Entrenamiento de una Red Neuronal Celular para procesamiento de imágenes utilizando el Algoritmo de Colonia de Abejas Artificial

José de Jesús Morales Romero
Department of Electrical Engineering
CINVESTAV-IPN
Mexico City, Mexico
jmoralesr@cinvestav.mx

***Abstract*—En este trabajo se presenta el Algoritmo de Colonia de Abejas para el entrenamiento de una Red Neuronal Celular. La Red Neuronal Celular se empleará como un procesador de imágenes. Las diferentes tareas que realizará la Red Neuronal Celular como procesador de imágenes son el extractor de bordes y removedor de ruido.**

***Index Terms*—Red Neuronal Celular, Algoritmo de Colonia de Abejas, procesamiento de imágenes, detector de bordes, removedor de ruido.**

I. Introducción

Leon O. Chua y L. Yang en el año de 1988 propusieron una nueva arquitectura de red neuronal llamada Red Neuronal Celular (CNN o CeNN por sus siglas en inglés de Cellular Neural Network) [1], en este trabajo utilizaremos la nomenclatura de CeNN para no confundir con las redes neuronales convolucionales (que también llevan como nomenclatura CNN). Desde entonces, se han desarrollado diversos estudios relacionados con este tipo de red neuronal. Como ejemplo de estos estudios, entre otros, tenemos los relacionados con la búsqueda y optimización de plantillas [2]–[4].

Las CeNNs se han utilizado para realizar diferentes tareas como son la resolución de ecuaciones diferencias, osciladores caóticos y como procesador de imágenes, entre otros. Dentro de estas tareas, el procesamiento de imágenes es una tarea atractiva [5].

Las CeNNs pueden realizar diferentes tipos de procesamiento de imágenes, como lo son la detección de bordes, removedor de ruido, detector de conectividad global, detector de sombras, detector de movimiento, operaciones booleanas sobre imágenes, etc [5].

Algunos ejemplos de utilización de la CeNN como procesador de imágenes se mencionan a continuación: en [6] encontramos una CeNN que procesa imágenes ruidosas; en [7] se realizó la segmentación de imágenes.

No solo las CeNN han sido implementadas en software sino también en hardware, esto los hace aún más atractivos. Por ejemplo, en [8] se ha implementado una CeNN en FPGA, esta red implementada en hardware es capaz de realizar diversas tareas sobre imágenes; en [9] se ha emulado una CeNN para el procesamiento de videos en tiempo real.

Para que una CeNN pueda realizar las diferentes tareas de procesamiento de imágenes, se necesita entrenar la red neuronal con algún método de optimización.

Los diferentes métodos de optimización se pueden dividir en dos clases principales: los métodos analíticos y los métodos heurísticos. Como ejemplo, de los métodos analíticos podemos encontrar el trabajo [3] y de los métodos heurísticos podemos encontrar el trabajo [4].

De los diferentes métodos de optimización heurísticos utilizados para el entrenamiento de CeNNs podemos encontrar PSO (del inglés Particle Swarms Optimization) [10], GA (de Genetic Algorithms) [2], ABC (de Artificial Bee Colony) [4] y NM (de Nelder-Mead) [11].

El algoritmo ABC, propuesto por Karaboga en el año 2005, se ha utilizado como método de optimización para diferentes problemas, lo cual por su facilidad de implementación se ha propuesto como una buena alternativa para la búsqueda de plantillas de las CeNNs [12].

Sin embargo, el algoritmo ABC se ha utilizado solo para una tarea de procesamiento de imágenes. Por lo cual, en este trabajo se presenta el entrenamiento de una CeNN para diferentes procesamientos de imágenes.

El resto del trabajo se divide de la siguiente forma: en la sección II se trata la teoría sobre las Redes Neuronales Celulares; en la sección III se da la teoría sobre el método de optimización de Algoritmo de Colonia de Abejas Artificial; después en la sección IV se hace la búsqueda de las plantillas de una CeNN para el procesamiento de imágenes utilizando el algoritmo ABC; en la sección V se muestran los resultados obtenidos durante el entrenamiento de la red neuronal; en la sección VI se habla de las conclusiones ha las que se ha llegado con este trabajo; finalmente en la sección VII se habla sobre los posibles trabajos a futuro que se pueden realizar a partir de este trabajo.

II. Teoría de la Red Neuronal Celular

Una CeNN está construida por una unidad básica llamada *célula*. La célula es replicada e interconectada en forma de arreglos que van desde una dimensión hasta n dimensiones.

Para el procesamiento de imágenes la CeNN es construida en dos dimensiones para el procesamiento de imágenes en escala de grises, y en tres dimensiones para el procesamiento de imágenes en escala de colores.

En este trabajo trataremos imágenes en escala de grises por lo cual utilizaremos una CeNN en arreglo bidimensional. En la Fig. 1, se muestra la estructura de la CeNN en dos dimensiones.

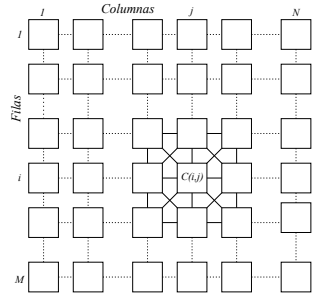


Fig. 1. Estructura de una CeNN de dos dimensiones para procesamiento de imágenes en escala de grises.

Cuando una CeNN procesa imágenes se requiere que la CeNN sea del mismo tamaño que la imagen a procesar. Esto es debido a que cada célula actúa sobre un píxel de la imagen. Por ejemplo, si la imagen a procesar es de tamaño 100×100 píxeles, la CeNN requerirá que sea del tamaño de 100×100 células.

En la Fig. 1, al conjunto de células que están conectadas directamente a la célula $C(i,j)$ se le llama *vecindario* $N_r(i,j)$, donde r es el radio del vecindario; así por ejemplo, si el $r = 1$ se tendrá un vecindario de 3×3 células. Este concepto de vecindario es importante comprenderlo, ya que este indicará el tamaño de las plantillas.

El comportamiento de cada célula esta definida por la ecuación (1), la cual es llamada ecuación de estado. Debido a que la ecuación original de estado es una ecuación diferencial, esta ha sido aproximada por el método de Euler para ser implementada de mejor forma en software y en hardware.

$$v_{xij}(n+1) = v_{xij}(n) + h[-v_{xij}(n) + f_{ij}(n) + I_{ij}] \quad (1)$$

Donde:

$$f_{ij}(n) = \sum_{C(k,l)\in N_r(i,j)} A(i,j;k,l)v_{ykl}(n)$$

y

$$I_{ij} = \sum_{C(k,l)\in N_r(i,j)} B(i,j;k,l)v_{ukl} + I$$

De la ecuación de estado (1), n es el paso de integración del método de Euler, h es la constante del paso de integración. Los valores v_{xij} y v_{ykl} y v_{ukl} es el estado, salida y entrada de la célula $C(i,j)$. Las matrices $A(i,j;k,l)$ y $B(i,j;k,l)$ son las plantillas de retroalimentación y control respectivamente. Finalmente, el valor I es el bias de la célula $C(i,j)$.

Al igual que el resto de las redes neuronales, las CeNN tienen a su salida una función de activación. Para el caso aquí presentado, donde se utiliza la CeNN como procesasor de imágenes, se utiliza la función *satlins*. En la ecuación (2) se define la *función de activación satlins*.

$$v_{yij}(n) = 0.5(|v_{xij}(n)+1| - |v_{xij}(n)-1|) \quad (2)$$

Las plantillas de retroalimentación y control junto con el valor del bias son los encargados de indicar la tarea que realizará la CeNN. Considerando una CeNN con un radio de $r = 1$ para procesar imágenes en escala de grises, se definen las plantillas de retroalimentación y control como en (3). Además se consideró las restricciones de estabilidad como se menciona en [1].

$$\mathbf{A} = \begin{bmatrix} a_0 & a_1 & a_2 \\ a_3 & a_4 & a_3 \\ a_2 & a_1 & a_0 \end{bmatrix}; \mathbf{B} = \begin{bmatrix} b_0 & b_1 & b_2 \\ b_3 & b_4 & b_3 \\ b_2 & b_1 & b_0 \end{bmatrix} \quad (3)$$

Como se vera más adelante, esta nomenclatura de las plantillas es conveniente para su implementación en software.

III. Teoría sobre el Algoritmo de Colonia de Abejas Artificial

El algoritmo **ABC** es considerado dentro de los llamados algoritmos bio-inspirados, debido a que se basa en el comportamiento de búsqueda de alimento de las abejas. Desde el punto de vista matemático, las fuentes de alimento son considaras como posibles soluciones al problema planteado.

Dentro del algoritmo ABC se definen tres grupos de abejas: el primero de ellos es el llamado grupo de abejas *abejas trabajadoras*, el segundo grupo es el llamado grupo de *abejas observadoras* y finalmente el tercer grupo es llamado grupo de *abejas exploradoras*.

El comportamiento de cada uno de estos grupos es descrito a continuación:

- Las ***abejas trabajadoras*** (EBs del inglés Employed Bees) componen la primera mitad de la colonia de abejas. Este grupo es enviado a las diferentes fuentes de alimento, además de que calculan la cantidad de nectar (o calidad de la fuente de alimento).
- Las ***abejas observadoras*** (OBs del inglés Onlooker Bees) componen la segunda mitad de la colonia de abejas. Este grupo es enviado a sus respectivas fuentes de alimento e igualmente que las EBs calculan la cantidad de nectar de la fuente de alimento.
- Las ***abejas exploradoras*** (SBs del inglés Scout Bees) representan las fuentes de alimento y son enviadas en búsqueda de nuevas fuentes de alimento basadas en un parámetro llamado *Limite* de SB. Aquí las fuentes de alimento representan las posibles soluciones al problema de optimización.

El algoritmo general de **ABC** [12] es mostrada en el Algoritmo 1.

Como fue mencionado anteriormente, el tamaño de la colonia de abejas es dividida en dos partes, las primera representa

Algorithm 1 ABC Algorithm

1: Se asigna los parámetros de control.
2: Se inicializa los valores de EBs, OBs, SBs y el *Limite*.
3: Se calcula el valor inicial del nectar de los EBs y memoriza el mejor valor encontrado.
4: **while** el criterio de parada es alcanzada **do**
5: Se envía a los eBs dentro de las fuentes de alimento y calcula la cantidad de nectar.
6: Se envía los OBs alrededor de las fuentes de alimento de EBs y calcula su la cantidad de nectar. Aquí es actualizado el mejor resultado utilizando una función del mejor.
7: Se envían las SBs a nuevas fuentes de alimentointo en forma aleatoria. Esto se realiza solo cuando el valor de *Limite* es alcanzado.
8: Se memoriza la mejor fuente de alimento encontrado hasta el momento.
9: **end while**

a las EBs y la segunda a las OBs. Desde el punto mátematico el conjunto de fuentes de alimento es producida aleatoriamente utilizando (4).

$$x_{i,j} = x_j^{min} + \phi(x_j^{max} - x_j^{min}) \tag{4}$$

De (4), $i = 1, \cdots, SB$; $j = 1, \cdots, D$; D representa la dimensión del problema de optimización; ϕ es un número escogido aleatoriamente con una distribución uniforme en el intervalo $[0, 1]$; finalmente x_j^{min} y x_j^{max} son los límites del área de búsqueda.

La búsqueda de alimento que realizan las EBs es en el vecindario de las SBs, esto se realiza utilizando (5).

$$v_{i,j} = x_{i,j} + \varphi(x_{i,j} - x_{k,j}) \tag{5}$$

De donde $k = 1, \cdots, SB$ y $k \neq j$; además j y k son índices elegidos aleatoriamente; φ es un número escogido aleatoriamente con distribución uniforme en el intervalor $[-1, 1]$. Finalmente, el valor $v_(i, j)$ debe de encontrarse dentro del área de búsqueda.

Las OBs escogen su fuente de alimento basada en alguna distribución de probabilidad. Para este trabajo realizado se ha implementado con la distribución de probabilidad definida en (6).

$$P_i = \frac{fit(x_i)}{\sum_{i=1}^{SB} fit(x_i)} \tag{6}$$

De (6), $fit(x_i)$ es llamada la función *fitness*, el cual es proporcional a la fuente de alimento de x_i. La función fitness depente del problema de optimización. En este paso, si la función *fitness* incrementa su valor, la probabilidad de visita de una nueva fuente de alimento se incrementa igualmente. Esta función se aplica para las OBs.

Sí durante la fase de SBs, una EB alcanza su *límite*, dicha fuente de alimento es abandonada y comienza una nueva búsqueda utilizando (4).

IV. Búsqueda de plantillas de una CeNN utilizando ABC

Como fue mencionado anteriormente, el algoritmo ABC es adecuado para la búsqueda de plantillas que realicen diferentes tareas de procesamientos de imágenes.

Para el entrenamiento de imágenes utilizando la CeNN, las plantillas de retroalimentación y control mostradas en (3) son colocadas como un vector. Este vector toma como ventaja que las matrices son simétricas. Dicho vector es mostrado en (7).

$$x_i = [a_0, a_1, a_2, a_3, a_4, b_0, b_1, b_2, b_3, b_4, I] \tag{7}$$

El método para el entrenamiento de la CeNN es supervisado, es decir, se compara las imágenes entregadas por la CeNN con las imágenes deseadas. De esta forma, se va corrigiendo el valor del vector mostrado en (7).

En la Fig. 2 se muestra el diagrama a bloques del método utilizado para el entrenamiento de la red neuronal.

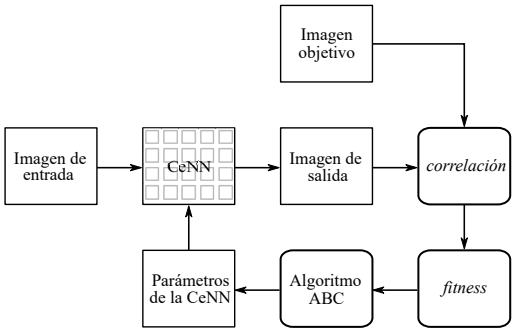


Fig. 2. Diagrama a bloques del método utilizado para el entrenamiento de una CeNN utilizando el algoritmo ABC

La función de correlación entre la imagen de salida (la imagen procesada por la CeNN) y la imagen objetivo (imagen de salida deseada) es realizada utilizando (8).

$$C = \frac{\sum_i^M \sum_j^N (xP_{i,j} - \overline{xP})(yP_{i,j} - \overline{yP})}{\sqrt{(\sum_i^M \sum_j^N (xP_{i,j} - \overline{xP})^2)(\sum_i^M \sum_j^N (yP_{i,j} - \overline{yP})^2)}} \tag{8}$$

De donde, $xP_{i,j}$ es el valor de cada píxel de la imagen procesada por la CeNN, $\overline{xP}$ es el promedio de los píxeles de la imagen procesada por la CeNN, $yP_{i,j}$ es el valor de cada píxel de la imagen objetivo, y $\overline{yP}$ es el promedio de los píxeles de la imagen objetivo.

El valor *fitness* requerido por el algoritmo ABC es calculado utilizando (9).

$$fit(C) = \begin{cases} 0.5 + |C| & C < 0 \\ \frac{1}{1+C} - 0.5 & C \geq 0 \end{cases} \tag{9}$$

De la ecuación anterior, el valor C es el valor de la correlación enre la imagen de salida y la imagen objetivo, la cual fue calculada en (8).

El funcionamiento para la búsqueda de plantillas de una CeNN que realiza el procesamiento de imágenes, basado en la Fig. 2 es el siguiente: Primero se establecen los parámetros iniciales del algoritmo ABC, es decir, primero se establecen aleatoriamente unas plantillas iniciales. Estas plantillas iniciales se ingresan como parámetros a la CeNN, la cual realizará el procesamiento de la imagen. Obtenido la imagen de salida de la CeNN, se realiza una *correlación* con la imagen objetivo.

Después de obtener el valor de correlación, se calcula el valor *fitness*. Este valor *fitness* es ingresado al algoritmo ABC. Con dicho valor, el algoritmo ABC se encarga de optimizar las plantillas anteriormente encontradas. Esto se repite en un ciclo.

La imagen de entrada y la imagen objetivo se establecen de acuerdo al procesamiento de imagen deseado. Por ejemplo, si se desea buscar contornos, la imagen objetivo contendrá los bordes de la imagen de entrada. Generalmente para el entrenamiento de la CeNN, se utilizan imágenes artificiales, es decir, imágenes no reales.

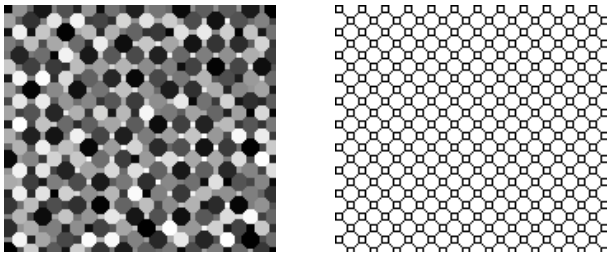
V. Resultados

El entrenamiento de la CeNN se realizó para dos tareas. El primero de ellos es como detector de bordes y el segundo como removedor de ruido.

A. CeNN como detector de bordes

Como fue mencionado anteriormente, para entrenar la CeNN se utilizó una imagen artificial como entrada y basada en dicha entrada se creo otra imagen como objetivo.

Las imágenes utilizadas para esta tarea se muestran la Fig. 3. En donde la Fig. 3a es la imagen ingresada al método de búsqueda de plantillas y en la Fig. 3b se muestra la imagen objetivo creada para la tarea de búsqueda de bordes.



(a) Imagen de entrada (b) Imagen objetivo

Fig. 3. Imágenes utilizadas para la detección de bordes

La Fig. 3 tiene un tamaño de 165×165 píxeles, y hay un paso entre grises de 2 con representación de grises entre 0 y 255. Finalmente la imagen tiene 129 tonos de grises lo que representa un 50.4% de cobertura de grises.

En la tabla I se muestran los valores obtenidos durante el entrenamiento de la CeNN utilizando el algoritmo ABC. Se establecieron cuatro diferentes tamaños de colonia, los cuales son 22, 44, 88 y 176. Además, se establecieron dos límites máximos de ciclos para el algoritmo ABC que fueron de 500 y 1000. Finalmente, el valor de *Límite* fue establecido en un valor de 10.

TABLE I
Rendimiento del algoritmo para búsqueda de contornos utilizando una CeNN

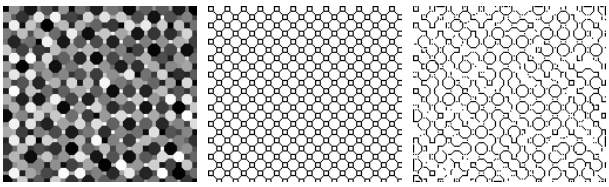
Ciclos		**Tamaño de la colonia**			
		22	44	88	176
500	*corr*	0.6368	0.6813	0.7065	**0.7859**
	fit	0.1109	0.0947	0.0859	**0.0599**
1000	*corr*	0.6763	0.6904	0.7165	0.7114
	fit	0.0965	0.0915	0.0825	0.0842

Los parámetros establecidos para la CeNN fueron de un paso de tiempo h de $0.03125s$ y un máximo tiempo de procesamiento $tmax$ de $1s$.

Las mejores plantillas encontradas fueron con un tamaño de colonia de 176 utilizando 500 ciclos. El valor de las plantillas se muestra en (10).

$$\begin{aligned} x = &[-10.396, 1.2147, -10.4482, -13.4523, \\ &62.4621, 4.6734, -20.8402, -7.4367, \\ &-12.8630, 61.9943, -35.1705] \end{aligned} \tag{10}$$

En la Fig. 4 se muestra el resultado obtenido de la detección de bordes utilizando las plantillas obtenidas utilizando el algoritmo ABC.



(a) Entrada (b) Objetivo (c) Objetivo

Fig. 4. Imágenes utilizadas para la detección de bordes y la imagen obtenida

Los datos de las imagen de entrada y la imagen objetivo son las mismas que las mencionadas anteriormente.

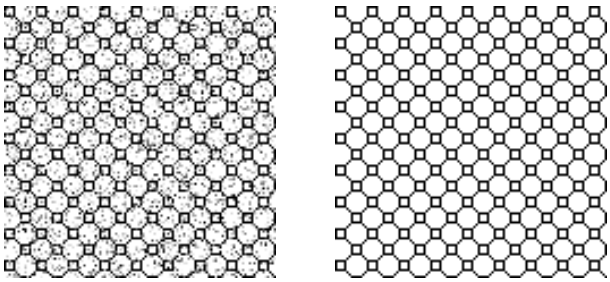
B. CeNN como removedor de ruido

Al igual que en la búsqueda de bordes, para entrenar una CeNN se utilizó una imagen artificial como entrada y finalmente se creo la imagen objetivo utilizada durante el entrenamiento.

Las imágenes utilizadas se muestran en la Fig. En donde la Fig. es la imagen ingresada al método de búsqueda de plantillas y en la Fig. se muestra la imagen objetivo para la tarea de removedor de ruido.

La Fig. 5 tiene un tamaño de 120×120. En donde la Fig. 5a es la imagen artificial utilizada como entrada y la Fig. 5b es la imagen utilizada como objetivo. Además, el ruido contiene un total de 65 tonos de grises y una cobertura del 3%.

En la tabla II se muestran los valores obtenidos durante el entrenamiento de la CeNN utilizando el algoritmo ABC



(a) Imagen de entrada (b) Imagen objetivo

Fig. 5. Imánes utilizadas para remover ruido

TABLE II
RENDIMIENTO DEL ALGORITMO PARA REMOVEDOR DE RUIDO UTILIZANDO UNA CENN

Ciclos		**Tamaño de la colonia**			
		22	44	88	176
200	*corr*	0.8495	0.8669	0.8715	0.8763
	fit	0.0406	0.0356	0.0343	0.0329
500	*corr*	0.8707	0.8805	0.8923	**0.8926**
	fit	0.0345	0.0317	0.0284	**0.0283**

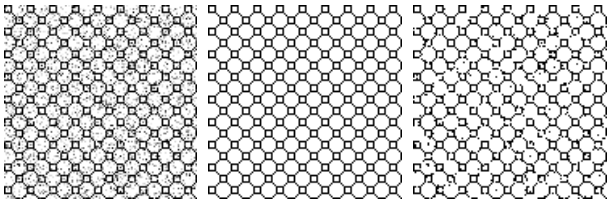
para removedor de ruido. Al igual que en la tarea anterior, se establecieron cuatro diferentes tamaños de colonia, los cuales son de 22, 44, 88, 176. Además, se establecieron dos límites máximos de ciclos para el algoritmo ABC que fueron de 200 y 500 ciclos. Finalmentes, el valor de *Límite* se estableció de 10.

Los parámetros para la CeNN fueron los siguientes: $h = 0.03125s$ y $tmax = 1s$.

Las mejores plantillas entonctradas para la CeNN como removedor de ruido fueron con los parámetros de 500 y un tamaño de colonia de 176. El valor de dichas plantillas se muestra en (11).

$$\begin{aligned} x = &[-9.1103, 3.7785, -14.1810, -12.0083, \\ &64.0000, 25.4679, 15.3484, 22.2699, \\ &28.4184, 59.4597, -7.5525] \end{aligned} \tag{11}$$

En la Fig. 6 se muestra el resultado obtenido para el removedor de ruido utilizando las plantillas obtenidas utilizando el algoritmo ABC.



(a) Entrada (b) Objetivo (c) Objetivo

Fig. 6. Imágenes utilizadas para la detección de bordes y la imagen obtenida

Los datos de las imagen de entrada y la imagen objetivo son las mismas que las mencionadas anteriormente.

VI. CONCLUSIONES

Como se ha observado durante el desarrollo de este trabajo, el algoritmo heurístico ABC es una buena alternativa para la búsqueda de plantillas para una CeNN. Los resultados entregados, como se puede observar el la sección V, son de buena calidad.

Aunque en este trabajo se realizaron únicamente dos tareas para el procesamiento de imágenes, se pueden realizar una mayor cantidad de tareas, como lo es extractor de bordes, detector de conectividad global, operaciones booleanas sobre imágenes, etc.

En este trabajo se utilizó la CeNN para procesar imágenes en escala de grises, sin embargo, es posible utilizar la CeNN para procesar imágenes en color. Además, es posible utilizar este tipo de red neuronal para procesar video. Para poder procesar video se requiere de realizar los cambios necesarios en la arquitectura de la red, como lo es el cambio en las plantillas.

Debido a que el algoritmo ABC es un método heurístico, se debe de elegir de forma experimental. Como se observa en las tablas de resultados, no en todos los casos se obtiene el mejor resultado utilizando, por ejemplo, el mayor tamaño de colonia.

Tomando en cuenta los resultados entregados por el algoritmo ABC, se puede llegar a utilizar este para el entrenamiento de otras arquitecturas de redes neuronales, como por ejemplo las redes pulsadas e inclusive las redes convolucionales.

VII. TRABAJO FUTURO

A pesar de que el algoritmo ABC entrega buenos resultados, sería buena idea comparar los resultados con otros métodos heurísticos o analíticos. Algunos de estos métodos para comparar podrían ser PSO , algoritmos geneticos y algoritmo de colonia de hormigas.

No solo revisar la calidad de los resultados entregados por la CeNN sino también cuál de los métodos anteriormente señalados entrenaría la red neuronal en menor tiempo, además de hacer la comparación con respecto a los recursos utilizados para este fin.

Como fue mencionado anteriormente, la CeNN es capaz de realizar diversas tareas. Es recomendable utilizarla para realizar otras tareas como el procesamiento de imágenes en color, inclusive realizar el procesamiento de video. Esto por su puesto entrenando la red con el algoritmo ABC.

REFERENCES

[1] L. C. L. Yang and L. Chua, "Cellular neural networks: Theory and applications," *IEEE Transactions on Circuits and Systems*, vol. 35, pp. 1257–1290, 1988.

[2] T. Kozek, T. Roska, and L. O. Chua, "Genetic algorithm for cnn template learning," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 6, pp. 392–402, 1993.

[3] M. Hanggi and G. S. Moschytz, "An exact and direct analytical method for the design of optimally robust cnn templates," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 46, no. 2, pp. 304–311, 1999.
[4] S. Parmaksızoğlu and M. Alçı, "A novel cloning template designing method by using an artificial bee colony algorithm for edge detection of cnn based imaging sensors," *Sensors*, vol. 11, no. 5, pp. 5337–5359, 2011.
[5] L. Chua and T. Roska, *Cellular Neural Networks and Visual Computing: Foundations and Applications*. Cambridge University Press, 2004.
[6] V. Murugesh, V. Arthy, and V. Agalya, "Edge detection of noisy images based on time-multiplexing cnn simulator," in *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, 2014, pp. 1–5.
[7] M. Duraisamy and F. M. M. Jane, "cellular neural network based medical image segmentation using artificial bee colony algorithm," in *2014 International Conference on Green Computing Communication and Electrical Engineering (ICGCCEE)*, 2014, pp. 1–6.
[8] J. J. Morales-Romero, M. A. Reyes-Barranca, and L. M. Flores-Nava, "Improved algorithm for time-multiplexing with digital cnn's applied in image processing, synthesized in a fpga," in *2019 16th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE)*, 2019, pp. 1–6.
[9] K. Kayaer and V. Tavsanoglu, "A new approach to emulate cnn on fpgas for real time video processing," in *2008 11th International Workshop on Cellular Neural Networks and Their Applications*, 2008, pp. 23–28.
[10] A. Giaquinto and G. Fornarelli, "Pso-based cloning template design for cnn associative memories," *IEEE transactions on neural networks*, vol. 20, no. 11, pp. 1837–1841, 2009.
[11] K. Nakai and A. Ushida, "Design technique of cellular neural network," *Electronics and Communications in Japan (Part III: Fundamental Electronic Science)*, vol. 78, no. 3, pp. 97–107, 1995.
[12] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer ..., Tech. Rep., 2005.