# Towards Arabic Character Recognition Using Bi-directional Associative Memory

Edris Amin

EECS Department,

University of Toledo, 2801 W. Bancroft

Toledo, Ohio 43606, USA

Devinder Kaur

EECS Department,

University of Toledo, 2801 W. Bancroft

Toledo, OH 43606, USA

***Abstract –*** This paper demonstrates the possibility of Arabic character recognition using a trained Bi-directional Associative Memory (BAM) Artificial Neural Network (ANN). The difficulties of such character recognition are presented and the use of BAM ANN is justified and demonstrated.

1. **Introduction**  
    The area of research involving western character recognition has been very active [1]. There are many reasons why character recognition hasn’t yet penetrated into eastern scripts. One of the reasons may be the fact that the computational know how was concentrated in the west, and the first computers were designed to display western characters as well as receive data in the form of western characters.   
     
    A key factor in why western characters were desirable for the first computers is because the western character set is mono-spaced meaning that all characters can be written to fit inside the same sized box. Arabic is a language whose written script has varying widths for characters depending on whether it appears in the beginning, middle, or end of a word. The Arabic character set is briefly presented in section II.

The Bi-directional Associative Memory technique and advantages is explained in section III. The project is implemented in MATLAB and is described in section IV. A summary of future work and conclusion is provided section V.

1. **Arabic Character Set**

The Arabic characters are utilized when writing script in the Arabic, Persian, and Urdu languages. The Arabic characters not only change size making it difficult to apply proper character recognition but many letters share the same fundamental shape. In table 1 the complete Arabic character set is shown. Please note that the characters presented in table 1 don’t represent the complete set of characters used in the Persian and Urdu languages, only Arabic. Those two languages adopted the Arabic characters and needed to add 4 extra characters for phonetic sounds which do not exist in the Arabic language like:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| P | پ, | Ch | ﭺ, | Zha | ژ, | and Ga | گ |

The columns of table 1 are represent how characters are written when in different parts of a word. Also note that many of the adjacent rows only differ by the number of dots sharing the same fundamental shape. Those rows are highlighted in the same color for better observation. Due to these challenges the goal of this project will be to develop a system which can at the very least identify fundamental features like its major direction (horizontal ب vs vertical ا ), whether is has a belly like س, whether it has an eye (or two) like م, the number and location of dots, and other features.

1. **Bi-directional Associative Memory**  
    Bart Kosko first introduced the idea of BAM in [2]. His attempt was one of the more popular ones during an era when many researchers were attempting to develop a psychological model for computation [3]. The ultimate goal for psychological computation is to create a model which requires a smaller training set and provides improved accuracy despite being provided a limited training set [3].

Other associative memory networks include the Hopfield and Hebbian network models [4, 5]. Both Hopfield and Hebbian networks associate inputs with outputs of equal size. Mapping elements to elements of equal size has some benefits including higher memory capacity. A weakness of mapping sets of equal size is that as the set size grows the number of matrix calculations grow exponentially.

The BAM network differs in that it maps two sets of arrays with unequal size, known as “hetroassociative” [5]. This has a smaller maximum capacity associated with the size of the smaller arrays in one of the sets. In the world of computing, a BAM network may be desirable because it requires fewer computations than a Hopfield network [5].

The Hebb Rule is commonly used for determining the weights of Associative Memory Networks. The Hebb Rule can be applied for networks with binary sets and bipolar sets [2, 3]. The algorithm considers associating set X with set Y.

|  |  |  |  |
| --- | --- | --- | --- |
| letter | initial | medial | final |
| ا | ا | ـا | ـا |
| ب | بـ | ـبـ | ـب |
| ت | تـ | ـتـ | ـت |
| ث | ثـ | ـثـ | ـث |
| ج | جـ | ـجـ | ـج |
| ح | حـ | ـحـ | ـح |
| خ | خـ | ـخـ | ـخ |
| د | د | ـد | ـد |
| ذ | ذ | ـذ | ـذ |
| ر | ر | ـر | ـر |
| ز | ز | ـز | ـز |
| س | سـ | ـسـ | ـس |
| ش | شـ | ـشـ | ـش |
| ص | صـ | ـصـ | ـص |
| ض | ضـ | ـضـ | ـض |
| ط | طـ | ـطـ | ـط |
| ظ | ظـ | ـظـ | ـظ |
| ع | عـ | ـعـ | ـع |
| غ | غـ | ـغـ | ـغ |
| ف | فـ | ـفـ | ـف |
| ق | قـ | ـقـ | ـق |
| ك | كـ | ـكـ | ـك |
| ل | لـ | ـلـ | ـل |
| م | مـ | ـمـ | ـم |
| ن | نـ | ـنـ | ـن |
| و | و | ـو | ـو |
| ه | هـ | ـهـ | ـه |
| ي | يـ | ـيـ | ـي |

Table 1. The Arabic Characters

A weight, w­ij is associated with the weight connecting element xi to element yj. This relationship is shown in figure 1.

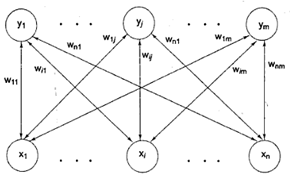


Fig. 1 An example of an association memory ANN [4]

Hebb Rule for calculating weights in a BAM network with P arrays to memorize [4];

(1)

In [3] it was mentioned that BAM networks can have binary or bipolar forms. In a binary BAM network each of the elements xi and yj have binary values 0, or 1. In a bipolar BAM each of the elements xi and yj have bipolar values -1, or 1. For our application bipolar patterns will be memorized.

1. **Application Technique**

My application was made in MATLAB. Initially 19 fundamental Arabic characters were digitized into 7 x 8 pixel bipolar matrices. These are shown in figure 2. The digitized characters were generated with the MATLAB script *drawletterV1BIPOLAR.m* the code of which is provided in the appendix.

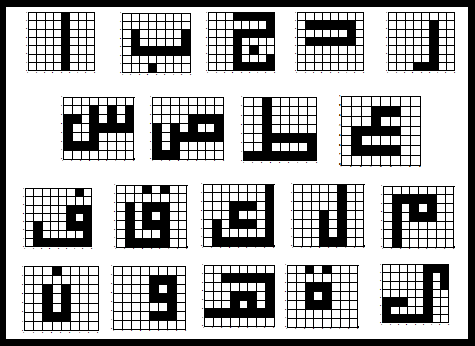


Figure 2. Digitized fundamental Arabic characters.

Once the fundamental characters were defined a bipolar 19 row identity target matrix was created such that each row was the target vector for each of the 19 characters.

The weight matrix for the BAM ANN was trained using the Hebb Rule in (1). Upon testing it was discovered that the weight matrix only converged on 2 targets for any pattern.

After eliminating various characters and features it was decided that for this project the BAM ANN would be designed to identify the four fundamental features shown in figure 4. Although the MATLAB code for *letterV4BIPOLAR.m* provided in the appendix contains 10 features only the four which produced an acceptable weight matrix were included. Feature 1 identifies a vertical character like ا . Feature 2 helps identify a character like ب . Feature 3 helps identify a character with a belly like ق. Feature 4 will identify a letter with three dots like ش. Including more features caused the BAM to erroneously converge to incorrect targets. The code for training and testing the BAM ANN is provided in the appendix for the script *letterrecogBIPOLARV4.m*

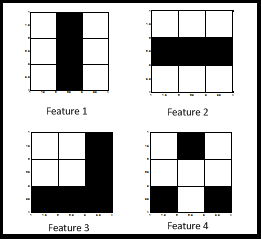


Figure 4. Identifiable features

1. **Conclusion and Future Work**

We have demonstrated that a BAM ANN can be successfully implemented to identify fundamental features of Arabic characters. The 10 features presented in the MATLAB script *letterV4BIPOLAR.m* are sufficient for identifying all the Arabic characters. Due to the limited time only 4 were successfully trained.

The algorithm requires more debugging so the BAM ANN can successfully converge for all 10 features. Once complete convergence is accomplished the ANN can be implemented in Arabic OCR software using the 10 features in a scanning window type reader. This will enable a computer for successful Arabic character recognition.

Refernces

[1] Ø. D. Trier, A.K. Jain, T. Taxt, “Feature extraction methods for character recognition—A survey”, *Pattern Recognition*, 29:4(1996), 641–662.

[2] B. Kosko, “Bidirectional associative memories” *IEEE Transactions on Systems, Man, and Cybernetics, SMC-17* 1987, pp. 49-60

[3] J.A. Anderson, “Cognitive and psychological computation with neural models” *IEEE Transactions on Systems, Man, & Cybernetics*, Vol 13(5), Sep-Oct 1983, 799-815.

[4] S.N. Sivanandam, S. Sumathi, S.N. Deepa, *Introduction to Neural Networks Using Matlab 6.0*, Tata McGraw-Hill Education, 2006

[5] M. Negnevitsky, *Artificial Intelligence A Guide to Intelligent Systems*, 2nd Edition,

Addison Wesley, 2005

**Appendix**

drawletterV1BIPOLAR.m

function [alif,ba,jeem,dal,ra,seen,Saad,Taa,a3en,fa,Qa,kaf,lam,meem,noon,waw,ha,tah,ya] = drawletterV1BIPOLAR()

%Alif-----------------

alif=[1 1 1 1 -1 1 1 1

1 1 1 1 -1 1 1 1

1 1 1 1 -1 1 1 1

1 1 1 1 -1 1 1 1

1 1 1 1 -1 1 1 1

1 1 1 1 -1 1 1 1

1 1 1 1 -1 1 1 1];%base

letter=alif;

draw(letter)

%Ba,--------------------

ba = [1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 -1 1 1 1 1 1 -1

1 -1 1 1 1 1 1 -1

1 -1 -1 -1 -1 -1 -1 -1%base

1 1 1 1 1 1 1 1

1 1 1 -1 1 1 1 1];

figure

letter=ba;

draw(letter)

%Jeem---------------------

jeem = [1 1 1 -1 -1 -1 -1 -1

1 1 1 1 1 1 1 -1

1 1 1 -1 -1 -1 -1 -1%base

1 1 1 -1 1 1 1 1

1 1 1 -1 1 -1 1 1

1 1 1 -1 1 1 1 -1

1 1 1 -1 -1 -1 -1 -1];

figure

letter=jeem;

draw(letter)

%dal------------------------------

dal = [1 1 1 1 1 1 1 1

1 -1 -1 -1 -1 -1 -1 1

1 1 1 1 1 1 -1 1

1 -1 -1 -1 -1 -1 -1 1%base

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1];

figure

letter=dal;

draw(letter)

%ra--------------------------

ra = [1 1 1 1 1 1 1 1

1 1 1 1 1 -1 1 1

1 1 1 1 1 -1 1 1

1 1 1 1 1 -1 1 1

1 1 1 1 1 -1 1 1%base

1 1 1 1 1 -1 1 1

1 1 1 -1 -1 -1 1 1];

figure

letter=ra;

draw(letter)

%seen-------------

seen = [1 1 1 1 1 1 1 1

1 1 1 -1 1 -1 1 -1

-1 -1 1 -1 1 -1 1 -1

-1 1 1 -1 -1 -1 -1 -1

-1 1 1 -1 1 1 1 1

-1 -1 -1 -1 1 1 1 1

1 1 1 1 1 1 1 1];

figure

letter=seen;

draw(letter)

%Saad-----------------------

Saad = [1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 1 1 -1 -1 -1 -1

-1 1 -1 1 -1 1 1 -1

-1 1 -1 -1 -1 -1 -1 -1

-1 1 -1 1 1 1 1 1

-1 -1 -1 1 1 1 1 1];

figure

letter=Saad;

draw(letter)

%Taa--------------------------

Taa = [1 1 -1 1 1 1 1 1

1 1 -1 1 1 1 1 1

1 1 -1 1 1 1 1 1

1 1 -1 1 1 1 1 1

1 1 -1 -1 -1 -1 -1 -1

1 1 -1 1 1 1 1 -1

-1 -1 -1 -1 -1 -1 -1 -1];%base

figure

letter=Taa;

draw(letter)

%a3en----------------

a3en= [1 1 1 1 1 1 1 1

1 1 1 -1 -1 -1 1 1

1 1 1 -1 1 1 1 1

1 -1 -1 -1 -1 -1 1 1

1 -1 1 1 1 1 1 1%base

1 -1 -1 -1 -1 -1 1 1

1 1 1 1 1 1 1 1];

figure

letter=a3en;

draw(letter)

%fa-------------------

fa = [1 1 1 1 1 1 -1 1

1 1 1 1 1 1 1 1

1 1 1 1 1 -1 -1 -1

1 1 1 1 1 -1 1 -1

1 -1 1 1 1 -1 -1 -1

1 -1 1 1 1 1 1 -1

1 -1 -1 -1 -1 -1 -1 -1];%base

figure

letter=fa;

draw(letter)

%Qa-------------------------

Qa = [1 1 1 -1 1 -1 1 1

1 1 1 1 1 1 1 1

1 -1 1 -1 -1 -1 1 1

1 -1 1 -1 1 -1 1 1

1 -1 1 -1 -1 -1 1 1

1 -1 1 1 1 -1 1 1%base

1 -1 -1 -1 -1 -1 1 1];

figure

letter=Qa;

draw(letter)

%kaf-------------------------

kaf = [1 1 1 1 1 1 1 -1

1 1 1 1 1 1 1 -1

1 1 1 -1 -1 -1 1 -1

1 1 1 -1 1 1 1 -1

1 -1 1 -1 -1 -1 1 -1

1 -1 1 1 1 1 1 -1

1 -1 -1 -1 -1 -1 -1 -1];%base

figure

letter=kaf;

draw(letter)

%lam----------------------------

lam = [1 1 1 1 1 -1 1 1

1 1 1 1 1 -1 1 1

1 1 1 1 1 -1 1 1

1 1 1 -1 1 -1 1 1

1 1 1 -1 1 -1 1 1

1 1 1 -1 1 -1 1 1%base

1 1 1 -1 -1 -1 1 1];

figure

letter=lam;

draw(letter)

%meem----------------------------

meem = [1 1 1 1 1 1 1 1

1 -1 -1 -1 -1 -1 1 1

1 -1 1 -1 1 -1 1 1

1 -1 1 -1 -1 -1 1 1%base

1 -1 1 1 1 1 1 1

1 -1 1 1 1 1 1 1

1 -1 1 1 1 1 1 1];

figure

letter=meem;

draw(letter)

%noon----------------------------

noon = [1 1 1 -1 1 1 1 1

1 1 1 1 1 1 1 1

1 1 -1 1 -1 1 1 1

1 1 -1 1 -1 1 1 1

1 1 -1 1 -1 1 1 1%base

1 1 -1 -1 -1 1 1 1

1 1 1 1 1 1 1 1];

figure

letter=noon;

draw(letter)

%waw------------------------------

waw = [1 1 1 1 1 1 1 1

1 1 1 1 -1 -1 -1 1

1 1 1 1 -1 1 -1 1

1 1 1 1 -1 -1 -1 1%base

1 1 1 1 1 1 -1 1

1 1 1 1 -1 -1 -1 1

1 1 1 1 1 1 1 1];

figure

letter=waw;

draw(letter)

%ha---------------------------------

ha = [1 1 1 1 1 1 1 1

1 1 -1 -1 -1 -1 -1 -1

1 1 1 1 1 1 1 -1

1 1 1 -1 -1 -1 1 -1

1 1 1 -1 1 -1 1 -1

-1 -1 -1 -1 -1 -1 -1 -1%base

1 1 1 1 1 1 1 1];

figure

letter=ha;

draw(letter)

%tah----------------------

tah= [1 1 -1 1 -1 1 1 1

1 1 1 1 1 1 1 1

1 1 -1 -1 -1 1 1 1

1 1 -1 1 -1 1 1 1

1 1 -1 -1 -1 1 1 1%base

1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1];

figure

letter=tah;

draw(letter)

%ya----------------------------------

ya = [1 1 1 1 1 -1 -1 -1

1 1 1 1 1 -1 1 -1

1 1 1 1 1 -1 1 -1%base

1 1 1 1 1 -1 1 1

-1 -1 -1 1 1 -1 1 1

-1 1 1 1 1 -1 1 1

-1 -1 -1 -1 -1 -1 1 1];

figure

letter=ya;

draw(letter)

function draw(letter);

[m n]=size(letter);

draw=[letter letter(:,[n])]';

draw=[draw draw(:,[m])]';

pcolor(draw)

colormap(gray)

axis('ij')

axis image

pause( -1.1)

function digit\_plot(digit);

[m n]=size(digit);

digit\_plot=[digit digit(:,[n])]';

digit\_plot=[digit\_plot digit\_plot(:,[m])]';

pcolor(digit\_plot)

colormap(gray)

axis('ij')

axis image

pause( -1.1)

letterV4BIPOLAR.m

%letters V4 Bipolar

%only 3 pixels wide

function [A, B, C, D, E, F, G, H, I, J] = letterV4BIPOLAR()

%A-----------------

%%alif, jeem1.2.3, ra1.2, taa1.2, kaf, lam, meem

A=[1 -1 1

1 -1 1

1 -1 1];

letter=A;

draw(letter)

%B--------------------

%ba1.2.3, fa, qa

B =[ 1 1 1

-1 -1 -1

1 1 1];

figure

letter=B;

draw(letter)

%C---------------------

%jeem1.2.3, dal1.2, Saad1.2, fa, qa, waw, ya

C = [-1 -1 -1

1 1 -1

-1 -1 -1];

figure

letter=C;

draw(letter)

%D------------------------------

%See1.2, Saad1.2, lam, noon

D = [-1 1 -1

-1 1 -1

-1 -1 -1];

figure

letter=D;

draw(letter)

%E--------------------------

%Taa, fa, qa, meem, waw, ha, tah

E = [-1 -1 -1

-1 1 -1

-1 -1 -1];

figure

letter=E;

draw(letter)

%F-------------

%ba1.2.3, r1.2, kaf

F = [ 1 1 -1

1 1 -1

-1 -1 -1];

figure

letter=F;

draw(letter)

%G-----------------------

%jeem1.2.3, ya

G = [-1 1 1

-1 1 1

-1 -1 -1];

figure

letter=G;

draw(letter)

%H--------------------------

%single dot

%ba1, jeem2, dal2, ra2, saad2, Taa2, fa, noon

H = [1 1 1

1 1 1

-1 1 1];%base

figure

letter=H;

draw(letter)

%I----------------

%double dot

%ba2, qa, tah, ya

I = [-1 1 -1

1 1 1

1 1 1];%base

figure

letter=I;

draw(letter)

%J----------------

%triple dot

%ba3, seen2

J = [ 1 -1 1

1 1 1

-1 1 -1];%base

figure

letter=J;

draw(letter)

function draw(letter);

[m n]=size(letter);

draw=[letter letter(:,[n])]';

draw=[draw draw(:,[m])]';

pcolor(draw)

colormap(gray)

axis('ij')

axis image

pause( 1.-1)

function digit\_plot(digit);

[m n]=size(digit);

digit\_plot=[digit digit(:,[n])]';

digit\_plot=[digit\_plot digit\_plot(:,[m])]';

pcolor(digit\_plot)

colormap(gray)

axis('ij')

axis image

pause( 1.-1)

letterrecogBIPOLARV4.m

function letterrecogBIPOLARV4

disp(' Hit any key to view bit map representaion of the fundemental Arabic character set.')

disp(' ')

pause

drawletterV1BIPOLAR()

disp(' Hit any key to view the ten fundemental features useful for identifying characters.')

pause

[A, B, C, D, E, F, G, H, I, J] = letterV4BIPOLAR();

p= [A(:), B(:), F(:), J(:)];%, G(F:), H(:), I(:), J(:)]

%p=[digit1(:),digit2(:),digit3(:),digit4(:),digit5(:),digit6(:),digit7(:),digit8(:),digit9(:),digit0(:)]

disp('Hit any key to define ten target vectors denoted by "t". ')

pause

t= [ 1 -1 -1 -1%01

-1 1 -1 -1%02

-1 -1 1 -1%03

-1 -1 -1 1]; %04

disp(' Hit any key to define the weight matrix which maps p to t.')

pause

W=[];

%build weight matrix W = sum((pi\*ti), i, 1, 10);

W = p(:,1)\*(t(1,:));

for i=2:4

W = W+ p(:,i)\*(t(i,:));

end

%testing

disp('Hit any key to test the weight matrix for the 4 selected features.')

pause

t1 = sign(W'\*p(:,1))

p1=sign(W\*t1)

t2 = sign(W'\*p(:,2))

p2=sign(W\*t2)

t3 = sign(W'\*p(:,3))

p3=sign(W\*t3)

t4 = sign(W'\*p(:,4))

p4=sign(W\*t4)