## Contents


- Loop to test if network will properly identify noisy E and Fs
- Plotting of original E and F, creation of examples of noisy E and F


```
%Project 1, Perceptron
%David Melanson

%%Input Section
clear all; clc; cla; clf; close all;
format short
rng('shuffle')
num = randi(randi(100));
for i = 1:num
    rng('shuffle')
end

% Binary inputs for letters E and F
Pe = [
    1 1 1 1 1, ...
    1 0 0 0 0, ...
    1 1 1 1 0, ...
    1 0 0 0 0, ...
    1 1 1 1 1
    ]';

Pf = [
    1 1 1 0 0, ...
    1 0 0 0 0, ...
    1 1 1 0 0, ...
    1 0 0 0 0, ...
    1 0 0 0 0
    ]';

P = cat(2, Pe, Pf);

% E target is 1, F is 0
T = [1 0];
%Input layer
[R, ~] = size(P);
[S, Q] = size(T);

%Initialize network parameters
%Change_Marker

%Initialize weights randomly
W = rand(S,R);
Wp = W(:, 1:R-1);

%watchon;
%cla;
E=1;


%sum squared error performance function
while (sse(E))
    Ai = hardlim(W*P);
    Ei = T-Ai;
```

```
    dWq = learnp(W, P, [], [], [], [], Ei, [], [], [], [], []);
    W = W+dWq;
    A = hardlim(W*P);
    E = T-A;
end
%watchoff;
fprintf('\nTarget is ')
fprintf('%2i', T)
fprintf('\nSolution reached ')
fprintf('%2i', A)
fprintf('\nWith weights\n')
fprintf('%1f %1f %1f %1f %1f\n', W)

Wp = W(:, 1:R-1);
Bp = W(:, R);
```

```
Target is  1 0
Solution reached  1 0
With weights
-0.392244 -0.440313 -0.283019 1.094566 1.702341
-0.701759 0.791722 0.968348 0.177816 0.933068
-0.752131 -0.277577 -0.565843 1.351970 0.428026
-0.739294 0.306217 0.988244 0.810358 0.009244
-0.796732 1.311096 1.708584 1.299051 1.760909
```

## Loop to test if network will properly identify noisy E and Fs

goes through 10000 iterations, flipping one more bit every loop until

```
%its tested up to 20 bits flipped

numBits = 20;
iterations = 10000;
identifiedE = [iterations, numBits];
identifiedF = [iterations, numBits];
for j = 1:numBits
    for i = 1:iterations

        testE = makeNoisy(Pe, j);
        resultE = hardlim(W*testE);
        if resultE > 0
            identifiedE(i, j) = 1;
        else
            identifiedE(i, j) = 0;
        end
        testF = makeNoisy(Pf, j);
        resultF = hardlim(W*testF);
        if resultF == 0
            identifiedF(i, j) = 1;
        else
            identifiedF(i, j) = 0;
        end
    end
end

identifiedE = sum(identifiedE)./(iterations/100);
identifiedF = sum(identifiedF)./(iterations/100);
```

## Plotting of original E and F, creation of examples of noisy E and F

and graphing of results of the above identifications

```
figure;
hintonw(reshape(Pe, [5,5])')
title('Original E')

figure;
for i = 1:4
    for j = 1:5
        num = (5*(i-1))+j;
        subplot(4, 5, num);
        example = makeNoisy(Pe, num);
        example = reshape(example, [5,5])';
        hintonw(example);
        title([num2str(num), ' bit(s) flipped'])
    end
end
sgtitle('E with bits flipped')

figure;
hintonw(reshape(Pf, [5,5])')
title('Original F')


figure;
for i = 1:4
    for j = 1:5
        num = j+(5*(i-1));
        subplot(4, 5, num);
        example = makeNoisy(Pf, num);
        example = reshape(example, [5,5])';
        hintonw(example);
        title([num2str(num), ' bit(s) flipped'])
    end
end
sgtitle('F with bits flipped');


figure;
subplot(1,2,1);
bar(1:numBits, identifiedE);
title('Perceptron Identification of the Letter E')
xlabel('Number of bits flipped')
ylabel('Correct Identifications (%)')
subplot(1,2,2)
bar(1:numBits, identifiedF);
title('Perceptron Identification of the Letter F')
xlabel('Number of bits flipped')
ylabel('Correct Identifications (%)')



fprintf('\n\nFor the letter "E":\n\n\n')
for i = 1:numBits
    fprintf('Identified %1g percent of inputs with %d bit(s) flipped.\n', identifiedE(i), i);
end
fprintf('\n\nFor the letter "F":\n\n\n')
for i = 1:numBits
```

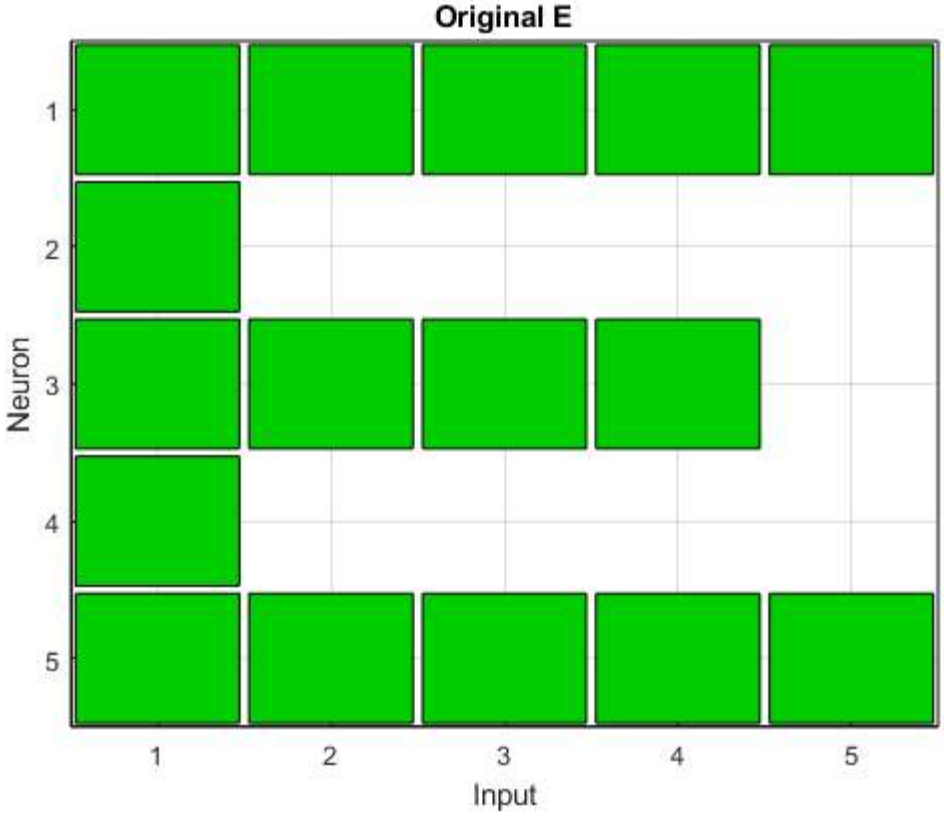
```
    fprintf('Identified %1g percent of inputs with %d bit(s) flipped.\n', identifiedF(i), i);
end

figure;
hintonw(reshape(makeNoisy2(Pf, 1), [5,5])')
title('F with bits flipped')


%function that takes an array and a number of bits to flip and does that
function retArr = makeNoisy(passArr, numBits)
    diffEl = 0;
    retArr = passArr;
    while(diffEl ~= numBits)
        row = randi(25);
        if passArr(row, 1) == 0
            retArr(row, 1) = 1;
        else
            retArr(row, 1) = 0;
        end
        diffEl = sum(sum(retArr ~= passArr));
    end
end

%function that takes an array and a number of bits to flip and does that
function retArr = makeNoisy2(passArr, numBits)
    diffEl = 0;
    retArr = passArr;
    while(diffEl ~= numBits)
        row = randi(25);
        retArr(row, 1) = -1*(passArr(row, 1));
        diffEl = sum(sum(retArr ~= passArr));
    end
end
```

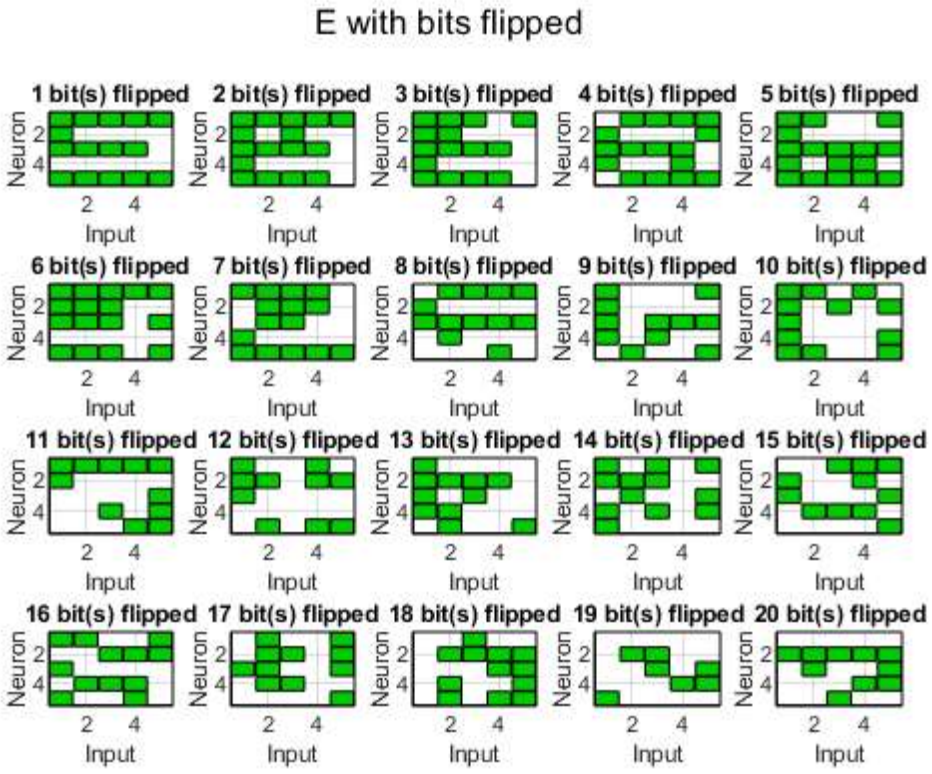
For the letter "E":

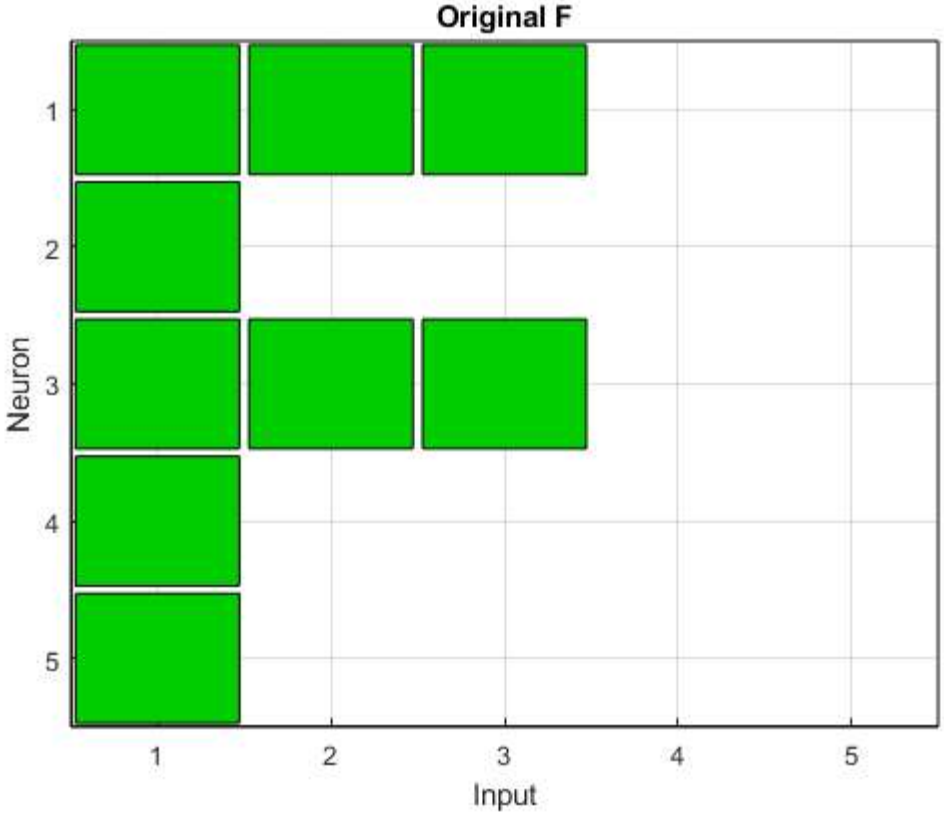
```
Identified 100 percent of inputs with 1 bit(s) flipped.
Identified 100 percent of inputs with 2 bit(s) flipped.
Identified 100 percent of inputs with 3 bit(s) flipped.
Identified 99.68 percent of inputs with 4 bit(s) flipped.
Identified 99.18 percent of inputs with 5 bit(s) flipped.
Identified 99.32 percent of inputs with 6 bit(s) flipped.
Identified 99.12 percent of inputs with 7 bit(s) flipped.
Identified 98.97 percent of inputs with 8 bit(s) flipped.
Identified 98.64 percent of inputs with 9 bit(s) flipped.
Identified 98.89 percent of inputs with 10 bit(s) flipped.
Identified 98.69 percent of inputs with 11 bit(s) flipped.
Identified 98.34 percent of inputs with 12 bit(s) flipped.
Identified 98.92 percent of inputs with 13 bit(s) flipped.
Identified 99.01 percent of inputs with 14 bit(s) flipped.
Identified 99.02 percent of inputs with 15 bit(s) flipped.
Identified 99 percent of inputs with 16 bit(s) flipped.
Identified 99.6 percent of inputs with 17 bit(s) flipped.
Identified 99.9 percent of inputs with 18 bit(s) flipped.
Identified 100 percent of inputs with 19 bit(s) flipped.
Identified 100 percent of inputs with 20 bit(s) flipped.
```

For the letter "F":

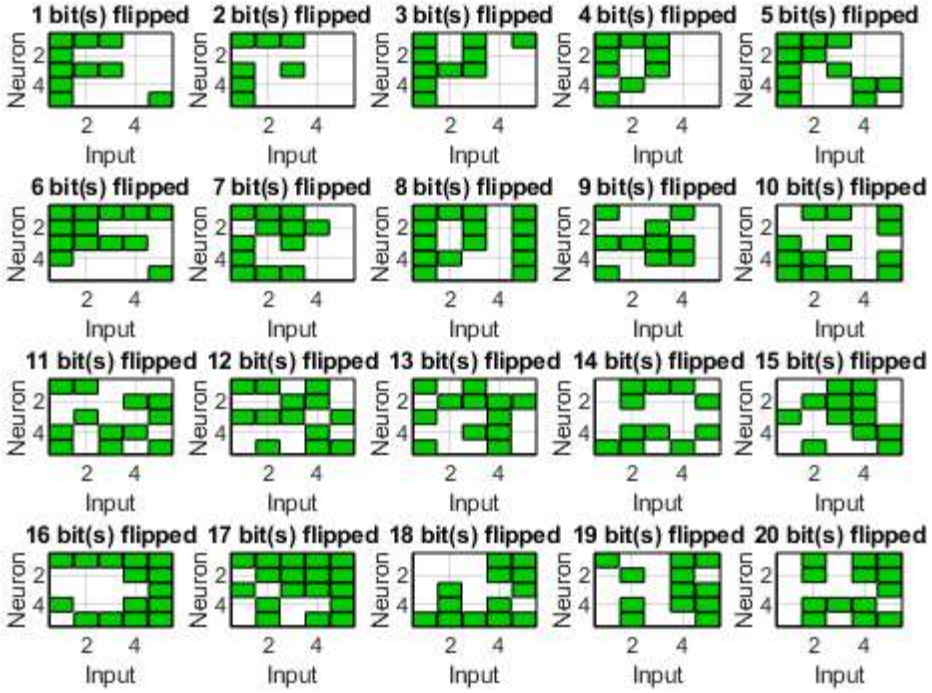
Identified 100 percent of inputs with 1 bit(s) flipped.
Identified 100 percent of inputs with 2 bit(s) flipped.
Identified 99.91 percent of inputs with 3 bit(s) flipped.
Identified 96.19 percent of inputs with 4 bit(s) flipped.
Identified 80.82 percent of inputs with 5 bit(s) flipped.
Identified 50.99 percent of inputs with 6 bit(s) flipped.
Identified 23.11 percent of inputs with 7 bit(s) flipped.
Identified 7.54 percent of inputs with 8 bit(s) flipped.
Identified 1.52 percent of inputs with 9 bit(s) flipped.
Identified 0.18 percent of inputs with 10 bit(s) flipped.
Identified 0.01 percent of inputs with 11 bit(s) flipped.
Identified 0 percent of inputs with 12 bit(s) flipped.
Identified 0 percent of inputs with 13 bit(s) flipped.
Identified 0 percent of inputs with 14 bit(s) flipped.
Identified 0 percent of inputs with 15 bit(s) flipped.
Identified 0 percent of inputs with 16 bit(s) flipped.
Identified 0 percent of inputs with 17 bit(s) flipped.
Identified 0 percent of inputs with 18 bit(s) flipped.
Identified 0 percent of inputs with 19 bit(s) flipped.
Identified 0 percent of inputs with 20 bit(s) flipped.

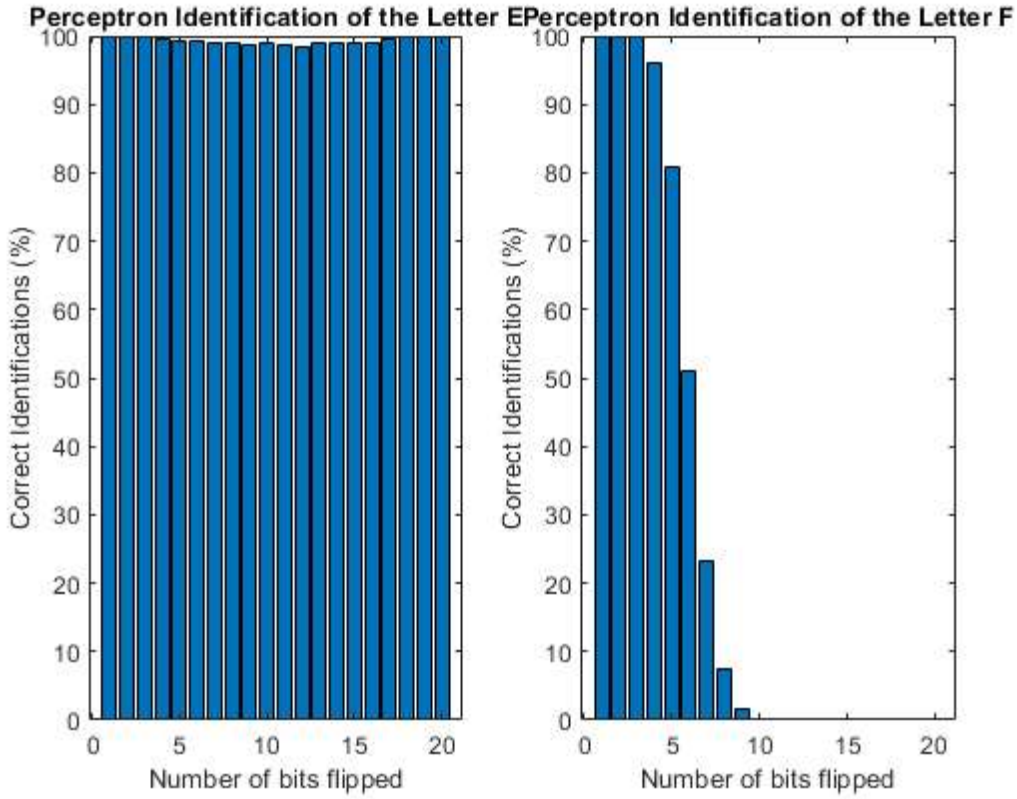

Original E
Neuron
Input
1
2
3
4
5

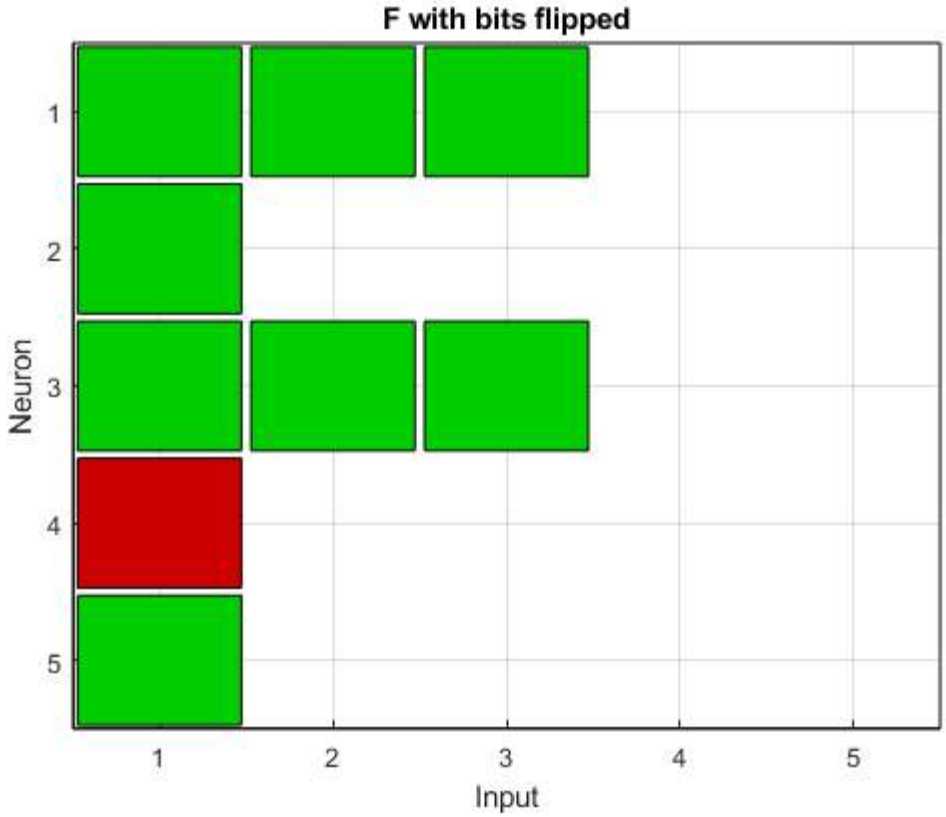

E with bits flipped
1 bit(s) flipped
2 bit(s) flipped
3 bit(s) flipped
4 bit(s) flipped
5 bit(s) flipped
6 bit(s) flipped
7 bit(s) flipped
8 bit(s) flipped
9 bit(s) flipped
10 bit(s) flipped
11 bit(s) flipped
12 bit(s) flipped
13 bit(s) flipped
14 bit(s) flipped
15 bit(s) flipped
16 bit(s) flipped
17 bit(s) flipped
18 bit(s) flipped
19 bit(s) flipped
20 bit(s) flipped
Neuron
Input




F with bits flipped






*Published with MATLAB® R2022a*