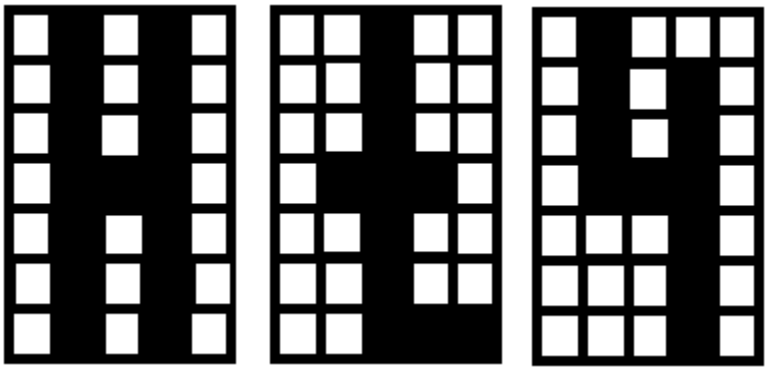
Sieć neuronowa - rozpoznawanie znaków.

Squiddy Skeith i Szymon

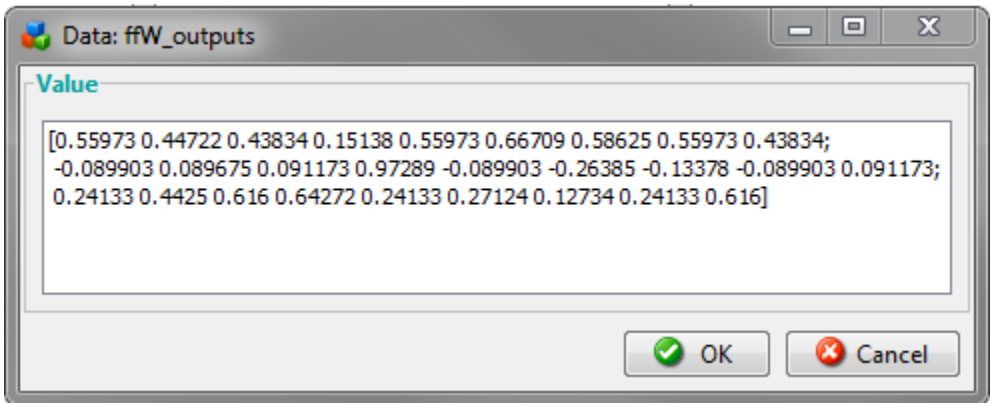
Znaki, które otrzymaliśmy do rozpoznania przez sieć:



```
We_H=[];
We_t=[];
We_4=[];
Wy_H=[0 0 1];
Wy_t=[0 1 0];
Wy_4=[1 0 0];
H=[0 1 0 1 0;0 1 0 1 0; 0 1 0 1 0 ;0 1 1 1 0; 0 1 0 1 0; 0 1 0 1 0; 0 1 0 1 0]
t=[0 0 1 0 0;0 0 1 0 0; 0 0 1 0 0 ;0 1 1 1 0; 0 0 1 0 0; 0 0 1 0 0; 0 1 1 1 1]
cztery=[0 0 0 0 0;0 1 1 1 0;0 0 0 1 0;0 0 0 1 0;0 0 0 1 0;0 0 0 1 0;0 0 0 1 0;0 0 0 0 0]

H =
    0    1    0    1    0
    0    1    0    1    0
    0    1    0    1    0
    0    1    1    1    0
    0    1    0    1    0
    0    1    0    1    0
    0    1    0    1    0

t =
    0    0    1    0    0
    0    0    1    0    0
    0    0    1    0    0
    0    1    1    1    0
    0    0    1    0    0
    0    0    1    0    0
    0    1    1    1    1

cztery =
    0    0    0    0    0
    0    1    1    1    0
    0    0    0    1    0
    0    0    0    1    0
    0    0    0    1    0
    0    0    0    1    0
    0    0    0    0    0
```

```
for i=1:1:5
   for j=1:1:7
      We_H=[We_H;H(j,i)];
      We_t=[We_t;t(j,i)];
      We_4=[We_4;cztery(j,i)];
   end
end

We_H1=We_H;
We_H1(10,1)=0
We_H2=We_H;
We_H2(20,1)=1;
We_t1=We_H;
We_t1(25,1)=1
We_t2=We_H;
We_t2(24,1)=0;
We_41=We_H;
We_41(18,1)=1
We_42=We_H;
We_42(20,1)=1;
Wy_H=[0 0 1]'
Wy_t=[0 1 0]'
Wy_4=[1 0 0]'

% W - wzór
W=[We_H We_H1 We_H2 We_t We_t1 We_t2 We_4 We_41 We_42];
out=[Wy_H Wy_H Wy_H Wy_t Wy_t Wy_t Wy_4 Wy_4 Wy_4];
```

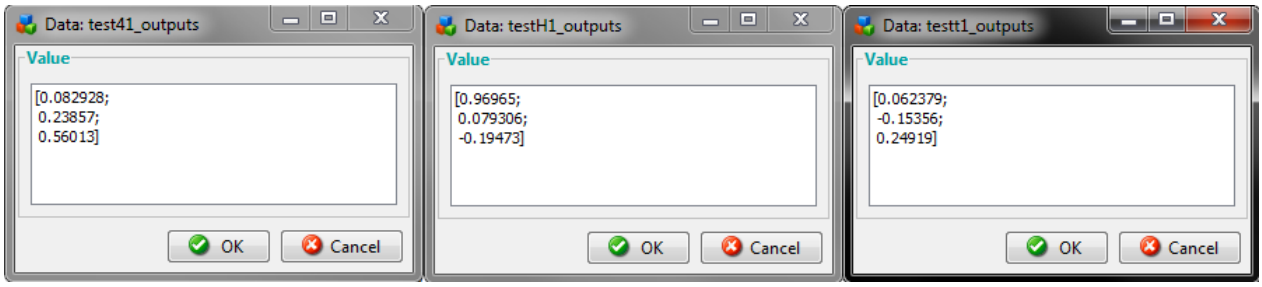
Następnie stworzyliśmy sieć feed forward i otrzymaliśmy następującą odpowiedź:



Aby wyniki równały się 0 lub 1 przekonwertowaliśmy je w podwójnej funkcji for.

```
for i=1:1:3
   for j=1:1:9
      if ffW_outputs(i,j)>0.5;
         ffW_outputs(i,j)=1;
      else
         ffW_outputs(i,j)=0;
      end
   end
end
ffW_outputs =

   1   0   0   0   1   1   1   1   0

   0   0   0   1   0   0   0   0   0

   0   0   1   1   0   0   0   0   1
```

Później przetestowaliśmy sieci przy pomocy niedokończonych znaków:



Tworzenie sieci Hopfielda, pracującej na wartościach -1 i 1:

```
for i=1:1:length(We_H)
   if(We_H(i)==0)
      We_H(i)=-1;
   end
end

for i=1:1:length(We_4)
   if(We_4(i)==0)
      We_4(i)=-1;
   end
end

for i=1:1:length(We_t)
   if(We_t(i)==0)
      We_t(i)=-1;
   end
end
```

Tworzenie macierzy:

```
We_Hop=[We_H We_4 We_t]';
We_Hop1=[We_H1 We_41 We_t1]';
Wy_Hop=[1 -1 -1;-1 1 -1; -1 -1 1]
```

Po utworzeniu sieci Hopfielda o nazwie znakihop i wprowadzenia komendy wynik=sim(znakihop,3,[],We_Hop) otrzymaliśmy wynik działania siec hopfielda. Wynik ten ma taką samą wartość, co wektor którym go uczyliśmy - wynik=We_Hop. Następnie wprowadziliśmy do tej sieci wadliwe wektory.

```
wynik_4=sim(znakihop,[1 15],[],We_41)
wynik_H=sim(znakihop,[1 15],[],We_H1)
wynik_t=sim(znakihop,[1 15],[],We_t1)
```

Wnioski:

Sieć Hopfielda świetnie radzi sobie z rozpoznawaniem znaków, przy wykonaniu zaledwie trzech taktów. Sieć Hopfielda jest łatwiejsza w obsłudze niż feed forward, lecz są równie skuteczne.