6. APÉNDICES

6.1. CÓDIGO MATLAB

En este apartado vamos a incluir las subrutinas de código en Matlab que son más significativas o que han conllevado un esfuerzo mayor en su implementación, realizando un papel primordial en el desarrollo del proyecto.

6.1.1. FUNCIÓN APRENDIZAJE

Esta subrutina implementa la actividad de aprendizaje en la neurona detectora de patrones de coincidencia. La función va acumulando en cada neurona los pesos de las conexiones y compara esta acumulación de pesos con el umbral de disparo de la neurona. Tras el disparo de la neurona, se actualizan los pesos sirviéndose de STDP y el acumulador neuronal se pone de nuevo a cero para reiniciar la cuenta.

La función recibe como entrada la variable bidimensional *matriz* que contiene todos los resultados obtenidos con los filtros de Gabor, codificados en el tiempo y simplificados a aquellas componentes con mayor valor. La otra variable *n_digitos* introduce el número de dígitos de entrada que estimulan a la neurona.

Como salida saca por pantalla varias imágenes:

-La primera imagen, representa los pesos acumulados finales para cada sinapsis (este valor muestra el aprendizaje de la neurona) según las 6 orientaciones de Gabor existentes.

-La segunda, tercera y cuarta imagen representan los pixeles activos comparándolos en el tiempo con la evolución del potencial acumulado por la neurona de salida. Unos ejemplos de estas ilustraciones aparecen en las Fig. 22 y 23. A su vez se representan los píxeles activos a la salida de los filtros de Gabor que codifican en el dominio del tiempo y que influirán directamente en el acumulador de la neurona de salida.

-La quinta ilustración, muestra todos los píxeles activos que codifican pulsos enviados a partir de las sinapsis que provocan el disparo de nuestra célula nerviosa entrenada.

-La sexta imagen, representa como va disminuyendo el tiempo entre disparos de la neurona detectora de coincidencia conforme el entrenamiento va siendo asimilado.

Podemos destacar en la subrutina la ayuda de dos funciones principales: *acumula_peso* y *actualizacion_peso*. La primera va acumulando el peso de las sinapsis que transportan pulsos a la neurona de salida (debido al proceso de estimulación neuronal) y modifica la variable *disparo* si la neurona detecta el patrón memorizado; la segunda función va actualizando los pesos de todas las sinapsis cuando la neurona detectora dispara. Sus códigos se explicarán más detenidamente en los siguientes apartados.

```
function out=aprendizaje(matriz, n_digitos);

[filas, columnas]=size(matriz);
timeslot =3e-3;                                                    %Tiempo de slot es 3 ms.
peso_inicial=0.01;
aumento_peso=peso_inicial*ones(filas,columnas/2);   %Matriz que acumula el aumento de los peso de cada sinapsis.
peso_total=0;                        %Variable que acumula el peso, se pone a cero cuando dispara la neurona de salida.
disparo=0;                                          %Variable binaria que indica si ha reconocido el patrón entrenado.
umbral=2;        %Umbral que debe superar “peso_total” para que se desborde el acumulador de la neurona de salida.
var_representacion=[];
var_representacion(1,1)=0;
var_representacion(1,2)=0;
k=2;
aferente_detector=[];            %Variable que me determina que pixel es el desencadenante del disparo de la neurona.
m=1;
tiempo_disparo=[];
tiempo=0;
indice=1;
for j=2:2:columnas
        for i=1:1:filas
                if(matriz(i,j)<inf & disparo<1)

                  [disparo,peso_total,var_representacion]=acumula_peso(aumento_peso...
…(matriz(i,j-1),j/2),peso_total,umbral,var_representacion,k);

                  var_representacion(k,2)=matriz(i,j);          %Registra el instante en el que se incrementa el peso.
                  k=k+1;
                  if(disparo>0)                                  %Se ha producido el reconocimiento de la imagen.
                    aumento_peso=actualizacion_peso(matriz,aumento_peso,i,j,(columnas/2),n_digitos,umbral);
```

```
                    aferente_detector(m,1)=matriz(i,j-1);
                    aferente_detector(m,2)=matriz(i,j);
                    m=m+1;
                  end;
                elseif (disparo>0 & i==filas)          Se ha producido disparo debido a una imagen intermedia.
                disparo=0;
                tiempo_disparo(indice)=aferente_detector(m-1,2)-tiempo;
                  if(j>2)
                    tiempo=((j/2)-1)*3e-3+2.5e-3;      %Actualiza el inicio de la ventana para el siguiente disparo.
                  else
                    tiempo=2.5e-3;                                  %Solo para el caso de la primera ventana.
                  end;
                  indice=indice+1;
                end;
          end;
end;
out=aumento_peso(:,columnas/2);
var_aux1=size(aumento_peso);
peso1=aumento_peso(:,var_aux1(2));
representa_imagen_peso(peso1,2);

%REPRESENTACION DE LAS VENTANAS
var_aux=zeros(2,2);                                 %Matriz que sirve para la representación de las ventanas.
var_aux(:,1)=umbral;
if n_digitos>75
  n_digitos_ventana=20;                             %Número de dígitos para las ventanas de representación.
  ventana=timeslot*n_digitos_ventana;

%PRIMERA VENTANA%
  var_aux1=find(var_representacion(:,2)<ventana);          %Acotación de la primera ventana de representación.
  var_aux2=find(aferente_detector(:,2)<ventana);
  var_aux(2,2)=ventana;                                    %La primera ventana tiene 20 slots de tiempo.
  primera_ventana(var_representacion(var_aux1,:),var_aux,n_digitos_ventana,...
...matriz,aferente_detector(var_aux2,:));

%SEGUNDA VENTANA ALTERNATIVA%
  mitad_digitos=5*n_digitos_ventana;
  mitad_tiempo=5*ventana;
  var_aux3=find((var_representacion(:,2)>mitad_tiempo)&(var_representacion(:,2)<(mitad_tiempo+ventana)));
  var_aux4=find((aferente_detector(:,2)>mitad_tiempo)&(aferente_detector(:,2)<(mitad_tiempo+ventana)));
  var_aux(1,2)=mitad_tiempo;
  var_aux(2,2)=mitad_tiempo+ventana;                  %La segunda_ventana también tiene 20 slots de tiempo.
  segunda_ventana(var_representacion(var_aux3,:),var_aux,mitad_tiempo,matriz(:,(2*mitad_digitos+1):...
...(2*mitad_digitos+40)) ,aferente_detector(var_aux4,:),n_digitos_ventana);

%TERCERA_VENTANA%
  inicio_ventana=timeslot*columnas/2-ventana;              %Inicio de la última ventana de representación.
```

```
   var_aux5=find(var_representacion(:,2)>inicio_ventana);
   var_aux6=find(aferente_detector(:,2)>inicio_ventana);
   var_aux(1,2)=inicio_ventana;
   var_aux(2,2)=timeslot*columnas/2;
   tercera_ventana(var_representacion(var_aux5,:),var_aux,...
... n_digitos_ventana,matriz(:,(2*n_digitos-39):(2*n_digitos)),...
...aferente_detector(var_aux6,:));
end;

%REPRESENTACION DEL PÍXEL QUE HACE QUE LA NEURONA DISPARE%
representa_aferente_detector(aferente_detector)

%DIFERENCIAS DE TIEMPO ENTRE DISPAROS DE LA NEURONA DE SALIDA%
matriz_diferencias=[];
[fil,col]=size(aferente_detector);
for i=1:1:fil
            if(i<fil)
              matriz_diferencias(i)=aferente_detector(i+1,2)-aferente_detector(i,2);
            end
end
variable=[1:fil-1];
diferencias_entre_picos(matriz_diferencias)
```

6.1.2. FUNCIÓN ACUMULA PESO

Esta subrutina es llamada en la función *aprendizaje* que realiza la operación de acumular el peso de las sinapsis que van transmitiendo los pulsos codificados en la primera etapa del sistema. Si el peso acumulado en la variable *peso_total* supera el umbral marcado en el experimento, la función modifica la variable binaria *disparo* pasando a tomar el valor "1", indicando al proceso que se ha producido desbordamiento del acumulador de la neurona detectora.

Las variables de entradas son:

-*peso_total*: Es la variable que ejerce la función de acumulador de peso a la entrada de la neurona de salida.

-*aumento_peso*: Variable que indica el peso de la sinapsis que transporta un pulso a la neurona de salida. Ayuda a actualizar la variable *peso_total*.

-*umbral*: Constante que marca el punto en el que la neurona de salida dispara (y por tanto reconoce un patrón entrenado). Si la variable *peso_total* supera el valor fijo marcado, nuestro detector reaccionará.

Las variables de salida son *disparo*, *peso_total* y *var_representacion* su misión consiste en ayudar en la actualización del sistema.

```
function [disparo,peso_total,var_representacion]=acumula_peso(aumento_peso,peso_total,umbral,...
...var_representacion,k)

peso_total=peso_total+aumento_peso;          %Actualización del peso total acumulado por la neurona de salida.
if (peso_total<umbral)
   disparo=0;                                %No se dispara, no se reconoce la imagen.
   var_representacion(k,1)=peso_total;
else
   disparo=1;                                %Se produce disparo, se reconoce la imagen.
   peso_total=0;                             %Como se ha disparado la neurona se reinicializa el acumulador de pesos.
   var_representacion(k,1)=2*umbral;
end;
```

6.1.3. FUNCIÓN ACTUALIZACIÓN PESO

Esta subrutina también es llamada en la función principal *aprendizaje*. Su cometido principal consiste en actualizar todos los pesos asociados a las sinapsis cuando la neurona de salida realiza un disparo. Esta actualización se rige según lo postulado en el mecanismo de aprendizaje bioinspirado STDP.

Las variables de entrada son:

-*matriz*: Esta variable contiene el valor de los pesos asociados a las sinapsis en el slot de tiempo en el que se produce el disparo de la neurona de salida.

-*aumento_peso*: Variable que proporciona los pesos de todas las sinapsis antes de actualizarse.

-*umbral*: Constante acordada como valor máximo del acumulador de la neurona de salida.

La salida de la función *actualizacion_peso* es la variable *aumento_peso* pero actualizada una vez aplicada las reglas descritas por el método STDP.

```
function aumento_peso=actualizacion_peso(matriz,aumento_peso,i,j,columnas,n_digitos,umbral)

p=j/2+1;                                                    %Variable auxiliar para índice.
if(p>n_digitos)
   p=n_digitos;
end;
aumento_peso(matriz(i,j-1),p)=aumento_peso(matriz(i,j-1),j/2)+incW(0);
if (aumento_peso(matriz(i,j-1),p)>(umbral/2))          %Ninguna actualización de peso supera el valor umbral.
   aumento_peso(matriz(i,j-1),p)=umbral/2;
end;
for k=1:1:i-1
        %Actualización positiva de los pesos.
        aumento_peso(matriz(k,j-1),p)=aumento_peso(matriz(k,j-1),j/2)+incW(matriz(k,j)-matriz(i,j));
        if (aumento_peso(matriz(k,j-1),p)>(umbral/2))
           aumento_peso(matriz(k,j-1),p)=umbral/2;
        %Para que la actualización de los pesos nunca sobrepase el umbral.
        end;
end;
for l=i+1:1:600
        if (matriz(l,j)<inf)
           %Actualización negativa de los pesos.
```

```
            aumento_peso(matriz(l,j-1),p)=aumento_peso(matriz(l,j-1),j/2)+incW(matriz(l,j)-matriz(i,j));
        end;
        if (aumento_peso(matriz(l,j-1),p)<0)
            aumento_peso(matriz(l,j-1),p)=0;
        %Para que la actualización de los pesos no sea inferior a cero.
        end;
end;
for j=p+1:1:columnas
    aumento_peso(:,j)=aumento_peso(:,p);          % Se actualiza la variable de salida en el siguiente slot de tiempo.
end;
```

6.1.4. FUNCIÓN CALCULO RATE

Esta función es una parte de la subrutina que calcula la tasa de éxito del experimento, considerando como éxito el que haya disparado la neurona deseada (aquella entrenada con el dígito que recibe a la entrada) antes que las demás:

$$Rate = \frac{succ}{total} \cdot 100$$

En esta ecuación, *succ* como se ha explicado en apartados anteriores se corresponde con el número de disparos que se producen correctamente. Es decir, aquellas entradas aplicadas a neuronas de salida que no han sido entrenadas con dicha entrada provocan un tiempo de disparo mayor que en el del detector que si ha sido entrenado con dicho entrada (sería el éxito de nuestro sistema). Por otro lado la variable *total* se corresponde con el número de disparos totales que se producen en el test. Esta subrutina nos va a devolver la variable *tiempo_disparo,* que muestra todos los tiempos en los que se produce los disparos de la neurona de salida a estudio.

Las variables de entrada a nuestra subrutina son:

-*matriz*: Son los pulsos en el dominio del tiempo enviados a cada neurona de salida a través de las sinapsis.

-*patrón*: Pesos finales de cada sinapsis (se corresponden cuando la neurona esta entrenada).

-*numero_disparos*: Números de disparos totales en el test.

Al igual que la función *aprendizaje* se ayuda de la función *acumula_peso* que nos ayuda actualizar el valor del acumulador de la neurona de salida.

```
function tiempo_disparo=calculo_rate(matriz,patron,numero_disparos)

[filas,columnas]=size(matriz);
peso_total=0;                    %Variable que actualiza el peso en un adicional producidos en un slot de tiempo.
disparo=0;                                                    %Indica si se ha reconocido la imagen o no.
umbral=2;
var_representacion=[];
var_representacion(1,1)=0;
```

```
var_representacion(1,2)=0;
k=2;
aferente_detector=[];
m=1;
tiempo_disparo=[];
tiempo=0;
indice=1;
for j=2:2:columnas
        for i=1:1:filas
                if(matriz(i,j)<inf && disparo<1)

                [disparo,peso_total,var_representacion]=acumula_peso(patrón(matriz(i,j-1)),peso_total,imbral,...
...var_representacion,k);

                var_representacion(k,2)=matriz(i,j);
                k=k+1;
                        if(disparo>0)                           %Se ha producido el reconocimiento de la imagen.
                          aferente_detector(m,1)=matriz(i,j-1);     %Función que marca el momento del disparo.
                          aferente_detector(m,2)=matriz(i,j);
                          m=m+1;
                        end;
                elseif (disparo>0 && i==filas)        %Se ha producido el disparo debido a una imagen intermedia.
                        disparo=0;
                        tiempo_disparo(indice)=aferente_detector(m-1,2)-tiempo;
                        if(j>2)
                          tiempo=((j/2)-1)*3e-3+2.5e-3;
                        else
                          tiempo=2.5e-3;
                        end;
                        indice=indice+1;
                end;
                if indice> numero_disparos
                  break;
%Se sale del "for" cuando supera el número de disparos que se pasan por línea de comandos.
                end
        end;
                if indice> numero_disparos
                  break;
                end
end;
```

6.2. ARTICULO URSI

Reconocimiento de Dígitos Usando Mecanismos Bioinspirados de Aprendizaje

J. A. Pérez-Carrasco, Eneko García-Arana, B. Acha, C. Serrano

jperez2@us.es, eneko.garcia.arana@gmail.com, bacha@us.es, cserrano@us.es

Dpto. Teoría de la Señal, ETSIT, Universidad de Sevilla. Avda de los Descubrimientos, s/n, CP41092

***Abstract*- This document presents a slightly bioinspired system for recognition of numeric digits. The system is composed by a Gabor filter bank with one scale and six orientations and one one-layer neural network trained using the biological training mechanism called STDP (*Spike-timing-dependent plasticity*). STDP is an unsupervised mechanism, where the network is able to learn autonomously when the inputs are repeated persistently. In the work presented here ten neurons have been trained to recognize ten different digits (0-9). To test and train our network, the MNIST database has been used. We have used 50000 images for training and 2000 images for testing.**

I. INTRODUCCIÓN

Cada vez es más habitual encontrar sistemas que tratan de implementar el procesamiento del cerebro para resolver problemas complejos. La evidencia constante de que el cerebro es capaz de realizar tareas casi impensables para la más compleja máquina desarrollada por el hombre a día de hoy está haciendo que la comunidad científica trate de emular el procesamiento biológico implementado en el cerebro. Recientemente los sistemas que están teniendo más éxito son aquéllos basados en redes neuronales, y de ellos, los basados en convoluciones que utilizan algoritmos de entrenamiento como el *backpropagation* están resultando más eficientes en el desarrollo de tareas como reconocimiento y seguimiento de objetos, clasificación, segmentación, etc [1][2]. A pesar de todos estos intentos, aún a día de hoy es difícil encontrar sistemas que además de emular la estructura de etapas del cerebro utilicen también los mecanismos de aprendizaje y comunicación mediante pulsos del mismo. El mecanismo de aprendizaje biológico STDP (*Spike-timing-dependent plasticity*) [3] es la hipótesis más comúnmente aceptada de mecanismo biológico de aprendizaje implementada en el cerebro humano. STDP es un proceso que ajusta la fuerza (pesos) de las conexiones entre neuronas en el cerebro basándose en los tiempos relativos entre los pulsos de salida de una neurona y los pulsos recibidos por la misma y que alteran su potencial.

Tratando de implementar un pequeño sistema bioinspirado hemos desarrollado en el presente trabajo un sistema compuesto de dos etapas. La primera de ellas es un banco de filtros de Gabor, que está bastante aceptado como una de las primeras etapas del procesamiento en el cerebro. La segunda etapa es una red neuronal de una etapa, compuesta por diez neuronas de salida, cuyas conexiones han sido entrenadas usando STDP. El sistema recibe como entrada un dígito numérico de 0 a 9, obtenido de la base de datos MNIST [4], y sólo la neurona de salida correspondiente a ese dígito de entrada será la que presente una mayor y rápida actividad a la salida.

II. IMPLEMENTACIÓN

A. Descripción del sistema

El sistema implementado en este trabajo está compuesto de dos etapas y se puede ver en la Fig. 1. La primera de ellas es un banco de filtros de Gabor y la segunda una red neuronal monoetapa compuesta por diez neuronas de salida. El tamaño de las imágenes de entrada es de 32x32, aunque para nuestra aplicación, estas imágenes han sido submuestradas para tener un tamaño de 16x16. Dichas imágenes son convolucionadas con un conjunto de 6 filtros de Gabor, que implementan una escala y seis orientaciones. El tamaño de los filtros de Gabor es 10x10.

Cada uno de estos filtros se obtiene mediante las siguientes expresiones [5]:

$$g(x,y)=\frac{1}{2\pi\cdot\sigma_x\sigma_y}\exp\left[-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2}+\frac{y^2}{\sigma_y^2}\right)+j2\pi\cdot Wx\right], \quad (1)$$

Realizando unas determinadas dilataciones y rotaciones a la función de transferencia obtenida *g(x, y)* obtenemos cada filtro codificando una orientación y escala:

$$\begin{aligned} g_{s,k}(x,y) &= a^{-s}g(x',y') \\ x' &= a^{-s}(x\cdot\cos\theta+y\cdot sen\theta) \\ y' &= a^{-s}(-x\cdot sen\theta+y\cdot\cos\theta) \end{aligned} \quad (2)$$

En estas expresiones θ representa la orientación y *S* la escala. Los parámetros $\{a,\theta,W,\sigma_x,\sigma_y\}$ son descritos en el método implementado por Manjunath [5].

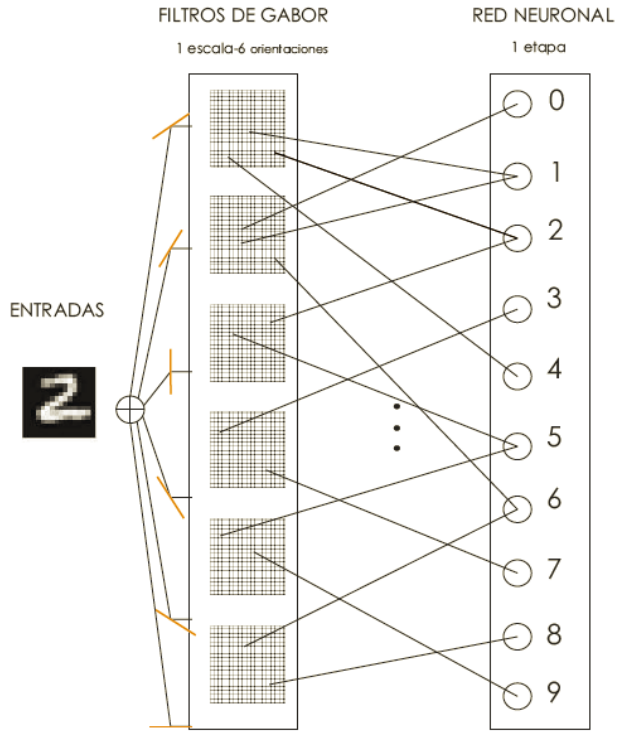

Fig. 1. Sistema de reconocimiento de dígitos con mecanismos bioinspirados de aprendizaje.

De cada imagen de salida tras la convolución hemos seleccionado la parte central de tamaño 10x10, una vez comprobado que en ese conjunto de píxeles estaba contenida la información con valor distinto de cero.

El mecanismo STDP es un mecanismo biológico de aprendizaje que se basa en los tiempos relativos entre los pulsos producidos por una neurona y los pulsos que recibe a su entrada y que pueden ser los que le han hecho activarse y disparar. Es por ello que en nuestra implementación necesitamos codificar el valor de cada píxel a la salida de la convolución con uno de los filtros de Gabor en pulsos. Para nuestra implementación hemos realizado una codificación intensidad-tiempo, de modo que hemos creado un pulso temporal para cada píxel con un tiempo de creación "inversamente proporcional" al valor del píxel. Así, un píxel con valor cercano a 0 producirá un pulso "tardío" y un píxel con valor cercano a 255 producirá un disparo casi instantáneo. El tiempo de creación de un pulso vendrá relacionado con su valor de píxel con la expresión:

$$tpo\_pulso = cte\_temp \times (1 - pixel\_value/255) \qquad (3)$$

En nuestra implementación *cte_temp* tiene un valor de 3*ms*. Un ejemplo ilustrativo de esta relación inversamente proporcional puede verse en la Fig. 2.

En el sistema, cada píxel de salida de cada filtro de Gabor está conectado a una de las de las 10 neuronas de salida, con lo cual tenemos 6x100x10 (6 componentes de Gabor, 100 píxeles de salida de cada filtro y 10 neuronas de salida) que hace un total de 6000 conexiones en nuestra etapa neuronal.

B. Mecanismo aprendizaje STDP

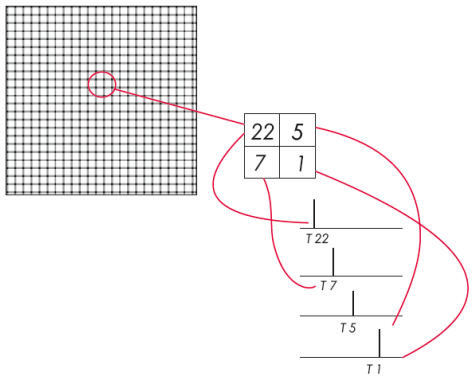

Fig. 2. Conversión de nivel de pixel en tiempo.

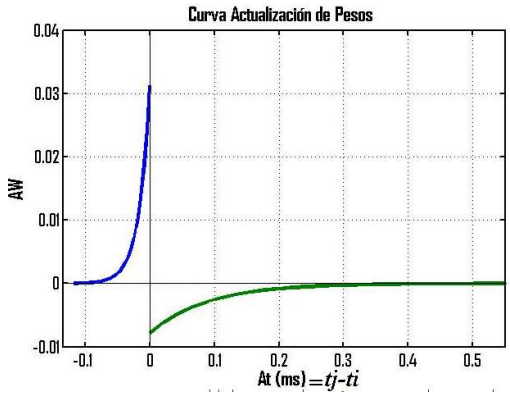

Fig. 3. Curva Aprendizaje STDP.

En nuestra implementación, las conexiones correspondientes a cada neurona de salida han sido entrenadas por separado. En una implementación futura y más eficiente se permitirá que todas las neuronas aprendan a la vez y el hecho de permitir inhibición entre ellas acabará haciendo que cada neurona aprenda un dígito diferente [3]. Por tanto, para la implementación aquí presentada, cada neurona de salida *i*, correspondiente al dígito de entrada *i* (*i*=0,..,9) ha sido entrenada por separado utilizando un conjunto de 5000 imágenes correspondientes al mismo dígito. Cuando la neurona *i* (sus conexiones) ha sido entrenada se procede a entrenar el resto de neuronas del mismo modo.

Pero nuestra pregunta sigue sin ser respondida, ¿cómo se implementa el aprendizaje bioinspirado basado en STDP? El mecanismo de aprendizaje STDP (*Spike Timing Depedent Plasticity*) [3] se basa en el principio de que las conexiones con las neuronas de salida ("pesos") se van reforzando o debilitando con la repetición de unos pulsos de entrada determinados, de modo que aquellas conexiones cuyos pulsos provoquen un disparo en la neurona de salida se verán reforzadas. En cambio, aquellas conexiones por las cuales viajen pulsos que lleguen a la neurona de salida posteriormente al disparo de la misma se verán debilitadas. STDP es un mecanismo no supervisado, es decir, las neuronas aprenden de manera autónoma ante la repetición de los patrones de entrada. Gracias a este mecanismo, cuando la neurona haya aprendido un patrón de entrada, le bastaran muy pocos pulsos en sólo algunas conexiones de entrada para producir un disparo. El mecanismo STDP se resume del siguiente modo: Inicialmente el peso de todas las conexiones es el mismo (o puede estar inicializado

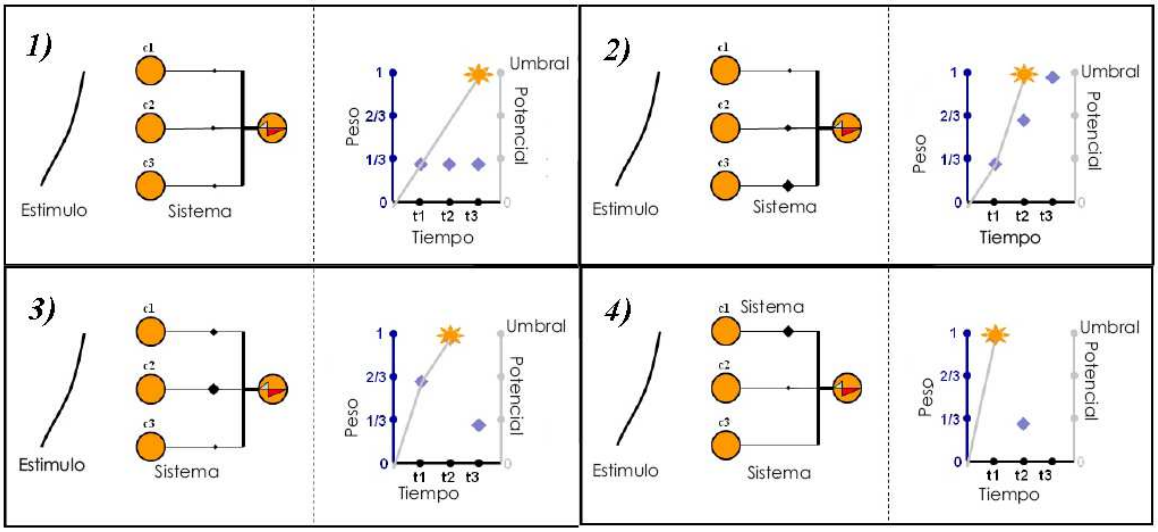

Fig. 4. Mecanismo Bioinspirado de Aprendizaje STDP.

aleatoriamente), pero cada vez que la neurona de salida produce un pulso de salida los pesos de las conexiones se actualizan de acuerdo a las siguientes expresiones [3]:

$$w_o = \Delta w_{inst} + w_{ant}$$

$$\Delta w_{inst} = \begin{cases} a^+ \cdot \exp\left(\dfrac{t_j - t_i}{\tau^+}\right) & t_j \le t_i \\ -a^- \cdot \exp\left(-\dfrac{t_j - t_i}{\tau^-}\right) & t_j > t_i \end{cases} \qquad (4)$$

En estas ecuaciones w_{ant} es el peso anterior en una de las conexiones, Δw_{inst} es el incremento de peso debido al disparo de la neurona final. Los parámetros a^+ y a^- y τ^+ y τ^- son constates predefinidas [3]. Por último, el valor $t_j - t_i$ representa la diferencia de tiempo entre el pulso producido por la neurona aferente en el instante t_j cuya conexión con la neurona de salida se está actualizando y el disparo producido por la neurona final en el instante t_i. La Fig. 3 representa la curva de actualización de pesos. En ella se puede ver que aquellas conexiones cuyos pulsos sean anteriores al disparo de la neurona de salida (y por tanto causantes del mismo) se verán reforzadas, sin embargo, los pesos de aquellas conexiones con pulsos posteriores al pulso de la neurona de salida se verán atenuados.

En la Fig. 4 se ilustra el mecanismo STDP de un modo simplificado. En esta figura, una neurona de salida está conectada a 3 neuronas que representamos como *c1*, *c2* y *c3* que producen un determinado patrón codificado en sus pulsos. Como se puede ver el orden de disparo de las neuronas es primero *c1*, después *c2* y finalmente *c3* (como viene representado por la curva de estímulo). El peso de las neuronas se representa en la figura como un punto negro en las conexiones (cuánto más grande mayor es su valor). Al principio (*paso 1*), todas las conexiones tienen igual valor (*pesos=1/3*), de modo que cada vez que llega un pulso a la neurona de salida, su potencial se ve incrementado por el peso correspondiente a cada

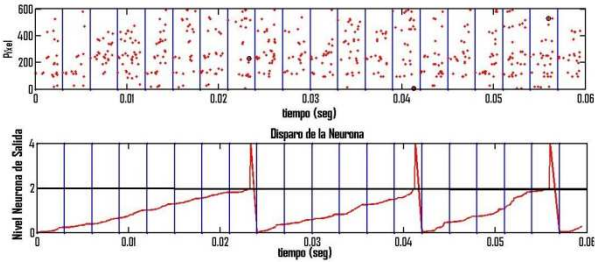

Fig. 5. Etapa de Entrenamiento de la Neurona '0' al inicio.

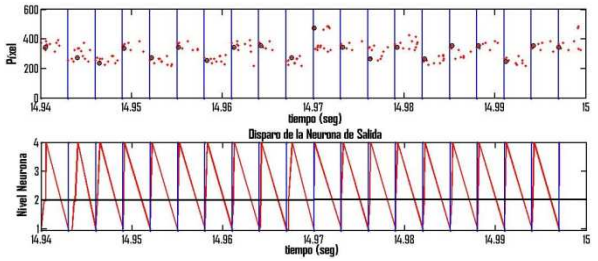

Fig. 6. Etapa de Entrenamiento de la Neurona '0' al final.

conexión. Se puede ver que en la primera iteración la neurona alcanza el umbral de disparo (de valor '1') cuando le llega el pulso procedente de la neurona *c3* (instante *t3*). En el primer caso, como el disparo de la neurona de salida se produce posteriormente a los pulsos de entrada, las 3 conexiones se ven reforzadas a valores 1/3, 2/3 y 1 respectivamente. En la segunda repetición del patrón de entrada (*paso 2*), será la neurona *c2* (por tener un *peso=2/3*) la que produzca la activación de la neurona de salida en el instante *t2*, con lo cual las conexiones con *c1* y *c2* se verán reforzadas y la conexión con *c3* atenuada. En una tercera iteración (*paso 3*), será la neurona *c2* de nuevo la causante de producir un disparo a la salida en el instante *t2*, con lo cual la conexión con *c1* será reforzada de nuevo y la conexión con *c3* se verá atenuada. En la última y cuarta iteración del ejemplo (*paso 4*), la conexión con la neurona *c1* ya tiene un peso de valor '1' y por tanto es suficiente para activar la neurona de salida en el instante *t1*. Las otras dos conexiones (con *c2* y *c3*) se ven atenuadas. En un estado estacionario estas dos conexiones estarán inhibidas y la conexión con la neurona *c1* bastará para producir la

Neurona	1º Disparo	2º Disparo	3º Disparo	4º Disparo	5º Disparo
0	5,37	4,58	4,64	5,17	5,01
1	0,51	0,63	0,61	0,60	0,75
2	1,92	4,75	3,50	2,47	2,93
3	1,85	6,59	0,69	2,53	2,57
4	4,67	2,30	3,78	2,77	3,50
5	1,77	1,43	2,38	3,57	4,04
6	1,15	5,63	1,81	4,88	3,77
7	3,11	4,03	4,97	2,07	7,35
8	0,27	1,55	0,60	0,75	2,13
9	1,27	1,15	2,80	2,84	1,92

Tabla 1. Disparos de las neuronas de salida ante el mismo estímulo (*ms*).

activación con la neurona de salida. Nótese que esto es lo que se esperaba pues la neurona *c1* es la más activa y la más rápida, y el algoritmo de aprendizaje STDP ha permitido que la neurona de salida aprenda esto. Quede constancia de que el ejemplo aquí descrito es muy sencillo y que sólo trata de ilustrar la idea que se esconde detrás del mecanismo biológico STDP.

III. Resultados Experimentales

Podemos definir dos etapas en nuestro experimento. La primera etapa es de entrenamiento de cada neurona de salida por separado. En la Fig. 5 hemos representado esta fase. El eje *y* de la gráfica superior codifica el píxel que dispara un pulso a la salida de cada convolución con el banco de filtros de gabor. Como recordamos había 600 píxeles posibles. El pulso producido por cada uno de estos píxeles (sólo se han representado aquellos con un valor superior a un nivel predefinido) se ha representado como un punto rojo. El eje *x* codifica el tiempo en segundos. Cada imagen utilizada de entrada produce una serie de pulsos en una ventana de duración 3ms (recordemos que es el tiempo utilizado en la codificación intensidad-tiempo). La gráfica inferior de la Fig. 5 representa el potencial acumulado por la neurona de salida que se está entrenando. En esta figura se está representando el entrenamiento correspondiente a la neurona '0'. Se puede observar en esta etapa que la neurona necesita cargar durante un tiempo largo (del orden de 22ms, ~8 imágenes) la información correspondiente a los patrones para producir un pulso de salida y descargarse. En el momento del disparo, entra en juego la actualización de los pesos de cada conexión, esto contribuye en un futuro a que la neurona de salida detecte rápidamente el patrón de entrada. En la Fig. 6 se muestra la etapa final del entrenamiento. Se puede observar que la neurona sólo necesita unos poquitos pulsos para activarse y producir un disparo, prueba de que ya se ha entrenado y ha reconocido el patrón a la entrada (del orden de décimas de ms).

Cada una de las diez neuronas ha sido entrenada con 5000 imágenes codificando uno de los dígitos. Estos dígitos han sido obtenidos de la base de datos MNIST [4]. A modo de ejemplo, en la Tabla 1 se muestra el tiempo que tarda en disparar cada una de las neuronas de salida una vez entrenadas cuando a la entrada se introducen cinco imágenes correspondientes al dígito '1'. Este caso (ilustrativo) no es óptimo ya que para dos de las cinco imágenes son otras neuronas de salida las que han disparado un pulso a la salida de un modo más rápido (en este caso la neurona que codifica el dígito '8').

El sistema ha sido testeado utilizando 200 imágenes correspondientes a cada dígito. Si consideramos éxito el que haya disparado la neurona deseada (aquella entrenada ante el dígito a la entrada) antes que las demás, podemos calcular la tasa de éxito como:

$$rate = \frac{succ}{total} \cdot 100 \qquad (5)$$

En esta ecuación, *succ* corresponde al número de entradas reconocidas correctamente y *total* corresponde al número de imágenes testeadas. En nuestro experimento la tasa de reconocimiento ha sido del 80.56%.

IV. Conclusiones

En el trabajo presentado se ha demostrado la posibilidad de incluir mecanismos de aprendizaje no supervisados bioinspirados en una aplicación sencilla de reconocimiento de dígitos. Nótese que en este trabajo el objetivo último no era alcanzar una tasa de reconocimiento elevada, sino demostrar cómo un sistema sencillo puede ser entrenado utilizando el algoritmo bioinspirado STDP y proporcionar resultados aceptables. Es posible alcanzar mejores resultados haciendo el sistema un poco más complejo utilizando tal vez mayor resolución en las imágenes, un banco de filtros mayor y añadiendo inhibición entre las neuronas de salida durante la etapa de entrenamiento, de modo que cuando una neurona de salida dispare un pulso, este pulso se utilice para inhibir e impedir que el resto de neuronas "aprendan" el patrón que ha hecho disparar a la primera de ellas [3].

Un sistema como el descrito puede ser implementado totalmente con módulos convolucionales bioinpirados, tales como módulos de convolución basados en el protocolo AER (*Address-Event-Representation*) [6].

Agradecimientos

Este trabajo ha sido financiado en parte por el proyecto MITRA (TEC2010-21619-C04-02), del Ministerio de Ciencia e Innovación.

Referencias

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.

[2] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio, "Robust object recognition with cortex-like mechanisms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 3, pp. 411–426, 2007.

[3] T. Masquelier, R. Guyonneau, and S. J. Thorpe, "Competitive STDP-Based Spike Pattern Learning, " *Neural Comput*, 2009, vol 21, pp. 1259-1276.

[4] Yann LeCun, Website: http://yann.lecun.com/exdb/mnist/

[5] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Trans. Pattern Anal. Mach. Intell.* vol.18, no. 8, pp. 837–842, Aug. 1996.

[6] J. A. Pérez-Carrasco, et al., "Advanced vision processing systems: Spike-based simulation and processing", *LNCS*, vol. 5807, pp. 640–651,2009.