

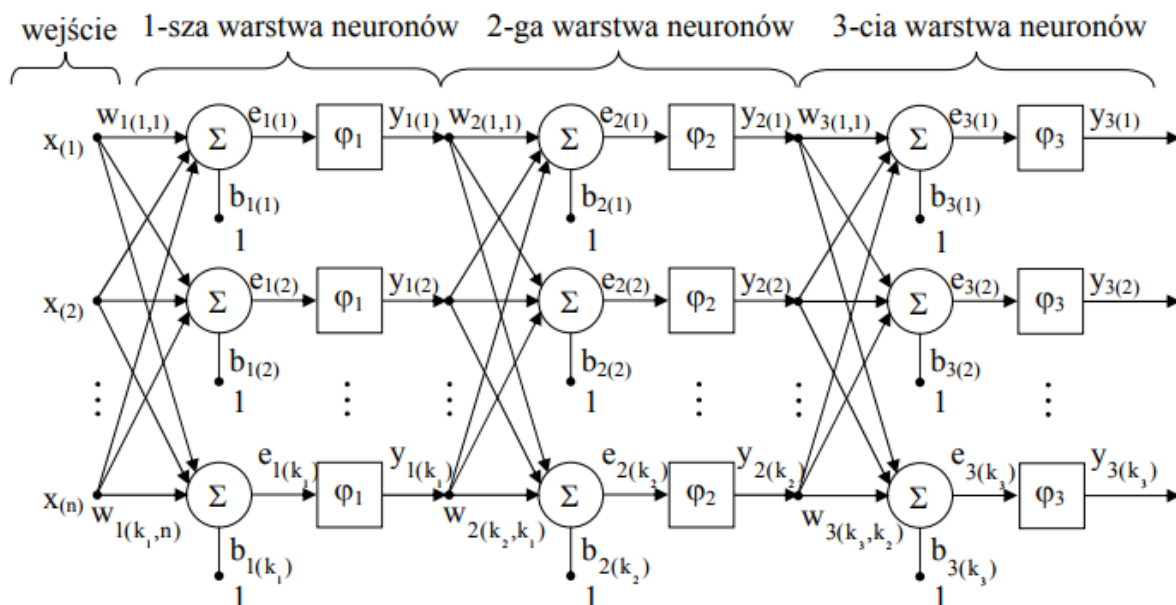
Podstawy sztucznej inteligencji Sprawozdanie ćwiczenie 3 Budowa i działanie sieci wielowarstwowej

Cel:

Celem ćwiczenia było poznanie budowy i działania wielowarstwowych sieci neuronowych poprzez uczenie z użyciem algorytmu wstecznej propagacji błędu rozpoznawania konkretnych liter alfabetu.

Wiedza:

W zadaniu wykorzystałam **trójwarstwową sieć neuronową**. Składa się ona ze zbioru neuronów logicznie rozmieszczonych w trzech warstwach, gdzie wyróżnia się *warstwę wejściową*, zbierającą dane, *warstwę wyjściową*, wysyłającą sygnał oraz pośredniczące pomiędzy warstwami wejściową i wyjściową *warstwy ukryte*. W sieci neuronowej trójwarstwowej występuje warstwa wejściowa, dwie warstwy ukryte oraz warstwa wyjściowa.



Schemat trójwarstwowej sieci neuronowej.

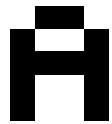
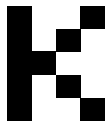
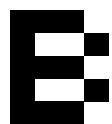
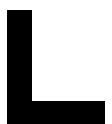
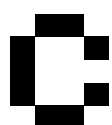
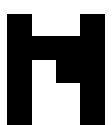
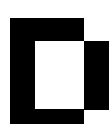
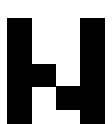

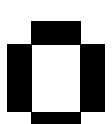

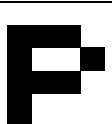

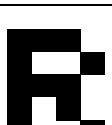
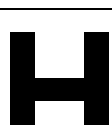
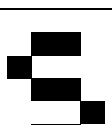



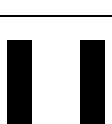
Jako algorytm uczenia zastosowałam **algorytm wstecznej propagacji błędu**. Metoda ta służy do obliczenia optymalnego zestawu wag.

Ogólny schemat procesu trenowania algorytmem wstecznej propagacji:

1. Ustalenie topologii sieci.
2. Losowa inicjacja wag na małe wartości.
3. Dla danych wejściowych ustalamy wartości wyjściowe, dla każdej warstwy.
4. Każdy neuron wyjściowy oblicza swój błąd, oparty na różnicy pomiędzy obliczoną odpowiedzią a poprawnym wynikiem.
5. Błędy propagowane są do wcześniejszych warstw.
6. Każdy neuron modyfikuje wagi na podstawie wartości błędu.
7. Wykonanie kroków od 3. dla wszystkich pozostałych danych uczących. Gdy wszystkie zostaną użyte, losowa zmienia ich kolejności i wykorzystywanie powtórne.
8. Zakończenie, gdy średni błąd na danych testowych przestanie maleć.

Zadania, które wykonałam w ramach ćwiczenia:

- Utworzyłam dane uczące i testujące, zawierające 20 dużych kolejnych liter alfabetu łacińskiego w postaci tablicy 4x5 pikseli dla jednej litery

	<table><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	0	1	1	0	1	0	0	1	1	1	1	1	1	0	0	1	1	0	0	1		<table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	0	1	1	0	1	0	1	1	0	0	1	0	1	0	1	0	0	1
0	1	1	0																																								
1	0	0	1																																								
1	1	1	1																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	1	0																																								
1	1	0	0																																								
1	0	1	0																																								
1	0	0	1																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	1	1	1	0	1	0	0	1	1	1	1	0	1	0	0	1	1	1	1	0		<table><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1	1	1
1	1	1	0																																								
1	0	0	1																																								
1	1	1	0																																								
1	0	0	1																																								
1	1	1	0																																								
1	0	0	0																																								
1	0	0	0																																								
1	0	0	0																																								
1	0	0	0																																								
1	1	1	1																																								
	<table><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	0	1	0	0	1	1	0	0	0	1	0	0	1	0	1	1	0		<table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	0	1	1	1	1	1	1	0	1	1	1	0	0	1	1	0	0	1
0	1	1	0																																								
1	0	0	1																																								
1	0	0	0																																								
1	0	0	1																																								
0	1	1	0																																								
1	0	0	1																																								
1	1	1	1																																								
1	0	1	1																																								
1	0	0	1																																								
1	0	0	1																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	1	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	1	1	1	0		<table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	0	1	1	0	0	1	1	1	0	1	1	0	1	1	1	0	0	1
1	1	1	0																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	0	1																																								
1	1	1	0																																								
1	0	0	1																																								
1	0	0	1																																								
1	1	0	1																																								
1	0	1	1																																								
1	0	0	1																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	0	0	0	1	1	1	0	1	0	0	0	1	1	1	1		<table><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	0	1	0	0	1	1	0	0	1	1	0	0	1	0	1	1	0
1	1	1	1																																								
1	0	0	0																																								
1	1	1	0																																								
1	0	0	0																																								
1	1	1	1																																								
0	1	1	0																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	0	1																																								
0	1	1	0																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr></table>	1	1	1	1	1	0	0	0	1	1	1	0	1	0	0	0	1	0	0	0		<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr></table>	1	1	1	0	1	0	0	1	1	1	1	0	1	0	0	0	1	0	0	0
1	1	1	1																																								
1	0	0	0																																								
1	1	1	0																																								
1	0	0	0																																								
1	0	0	0																																								
1	1	1	0																																								
1	0	0	1																																								
1	1	1	0																																								
1	0	0	0																																								
1	0	0	0																																								
	<table><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	0	1	0	0	0	1	0	1	1	1	0	0	1	0	1	1	0		<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	1	1	0	1	0	0	1	1	1	1	0	1	0	1	0	1	0	0	1
0	1	1	0																																								
1	0	0	0																																								
1	0	1	1																																								
1	0	0	1																																								
0	1	1	0																																								
1	1	1	0																																								
1	0	0	1																																								
1	1	1	0																																								
1	0	1	0																																								
1	0	0	1																																								
	<table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr></table>	1	0	0	1	1	0	0	1	1	1	1	1	1	0	0	1	1	0	0	1		<table><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	0	1	1	0	1	0	0	0	0	1	1	0	0	0	0	1	0	1	1	0
1	0	0	1																																								
1	0	0	1																																								
1	1	1	1																																								
1	0	0	1																																								
1	0	0	1																																								
0	1	1	0																																								
1	0	0	0																																								
0	1	1	0																																								
0	0	0	1																																								
0	1	1	0																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr></table>	1	1	1	0	0	1	0	0	0	1	0	0	0	1	0	0	1	1	1	0		<table><tr><td>1</td><td>1</td><td>1</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td><td>0</td></tr></table>	1	1	1	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0
1	1	1	0																																								
0	1	0	0																																								
0	1	0	0																																								
0	1	0	0																																								
1	1	1	0																																								
1	1	1	0																																								
0	1	0	0																																								
0	1	0	0																																								
0	1	0	0																																								
0	1	0	0																																								
	<table><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	0	0	0	1	0	0	0	1	1	0	0	1	1	1	1	1		<table><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td><td>0</td></tr></table>	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	0	1	1	0
1	1	1	1																																								
0	0	0	1																																								
0	0	0	1																																								
1	0	0	1																																								
1	1	1	1																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	0	1																																								
1	0	0	1																																								
0	1	1	0																																								

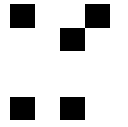
Tablice z graficznym i binarnym przedstawieniem danych uczących.

Tablice składające się z 20 pól, czarnych - 1 lub białych - 0.

Czarne pole oznacza, że w danym miejscu występuje element litery.

- Implementacja
 - Zainicjowałam zmienną *start*, która reprezentuje maksymalną oraz minimalną ilość wejść do sieci.
 - Ustawiłam ilość neuronów w każdej warstwie sieci w zmiennej *wyjścia_s*
 - Wprowadziłam dane uczące w postaci binarnej zapisanej kolumnowo do zmiennej *WEJSCIE*.
 - Wygenerowałam trójwarstwową sieć neuronową przy użyciu narzędzi z biblioteki Matlaba *Neural Network Toolbox*.
 - Zainicjowałam zmienną *WYJSCIE*, której przypisywane były wyniki trafienia żądanej litery.
 - Użyte metody i zmienne w implementacji:
 - funkcja *newff(start, wyjścia_s, {'tansig', 'tansig', 'tansig'}, 'traingda')*, która tworzy sieć neuronową
 - zmienna *start*, która reprezentuje maksymalną oraz minimalną ilość wejść do sieci.
 - zmienna *wyjścia_s* z liczbą elementów wektora wyjściowego
 - parament *tansig*, określający funkcję aktywacji – tangens hiperboliczny, użyty trzykrotnie, do stworzenia trójwarstwowej sieci
 - funkcja *traingda*, która analizuje wartości wag oraz odchylenia zgodnie ze spadkiem gradientu z adaptacyjną szybkością uczenia
 - zmienna *net*, do której przypisywana jest nowo tworzona sieć
 - *net.trainParam.x* – ustawianie wartości parametrów treningu
 - dane uczące *WEJSCIE*, dane wyjściowe odpowiadające danym uczącym (1 – trafienie, 0 – błąd) *WYJSCIE*
 - funkcja *train(net, WEJSCIE, WYJSCIE)*, do uczenia sieci z wykorzystaniem danych
- Przeprowadziłam proces uczenia dla różnych wartości parametrów oraz przetestowałam działanie sieci.

W celach testowych jedną z wprowadzonych zmiennych zniekształciłam, aby sprawdzić jak sieć poradzi sobie z realizacją tego zadania.

	1 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 1 0 1 0
---	---

Graficznie przedstawiona błędna zmienna wprowadzona dla testów.

Dla różnych wartości współczynnika uczenia, algorytm wskazywał różne litery. Liczbę epok ustawiłam jednakową dla każdego testu (10000).

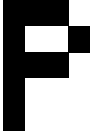
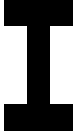
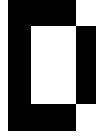
Współczynnik uczenia	0,001		0,01		0,1	
Interpretowana litera	P - 0,8770		I - 0,8987		D - 0,9221	
		1 1 1 0 1 0 0 1 1 1 1 0 1 0 0 0 1 0 0 0		1 1 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 1 1 0		1 1 1 0 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 0

Tabela przedstawiające wartości najbliższe 1 (podobieństwo do uczonej litery) dla danego współczynnika uczenia.

Przetestowałam działanie procesu uczenia dla funkcji *newff* dla losowo wybranych liter. Im wartość bliższa 1, to testowana litera jest literą szukaną przez użytkownika. Na początku wartości sprawdziłam dla współczynnika uczenia równego 0,1. Zmniejszanie współczynnika nieznaczaco zmieniało wartości. Wszystkie litery, które były wprowadzone w danych uczących są rozpoznawalne prawidłowo. W poniższej tabeli umieściłam wyniki zmiennej *efekt*.

	Wartości dla żądanych liter				
	C	F	I	L	O
A	0,0001	0,0083	0	0	0,0004
B	0,0006	0,0061	0,0007	0	0,001
C	0,9757	0,0034	0,0002	0	0,0005
D	0,0007	0,003	0,0007	0,0005	0,0012
E	0,0001	0,007	0,0002	0,0016	0,0007
F	0,0004	0,981	0,0009	0,0002	0,0007
G	0,0012	0,0009	0,0004	0,0008	0,0001
H	0,0009	0,0062	0,0001	0,0021	0,0011
I	0,0006	0,003	0,9709	0,0001	0,0018
J	0,0018	0,0022	0,0006	0,0002	0,0001
K	0,0025	0,0047	0,0005	0,0005	0,0008
L	0,001	0,0019	0,0001	0,9858	0,0009
M	0,0005	0,0013	0	0,0007	0,001
N	0,0002	0,0048	0,0002	0,0009	0,0005
O	0,0014	0,0058	0,0002	0,0006	0,9429
P	0,0011	0,0018	0,0003	0,0001	0,0054
R	0,0013	0,004	0,0002	0,0008	0,0001
S	0,0001	0,0005	0,0002	0,0001	0,003
T	0,0003	0,0018	0,0007	0,0001	0,0015
U	0,0001	0,0011	0,0001	0,0003	0,0001

Tabela przedstawia wartości zmiennej *efekt* dla losowo wybranych liter przy użyciu funkcji *newff*.

Wnioski:

Na podstawie testów można zauważyć, że program z użyciem funkcji *newff* skutecznie radzi sobie z rozpoznawaniem podanych liter. Natomiast, gdy zniekształcimy litery lub wprowadzimy literę spoza zakresu danych uczących program nie jest w stanie przyporządkować jej poprawnie, przypisuje najbardziej prawdopodobną literę, czasami w sposób losowy.

Wartość współczynnika uczenia wpływa na czas oraz precyzję działania. Im wyższy tym sieć uczy się szybciej, natomiast może generować więcej błędów.

Uczenie liter przez sieć przebiega cyfra po cyfrze reprezentacji binarnej danej litery, a następnie wartości są porównywalne z danymi testowymi. Wynikiem jest wypisanie na ekran litery, której reprezentacja binarna w największym stopniu jest podobna do reprezentacji litery w zbiorze danych uczących.

Listing kodu programu wraz z komentarzami:

```
close all; clear all; clc;
```

```
%wejścia do sieci oraz minimalne oraz maksymalne wartości wejść
```

```
start = [0 1; 0 1; 0 1; 0 1; 0 1;  
         0 1; 0 1; 0 1; 0 1; 0 1;  
         0 1; 0 1; 0 1; 0 1; 0 1;  
         0 1; 0 1; 0 1; 0 1; 0 1];
```

```
%wyjścia sieci (ilość neuronów w każdej z warstw sieci)
```

```
wyjścia_s = [40 20 20];
```

```
%użycie algorytmu newff
```

```
net = newff(start, wyjścia_s, {'tansig', 'tansig', 'tansig'}, 'traingda');
```

```
%kolumnowa reprezentacja binarna 20 dużych liter dla tablicy 4x5
```

```
%A B C D E F G H I J K L M N O P R S T U  
WEJSCIE = [0 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0 1 1;  
           1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 0;  
           1 1 1 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 0;  
           0 0 0 0 1 1 0 1 0 1 1 0 1 1 0 0 0 0 0 1;  
           1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1;  
           0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 1 0;  
           0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0;  
           1 1 1 1 0 0 0 1 0 1 0 0 1 1 1 1 1 1 0 0 1;  
           1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 1;  
           1 1 0 0 1 1 0 1 1 0 1 0 0 1 0 1 1 1 1 1 0;  
           1 1 0 0 1 1 1 1 1 0 0 0 0 1 0 0 1 1 1 0 0;  
           1 0 0 1 0 0 1 1 0 1 0 0 1 1 1 0 0 0 0 0 1;  
           1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 0 1;  
           0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0;  
           0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 1 0 0 0 0;  
           1 1 1 1 0 0 1 1 0 1 0 0 1 1 1 0 0 1 0 1 0;  
           1 1 0 1 1 1 0 1 1 0 1 1 1 1 1 0 1 1 0 0 0;  
           0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 1 1 1;  
           0 1 1 1 1 0 1 0 1 1 0 1 0 0 1 0 0 1 0 1 1;  
           1 0 0 0 1 0 0 1 0 1 1 1 1 1 0 0 1 0 0 0 0];
```

```
%A B C D E F G H I J K L M N O P R S T U  
WYJSCIE = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %A  
           0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %B  
           0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %C  
           0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %D  
           0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %E  
           0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0; %F  
           0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0; %G  
           0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0; %H  
           0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0; %I  
           0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0; %J  
           0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0; %K  
           0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0; %L  
           0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0; %M  
           0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0; %N  
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0; %O  
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0; %P  
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0; %R  
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0; %S  
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0; %T  
           0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]; %U  
%A B C D E F G H I J K L M N O P R S T U
```

%PARAMETRY TRENINGU SIECI:

net.trainParam.epochs = 10000;

%maksymalna ilosc epok

net.trainParam.mu = 0.001;

%wspolczynnik uczenia sieci

net = train(net, WEJSCIE, WYJSCIE);

%uczenie sieci

%DANE TESTOWE

test_A = [0; 1; 1; 0;

1; 0; 0; 1;

1; 1; 1; 1;

1; 0; 0; 1;

1; 0; 0; 1];

test_B = [1; 1; 1; 0;

1; 0; 0; 1;

1; 1; 1; 0;

1; 0; 0; 1;

1; 1; 1; 0];

test_C = [0; 1; 1; 0;

1; 0; 0; 1;

1; 0; 0; 0;

1; 0; 0; 1;

0; 1; 1; 0];

test_D = [1; 1; 1; 0;

1; 0; 0; 1;

1; 0; 0; 1;

1; 0; 0; 1;

1; 1; 1; 0];

test_E = [1; 1; 1; 1;

1; 0; 0; 0;

1; 1; 1; 0;

1; 0; 0; 0;

1; 1; 1; 1];

test_F = [1; 1; 1; 1;

1; 0; 0; 0;

1; 1; 1; 0;

1; 0; 0; 0;

1; 0; 0; 0];

test_G = [0; 1; 1; 0;

1; 0; 0; 0;

1; 0; 1; 1;

1; 0; 0; 1;

0; 1; 1; 0];

test_H = [1; 0; 0; 1;

1; 0; 0; 1;

1; 1; 1; 1;

1; 0; 0; 1;

1; 0; 0; 1];

test_I = [1; 1; 1; 0;

0; 1; 0; 0;

0; 1; 0; 0;

0; 1; 0; 0;

1; 1; 1; 0];

test_J = [1; 1; 1; 1;

0; 0; 0; 1;

0; 0; 0; 1;

1; 0; 0; 1;

0; 1; 1; 1];

test_K = [1; 0; 0; 1;

1; 0; 1; 0;

1; 1; 0; 0;

1; 0; 1; 0;

1; 0; 0; 1];

test_L = [1; 0; 0; 0;

1; 0; 0; 0;

1; 0; 0; 0;

1; 0; 0; 0;

1; 1; 1; 1];

test_M = [1; 0; 0; 1;


```

        1; 1; 1; 1;
        1; 0; 1; 1;
        1; 0; 0; 1;
        1; 0; 0; 1];
test_N = [1; 0; 0; 1;
        1; 0; 0; 1;
        1; 1; 0; 1;
        1; 0; 1; 1;
        1; 0; 0; 1];
test_O = [0; 1; 1; 0;
        1; 0; 0; 1;
        1; 0; 0; 1;
        1; 0; 0; 1;
        0; 1; 1; 0];
test_P = [1; 1; 1; 0;
        1; 0; 0; 1;
        1; 1; 1; 0;
        1; 0; 0; 0;
        1; 0; 0; 0];
test_R = [1; 1; 1; 0;
        1; 0; 0; 1;
        1; 1; 1; 0;
        1; 0; 1; 0;
        1; 0; 0; 1];
test_S = [0; 1; 1; 0;
        1; 0; 0; 0;
        0; 1; 1; 0;
        0; 0; 0; 1;
        0; 1; 1; 0];
test_T = [1; 1; 1; 0;
        0; 1; 0; 0;
        0; 1; 0; 0;
        0; 1; 0; 0;
        0; 1; 0; 0];
test_U = [1; 0; 0; 1;
        1; 0; 0; 1;
        1; 0; 0; 1;
        1; 0; 0; 1;
        0; 1; 1; 0];
test_BLAD= [1; 0; 0; 1;
        0; 1; 1; 0;
        0; 0; 0; 0;
        0; 0; 0; 0;
        1; 0; 1; 0];

%testowanie dzialania sieci
efekt = sim(net, H_testowe);

%szukanie najwiekszej wartosci wyjscia
maks = 1;

for i=1:1:20
    if (efekt(maks)<efekt(i))
        maks = i;
    end;
end

%wypisywanie wartosci liter - wynik
disp('A: '), disp(efekt(1)); disp('B: '), disp(efekt(2));
disp('C: '), disp(efekt(3)); disp('D: '), disp(efekt(4));
disp('E: '), disp(efekt(5)); disp('F: '), disp(efekt(6));
disp('G: '), disp(efekt(7)); disp('H: '), disp(efekt(8));
disp('I: '), disp(efekt(9)); disp('J: '), disp(efekt(10));
disp('K: '), disp(efekt(11)); disp('L: '), disp(efekt(12));
disp('M: '), disp(efekt(13)); disp('N: '), disp(efekt(14));
disp('O: '), disp(efekt(15)); disp('P: '), disp(efekt(16));
disp('R: '), disp(efekt(17)); disp('S: '), disp(efekt(18));
disp('T: '), disp(efekt(19)); disp('U: '), disp(efekt(20));

```

```
switch maks
    case 1
        disp('Wprowadzono A')
    case 2
        disp('Wprowadzono B')
    case 3
        disp('Wprowadzono C')
    case 4
        disp('Wprowadzono D')
    case 5
        disp('Wprowadzono E')
    case 6
        disp('Wprowadzono F')
    case 7
        disp('Wprowadzono G')
    case 8
        disp('Wprowadzono H')
    case 9
        disp('Wprowadzono I')
    case 10
        disp('Wprowadzono J')
    case 11
        disp('Wprowadzono K')
    case 12
        disp('Wprowadzono L')
    case 13
        disp('Wprowadzono M')
    case 14
        disp('Wprowadzono N')
    case 15
        disp('Wprowadzono O')
    case 16
        disp('Wprowadzono P')
    case 17
        disp('Wprowadzono R')
    case 18
        disp('Wprowadzono S')
    case 19
        disp('Wprowadzono T')
    case 20
        disp('Wprowadzono U')
    otherwise
        disp('Wprowadzono bledna wartosc')
end
```