Clasificación de granos de café usando FPGA

Jorge Hernández*, Flavio Prieto*[§],

** Grupo de Percepción y Control Inteligente (PCI)*
Universidad Nacional de Colombia, sede Manizales
[§] *e-mail: faprietoo@unal.edu.co*

(Recibido: Diciembre 16 de 2004 - Aceptado: Noviembre 29 de 2005)

Resumen

Este artículo presenta el desarrollo de un sistema para la clasificación de granos de café, según la etapa de maduración, utilizando Redes Neuronales Artificiales (RNAs). Como herramienta de clasificación se utilizaron dos estructuras de RNAs, Perceptrón Multi-Capa (MLP) y el modelo de red neuronal basado en bloques (BBNN). La estructura MLP fue diseñada e implementada sobre C++ usando el algoritmo de aprendizaje con retro-propagación del error. Para aumentar la velocidad de ejecución de la RNA se implementó bajo dispositivos electrónicos utilizando su paralelismo natural. El modelo BBNN consiste en un arreglo bidimensional de bloques fundamentales y pesos enteros, dirigidos a la fácil implementación sobre dispositivos electrónicos configurables como los arreglos de compuerta programable por campo (FPGAs). La optimización de la estructura usa un algoritmo genético. Esta estructura ha sido implementada y sintetizada sobre la tarjeta Altera Flex 10K FPGAs. El porcentaje de efectividad para la estructura MLP fue 91.7% y para el modelo BBNN fue 89.5%.

Palabras Clave: Redes Neuronales Artificiales, RNAs, BBNN, VHDL, FPGA, Clasificación de Patrones.

Abstract

This work presents the development of a system for the classification of coffee beans based on the maturity stages using Artificial Neural Network Systems (ANNs). As classification tool two RNAs' structures were used, Multi-Layer Perceptron (MLP) and the block-based neural network model (BBNN). Multilayer perceptron structure (MLP) has been designed and implemented on C++ using the back-propagation learning algorithm. To increase the execution speed for ANN, it has been implemented on hardware using natural parallelism. The block-based neural network (BBNN) model consists of a two-dimensional array of fundamental blocks and integer weights in order to allow easier implementation with reconfigurable hardware such as field programmable gate array (FPGAs). The architecture is globally optimized using a genetic algorithm. This architecture has been implemented and synthesized on Altera Flex 10K FPGAs. The percentage of effectiveness for MLP structure was 91.7\% and for BBNN model was 89.5\%.

Keywords: Artificial neural network systems, ANNs, BBNN, VHDL, FPGA, Pattern classification.

1. Introducción

En la actualidad la recolección de café es realizada de forma manual; sin embargo al plantear una cosecha automatizada se obtiene como resultado un café poco homogéneo a la hora de iniciar el proceso de beneficio. La poca selectividad del café, según su etapa de maduración, granos verdes, maduros y sobremaduros, entre otros, disminuye su calidad considerablemente. Dependiendo del estado de maduración en que se encuentra el café se producen diferentes sabores y aromas **[1]**. Esta caída en la calidad trae como consecuencia una disminución en su precio frente al mercado mundial. La clasificación de los granos antes del proceso de beneficio depende de la subjetividad del recolector. Este proceso puede

ser reemplazado por los sistemas de visión artificial, dado que estos últimos se basan en el análisis de diferentes características tales como el tamaño y el brillo.

Por esta razón se propone un sistema de visión artificial antes del proceso de beneficio para la clasificación de granos de café, según la etapa de maduración, utilizando como característica principal la intensidad del color de su epidermis.

Como herramienta de clasificación se propone un clasificador con redes neuronales artificiales (RNAs). Las RNAs son capaces de entender el significado de diferentes formas visuales o de distinguir entre varias clases de objetos a altas velocidades por su funcionamiento en paralelo, simulando el funcionamiento complejo del cerebro humano. Ellas logran su mayor rendimiento en plataformas paralelas que requieren gran capacidad de procesamiento, como los procesadores en paralelo y los dispositivos lógicos programables (FPGAs), entre otros. Estos últimos presentan una alternativa viable para la implementación de las RNAs en hardware. Las RNAs clásicas, Perceptrón Multi-Capa, comprenden algunas operaciones que se convierten en un problema a la hora de la implementación en hardware, como el almacenamiento de pesos reales (punto flotante), multiplicación sináptica y la función de actividad no-lineal, las cuales requieren una capacidad circuital muy grande **[2]**.

Aunque existen diferentes topologías de RNAs diseñadas para la implementación sobre plataformas digitales que se basan en técnicas como bloques fundamentales **[3]**, modulación de frecuencia **[4]**, computación estocástica **[5]**, redes rápidas sin multiplicadores **[6]**, entre otras, se implementó la red neuronal basada en bloques por su propiedad de pesos enteros y función de activación lineal.

En un comienzo el clasificador se desarrolla en C++ bajo DOS en un procesador de 1.67Ghz, con el fin de realizar el entrenamiento de las RNAs, cálculo de los pesos sinápticos, los algoritmos de optimización respectivos y las topologías empleadas antes de ser implementadas en el FPGA; además, para tener un referente con el cual comparar los tiempos de ejecución y el desempeño del sistema. El diseño para la programación del FPGA es desarrollado en VHDL y simulado bajo el entorno MAX+PLUS II de Altera versión 10.1 y sintetizado en la tarjeta UP2 del programa universitario de Altera en el dispositivo de la familia FLEX10K EPF10K70RC240-4 **[7]**, que cuenta con 70.000 compuertas y 2Mb de memoria RAM y además trabaja con un reloj de 25.175 MHz.

Este artículo se encuentra organizado de la siguiente forma. En la sección 2 se realiza la descripción general de las técnicas y procedimientos empleados en cada uno de los módulos que conforman el sistema. La sección 3 trata acerca de la topología utilizada en el sistema de clasificación según la plataforma de implementación. La sección 4 trata acerca de la implementación del sistema de clasificación. En la sección 5 se exponen los resultados del sistema de clasificación implementado. Finalmente, en la sección 6, se presentan las conclusiones y recomendaciones.

2. Desarrollo

2.1. Adquisición

El proceso de adquisición fue realizado tomando imágenes con un único grano y diferentes condiciones de iluminación, fondo y cámara **[8]**, **[9]**. El estudio se realizó en 11 muestras que comprenden 8 semanas, las cuales se separaron en *n* etapas de maduración (EM), como lo muestra la Tabla 1. Se tomaron un total de 5.800 imágenes con seis tipos de iluminaciones, tres tipos de fondo (blanco, azul oscuro y azul claro) y dos tipos de cámara (cámara de vídeo JVC y cámara fotográfica digital logitec).

Tabla 1: Relación entre: Muestras, Semanas y Etapas de Maduración (EM)

Muestra	1	2	3	4	5	6	7	8	9	10	11
Semanas	1	2	3	4	5	5	6	6	7	7	8
8 EM	1	2	3	4	5	6		7		8	
7 EM	1		2	3	4	5		6		7	
6 EM	1		2		3	4		5		6	
5 EM	1		2			3		4		5	
4 EM	1		2			3			4		
3 EM		1				2			3		

2.2. Segmentación

En un sistema de visión artificial, la segmentación es una etapa determinante en el procesamiento de imágenes, que busca aislar los objetos de interés, para luego realizar el análisis de sus características **[10]**. Existen una gran cantidad de algoritmos para realizar la segmentación de objetos en una imagen, pero su utilización depende de la aplicación específica.

Para las aplicaciones sobre la plataforma digital, y teniendo en cuenta que la imagen a segmentar es cuadrada y el grano se encuentra completamente centrado, se normaliza la imagen a un cuadrado de dimensiones $N_n \times N_n$ y se toma un segmento cuadrado de la parte interna de la imagen de dimensiones $N_s \times N_s$ (Fig. 1), múltiplo de 2, para facilitar las operaciones digitales en el proceso de caracterización.

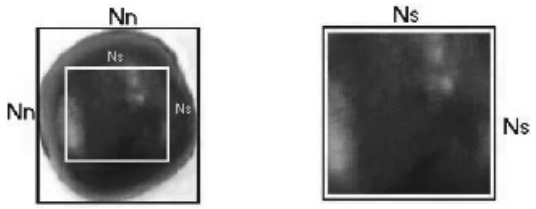


Figura 1: (a) Imagen Normalizada, (b) Ventana de Segmentación

Para establecer el valor de N_n y N_s se observa el porcentaje de grano y no grano seleccionado por la ventana de segmentación tal que la Ec 1 se cumpla y desde una longitud de N_n mayor o igual a N_s.

$$\max\left(\frac{N_n}{N_s}\right) \quad Si \quad \%NoGrano\left(N_s^2\right) = 0 \qquad (1)$$

Tabla 2: Porcentaje de Grano y No Grano dependiendo de la relación Ns:Nn

N_s:N_n	% Grano	% No Grano
1:1.25	88.9904	8.6339
1:1.5	67.9680	1.8937
1:1.75	51.9506	0.6090
1:2	41.1943	0
1:2.25	33.8147	0

De esta forma, después de probar diferentes relaciones, se seleccionó la relación N_s:N_n = *1:2* (Tabla 2).

2.3. Caracterización

La caracterización se refiere al proceso de extracción de algunas medidas numéricas de la imagen segmentada. Para este caso el color de la epidermis del grano es la más representativa, ésta es descompuesta en sus respectivas componentes del modelo RGB.

Para determinar el poder discriminante se utilizó el criterio de lambda de Wilks(Λ) **[11]**, el cual toma valores entre 0 y 1, cuanto más cerca de 0 esté, mayor es el poder discriminante de las variables seleccionadas.

$$\Lambda_{Wilks} = \frac{|I|}{|I+E|} \qquad (2)$$

Donde, I es la matriz de la suma de los cuadrados de la variabilidad intra-grupos y E es la matriz de la suma de los cuadrados de la variabilidad entre-grupos

Tabla 3: Criterio de Lambda de Wilks.

Características	Poder Discriminante (Λ)
R	0.00783866081631
G	0.00517518580809
B	0.09405113322486
R, G	0.00000609099898
R, B	0.00008474553955
G, B	0.00002583203295
R, G, B	0.00000035649235

La Tabla 3 muestra el poder discriminante de las componentes *RGB* y sus posibles combinaciones; observando que la utilización de las tres componentes ofrece la mayor información no redundante.

2.4. Clasificación

Los clasificadores con RNAs ofrecen ventajas, comparados con otras herramientas de clasificación

tales como adaptabilidad, procesamiento paralelo masivo y tolerancia a los errores. Las RNAs han sido implementadas con éxito en actividades de inspección de calidad y clasificación de diferentes productos agrícolas, debido a que las características que definen a estos productos no siguen una función matemática determinada.

Las RNAs presentan varias topologías, dependiendo de la aplicación y la plataforma de implementación, entre otros. Para aplicaciones de clasificación y reconocimiento de patrones, las topologías más utilizadas son el Perceptrón Multi-Capa *(MLP)* y Redes de Funciones de Base Radial *(RBFN)*. Para nuestra aplicación utilizamos una red MLP en la simulación bajo software, teniendo en cuenta que su tiempo de entrenamiento es mucho menor que una red RBFN.

Por otra parte, las RNAs clásicas comprenden algunas operaciones que se convierten en un problema a la hora de la implementación en hardware, como el almacenamiento de pesos reales debido al punto flotante, multiplicación sináptica y la función de actividad no-lineal, las cuales requieren una capacidad circuital muy grande **[2]**. Entre las diferentes topologías que buscan mejorar estos problemas se encuentran las RNAs, que son rápidas sin multiplicadores **[6]**, basadas en la computación estocástica **[5]**, en la modulación de frecuencia **[4]** y en bloques **[3]**.

Las redes rápidas sin multiplicadores, aunque solucionan el problema de la multiplicación sináptica, presenta pesos sinápticos con valores flotantes (64 bits) con función de activación no-lineal y continua de implementación robusta. Las redes basadas en la computación estocástica trabajan con pesos sinápticos con 128 bits, para lograr una distribución de probabilidad apropiada, la función de activación es no-lineal y como clasificadores no presenta líneas de decisión sino áreas de decisión. Las redes basadas en la modulación de frecuencia trabajan con funciones de integración lógicas, los pesos sinápticos con 128 bits, la función de activación es no-lineal tipo escalón y necesitan una señal de reloj para hacer evaluar la trama de pesos sinápticos. Las redes neuronales basadas en bloques son estructuras modulares con pesos sinápticos con valores enteros (8-16 bits) y funciones de activación lineales.

Dadas las condiciones y las restricciones presentes en las diferentes topologías estudiadas, se optó por la implementación de las redes neuronales basadas en bloques (BBNN). Esta topología presenta operaciones entre números enteros con sólo 8 bits, evitando la implementación y utilización de un procesador de punto flotante, el cual usaría gran parte del dispositivo.

3. Red Neuronal Basada en Bloques

3.1. Arquitectura

La arquitectura BBNN consiste en un arreglo bidimensional (2D), de bloques fundamentales con cuatro variables de entrada y/o salida y conectadas con pesos. Cada bloque puede tener una de las cuatro diferentes configuraciones internas escogidas, dependiendo de la estructura (Fig. 2). Los bloques intermedios son interconectados directamente con sus bloques vecinos. El modelo presenta pesos de tipo entero para fácil implementación en hardware reconfigurable como arreglos lógicos programables. Cada nodo es caracterizado por una función de activación *g()*. El nodo de salida usa una función de activación lineal, simétrica y saturada para implementación en hardware.

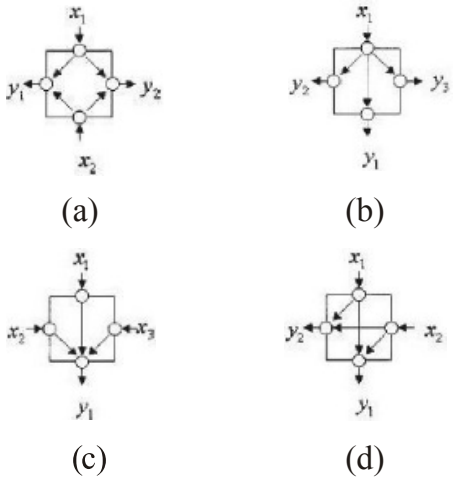


Figura 2: Cuatro tipos, diferentes configuraciones internas de bloques básicos: (a) 1/3, (b) 3/1, (c) 2/2, (d) 2/2

La estructura BBNN tiene características modulares. El tamaño puede ser expandido adicionando más bloques básicos. La estructura de la red simplemente

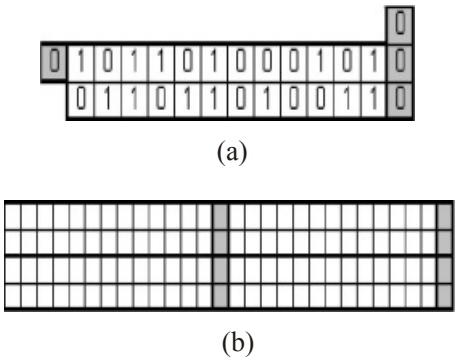
corresponde a la determinación de las señales de flujo. Los pesos de conexión son entrenados por el proceso de optimización basado en algoritmos genéticos.

3.2. Optimización de la Estructura y los Pesos

La optimización del BBNN está compuesta de dos partes: optimización de estructura y optimización de parámetros. La optimización de la estructura corresponde a la determinación de la configuración interna de cada bloque. La optimización de parámetros o proceso de aprendizaje determina el valor de los pesos entre las conexiones entrada-salida. Los algoritmos genéticos encuentran una solución más óptima en busca de un espacio discreto, consistente de cromosomas de acuerdo con la función de aptitud "*fitness*".

3.3. Codificación de la Estructura y los Pesos

La estructura del BBNN determina las señales de flujo entre los bloques, y ésta a su vez determina la configuración interna o el modo entrada-salida de cada bloque. La estructura y los pesos de conexión se optimizan simultáneamente y deben ser codificados en un cromosoma al mismo tiempo. Por esta razón es necesario utilizar una codificación multidimensional, donde cada gen es asignado a una dimensión. En este modelo (2D), la estructura y los pesos de la red de cada bloque son asignados a un cromosoma bidimensional, lo cual mejora la ejecución del operador de cruce en el algoritmo genético.



(a)

(b)

Figura 3: Codificación Bidimensional: (a) Cromosoma de un bloque básico, (b) Codificación de un BBNN 2 × 2

Por ejemplo en la Fig. 3 todos los pesos son codificados por números binarios de cuatro bits. La Fig. 3(a) representa un cromosoma de un bloque básico de una estructura BBNN. La Fig. 3(b) muestra el resultado bidimensional de una estructura de *2 × 2*.

3.3.1. Operadores genéticos

Los algoritmos genéticos son un entrenamiento supervisado a partir de la salida deseada para una entrada determinada, de esta forma se compara con la salida resultante para determinar un error.

Para la optimización con el algoritmo genético, el BBNN requiere 4 parámetros principales. La Tabla 4 muestra los parámetros y los valores recomendados **[3], [12]**

Tabla 4: Parámetros escogidos para la optimización BBNN

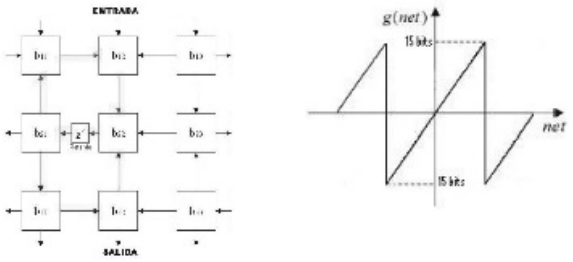
Parámetro	Valor
Probabilidad de cruce (p_c)	0.5
Prob. de mutación de pesos (p_{mw})	0.001
Prob. de mutación de pesos (p_{ms})	0.005
Población	100

4. Implementación

Teniendo en cuenta que el entrenamiento de dicha estructura se realiza a partir de algoritmos genéticos, se desarrolló la topología BBNN bajo C++ antes de realizar una implementación sobre la FPGA, la implementación y sintetización sobre la FPGA se realiza mediante un lenguaje de descripción de hardware como lo es el VHDL. De esta forma se diseñó una estructura lo más cercana posible al desarrollo bajo VHDL, utilizando funciones de desborde y de acarreo presentes en plataformas digitales.

Al simular la estructura sobre el proceso de entrenamiento, surgían redes cuyos flujos formaban trayectorias cerradas, las cuales presentaba características de sistemas inestables. No obstante, la utilización de un retardo en un flujo de la trayectoria reduce la inestabilidad por el truncamiento de los lazos algebraicos **[13]**. Sobre VHDL, el retardo se realiza con un registro, siendo necesaria la utilización de una señal de reloj. En la Fig. 4(a) se observa una

red inestable con dos trayectorias cerradas con un flujo en común, sobre el cual se debe colocar el retardo, solucionando así el problema de inestabilidad.



(a) (b)

Figura 4: (a) Estructura del BBNN, (b) Función Activación tipo Módulo

Adicionalmente, se implementó una función de activación con la operación módulo de 16 bits, y la pendiente de la sección lineal se reguló con una función de corrimiento (Fig. 4(b)). El número de bits del corrimiento se estableció con varias pruebas en el entrenamiento, dando como resultado una dependencia muy directa con la mitad del número de bits utilizados en la codificación de los pesos, es decir, si los pesos son de 8 bits, los bits de corrimiento son 4, dando una pendiente $1/2^4=1/16$.

Teniendo en cuenta el tipo de función de activación, se puede simplificar la sintetización de la estructura en la FPGA, si el número de bits utilizados en la función de corrimiento es par. Se procede intercambiando el orden de la operación módulo con la operación de corrimiento, de esta forma se aplica la operación de corrimiento a la entrada del bloque, utilizando así la mitad de los bits requeridos. A la salida se usa la operación de módulo. Además, este tipo de función de activación, mejora considerablemente los problemas de estabilidad que presentan las trayectorias cerradas. Sin embargo, se sugiere el uso de un registro en la salida de cada red, para evitar los estados transitorios de la estructura (Fig 5).

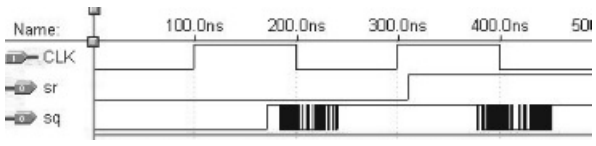


Figura 5: Señal: Reloj (CLK), Salida con Registro (sr) y Salida sin Registro (sq)

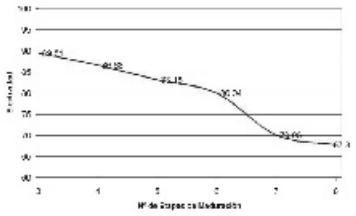
Finalmente, para la implementación sobre VHDL, los valores iniciales de los bloques laterales se hacen igual a cero. De esta forma los pesos entre esas entradas y sus respectivas salidas son redundantes y pueden ser ahorrados a la hora de realizar la sintetización.

5. Resultados

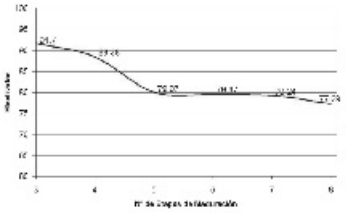
Las topologías MLP y BBNN fueron implementadas como clasificadores en varias etapas de maduración, agrupando las muestras según la semana (Tabla 1). Se tomó el *60%* de las muestras para el entrenamiento y el *40%* para la validación.

La evaluación de resultados se realizó a partir de la matriz de confusión **[14]**, la cual es una herramienta muy utilizada para la presentación y el análisis del resultado de una clasificación. Es una matriz cuadrada de orden n igual al número de clases; en las filas se representan las clases reales mientras que en las columnas se representan las clases asignadas por el clasificador. A partir de la matriz se pueden determinar las métricas de desempeño (MDs)**[15]**, efectividad, especificidad y precisión de cada clasificador.

En la Fig. 6 se muestra la disminución de la efectividad a medida que el número etapas de clasificación aumenta, según la Tabla 1. Para el clasificador con la topología BBNN la efectividad es mayor, con respeto a la topología MLP, cuando se diseña para 5 y 6 etapas de maduración. En los otros casos, el clasificador implementado con la topología MLP presenta mejores índices de clasificación que el clasificador implementado con la topología BBNN. Al aumentar hasta 7 y 8 etapas de maduración en la topología BBNN, la efectividad disminuye considerablemente (67.8% aprox.), debido a la poca selectividad que presentan los pesos enteros entre clases adyacentes.



(a)



(b)

Figura 6: Efectividad vs No de etapas de maduración: (a) BBNN, (b) MLP

Los resultados presentados sobre C++ de la estructura BBNN son iguales que los resultados obtenidos en la simulación sobre VHDL según las métricas de desempeño. Sin embargo, los tiempos de ejecución (Tabla 5) son diferentes.

Tabla 5: Tiempos de ejecución

Clasificador	Tiempo (μs)	Plataforma
MLP	156.25	AMD Athlon 1.67 GHz
BBNN	932.75	AMD Athlon 1.67 GHz
BBNN	0.57413	FPGA 25.175MHz

Los tiempos presentados en la Tabla 5 son tiempos promedio entre las diferentes implementaciones realizadas para cada topología. Los tiempos de ejecución en C++ de la topología MLP son menores a los tiempos de ejecución de la topología BBNN (Tabla 5), esto se debe a los lazos de realimentación presentados en la topología BBNN. El tiempo de 0.57413 muestra que topología BBNN toma aproximadamente 11 ciclos de reloj para llegar a un estado estable.

Otro punto a tener en cuenta es la cantidad de bits utilizados por cada clasificador. La cantidad de bits la determina el número de pesos (Tabla 6) necesarios por cada red y el tamaño de cada peso. Para las redes implementadas en C++ (MLP) cada peso es de tipo flotante de 64 bits mientras para las redes implementadas en VHDL (BBNN) cada peso es de tipo byte (8 bits).

Tabla 6: Número de pesos y bits utilizados en el clasificador

Clasificador	N° de Pesos	N° de Bits
MLP	608	38912
BBNN	114	912

6. Conclusiones y Recomendaciones

La discriminación entre clases fue realizada a partir de la información ofrecida por cada componente de brillo del espacio RGB. La característica y el plano de características más discriminante fue la componente Verde (G) y Rojo-Verde (R, G). La efectividad de la topología BBNN disminuye considerable con respecto a la topología MLP al aumentar el número de clases a clasificar, esto se debe por la representación de los pesos sinápticos en cada una de las topologías presentadas. El traslape de las muestras con las características seleccionadas es la principal causa del porcentaje de efectividad aceptable (90% aprox.) en los dos clasificadores implementados para 3 etapas de maduración. Las etapas de maduración que presentan mayor traslape son las etapas intermedias (4 y 5). Las mayores efectividades se presentaron con los clasificadores con menores etapas de clasificación para la topología MLP con una efectividad del promedio del 91.7% y con la topología BBNN con una efectividad promedio 89.5%. El tiempo promedio de ejecución de la topología implementada sobre la FPGA es aceptable, aclarando que el clasificador de 8 clases requiere más ciclos de reloj que el clasificador de sólo 3, dado que el primero presenta mayor número de lazos de realimentación. Se propone la implementación de un clasificador utilizando la topología de redes basada en la modulación de frecuencia y un clasificador a partir de máquinas de vector soporte (SVM) sintetizadas de la misma forma sobre la FPGA.

7. Agradecimientos

Se agradece a la Universidad Nacional de Colombia Sede Manizales y al Grupo de Investigación de Percepción y Control Inteligente.

8. Referencias bibliográficas

[1] Arcila, J. Fisiología del cafeto: crecimiento, desarrollo, floración y producción. Cenicafé, Chinchiná (Colombia), 2000.

[2] White, B.A. And Elmasry, M.I. The dig-neocognitron: A digital neocognitron neural model for vlsi. IEEE Transactions On Neural Networks, 3(73-85), Jan 1992.

[3] Sang-Woo Mon and Seong-Gon. Block-base neural networks. IEEE Transactions On Neural Networks, 12(2):307316, March 2001.

[4] Hikawa, H. Frecuency-based multilayer neural network on-chip learning and enhanced neuron characteristics. IEEE Transactions On Neural Networks, 10(3):545553, May 1999.

[5] Bade, S. and Hutchings, B. L. Fpga-based stochastic neural networks: Implementation. Proceedings of IEEE Workshop on FPGAs for Custom Computing Machines, pag 189198, April 1994.

[6] Piazza, F. Marchesi, M. Orlandi, G Uncini, A. Fast neural networks without multipliers. IEEE Transactions On Neural Networks, 4(1):5362, Jan 1993.

[7] Handlen, J. O. Furman, M. D. Rapid Prototyping of Digital Systems Tutorial Approach. 2003.

[8] Sandoval, Z. y Prieto, F. Caracterización y clasificación de café cereza por medio de visión artificial. 2003.

[9] Castaño, L. y Prieto, F. Sistema de visión artificial para clasificación de granos de café basado en fpga. Encuentro De Investigación Sobre Tecnologías De Información Aplicadas A La Solucion De Problemas, pag 137140, 2003.

[10] Montes-Castrillón, N. L. Segmentación de Imágenes de Frutos de Café en el Proceso de Beneficio. Universidad Nacional de Colombia, 2003.

[11] Figueras, S. Análisis discriminante [en línea].Http://www.5campus.com/lección /Discri. 2000.

[12] Gutiérrez, J. M. Introducción a la inteligencia artificial [en línea]. http://personales.unican.es/gutierjm.

[13] Shankar Krishnan and Dinesh Manocha. Algebraic Loop Detection and Evaluation Algorithms. Canadian Human-Computer Communications Society, 1996.

[14] Cortijo-Bon, F. J. Estimación del error de clasificación y verificación de resultados [en línea] http://www-etsi2.ugr.es/depar/ccia/rf/www /Tema1. Oct 2001.

[15] Ferri, C., Hernández-Orallo J., Salido, M.A. Volume under the roc surface for multi-class problems. Exact computation and evaluation of approximations. April 2003.