Red Neuronal Artificial Backpropagation aplicada al reconocimiento de dígitos hexadecimales

Víctor Manuel Gutiérrez Corzo, Darío González Galindo
Miguel Ángel Ruiz Pinto, * Francisco Ronay López Estrada, * Joaquín Eduardo Domínguez Zenteno

Maestría en Tecnologías en Informática, Universidad Autónoma de Chiapas, 29050, México
(email: victor.gutierrez@unach.mx, dario.gonzalezg01@gmail.com, unoconmigo@hotmail.com)
*Tecnológico Nacional de México, Instituto Tecnológico de Tuxtla Gutiérrez. TURIX Grupo de Diagnóstico y control de Sistemas Dinámicos, Carretera Panamericana km 1080, Tuxtla Gutiérrez, Chiapas.

Resumen: El presente artículo detalla la implementación de un sistema de identificación de patrones numéricos utilizando Redes Neuronales Artificiales con algoritmo BackPropagation, el sistema es capaz de reconocer y representar los números del 0 al 9 y las letras por medio de una interfaz de usuario montado en un sistema embebido Arduino. La imagen se capturo a través de Matlab por medio de una cámara Web y considerando diferentes condiciones de iluminación para garantizar la robustez del algoritmo. Finalmente, este algoritmo se programó en un sistema embebido de bajo costo, lo cual demuestra su aplicabilidad y portabilidad.

.

Palabras claves: Red Neuronal Artificial (RNA), Backpropagation (BP), Reconocimiento de patrones, Sistema embebido.

1. INTRODUCCIÓN

En los últimos años las redes neuronales artificiales (RNA) han sido ampliamente utilizadas en el reconocimiento de patrones principalmente por la cantidad de información que puede ser analizada y la velocidad con que estas son procesadas (Ramírez *et al*, 2011). Por ejemplo, ha sido utilizado en el reconocimiento de patrones para detectar enfermedades en plantas (Zhang *et al*, 2013), análisis de rayos X (Rojas-Espinoza & Ortíz-Iribarren, 2011), letras (Beigi, 2015), mecánica de visión (Lancheros-Cuesta, 2015), entre otros; lo cual demuestra su aplicabilidad para solución de problemas reales.

La eficiencia de la RNA es muy variada, debido a ello se han propuesto diferentes metodologías con el fin de optimizar los tiempos de cómputo y el desempeño de los algoritmos, por ejemplo, en Cruz, (2011) se propone el método basado en el enfoque Backpropagation que considera el entrenamiento de una red multicapa, bajo aprendizaje supervisado con las instrucciones que describan cada salida y el valor esperado en cada una de ellas. En (Beale *et al,* 2014) Real-Time Recurrent Learning Algorithm, consistente en un algoritmo de aprendizaje de redes completamente recurrentes que se ejecutan continuamente en el tiempo de muestreo y Widrow-Hoff learning, que básicamente consiste en la implementación iterativa de la regresión lineal para reducir el error cuadrado de un ajuste lineal, entre otros. Entre ellos la técnica Backpropagation (BP) ha resaltado como una técnica poderosa y ampliamente utilizada para redes neuronales supervisadas, ya que proporciona un método computacionalmente eficiente para el entrenamiento de perceptrón multicapa (Rajini y Reddy, 2010). En este trabajo se considera el enfoque BP debido que puede programarse fácilmente y a que ha demostrado una amplia aplicabilidad en el reconocimiento de patrones.

El reconocimiento de patrones consiste en la forma que percibimos al mundo; el mecanismo con que contamos para distinguir los diferentes elementos que se encuentran a nuestro alrededor, y para realizarlo de manera automática se pueden emplear diferentes técnicas: Reconocimiento estadístico de patrones, Reconocimiento sintáctico de patrones, Redes Neuronales, Reconocimiento Lógico Combinatorio, entre otros (consultar Jesús y José, 2011) y las referencias que ahí se presentan).

Esta aplicación se ha incrementado en los últimos años debido al poder de procesamiento de las computadoras, particularmente en el manejo de grandes cantidades de información (Szeliski, 2010). Para llevar a cabo este proceso se utilizan diferentes imágenes y complejos algoritmos matemáticos; los cuales han sido adaptados para el desarrollo de las RNA's como se analiza en Hagan, (2014). Una de las ventajas de una RNA con BP es que puede ser programadas en pequeños sistemas embebidos, como lo son sistemas Arduino (Ortega-Zamorano *et al*, 2016), Raspberry Pi (Colombo, 2016), BeagleBone (Mendoza, 2015), entre otros.

Los sistemas embebidos no cuentan con un poder de cómputo equivalente a los sistemas de cómputo tradicionales, sin embargo, están diseñados para ser utilizados aplicaciones específicas y su bajo costo los hacen sumamente útiles en múltiples aplicaciones, como en el campo automotriz, teléfonos móviles, iPad, reproductores Blu-Ray, refrigeradores, alarmas de casas, lavadoras, cámaras fotográficas, instrumentación industrial, equipos médicos, Set Top Boxes como se analiza en el trabajo de Salas Arriarán, (2015) y las referencias ahí citadas. Debido a ello este trabajo

considera el uso de un sistema embebido de bajo costo y de fácil programación como herramienta computacional para probar los algoritmos propuestos.

En este trabajo se desarrolla una aplicación de RNA Backpropagation para el reconocimiento de números hexadecimales a través de la imagen capturada por una cámara web y por una botonera matricial implementada en el Arduino. El sistema tiene una salida de 4 bits como una salida parcial y como salida final una salida numérica en un display el número digital identificado. Se desarrolló una red perceptrón multicapa con funciones de activación sigmoidal entrenados por los algoritmos de aprendizaje. Con los datos de entrenamiento para el diseño implementado de la RNA BP y las funciones de transferencias logsig de Matlab se facilita el aprendizaje de la RNA desarrollada. Los resultados muestran la aplicabilidad del método en la aplicación de reconocimiento de patrones con un alto nivel de desempeño.

2. IMPLEMENTACIÓN DE LA RNA

El diagrama de identificación de patrones implementado se muestra en la Fig. 1. El sistema está compuesto por tres bloques: el primer bloque es la adquisición de la imagen compuesta por una cámara web y una fuente de luz externa, el segundo bloque es la arquitectura de la RNA utilizada en Matlab y Arduino; el último bloque es el reconocimiento y representación a través del dispositivo embebido con un LCD de 7 segmentos.

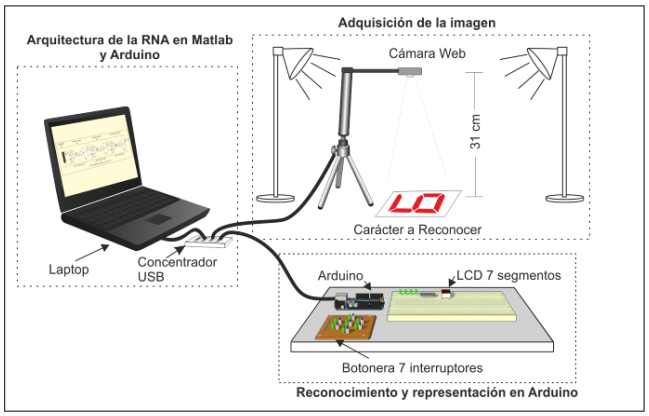

Fig. 1. Diagrama general del sistema RNA.

2.1 Adquisición de la imagen

Este bloque está compuesto de dos secciones: la primera corresponde al pre-procesamiento de la imagen en donde se lleva a cabo la segmentación de la imagen, la segunda sección es donde se obtienen las características de cada digito.

2.1.1 Pre-procesamiento de la imagen

El pre-procesamiento se realizó para eliminar la variabilidad que está presente en la adquisición de la imagen, en esencia el fin de esta sección es mejorar la calidad y adecuarla para la siguiente etapa. A continuación, se presenta un ejemplo del digito 'numeral 9', estas operaciones fueron realizadas en Matlab.

Captura de la imagen. Se utilizó la función “videoinput” esta permite la conexión entre MATLAB® y un dispositivo de adquisición de imágenes en particular, para capturar la imagen se utilizó una cámara de video USB con una resolución de 640x480, la imagen capturada se puede observar en la Fig. 2(a).

Conversión a escala de grises. Se convirtió en formato de escala de grises mediante el uso de la función de "rgb2gray", la imagen resultante se muestra en la Fig. 2 (b).

Conversión a binario. Mediante la función “imbinarize” se creó una imagen binaria reemplazando todos los valores por encima de un umbral determinado globalmente con “1” y estableciendo todos los demás valores en “0”. El objetivo de la binarización es minimizar la información no deseada que pueda estar presente en la imagen como se observa en la Fig. 2(c).

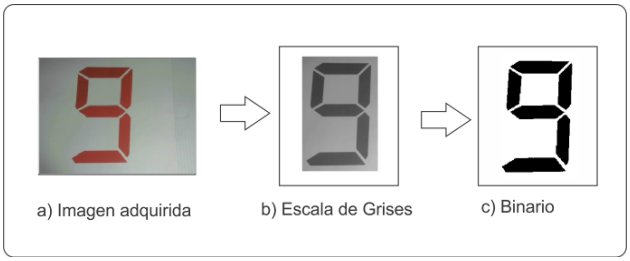

Fig. 2. Pasos del pre-procesamiento de imagen.

2.1.2 Extracción de la matriz para entrenamiento

Se definieron las características que conforman los elementos delimitados de la imagen binaria, en donde un '1' indica la presencia de un píxel blanco y un '0' representa la presencia de un píxel negro como se muestra en la representación matricial binaria del digito '9' en la Fig. 3(a). Posteriormente se redimensionó a una matriz 5 × 7 en donde se invierte la lógica de los pixeles, tal como se muestra en la Fig. 3(b). Esta matriz binaria de tamaño 5x7 se reformó en una matriz binaria de tamaño 35 × 1 como se observa en la Fig. 3(c). Del mismo modo, todos los vectores de características de los 16 dígitos del LCD de 7 segmentos (0-F) se crean en forma de matriz de columna binaria siguiendo este mismo método.

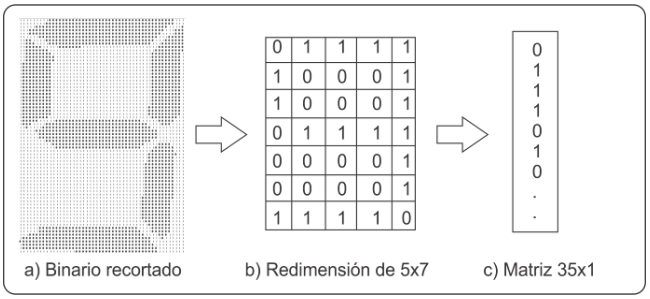

Fig. 3. Pasos del pre-procesamiento de imagen.

2.2 Arquitectura de la RNA Matlab y Arduino

La RNA BP diseñada en Matlab para el reconocimiento de imagen, está compuesta en la entrada por una primera capa de 16 neuronas, una capa oculta de 30 y una capa de salida de 4. La primera capa corresponde a los 16 dígitos a reconocer, la capa oculta se determinó por métodos heurísticos, la capa de salida está compuesta por 4 neuronas la cual nos proporcionará el resultado final, la función de entrenamiento utilizada en el diseño fue la Levenberg-Marquardt backpropagations (trainlm). La función de transferencia que se utilizó en toda la RNA fue logsig, en la Fig. 4 se muestra la RNA en su modelo matemático en Matlab. Y para el caso de la botonera matricial la RNA implementada en el Arduino se compone de 3 capas: la 1a. capa de 7 neuronas, una capa oculta de 6 neuronas y la capa de salida de 4, utilizando la función de transferencia tansig, logsig y purelin con entrenamiento trainlm. La Fig. 5 se muestra la RNA en su modelo matemático.

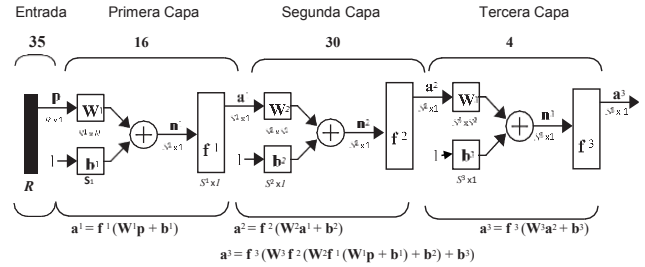

Fig. 4.- Modelo Matemático RNA en Matlab.

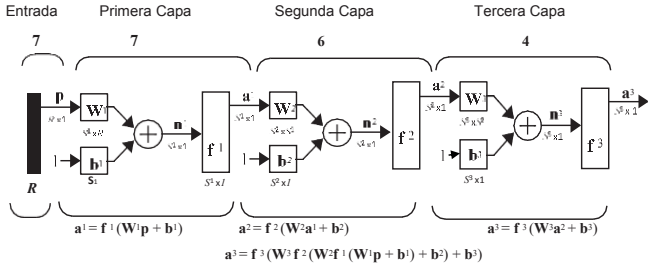

Fig. 5.- Modelo Matemático RNA en Arduino.

2.3 Modelado matemático de la RNA en el Arduino.
De acuerdo al modelo matemático se desarrollan las siguientes ecuaciones:

$$a^1 = f^1(W^1 \mathrm{p} + b^1) \quad (1)$$

Primera Capa 7 neuronas.

$$a1_1 = (P1 * W1_{1,1}) + (P2 * W1_{2,2}) + (P3 * W1_{3,3}) + (P4 * W1_{4,4}) + (P5 * W1_{5,5}) + (P6 * W1_{6,6}) + (P7 * W2_{7,7}) + b1_1$$

.

.

$$a1_7 = (P1 * W1_{1,1}) + (P2 * W1_{2,2}) + (P3 * W1_{3,3}) + (P4 * W1_{4,4}) + (P5 * W1_{5,5}) + (P6 * W1_{6,6}) + (P7 * W2_{7,7}) + b1_7 \quad (2)$$

Aplicando a la salida de esta primera neurona la función de transferencia tansig:

$$a = \frac{e^n - e^{-n}}{e^n + e^{-n}} \quad (3)$$

Obtenemos las primeras 7 salidas:

$$S_1 = tansig(a1_1)$$

.

.

$$S_7 = tansig(a7_1) \quad (4)$$

Siguiendo con el modelo matemático aplicamos una capa oculta de 6 Neuronas de la cual obtenemos:

$$a^2 = f^2(W^2 a^1 + b^2) \quad (5)$$

$$a2_1 = (S1_1 * W2_{1,1}) + (S1_2 * W2_{2,2}) + (S1_3 * W2_{3,3}) + (S1_4 * W2_{4,4}) + (S1_5 * W2_{5,5}) + (S1_6 * W2_{6,6}) + (S1_7 * W2_{7,7}) + b2_1$$

.

.

$$a2_6 = (S1_1 * W2_{1,1}) + (S1_2 * W2_{2,2}) + (S1_3 * W2_{3,3}) + (S1_4 * W2_{4,4}) + (S1_5 * W2_{5,5}) + (S1_6 * W2_{6,6}) + (S1_7 * W2_{7,7}) + b2_6 \quad (6)$$

Aplicamos la función de transferencia logsig a las salidas de la capa oculta de 6 neuronas:

$$a = \frac{1}{1+e^{-n}} \quad (7)$$

Obtenemos las siguientes salidas:

$$S1_2 = logsig(a2_1)$$

.

.

$$S6_2 = logsig(a2_6) \quad (8)$$

Continuando la descripción del modelo matemático aplicamos la capa de salida con 4 Neuronas de la cual obtenemos:

$$a^3 = f^3(W^3 a^2 + b^3) \quad (9)$$

$$a3_1 = (S2_1 * W3_{1,1}) + (S2_2 * W3_{2,2}) + (S2_3 * W3_{3,3}) + (S2_4 * W3_{4,4}) + (S2_5 * W3_{5,5}) + (S2_6 * W3_{6,6}) + (S2_7 * W3_{7,7}) + b3_1$$

.

.

$$a3_4 = (S2_1 * W3_{1,1}) + (S2_2 * W3_{2,2}) + (S2_3 * W3_{3,3}) + (S2_4 * W3_{4,4}) + (S2_5 * W3_{5,5}) + (S2_6 * W3_{6,6}) + (S2_7 * W3_{7,7}) + b3_4 \quad (10)$$

A esta última capa de salida se le aplica la función de transferencia purelin.

$$a = n \quad (11)$$

Por lo tanto, obtenemos las siguientes ecuaciones de salida de nuestro sistema RNA:

$$S_{1=purelin(a3_1)}$$
$$S_{2=purelin(a3_2)}$$
$$S_{3=purelin(a3_3)}$$
$$S_{4=purelin(a3_4)} \quad (12)$$

Posterior al modelado matemático y en base a las ecuaciones desarrolladas se implementa el código correspondiente en la tarjeta del Arduino.

2.4 Reconocimiento y representación en sistema embebido

Una vez que el dígito ha sido reconocido en Matlab, se envía por el puerto serial una cadena de 7 valores, correspondientes a los segmentos del display.

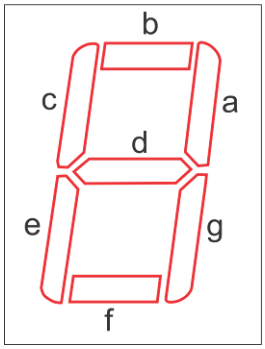

Fig. 6.- Disposición de los segmentos del display

Una cadena de texto es recibida por la tarjeta Arduino, esta es almacenada en una matriz, los valores de cada vector son extraídos y convertidos a número entero de forma individual, y almacenados en las variables de entrada de la RNA programada en la tarjeta embebida. Este proceso se lleva a cabo con cada uno de los dígitos reconocidos por MatLab, o bien sean introducidos por medio de una botonera de 7 interruptores. Terminada esta etapa se obtienen 4 salidas, las cuales son enviadas a puertos digitales, en donde están conectados 4 led's, a manera de monitor; y de igual forma las salidas son representadas en un LCD de 7 segmentos, como puede observarse en la Fig. 7.

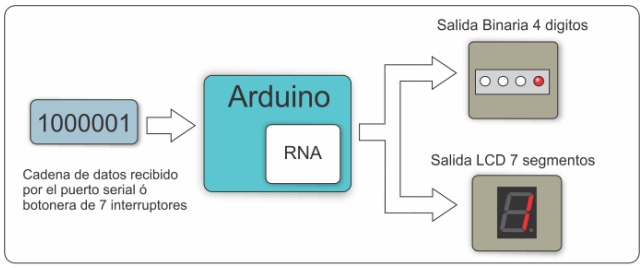

Fig. 7. – Diagrama de bloques comunicación Matlab con Arduino.

3. RESULTADOS

Se realizó el entrenamiento de forma supervisado, donde el conjunto conocido de datos de entrada-salida se utilizó para, iterativamente ir ajustando los pesos de la red. Se configuró el net.trainParam.goal a 1e-9, para obtener mayor precisión, en la Fig. 8 se puede observar el desempeño obtenido de la red en Matlab, en 24 épocas obtuvimos la precisión deseada que va en función de las neuronas configuradas, a mayor cantidad neuronas se reduce el número de épocas de entrenamiento, sin llegar al sobre entrenamiento.

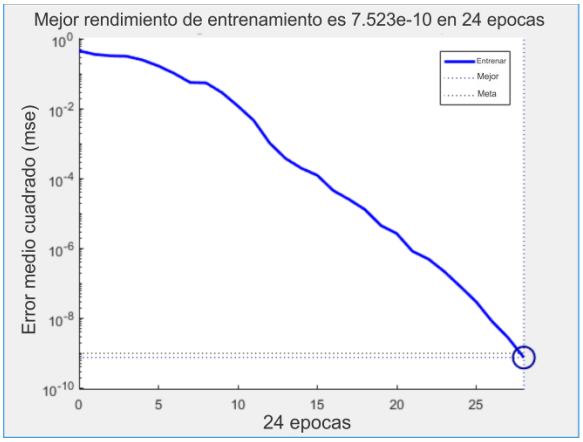

Fig. 8. – Rendimiento de la red entrenada.

De igual forma se realizó la prueba de matriz de confusión para determinar el porcentaje de precisión de la red entrenada, se obtuvieron los datos observados en la Fig.10. Se puede analizar que para el caso de la salida 1, se obtuvo un 43.8% de precisión, para la salida 2, un 18.8% de precisión, para la salida 3 un 12.5% de precisión, para la salida 4 un 6.3% de precisión, en general la RNA tiene un 81.3% de precisión, esta se puede mejorar si se incrementa el tamaño de las neuronas, así como el número de muestras a entrenar.

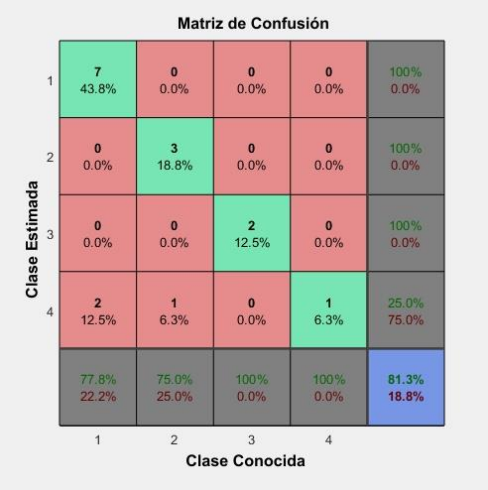

Fig. 9. – Matriz de confusión de la RNA.

Para determinar con qué porcentaje de error realiza el reconocimiento, se realizó un segundo entrenamiento con datos con ruido como la iluminación, calidad de la cámara WEB, calidad y color de los patrones a reconocer y se comparó con la capacidad de reconocimiento de la red diseñada. De esta manera se crearon 16 copias con ruido por cada dígito, los valores son limitados por * min y * máx., para que los valores estén entre 0 y 1. Los objetivos de salida también fueron definidos. Se puede observar el resultado de la comparación.

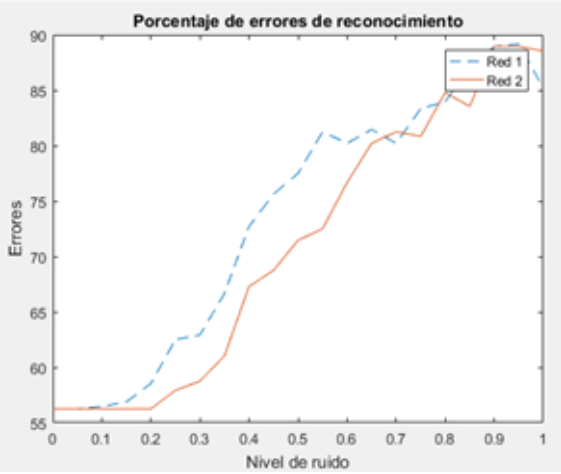

Fig. 10. – Porcentaje de error de la RNA con ruido.

Se observa que la Red 1 presenta más errores que la Red 2, debido a que la primera fue entrenada sin ruido a diferencia de la segunda donde sí se consideró la perturbación.

La red diseñada reconoce los dígitos entrenados, se obtuvo los resultados esperados, el sistema realizó el reconocimiento de los 16 dígitos tipo LCD de 7 segmentos. Para facilidad de uso se diseñó una interfaz de usuario en donde se realiza el proceso de captura y selección de la imagen a reconocer, como puede observarse en la Fig. 11.

Fig. 11. Interfaz de usuario del sistema RNA.

Como se menciona en la sección anterior, el proyecto se implementó en un sistema embebido tal como se muestra en la Fig. 13, donde se pueden observar los elementos físicos tales como la cámara web USB, la botonera, el sistema embebido, la placa de desarrollo (protoboard) y el display LCD de 7 segmentos.

Fig. 12. Sistema RNA implementado.

Sistema neuronal con sus dos formas de adquisición de datos: por imagen y botonera matricial, una vez que se obtuvieron los datos de la imagen capturada por la cámara Web fueron procesados en la Red Neuronal en Matlab y en el sistema embebido. Los datos de la imagen se recibieron a través de la interface serial del Arduino, reconociéndose los dígitos tipo LCD de 7 segmentos los cuales constituyen una entrada. Para el caso de la botonera matricial se conectaron 7 entradas digitales directamente al sistema embebido el cual reconoció cada segmento otorgado. Para ambos casos, la salida de la RNA entregó 4 bits, los cuales fueron visualizados en una interfaz (PC) y en el LCD del sistema embebido.

4. CONCLUSIONES

En este trabajo se propuso la implementación de un sistema de reconocimiento de dígitos hexadecimales en Matlab y en un sistema embebido; el sistema presentó un funcionamiento adecuado en el reconocimiento de la imagen de los dígitos, dando la opción de que se pueda incluso reconocer el color de los mismos; toda esta información se almacena en una base de datos. La función de entrenamiento trainlm fue rápida, los requisitos de memoria fueron relativamente pequeños al transferir los datos por puerto serial. El tiempo de convergencia para el Arduino fue imperceptible.

Para garantizar la efectividad y robustez del método, las imágenes estuvieron expuestas a ruido ambiental tales como la iluminación, la calidad y color de los patrones a reconocer, es por esto que fue importante realizar la comparación de la red entrenada contra otra con ruido, con esto se determinó que la red entrenada obtuvo un 81.3% de precisión. El algoritmo desarrollado fue eficientemente implementado en un sistema embebido, y se construyó un montaje físico que demostró su aplicabilidad.

Futuros trabajos se desarrollarán para aplicar está técnica al reconocimiento de plagas en árboles frutales e incrementar el

nivel de sensibilidad de la RNA para reducir el porcentaje de error.

REFERENCIAS

Jesús Ariel Carrasco Ochoa y José Francisco Martínez Trinidad. *Reconocimiento de patrones*. Komputer Sapiens. Año III, Vol. II,2011, Sociedad Mexicana de Inteligencia Artificial, A.C.

Colombo Vlaeminch, R. (2016). *Deep Learning para reconocimiento de imágenes en Raspberry Pi 2*. Bajo Licencia de Creative Commons.

Cruz, P. P. (2011). *Inteligencia artificial con aplicaciones a la ingeniería*. Alfaomega.

Demuth, H. B., Beale, M. H., De Jess, O., & Hagan, M. T. (2014). *Neural network design*. Martin Hagan.

Huerta, H. V., Vásquez, A. C., Dueñas, A. M. H., Loayza, L. A., & Naupari, P. J. R. (2009). *Reconocimiento de patrones mediante redes neuronales artificiales*. Revista de investigación de Sistemas e Informática, 6(2), 17-26.

Hugo Vega Huerta, Augusto Cortez Vásquez, Ana María Huayna, Luis Alarcón Loayza, Pablo Romero Naupari. *Revista de Ingeniería de Sistemas e Informática Vol . 6, No 2, 2009*

Beigi, M. V. (2015). *Handwritten Character Recognition Using BP NN, LAMSTAR NN and SVM*. Northwestern University, SPRING.

Mark Hudson Beale, Martin T. Hagan, Howard B. Demuth. The MathWorks, Inc. (2017a). *Neural Network Toolbox™ Reference. r2017a*. Recuperado 6 junio, 2017

Mendoza Ticona, Freddy Rodrigo. (2015). *Sistema embebido para el reconocimiento de escritura sobre plataforma Linux embebido*. Journal of Research of the University of Quindio, Vol. 27 Issue 2, p29-33. 5p

Ortega-Zamorano, F., Jerez, J. M., Muñoz, D. U., Luque-Baena, R. M., & Franco, L. (2016). *Efficient implementation of the backpropagation algorithm in fpgas and microcontrollers*. IEEE transactions on neural networks and learning systems, 27(9), 1840-1850.

Ramírez Q. Juan A. y Chacón M. Mario I (2011). *Redes neuronales artificiales para el procesamiento de imágenes, una revisión de la última década*. RIEE&C, Revista de Ingeniería Eléctrica, Electrónica y Computación, Vol. 9 No. 1, JULIO 2011.

Rajini, G. K., & Reddy, G. R. (2010, February). *Performance evaluation of neural networks for shape identification in image processing. In Signal Acquisition and Processing*, 2010. ICSAP'10. International Conference on (pp. 255-258). IEEE.

Richard Szeliski (2010). *Computer Vision: Algorithms and Applications*. 2010 Springer. http://szeliski.org/Book/.

Salas Arriarán, S. (2015). *Todo sobre sistemas embebidos*. 1st ed. Lima: Editorial UPC.

Sánchez E, Alanis A. 2006. *Redes Neuronales: Conceptos fundamentales y aplicaciones a control automático*. Madrid. Prentice-Hall. 210.

Valencia Reyes, M. A. (2007). *Algoritmo Backpropagation para Redes Neuronales: conceptos y aplicaciones*. Instituto Politécnico Nacional. Centro de Investigación en Computación.

Zhang, W., Teng, G., & Wang, C. (2013). *Identification of jujube trees diseases using neural network*. Optik-International Journal for Light and Electron Optics, 124(11), 1034-1037.

Rojas-Espinoza, G., & Ortíz-Iribarren, O. (2011). *Identification of knotty core in Pinus radiata logs from computed tomography images using artificial neural network*. Maderas. Ciencia y tecnología, 12(3).

Lancheros-Cuesta, D., Schlenker, C. S., & Morales, L. F. (2015, June). *System based on machine vision for translation of fingerspelling alphabet to latin alphabet*. In Information Systems and Technologies (CISTI), 2015 10th Iberian Conference on (pp. 1-5). IEEE.