y considerando que una característica propia de las bases de datos estudiadas es su gran dimensionalidad, los valores obtenidos por estas fórmulas son fraccionarios o menores a 1, por lo cual no tiene sentido su aplicación. Las fórmulas F1, F2 , F11, F12 y F14 (ver Tabla I) retornan un intervalo de valores para N_h, por lo que para este trabajo se tomaron únicamente los valores más pequeños. La fórmula F12 es descartada porque solo indica que N_h es cualquier valor contenido en el intervalo $[0, N_{tr}]$, lo cual no es suficiente para los propósitos de este trabajo. La constante a de la fórmula F7, representa el primer momento absoluto de la distribución de la magnitud de Fourier de la función objetivo, la cual es desconocida como se menciona en [24], por lo que se asigna $a = 1$ para todas las bases de datos. En la Tabla III, se observa que F3, F4 y F6 obtienen los valores más pequeños para N_h, debido a que basan su cálculo en el número de muestras (F3 y F4), y en el número de clases (F6). F13 hace una mediación entre número de muestras y clases, obteniendo valores no muy grandes ni muy pequeños, de manera similar a F14; sin embargo, esta última solo usa los valores de entrada o características. La Tabla IV muestra los mejores resultados en exactitud o acierto AC obtenidos de clasificar los conjuntos de datos empleados en este estudio, y aplicando únicamente las fórmulas F3, F4, F6, F13, y F14. Solo se incluyen estas fórmulas porque son las que cumplen con la condición $1 < N_h < N_i$. En esta tabla se observa que con los conjuntos de datos BC y CN se obtienen los peores valores de AC en la mayoría de las fórmulas y con el conjunto de datos original (es decir, sin usar la RNAC), en el resto de las bases de datos, la mayoría de los valores de AC son mayores a 0.8. En otras palabras, los conjuntos de datos BC (24481 atributos) y CN (7129) son los más difíciles de clasificar correctamente, y no se observa una tendencia en el sentido de que sean más difíciles de clasificar por tener un mayor número de características. Los conjuntos de datos LC, PT y OT, los cuales tiene un número mayor de características que CN, no muestran dificultad en la clasificación, se observa en estas bases de datos valores de AC superiores a 0.9, lo cual confirma lo presentado en otros trabajos [46], [47], en el sentido de que no necesariamente es la cantidad de características quien define un valor de clasificación correcta alta. De acuerdo con los *rank* de Friedman, la mejor clasificación se obtiene cuando la base de datos no pasa por la RNAC, consiguiendo un *rank* promedio de 2.31, debido, posiblemente, a que no haypérdida de información, sin embargo, las F13 y F14 muestran resultados muy competitivos, mostrando *rank* de 2.37 y 2.43, respectivamente. En el caso particular de la F13, la efectividad obtenida en la clasificación es muy similar a la obtenida con el conjunto de datos original, aunque, con una disminución considerable en el número de características del conjunto de datos utilizado para el entrenamiento de la RNAP (véase la Tabla III). La peor fórmula es la F6, con un *ranking* promedio de 5.56, y coincide con el hecho de que usa la menor cantidad de neuronas ocultas con $N_h = 2$, por lo cual se pierde suficiente información de las muestras mostrando un resultado negativo en la posterior clasificación por la red neuronal artificial. F6 se basa únicamente en el número de clases en el conjunto de datos para obtener el número apropiado de neuronas ocultas, lo cual se considera que no es efectivo en el contexto de los conjuntos de datos microarrays de expresión genética, o en general, donde se tienen muy pocas clases. Hasta ahora, se ha discutido que el uso de la F13 para calcular las neuronas ocultas de una RNAC es bastante prometedor. Primero, se observa un desempeño muy semejante al obtenido al usar el conjunto de datos entero u original, pero con un conjunto de datos reducido significativamente. Por estas razones, es necesario el uso de una prueba estadística que permita determinar si en realidad la diferencia entre los valores obtenidos con los conjuntos de datos enteros y los reducidos es insignificante. Para ello, en este trabajo se usan las pruebas de Friedman e

TABLA IV
RESULTADOS AC Y RANKS DE FRIEDMAN

Base	Orig.	F13	F14	F3	F4	F6
BC	**0.67**	0.63	**0.67**	**0.67**	**0.67**	0.63
CN	0.67	**0.78**	0.61	0.56	0.72	0.61
CT	0.79	**0.89**	0.84	0.79	0.68	0.68
DL	**0.93**	**0.93**	**0.93**	0.87	0.73	0.80
LC	**1.00**	**1.00**	**1.00**	0.93	0.91	0.84
LM	**1.00**	**1.00**	**1.00**	**1.00**	0.97	0.93
OT	**1.00**	**1.00**	**1.00**	0.93	0.92	0.87
PT	**0.95**	0.83	0.90	0.71	0.76	0.66
Rank	2.31	2.37	2.43	3.93	4.37	5.56

Iman–Davenport, las cuales indicaran si existe o no diferencia estadística con un nivel de significancia del 95% (ver sección VI). El valor reportado, de acuerdo a Friedman considerando una distribución χ^2_F con 5 grados de libertad, es de 20.60, y el valor p calculado por la prueba de Friedman es 9.60 $\times 10^{-4}$. Por otro lado, los valores reportados para la prueba de Iman–Davenport, considerando una distribución F_F con 5 y 35 grados de libertad, es de 7.43, y el calculado por el estadístico Iman–Davenport es un valor $p = 7.65 \times 10^{-5}$. Las pruebas de Friedman e Iman–Davenport rechazan la hipótesis nula, es decir, existe una diferencia significativa en los resultados presentados en la Tabla IV. Por esta razón, es necesario el uso de un estadístico post-hoc para determinar qué pares de métodos son diferentes estadísticamente. En este trabajo se usa la prueba de Holm, cuyos resultados se muestran en la Tabla V, donde la hipótesis nula H_0 indica que el par de fórmulas analizadas tienen el mismo acierto (AC). H_0 será rechazada cuando $p \leq 0.004167$. Aquellos pares de fórmulas donde se rechaza H_0 son marcados en negritas. Los resultados de esta tabla indican que no hay una diferencia estadísticamente significativa al aplicar una RNAC con las fórmulas F13, F14, F3 y F4 respecto a la base de datos original. Las fórmulas F3, F4 y F14, muestran un comportamiento aceptable, sin embargo, la F13 muestra un comportamiento muy positivo, al tener resultados muy similares a los obtenidos al usar la base de datos original, y además de no existir diferencia estadística significante. Por otro lado, existe diferencia significativa entre los métodos: Original vs F6; F13 vs F6; F14 vs F6, en otras palabras, la F6 debe ser descartada por tener un desempeño negativo y con significancia estadística. Las fórmulas F3 y F4 basan su cálculo en el número de muestras de entrenamiento N_{tr}, por lo que en problemas con un mayor número de muestras o métodos de validación diferentes, el valor de N_h se incrementa. La F14 no toma en cuenta el tamaño de la muestra de entrenamiento y solo el número de características N_i, así que es invariable ante las diferentes particiones que se generen en el método de partición (en este caso *Hold-out*, ver sección VI) usado para evaluar el modelo, y al número total de muestras existentes. La F13 determina el valor N_h tomando en cuenta tanto la cantidad de muestras como de características, lo que podría justificar su buen desempeño. Continuando con el estudio de posibilidades de las fórmulas estudiadas, en el contexto de bases de datos de alta dimensionalidad, se presenta un estudio del costo computacional asociado, el cual incluye el pre-procesamiento realizado por la RNAC. Para ello, se

TABLA V
VALORES p DE LA PRUEBA DE HOLM

fórmula	Orig	F13	F14	F3	F4	F6
Orig	-	0.0500	0.0167	0.0063	0.0042	**0.0033**
F13	0.0500	-	0.0250	0.0071	0.0045	**0.0036**
F14	0.0167	0.0250	-	0.0083	0.0050	**0.0038**
F3	0.0063	0.0071	0.0083	-	0.0125	0.0056
F4	0.0042	0.0045	0.0050	0.0125	-	0.0100
F6	**0.0033**	**0.0036**	**0.0038**	0.0056	0.0100	-

utiliza la cantidad de conexiones necesarias para entrenar las redes neuronales (véase la sección VI). Es evidente, como se observa en la Tabla VI, que agregar un pre-procesamiento previo para reducir al dimensionalidad del conjunto de datos genera un costo computacional adicional. Lo mismo sucede si aplica un PCA o alguna otra RNAC, o si se realiza una selección de características por cualquier otro método; sin embargo, el principal aporte de este trabajo es la validación de algunos métodos del estado del arte que automatizan la decisión de a cuántas características reducir el conjunto de datos, y de este modo obtener un medio para conocer la cantidad de neuronas ocultas en la RNAC o características (en este contexto) a la que será disminuida la dimensionalidad del conjunto de datos de entrenamiento, evitando recurrir a un proceso empírico como el método de prueba y error. Al

TABLA VI
GANANCIA O PÉRDIDA EN COSTO COMPUTACIONAL

Base	Orig.	F13	F14	F3	F4
BC	0.00	54.38	1254.18	0.23	**-0.08**
CN	0.00	25.14	364.34	**-0.08**	**-0.23**
CT	0.00	6.22	101.42	0.08	**-0.23**
DL	0.00	17.75	205.26	**-0.08**	**-0.23**
LC	0.00	15.00	641.57	0.69	1.08
LM	0.00	15.30	364.34	0.23	**-0.08**
OT	0.00	12.23	775.87	1.00	0.08
PT	0.00	38.68	1253.99	0.54	0.08

analizar el costo computacional, estimado como se indica en la (Ec. 2), se encontró que las fórmulas F3 [20] y F4 [21] no lo incrementan (incluyendo el costo de la RNAC y el de la RNAP), y para algunas de las bases de datos utilizadas, incluso se reduce. La reducción del costo computacional se manifiesta con una cifra negativa en la Tabla VI, puesto que el número de pesos sinápticos total, incluyendo los requeridos por la RNAC y la RNAP es menor al utilizado por la RNAP entrenada con la base de datos original. Por otra parte, se observó que la aplicación de las fórmulas F13 y F14, si bien no afectan significativamente la efectividad del clasificador, si aumentan el costo computacional al usar una RNAC como una etapa previa. La fórmula con mejores posibilidades de éxito es la F13, la cual considera tanto las muestras contenidas en el conjunto de datos como su dimensionalidad. La disminución en la dimensionalidad con la fórmula F4 en la capa de entrada de la RNAP, reduce el número de pesos neuronales por calcular en un promedio del 17% en 5 de las 8 bases de datos, lo que se traduce en una importante reducción del costo computacional del algoritmo; en los tres conjuntos de datos restantes no se incrementa de manera importante el costo

computacional. Es notorio que el uso de la RNAC incrementa el costo computacional en algunas de las bases de datos (Tabla VI), sin embargo, en situaciones donde es necesario disminuir el número de características de los conjuntos de datos, el uso de alguna fórmula como la F13 (que considera tanto las características como el número de clases) es una alternativa muy atractiva por que evita el costo computacional asociado al proceso de prueba y error requerido para encontrar la configuración adecuada, el cual puede ser tan alto o bajo, dependiendo del nivel de experiencia del diseñador de la red neuronal. El costo computacional, establecido en términos del número de pesos sinápticos (Ec. 2), permite identificar el número de cálculos que se requieren para ejecutar las redes neuronales, repercutiendo positivamente en el software en donde se implementan; sin embargo, potencialmente puede también favorecer los recursos de hardware. Por ejemplo, los sistemas de cómputo embebido que hacen uso de FPGA (Field Programmable Gate Array) o ASIC (Application-Specific Integrated Circuit) [48], requieren un menor número de arreglos lógicos necesarios para el entrenamiento y procesamiento de las redes neuronales, en tanto menor sea el número de pesos sinápticos que deben calcular. En algunas tecnologías de cómputo móvil se ha presentado la necesidad de recurrir a aceleradores de hardware, en conjunto con plataformas como TensorFlow para aprovechar los limitados recursos de hardware que usualmente acompañan a estos dispositivos, en comparación con los sistemas de cómputo dedicados [49]; por lo que la reducción y optimización en el número de conexiones neuronales potencialmente reduce el tiempo de respuesta y aprovechamiento de recursos. En aplicaciones para redes eléctricas inteligentes (SME, Smart Energy Grids), la propuesta de este trabajo puede ser útil debido a que es persistente la búsqueda de alternativas para la optimización de métodos y modelos de programación de FPGA [50].

VIII. Conclusión

En este trabajo se presentó el estudio de 14 fórmulas o métodos del estado del arte para el cálculo óptimo del número de neuronas en la capa oculta de una red neuronal artificial. Este estudio fue dirigido a tratar de reducir la dimensionalidad de 8 conjuntos de datos (de naturaleza biomédica) que se encuentran en escenarios de *big data*. Se usó una Red Neuronal Auto-Codificadora (RNAC) como reductor y se evalúo al utilizar los vectores codificados por la RNAC como entrada de una Red Neuronal de Aprendizaje Profundo (RNAP) basado en un perceptrón multicapa.

El porcentaje de acierto en la clasificación se aplicó como criterio de medida de la efectividad de los métodos estudiados, asimismo, se usaron los $ranks$ de Friedman y una prueba estadística no paramétrica para determinar la significancia de los resultados. Los resultados mostraron que las fórmulas F13 [20], [39] y F14 [38] compiten muy de cerca con la efectividad en la clasificación obtenida con la base de datos entera, mientras que F3 [20] y F4 [21] tienen un desempeño inferior. Sin embargo, al realizar la prueba post-hoc de Holm, se determinó que no existe una diferencia estadísticamente significativa entre los resultados de emplear un conjunto de datos reducido por la RNAC con las fórmulas F3, F4, F13 y F14, con respecto a la base de datos completa. El análisis del costo computacional permite observar que las fórmulas F3 y F4 no lo incrementan, aun considerando la incorporación de ambas redes: RNAC y RNAP, logrando incluso una ligera reducción en algunas de las bases de datos. En situaciones donde es fundamental la reducción de la dimensionalidad de un conjunto de datos, la combinación de una RNAC y la F13, tiene grandes posibilidades, dado que no requiere de un proceso de prueba y error para decidir hasta donde la dimensionalidad de la base de datos debe ser reducida, y se obtienen resultados de clasificación muy semejantes a los generados al usar el conjunto de datos completo.

REFERENCIAS

[1] W. Alves, D. Martins, U. Bezerra, and A. Klautau, "A hybrid approach for big data outlier detection from electric power scada system," *IEEE Latin America Transactions*, vol. 15, no. 1, pp. 57–64, 2017.

[2] R. Elshawi, S. Sakr, D. Talia, and P. Trunfio, "Big data systems meet machine learning challenges: Towards big data science as a service," *Big Data Research*, vol. 14, pp. 1 – 11, 2018.

[3] S. Shilo, H. Rossman, and E. Segal, "Axes of a revolution: challenges and promises of big data in healthcare," *Nature Medicine*, vol. 26, no. 1, pp. 29–38, 2020.

[4] B. Deebak and F. Al-Turjman, "A novel community-based trust aware recommender systems for big data cloud service networks," *Sustainable Cities and Society*, vol. 61, p. 102274, 2020.

[5] A. Mumtaz, A. Bux Sargano, and Z. Habib, "Fast Learning Through Deep Multi-Net CNN Model For Violence Recognition In Video Surveillance," *The Computer Journal*, 07 2020.

[6] V. Morfino, S. Rampone, and E. Weitschek, *A Comparison of Apache Spark Supervised Machine Learning Algorithms for DNA Splicing Site Prediction*. Singapore: Springer Singapore, 2020, pp. 133–143.

[7] R. L. Rodrigues, J. L. C. Ramos, J. C. S. Silva, and A. S. Gomes, "Discovery engagement patterns moocs through cluster analysis," *IEEE Latin America Transactions*, vol. 14, no. 9, pp. 4129–4135, 2016.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[9] M. Zaharia, et al., "Apache spark: A unified engine for big data processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, 2016.

[10] M. Abadi, et al, "Tensorflow: A system for large-scale machine learning," in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16. Berkeley, CA, USA: USENIX Association, 2016, pp. 265–283.

[11] A. K. Dwivedi, "Artificial neural network model for effective cancer classification using microarray gene expression data," *Neural Computing and Applications*, vol. 29, no. 12, pp. 1545–1554, 2018.

[12] H. Liu and H. Motoda, *Feature Extraction, Construction and Selection: A Data Mining Perspective*. USA: Kluwer Academic Publishers, 1998.

[13] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[14] Y. Bengio, A. Courville, and V. P., "Representation learning: a review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[15] U. G. Mangai, S. Samanta, and P. R. Das S. Chowdhury, "A survey of decision fusion and feature fusion strategies for pattern classification," *IETE Tech. Rev.*, vol. 27, no. 4, pp. 293–307, 2010.

[16] D. Charte, F. Charte, S. García, M. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Information Fusion*, vol. 44, pp. 78 – 96, 2018.

[17] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[18] D. A. Sprecher, "A universal mapping for kolmogorov's superposition theorem," *Neural Networks*, vol. 6, pp. 1089–1094, 1993.

[19] L. L. Rogers and F. U. Dowla, "Optimization of groundwater remediation using artificial neural networks with parallel solute transport modeling." *Water Resources Research*, vol. 30, no. 2, pp. 457–481, 2 1994.

[20] E. B. Baum, "On the capabilities of multilayer perceptrons," *Journal of Complexity*, vol. 4, pp. 193–215, 1988.

[21] N. Wanas, G. Auda, M. S. Kamel, and F. Karray, "On the optimal number of hidden nodes in a neural network," Systems Design Engineering Department, University of Waterloo, Tech. Rep., 1998.
[22] K. Shibata and Y. Ikeda, "Effect of number oh hidden neurons on learning in large-scale layered neural networks," in *2009, ICROS-SICE International Joint Conference*, J. Fukuoka International Congress Center, Ed., 2009.
[23] K. G. Sheela and S. Deepa, "A new algorithm to find number of hidden neurons in radial basis function networks for wind speed prediction in renewable energy systems," *Journal of Control Engineering and Applied Informatics*, vol. 15, no. 3, 2013.
[24] S. Xu and L. Chen, "A novel approach for determining the optimal number of hidden layer neurons for fnn's and its application in data mining," in *5th International Conference on Information Technology and Applications (ICITA 2008)*, 2008.
[25] A. R. Barron, "Approximation and estimation bounds for artificial neural networks," *Machine Leraning*, vol. 14, pp. 115–133, 1994.
[26] R. Majalca and P. R. Acosta, "Convex hulls and the size and the size of the hidden layer in a mlp based classifier," *IEEE Latin America Transactions*, vol. 17, no. 06, pp. 991–999, 2019.
[27] Q. Fournier and D. Aloise, "Empirical comparison between autoencoders and traditional dimensionality reduction methods," in *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, 2019, pp. 211–214.
[28] D. C. Ferreira, F. I. Vázquez, and T. Zseby, "Extreme dimensionality reduction for network attack visualization with autoencoders," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–10.
[29] J. R. Muh. Ibnu Choldun, Santoso and K. Surendro, "Determining the Number of Hidden Layers in Neural Network by Using Principal Component Analysis," School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Tech. Rep., 2019.
[30] O. Meriwani, "Enhancing Deep Neural Network Perforamnce on Small Datasets by the using Deep Autoencoder." *An Assignment in Data Science*, 2019.
[31] R. Maulik, B. Lusch, and P. Balaprakash, "Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders," 2020.
[32] T. Vuji, T. Matijevi, J. Ljucovi, A. Balota, and Z. Evarac, "Comparative Analysis of Methods for Determining Number of Hidden Neurons in Artificial Neural Network," in *Faculty of Organization and Informatics*. Central European Conference on Information and Intelligent Systems, 2016, pp. 219–256.
[33] A. Lunt and S. Xu, "An Empirically-Sourced Heuristic for Predetermining the Size of the Hidden Layer of a Multi-layer Preceptron for Large Datasets," School of Engineeringn and ICT, University of Tasmania, Tech. Rep., 2016.
[34] R. Duda, P. Hart, and D. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.
[35] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
[36] J. Denker, D. Schwartz, B. Wittner, S. Solla, R. Howard, L. Jackel, and J. Hopfield, "Large automatic learning, rule extraction, and generalization," *Complex Systems*, pp. 877–922, 1987.
[37] P. Gallinari, S. Thiria, and F. Soulie, "Multilayer perceptrons and data analisys," in *IEEE International Conference on Neural Networks*, vol. 391, 1988, pp. 391–399.
[38] M. Arai, "Bounds on the number of hidden units in binary-valued three-layer neural networks," *Neural Networks*, vol. 6, pp. 855–860, 1993.
[39] S. Huang and Y. Huang, "Bounds on the number of hidden neurons in multilayer perceptrons," *IEEE Transactions on Neural Networks*, vol. 2, no. 1, pp. 47–55, 1991.
[40] K. Ridge, "Kent ridge biomedical data set repository," 2005.
[41] U. Markowska-Kaczmar and M. Koldowski, "Spiking neural network vs multilayer perceptron: who is the winner in the racing car computer game," *Soft Computing*, vol. 19, no. 12, pp. 3465–3478, 2015.
[42] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance." *Journal of the American Statistical Association*, vol. 32, 1937.
[43] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the friedman statistic," *Communications in Statistics - Theory and Methods*, vol. 9, no. 6, pp. 571–595, 1980.
[44] S. Holm, "A simple sequentally rejective multiple test procedure." *Scand J Statist*, vol. 6, 1979.
[45] I. Triguero, S. Gonzalez, J. Moyano, S. Garcia, J. Alcala-Fdez, J. Luengo, A. Fernandez, M. del Jesus, L. Sanchez, and F. Herrera, "Keel 3.0: An open source software for multi-stage analysis in data mining," *International Journal of Computational Intelligence Systems*, vol. 10, pp. 1238–1249, 2017.
[46] A. Reyes-Nava, J. Sánchez, R. Alejo, A. Flores-Fuentes, and E. Rendón-Lara, "Performance analysis of deep neural networks for classification of gene-expression microarrays," in *Pattern Recognition, MCPR 2018*, vol. 10880, June 2018, pp. 105–115.
[47] A. Reyes-Nava, et al., "Using deep learning to classify class imbalanced gene-expression microarrays datasets," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Cham: Springer International, 2019, pp. 46–54.
[48] A. Shawahna, S. M. Sait, and A. El-Maleh, "Fpga-based accelerators of deep learning networks for learning and classification: A review," *IEEE Access*, vol. 7, pp. 7823–7859, 2019.
[49] P. N. Whatmough, C. Zhou, P. Hansen, S. K. Venkataramanaiah, J. sun Seo, and M. Mattina, "Fixynn: Efficient hardware for mobile computer vision via transfer learning," 2019.
[50] C. Sandoval-Ruiz, "Lfsr-fractal ann model applied in r-ieds for smart energy," *IEEE Latin America Transactions*, vol. 18, no. 04, pp. 677–686, 2020.

Héctor Vega Nacido en la ciudad de Toluca en 1960, estudió ingeniería en Electrónica en la Universidad Autónoma Metropolitana Azcapotzalco y ahí descubrió su pasión por las computadoras lo que lo llevó a dedicar su actividad profesional a ellas. Maneja varios lenguajes de programación (Fortran, C, Python, Ensambladores). Actualmente reside en Metepec, México y está terminando la Maestría en Ciencias de la ingeniería en el Instituto Tecnológico de Toluca.

Carlos Castorena Ingeniero Mecatrónico titulado en 2015, actualmente cursa la Maestría en Ciencias de la ingeniería, ambos en el Instituto Tecnológico de Toluca (México), dentro de sus intereses de investigación se destacan: la inteligencia artificial y las redes neuronales artificiales en un contexto de big data.

Roberto Alejo Doctor en Sistemas Informáticos Avanzados por la Universitat Jaume I, España (2011), adscrito a la División de Estudios de Posgrado e Investigación del Tecnológico Nacional de México, campus Toluca, con un profundo interés científico en la aplicación de la inteligencia artificial a la solución de problemas reales. Asimismo es especialista en redes neuronales artificiales, aprendizaje automático y minería de datos.

Everardo E. Granda-Gutiérrez Doctor en Ciencias en ingeniería Electrónica por el Instituto Tecnológico de Toluca, México (2008). Es profesor en la Maestría en Ciencias de la Computación, adscrito al Centro Universitario UAEM Atlacomulco, de la Universidad Autónoma del Estado de México. Como área de interés central se destaca la aplicación de sistemas inteligentes en la instrumentación y control de procesos de manufactura no convencional.