**Hopfield network**

**Research Background:**

Hopfield neural network was invented by Dr. John J. Hopfield in 1982. It consists of a single layer which contains one or more fully connected recurrent neurons. The Hopfield network is commonly used for auto-association and optimization tasks.

**Discrete Hopfield Network:**

A Hopfield network which operates in a discrete line fashion or in other words, it can be said the input and output patterns are discrete vector, which can be either binary (0,1) or bipolar (+1, -1) in nature. The network has symmetrical weights with no self-connections i.e., **wij = wji** and **wii = 0**.

**Architecture:**

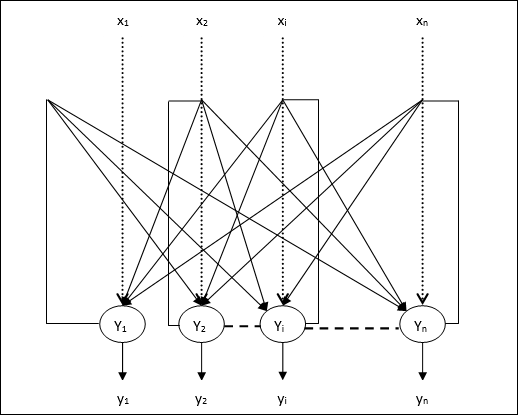
Following are some important points to keep in mind about discrete Hopfield network −

* This model consists of neurons with one inverting and one non-inverting output.
* The output of each neuron should be the input of other neurons but not the input of self.
* Weight/connection strength is represented by **wij**.
* Connections can be excitatory as well as inhibitory. It would be excitatory, if the output of the neuron is same as the input, otherwise inhibitory.
* Weights should be symmetrical, i.e. **wij = wji**

The output from **Y1** going to **Y2**, **Yi** and **Yn** have the weights **w12**, **w1i** and **w1n** respectively. Similarly, other arcs have the weights on them.

**Training Algorithm:**

During training of discrete Hopfield network, weights will be updated. As we know that we can have the binary input vectors as well as bipolar input vectors. Hence, in both the cases, weight updates can be done with the following relation



**Research Method:**

* **Case 1** − Binary input patterns

For a set of binary patterns **s(p), p = 1 to P**

Here, **s(p) = s1(p), s2(p),..., si(p),..., sn(p)**

Weight Matrix is given by

wij=∑p=1P[2si(p)−1][2sj(p)−1]fori≠jwij=∑p=1P[2si(p)−1][2sj(p)−1]fori≠j

* **Case 2** − Bipolar input patterns

For a set of binary patterns **s(p), p = 1 to P**

Here, **s(p) = s1(p), s2(p),..., si(p),..., sn(p)**

Weight Matrix is given by

wij=∑p=1P[si(p)][sj(p)]fori≠j

**Source code:**

function varargout = hopfieldNetwork(varargin)

% Begin initialization code

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @hopfieldNetwork\_OpeningFcn, ...

'gui\_OutputFcn', @hopfieldNetwork\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code

% --- Executes just before hopfieldNetwork is made visible.

function hopfieldNetwork\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to hopfieldNetwork (see VARARGIN)

% Choose default command line output for hopfieldNetwork

handles.output = hObject;

N = str2num(get(handles.imageSize,'string'));

handles.W = [];

handles.hPatternsDisplay = [];

% Update handles structure

guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.

function varargout = hopfieldNetwork\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in reset.

function reset\_Callback(hObject, eventdata, handles)

% cleans all data and enables the change of the number of neurons used

for n=1 : length(handles.hPatternsDisplay)

delete(handles.hPatternsDisplay(n));

end

handles.hPatternsDisplay = [];

set(handles.imageSize,'enable','on');

handles.W = [];

guidata(hObject, handles);

function imageSize\_Callback(hObject, eventdata, handles)

% hObject handle to imageSize (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

num = get(hObject,'string');

n = str2num(num);

if isempty(n)

num = '32';

set(hObject,'string',num);

end

if n > 32

warndlg('It is strongly recomended NOT to work with networks with more then 32^2 neurons!','!! Warning !!')

end

% --- Executes during object creation, after setting all properties.

function imageSize\_CreateFcn(hObject, eventdata, handles)

% hObject handle to imageSize (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc

set(hObject,'BackgroundColor','white');

else

set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));

end

% --- Executes on button press in loadIm.

function loadIm\_Callback(hObject, eventdata, handles)

[fName dirName] = uigetfile('\*.bmp;\*.tif;\*.jpg;\*.tiff');

if fName

set(handles.imageSize,'enable','off');

cd(dirName);

im = imread(fName);

N = str2num(get(handles.imageSize,'string'));

im = fixImage(im,N);

imagesc(im,'Parent',handles.neurons);

colormap('gray');

end

% --- Executes on button press in train.

function train\_Callback(hObject, eventdata, handles)

Npattern = length(handles.hPatternsDisplay);

if Npattern > 9

msgbox('more then 10 paterns isn''t supported!','error');

return

end

im = getimage(handles.neurons);

N = get(handles.imageSize,'string');

N = str2num(N);

W = handles.W; %weights vector

avg = mean(im(:)); %removing the cross talk part

if ~isempty(W)

%W = W +( kron(im,im))/(N^2);

W = W + ( kron(im-avg,im-avg))/(N^2)/avg/(1-avg);

else

% W = kron(im,im)/(N^2);

W = ( kron(im-avg,im-avg))/(N^2)/avg/(1-avg);

end

% Erasing self weight

ind = 1:N^2;

f = find(mod(ind,N+1)==1);

W(ind(f),ind(f)) = 0;

handles.W = W;

% Placing the new pattern in the figure...

xStart = 0.01;

xEnd = 0.99;

height = 0.65;

width = 0.09;

xLength = xEnd-xStart;

xStep = xLength/10;

offset = 4-ceil(Npattern/2);

offset = max(offset,0);

y = 0.1;

if Npattern > 0

for n=1 : Npattern

x = xStart+(n+offset-1)\*xStep;

h = handles.hPatternsDisplay(n);

set(h,'units','normalized');

set(h,'position',[x y width height]);

end

x = xStart+(n+offset)\*xStep;

h = axes('units','normalized','position',[x y width height]);

handles.hPatternsDisplay(n+1) = h;

imagesc(im,'Parent',h);

else

x = xStart+(offset)\*xStep;

h = axes('units','normalized','position',[x y width height]);

handles.hPatternsDisplay = h;

end

imagesc(im,'Parent',h);

set(h, 'YTick',[],'XTick',[],'XTickMode','manual','Parent',handles.learnedPaterns);

guidata(hObject, handles);

% --- Executes on button press in addNoise.

function addNoise\_Callback(hObject, eventdata, handles)

im = getimage(handles.neurons);

% N = get(handles.imageSize,'string');

% N = floor(str2num(N)/2)+1;

noisePercent = get( handles.noiseAmount, 'value' );

N = round( length(im(:))\* noisePercent );

N = max(N,1); %minimum change one neuron

ind = ceil(rand(N,1)\*length(im(:)));

% im(ind) = -1\*im(ind); %!!!!

im(ind) = ~im(ind);

imagesc(im,'Parent',handles.neurons);

colormap('gray');

% --- Executes on button press in run.

function run\_Callback(hObject, eventdata, handles)

im = getimage(handles.neurons);

[rows cols] = size(im);

if rows ~= cols

msgbox('I don''t support non square images','error');

return;

end

N = rows;

W = handles.W;

if isempty(W)

msgbox('No train data - doing nothing!','error');

return;

end

%figure; imagesc(W)

mat = repmat(im,N,N);

mat = mat.\*W;

mat = im2col(mat,[N,N],'distinct');

networkResult = sum(mat);

networkResult = reshape(networkResult,N,N);

im = fixImage(networkResult,N);

imagesc(im,'Parent',handles.neurons);

% hObject handle to run (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

function im = fixImage(im,N)

% if isrgb(im)

if length( size(im) ) == 3

im = rgb2gray(im);

end

im = double(im);

m = min(im(:));

M = max(im(:));

im = (im-m)/(M-m); %normelizing the image

im = imresize(im,[N N],'bilinear');

%im = (im > 0.5)\*2-1; %changing image values to -1 & 1

im = (im > 0.5); %changing image values to 0 & 1

% --- Executes on slider movement.

function noiseAmount\_Callback(hObject, eventdata, handles)

% hObject handle to noiseAmount (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

percent = get(hObject,'value');

percent = round(percent\*100);

set(handles.noisePercent,'string',num2str(percent));

% Hints: get(hObject,'Value') returns position of slider

% get(hObject,'Min') and get(hObject,'Max') to determine range of slider

% --- Executes during object creation, after setting all properties.

function noiseAmount\_CreateFcn(hObject, eventdata, handles)

% hObject handle to noiseAmount (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background, change

% 'usewhitebg' to 0 to use default. See ISPC and COMPUTER.

usewhitebg = 1;

if usewhitebg

set(hObject,'BackgroundColor',[.9 .9 .9]);

else

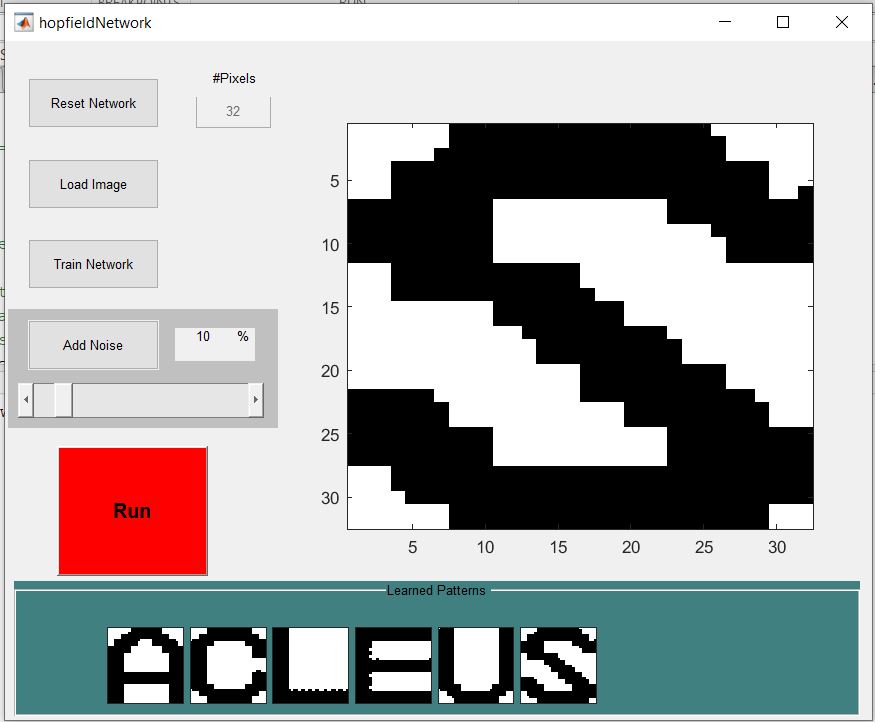
set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));

end

**Results:**

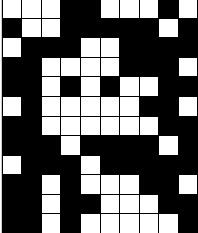
**Training Process**

I trained the letters **A, C, L, E, U, S**

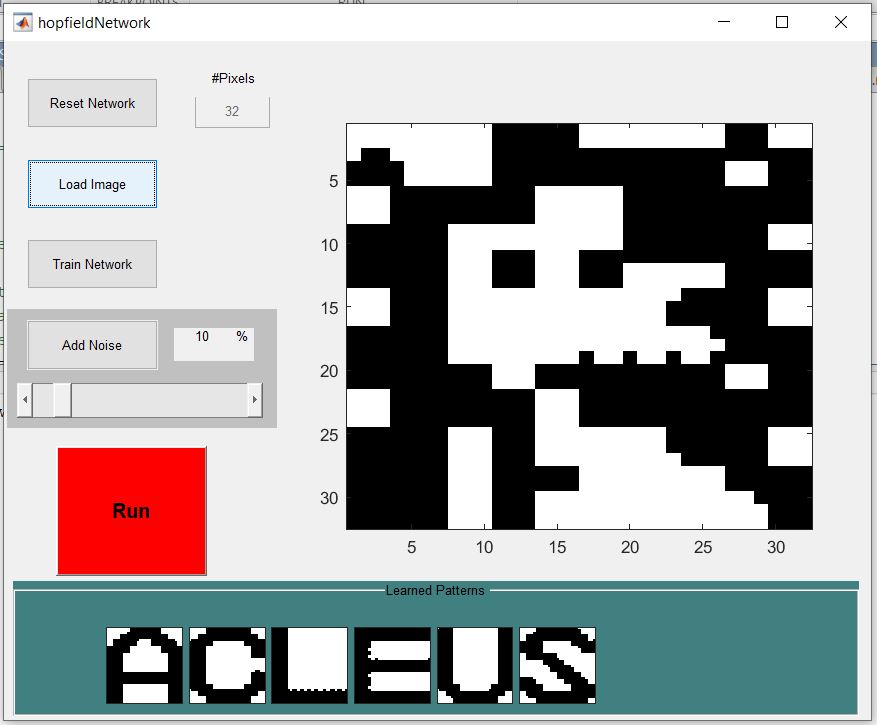
****

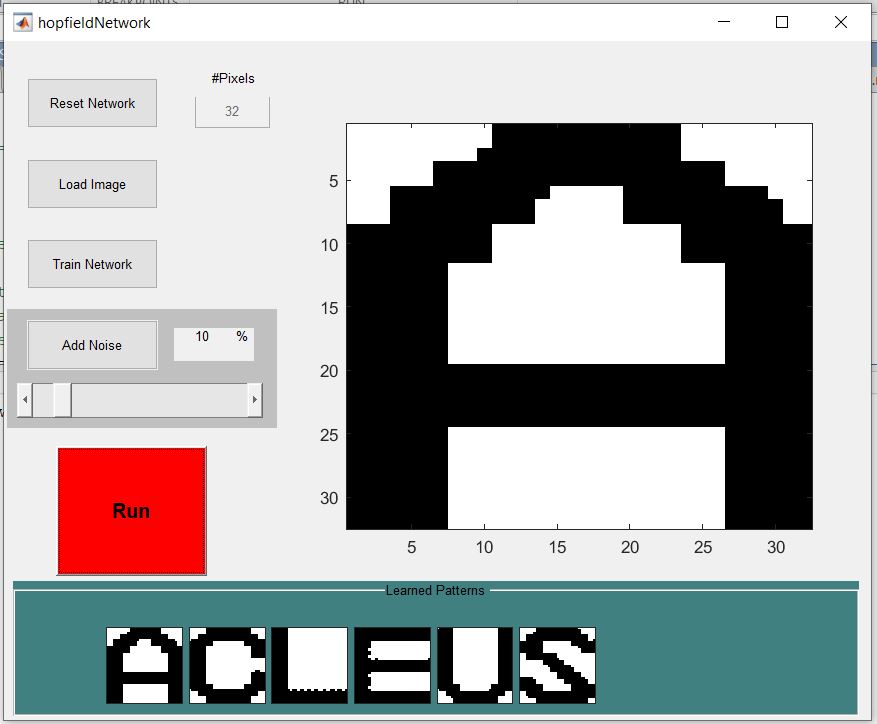
**Testing Process:**

**Load Image 1:**



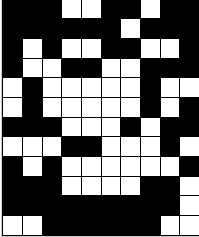
**Input: Image 1 Output: A**

****

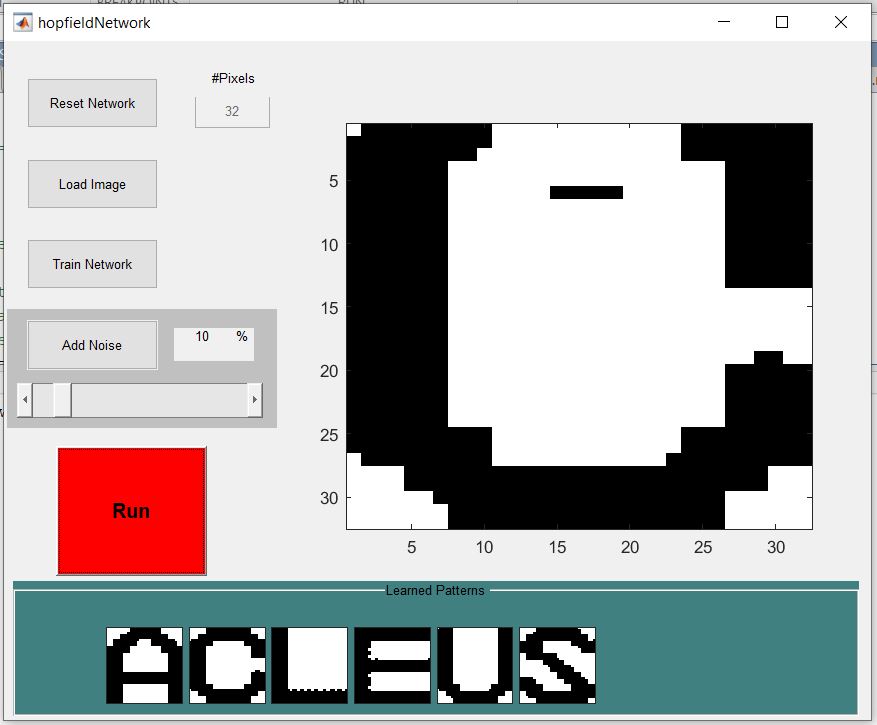
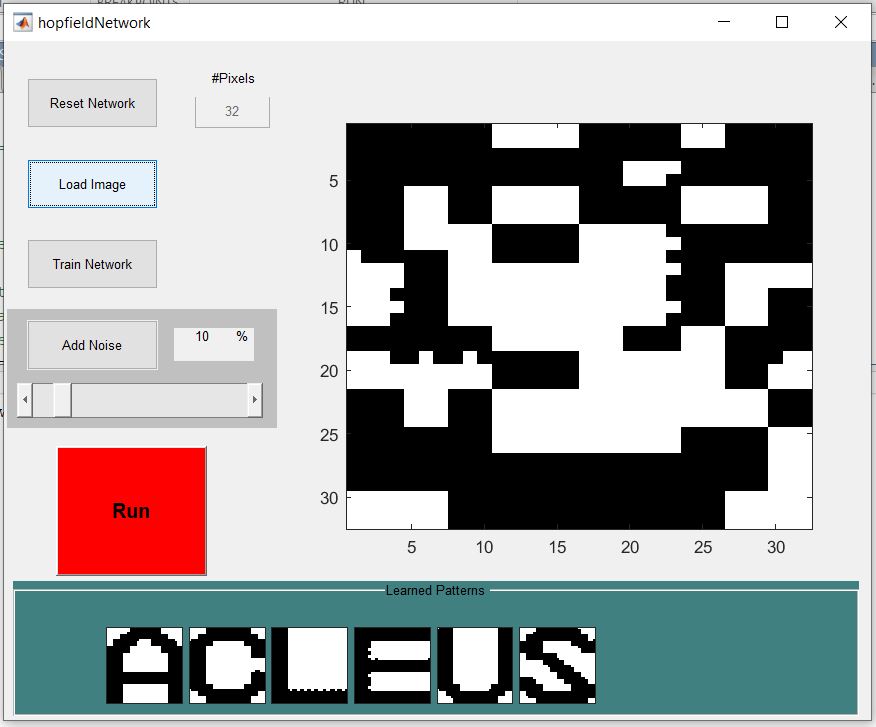


Thus image 1 was identified as ‘A’ by the Hopfield network.

**Load Image 2:**

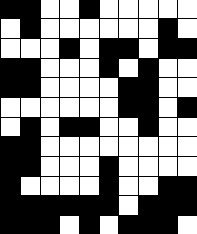


**Input: Image 2 Output: C**

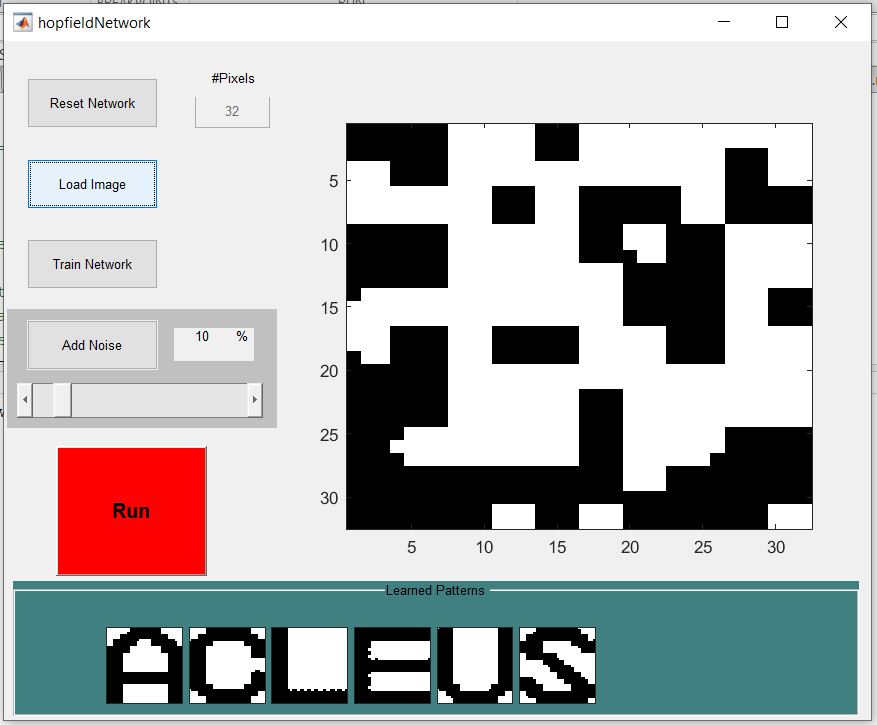
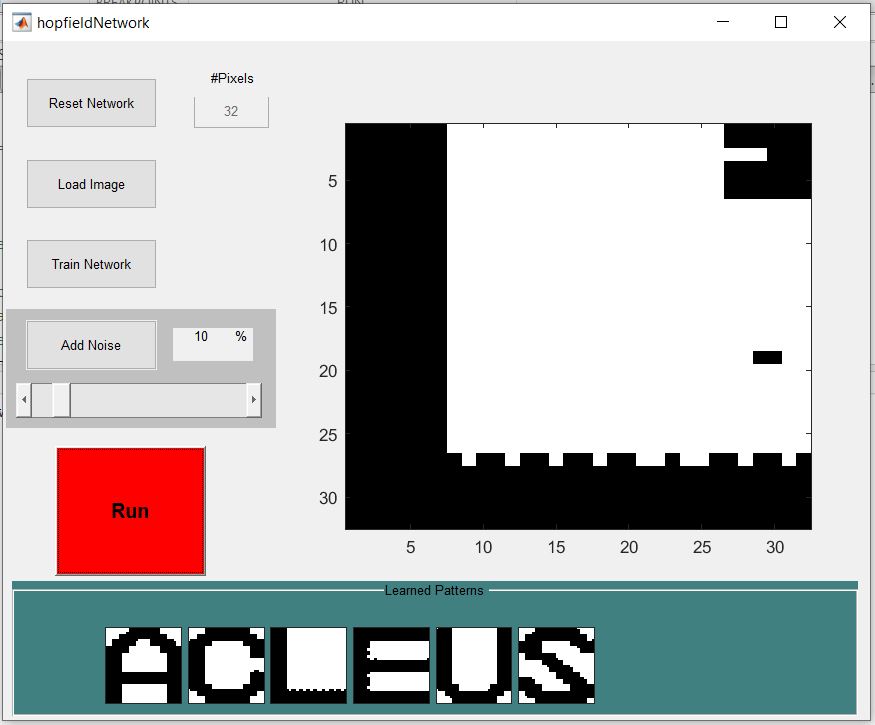


Thus image 2 was identified as ‘C’ by the Hopfield network.

**Load Image 3:**



**Input: Image 3 Output: L**

** **

Thus image 3 was identified as ‘L’ by the Hopfield network.